SC8F073 User Manual

Enhanced flash memory 8-bit CMOS microcontrollers

Rev. 1.0.5

Please note the following CMS IP policy

＊China Micro Semicon Co., Ltd. (hereinafter referred to as the Company) has applied for patents and holds absolute legal rights and interests. The patent rights associated with the Company's MCUs or other products have not been authorized for use, and any company, organization, or individual who infringes the Company's patent rights through improper means will be subject to all possible legal actions taken by the Company to curb the infringement and to recover any damages suffered by the Company as a result of the infringement or any illegal benefits obtained by the infringer.

＊The name and logo of Cmsemicon are registered trademarks of the Company.

＊The Company reserves the right to further explain the reliability, functionality and design improvements of the products in the data sheet. However, the Company is not responsible for the use of the Specification Contents. The applications mentioned herein are for illustrative purposes only and the Company does not warrant and does not represent that these applications can be applied without further modification, nor does it recommend that its products be used in places that may cause harm to persons due to malfunction or other reasons. The Company's products are not authorized for use as critical components in lifesaving, life-sustaining devices or systems. The Company reserves the right to modify the products without prior notice. For the latest information, please visit the official website at www.mcu.com.cn.

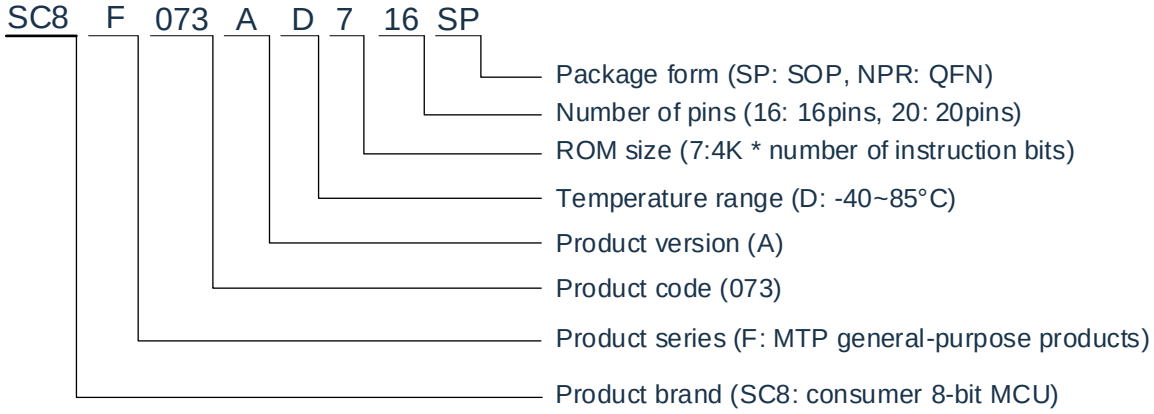
Table of Contents

1. Product Description

1.1 Features

- Memory
  - ROM: 4K×16Bit
  - Universal RAM: 256×8Bit
- 8-level stack buffer
- Short and clear instruction system (66 instructions)
- Low voltage detection circuit
- WDT
- Interrupt sources
  - 3 timer interrupts
  - RA, RB ports interrupt on change
  - Other peripheral interrupts
- 128-byte program EEPROM
  - 10,000 times rewritable
- Timers
  - 8-bit timers: TIMER0, TIMER2
  - 16-bit timer: TIMER1
  - TIMER0, TIMER1, TIMER2 can select external 32.768Khz oscillating clock sources
- 2-channel comparator module
  - COMP1 positive: RB1/resistor divider outputs
  - COMP1 negative: RB1/RB2/RB4/RB5/BG/resistor divider outputs
  - COMP2 positive: RA1/resistor divider outputs
  - COMP2 negative: RA1/RA2/RB0/RB1/BG/resistor divider outputs
- Operating voltage: V_{LVR3}~5.5V@16MHz/2T
  V_{LVR1}~5.5V@16MHz/4T
  Operating temperature: -40℃~85℃
- Internal RC oscillation: designed frequency of 16MHz
- Instruction period (single instruction or dual instruction)
- PWM module
  - 5-channel PWM with 2-channel complementary output and selectable polarity
  - 4-channel PWM common period, independent duty cycle
  - 1-channel PWM independent period, independent duty cycle
  - 10-bit PWM accuracy
- LVR can be selected from 1.8V/2V/2.5V/3V
- High accuracy 12-bit ADC
  - Built-in high accuracy 1.2V reference voltage
  - Optional internal reference sources: 2.0V/2.4V/3.0V
- 1-channel USART communication module
  - Supports synchronous master-slave and asynchronous full-duplex modes
  - Communication ports can be configured on RA3/RA4, RB0/RB1

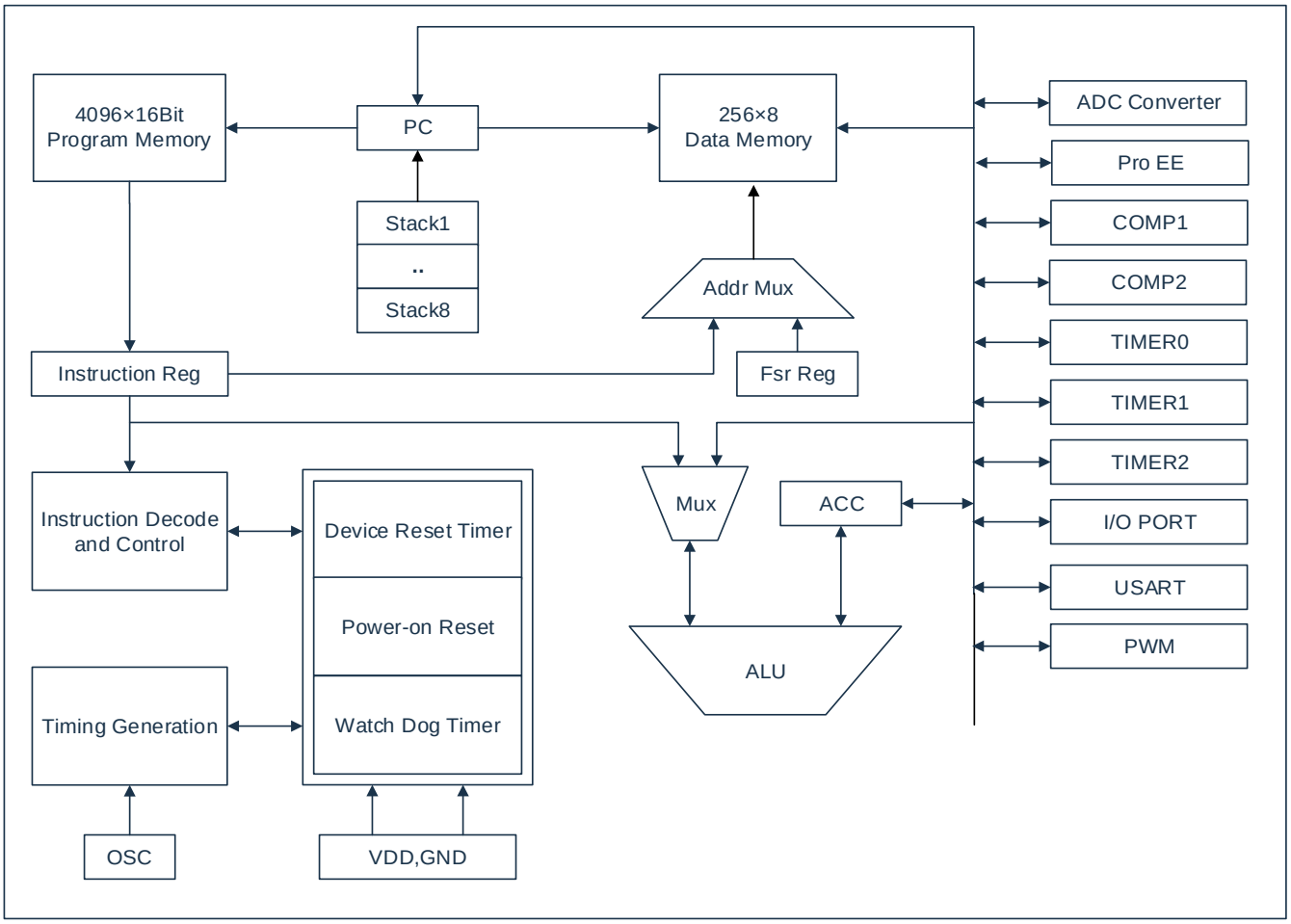
1.2 Product model list



Model description

PRODUCT	ROM	RAM	Pro EE	PWM	ACOMP	I/O	ADC	TIMER	USART	PACKAGE
SC8F073AD716SP	4K×16	256×8	128×16	5	2	14	12Bit×14	3	1	SOP16
SC8F073AD716NPR	4K×16	256×8	128×16	5	2	14	12Bit×14	3	1	QFN16
SC8F073AD720SA	4K×16	256×8	128×16	5	2	18	12Bit×18	3	1	TSSOP20
SC8F073AD720NPR	4K×16	256×8	128×16	5	2	18	12Bit×18	3	1	QFN20

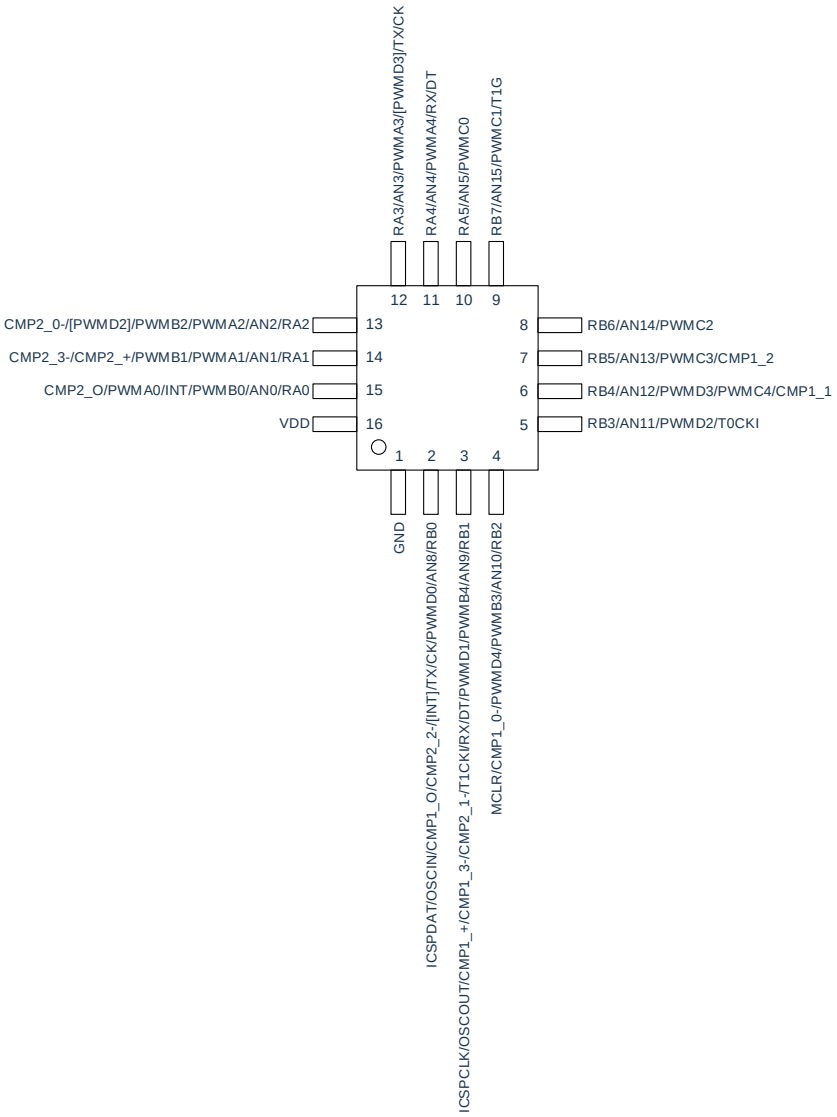
1.3 System structure diagram



1.4 Top view

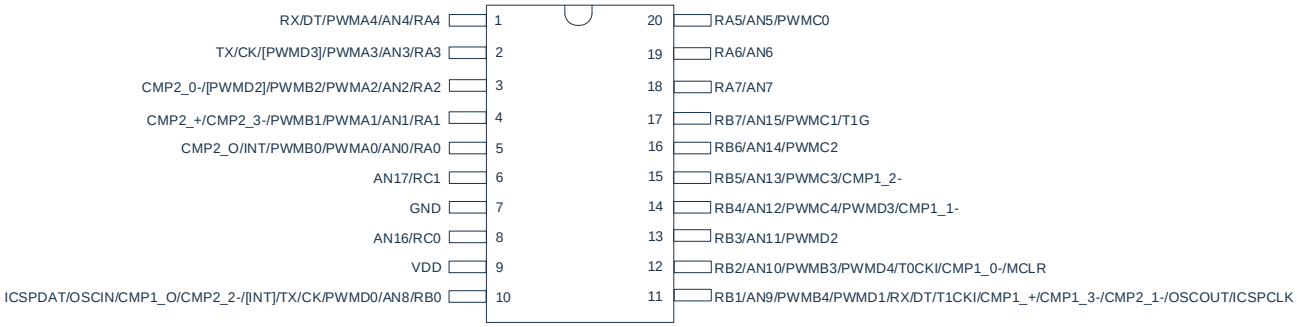
1.4.1 SC8F073AD716SP

1.4.2 SC8F073AD716NPR



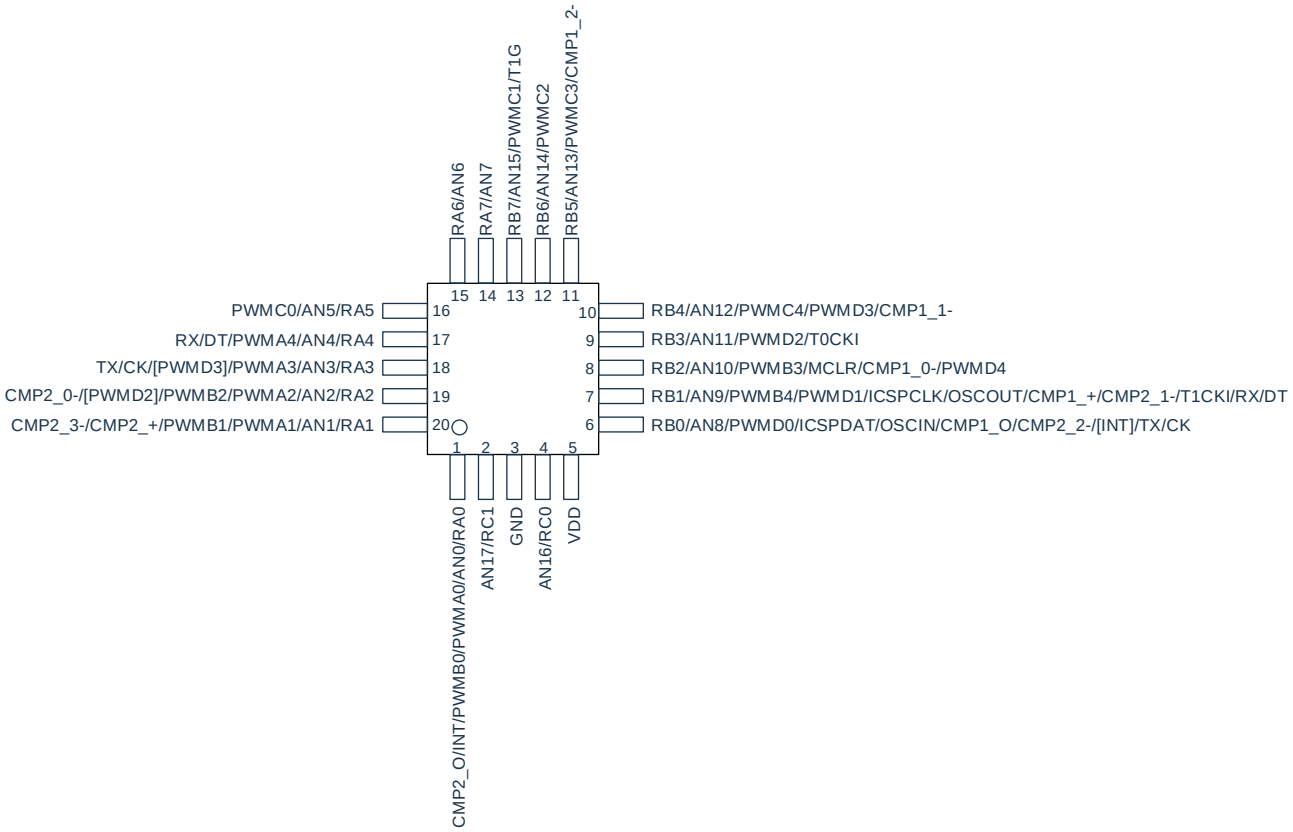
SC8F073AD716NPR

1.4.3 SC8F073AD720SA



SC8F073AD720SA

1.4.4 SC8F073AD720NPR



SC8F073AD720NPR

Note:

1) The serial port function of RA3/RA4 and RB0/RB1 is set by the CONFIG.
2) The INT function of RA0 and RB0 is set by the CONFIG.
3) The PWMD group function of RA2/RA3 and RB3/RB4 is set by the CONFIG.

SC8F073 pin description:

Pin name	IO type	Pin description
VDD,GND	P	Supply voltage input pin, ground pin
RA0-RA7	I/O	Programmable as input pin, push-pull or open-drain output pin, with pull-up and pull-down resistor function, and interrupt-on-change function.
RB0-RB7	I/O	Programmable as input pin, push-pull or open-drain output pin, with pull-up and pull-down resistor function, and interrupt-on-change function.
RC0-RC1	I/O	Programmable as input pin, push-pull output pin, with pull-up resistor function.
ICSPCLK/ICSPDAT	I/O	Programmable clock/data pin
PWMA0-PWMA4	O	PWMA group output pin
PWMB0-PWMB4	O	PWMB group output pin
PWMC0-PWMC4	O	PWMC group output pin
PWMD0-PWMD4	O	PWMD group output pin
AN0-AN17	I	12-bit ADC output pin
INT	I	External interrupt input pin
CMP1_+	I	Comparator 1 positive input pin
CMP2_+	I	Comparator 2 positive input pin
CMP1_0-, CMP1_1-, CMP1_2-, CMP1_3-	I	Comparator 1 negative input pin
CMP2_0-, CMP2_1-, CMP2_2-, CMP2_3-	I	Comparator 2 negative input pin
CMP1_O	O	Comparator 1 result output pin
CMP2_O	O	Comparator 2 result output pin
T0CKI	I	TIMER0 external clock input pin
T1CKI	I	TIMER1 external clock input pin
T1G	I	TIMER1 gate input pin
TX/CK	I/O	USART asynchronous transmit output pin/synchronous clock input/output pin
RX/DT	I/O	USART asynchronous receive input pin/synchronous data input/output pin
OSCIN/OSCOUT	I/O	32.768K crystal oscillator input pin/output pin
MCLR	I	External reset input pin

1.5 System configuration register

The System Configuration Register (CONFIG) is an MTP option for the initial condition of the MCU. It can only be written by the SC programmer and cannot be accessed or manipulated by the user. It contains the following contents.

1. WDT (watchdog selection)
   - ENABLE Enable WDT
   - DISABLE Disable WDT
2. PROTECT (encrypted)
   - DISABLE ROM code is not encrypted
   - ENABLE ROM code is encrypted, and the value read out by the programmed emulator will be uncertain after encryption
3. LVR_SEL (low-voltage detection selection)
   - 1.8V
   - 2.0V
   - 2.5V
   - 3.0V
4. F_{CPU}_DIV (instruction clock divison)
   - 4T Divided by 4, $F_{CPU}=F_{SYS}/4$
   - 2T Divided by 2, $F_{CPU}=F_{SYS}/2$
5. WDT_DIV (WDT prescaler factor control)
   - DISABLE The WDT prescaler can be selected from 1:128 via the OPTION_REG register.
   - ENABLE The WDT prescaler can be selected from 3:384 via the OPTION_REG register.
6. ICSP (Emulation port function selection)
   - ICSP The ICSPCLK and DAT ports remain as emulation ports, and all functions cannot be used.
   - NORMAL ICSPCLK, DAT ports are general function ports
7. USART_PORT_SEL (USART port selection)
   - RA3/RA4 Select RA3 as TX port, RA4 as RX port
   - RB0/RB1 Select RB0 as TX port, RB1 as RX port.
8. INT_PORT_SEL (INT port selection)
   - RA0 Select RA0 as the INT port
   - RB0 Select RB0 as the INT port
9. PWMD_PORT_SEL (PWMD group port selection)
   - PWMD0~PWMD4 Allocated to RB0/RB1/RB3/RB4/RB2
   - PWMD0~PWMD4 Allocated to RB0/RB1/RA2/RA3/RB2
10. EXT_RESET (external reset port selection)
    - DISABLE Disable external reset function, RB2 is used as a normal IO port.
    - ENABLE Enable external reset function, RB2 is used as an external reset port.

1.6 Online serial programming

The microcontroller can be programmed serially in the final application circuit. Programming can be done simply with the following 4 lines.

- Power line
- Ground line
- Data line
- Clock line

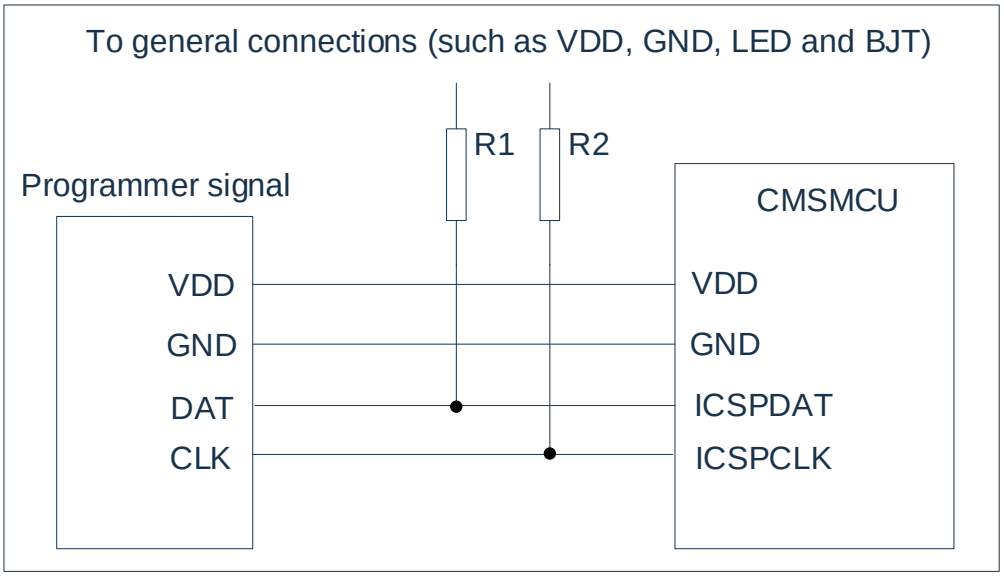


Figure 1-1: Typical connection for online serial programming

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

1.7 Integrated development environment

- On-Chip Debug (OCD), ISP
- 4 hardware breakpoints
- Software reset, pause, single step, run, etc.

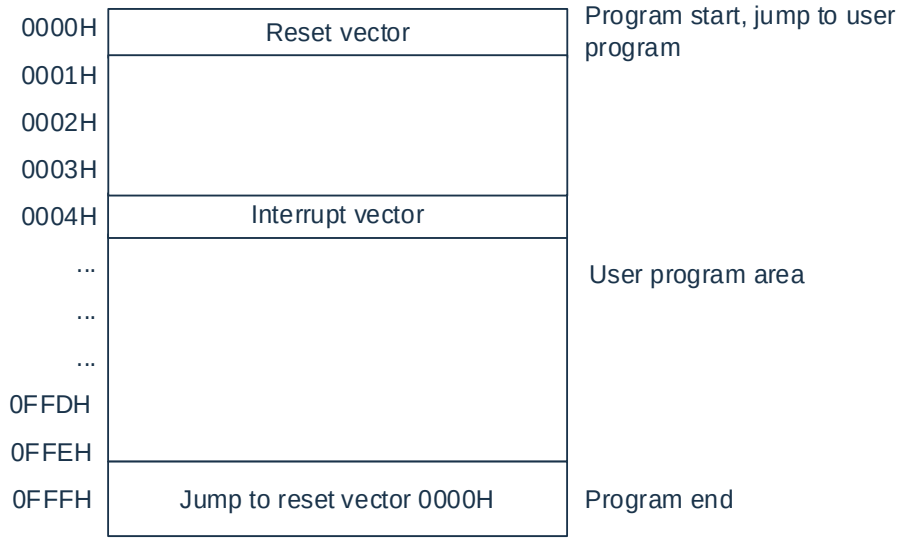
2. Central Processing Unit (CPU)

2.1 Memory

2.1.1 Program memory

SC8F073 program memory space

MTP: 4K



2.1.1.1Reset vector (0000H)

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

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

After any of the above resets, program execution will restart from 0000H and the system registers will be restored to their default values. The system reset mode can be determined by the PD and TO flags of the STATUS register. The following program demonstrates how to define a reset vector in MTP.

Example: define a reset vector

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

2.1.1.2Interrupt vector

The address for interrupt vector is 0004H. Once the interrupt responds, the current value for program counter (PC) will be saved to stack buffer and jump to 0004H to execute interrupt service program. All interrupt will enter 0004H. Users will determine which interrupt to execute according to the bit of the interrupt request flag bit register. The following program illustrates how to write interrupt service program.

Example: Define an interrupt vector, and the interrupt program is placed after the user program.

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

Note: Since the microcontroller does not provide specific instructions for stack operations, users need to protect the interrupt context themselves.

Example: interrupt-in protection

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

Example: interrupt-out restore

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

2.1.1.3Jump table

The jump table enables multiple address jumps. Since the PCL and ACC values can be added together to get a new PCL, multiple address jumps can be realized by adding different ACC values to the PCL. If the value of ACC is n, then PCL+ACC represents the current address plus n. After the execution of the current instructions, the value of PCL will add 1 (refer to the following examples). If PCL+ACC overflows, then PC will not carry. As such, users can achieve multi-address jumps by changing the value of ACC.

PCLATH is the PC high buffer register and must be assigned first when operating on the PCL.

Example: correct multi-address jump

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

Example: wrong multi-address jump

```
MTP 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 address 0000H
0101H:          JP          LOOP5           ;ACC=4, jump to address 0001H
0102H:          JP          LOOP6           ;ACC=5, jump to address 0002H
```

Note: Since PCL overflow does not automatically carry into the high byte, when using PCL for multiple-address jumps, it is important to ensure that this section of code is not placed at the boundary of MTP memory pages.

2.1.2 Data memory

SC8F073 data memory list

Addr.	BANK0	Addr.	BANK1	Addr.	BANK2	Addr.	BANK3
00H	INDF	80H	INDF	100H	INDF	180H	INDF
01H	OPTION_REG	81H	TMR0	101H	OPTION_REG	181H	TMR0
02H	PCL	82H	PCL	102H	PCL	182H	PCL
03H	STATUS	83H	STATUS	103H	STATUS	183H	STATUS
04H	FSR	84H	FSR	104H	FSR	184H	FSR
05H	TRISB	85H	TRISA	105H	TRISC	185H	-----
06H	PORTB	86H	PORTA	106H	PORTC	186H	-----
07H	WPDB	87H	WPDA	107H	-----	187H	-----
08H	WPUB	88H	WPUA	108H	WPUC	188H	-----
09H	IOCB	89H	IOCA	109H	ANSEL2	189H	-----
0AH	PCLATH	8AH	PCLATH	10AH	PCLATH	18AH	PCLATH
0BH	INTCON	8BH	INTCON	10BH	INTCON	18BH	INTCON
0CH	ODCONB	8CH	ODCONA	10CH	TMR1L	18CH	-----
0DH	PIR1	8DH	EECON1	10DH	TMR1H	18DH	-----
0EH	PIE1	8EH	EECON2	10EH	T1CON	18EH	-----
0FH	CMP1CON0	8FH	EEDAT	10FH	PIR2	18FH	-----
10H	CMP1CON1	90H	EEDATH	110H	PIE2	190H	-----
11H	PR2	91H	EEADR	111H	-----	191H	-----
12H	TMR2	92H	EEADRH	112H	-----	192H	-----
13H	T2CON	93H	ANSEL0	113H	-----	193H	-----
14H	OSCCON	94H	ANSEL1	114H	-----	194H	-----
15H	PWMCON0	95H	ADCON0	115H	-----	195H	-----
16H	PWMCON1	96H	ADCON1	116H	-----	196H	-----
17H	PWMTL	97H	-----	117H	-----	197H	-----
18H	PWMTH	98H	ADRESL	118H	TXSTA	198H	-----
19H	PWMD0L	99H	ADRESH	119H	RCSTA	199H	
1AH	PWMD1L	9AH	CMP2CON0	11AH	SPBRG	19AH	
1BH	PWMD4L	9BH	PWMD2L	11BH	TXREG	19BH	
1CH	PWMT4L	9CH	PWMD3L	11CH	RCREG	19CH	
1DH	PWMCON2	9DH	PWM23DT	11DH	-----	19DH	
1EH	PWMD01H	9EH	PWMD23H	11EH	-----	19EH	
1FH	PWM01DT	9FH	CMP2CON1	11FH	-----	19FH	
20H	Universal register 96 bytes	A0H	Universal register 80 bytes	120H	Universal register 80 bytes	1A0H	-----
		EFH		16FH		1EFH	
		F0H	Rapid storage area 70H-7FH	170H	Rapid storage area 70H-7FH	1F0H	Rapid storage area 70H-7FH
		--		--		--	
7FH		FFH		17FH		1FFH	

Data memory is divided into two functional areas: special function registers and universal data memory. Most of the data memory cells are readable/writable, but some are read-only. Special function registers are addressed from 00H to 1FH, 80 to 9FH, and 100H to 11FH.

SC8F073 special function register summary Bank0

Addr.	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
00H	INDF	Addressing this unit will address data memory (not a physical register) using the contents of the FSR.								xxxxxxxx
01H	OPTION_REG	T0LSE_EN	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	01111011
02H	PCL	Program counter low byte								00000000
03H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
04H	FSR	Indirect data memory address pointer								xxxxxxxx
05H	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	11111111
06H	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxxxxxx
07H	WPDB	WPDB7	WPDB6	WPDB5	WPDB4	WPDB3	WPDB2	WPDB1	WPDB0	00000000
08H	WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0	00000000
09H	IOCB	IOCB7	IOCB6	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	IOCB0	00000000
0AH	PCLATH	----	----	----	----	----	Write buffer for the high 3 bits of the program counter			-----000
0BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
0CH	ODCONB	ODCONB7	ODCONB6	ODCONB5	ODCONB4	ODCONB3	ODCONB2	ODCONB1	ODCONB0	00000000
0DH	PIR1	RCIF	TXIF	CMP1IF	PWMIF	RAIF	TMR1IF	TMR2IF	ADIF	--000-00
0EH	PIE1	RCIE	TXIE	CMP1IE	PWMIE	RAIE	TMR1IE	TMR2IE	ADIE	--000-00
0FH	CMP1CON0	CMP1EN	CMP1PS	CMP1NS2	CMP1NS1	CMP1NS0	CMP1NV	CMP1OUT	CMP1OEN	00000000
10H	CMP1CON1	CMP1IM	AN1_EN	RBIAS1_H	RBIAS1_L	LVDS1<3:0>				00000000
11H	PR2	TIMER2 period register								11111111
12H	TMR2	TIMER2 module register								00000000
13H	T2CON	CLK_SEL	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	00000000
14H	OSCCON	----	IRCF2	IRCF1	IRCF0	----	----	SWDTEN	-----	-101--1-
15H	PWMCON0	CLKDIV<2:0>			PWM4EN	PWM3EN	PWM2EN	PWM1EN	PWM0EN	00000000
16H	PWMCON1	PWMIO_SEL<1:0>		PWM2DTEN	PWM0DTEN	----	----	DT_DIV<1:0>		0000--00
17H	PWMTL	PWM0~PWM1 period low 8-bit registers								00000000
18H	PWMTH	----	----	PWM4D<9:8>		PWM4T<9:8>		PWMT<9:8>		--000000
19H	PWMD0L	PWM0 duty cycle low 8 bits								00000000
1AH	PWMD1L	PWM1 duty cycle low 8 bits								00000000
1BH	PWMD4L	PWM4 duty cycle low 8 bits								00000000
1CH	PWMT4L	PWM4 period low 8-bit register								00000000
1DH	PWMCON2	----	----	----	PWM4DIR	PWM3DIR	PWM2DIR	PWM1DIR	PWM0DIR	---00000
1EH	PWMD01H	----	----	PWMD1<9:8>		----	----	PWMD0<9:8>		--00--00
1FH	PWM01DT	----	----	PWM01DT<5:0>						--000000

SC8F073 special function register summary Bank1

Addr.	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
80H	INDF	Addressing this unit will address data memory (not a physical register) using the contents of the FSR.								xxxxxxxx
81H	TMR0	TIMER0 data register								xxxxxxxx
82H	PCL	Program counter low byte								00000000
83H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
84H	FSR	Indirect data memory address pointer								xxxxxxxx
85H	TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	11111111
86H	PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	xxxxxxxx
87H	WPDA	WPDA7	WPDA6	WPDA5	WPDA4	WPDA3	WPDA2	WPDA1	WPDA0	00000000
88H	WPUA	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0	00000000
89H	IOCA	IOCA7	IOCA6	IOCA5	IOCA4	IOCA3	IOCA2	IOCA1	IOCA0	00000000
8AH	PCLATH	----	----	----	----	Write buffer for the high 4 bits of the program counter				----0000
8BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
8CH	ODCONA	ODCONA7	ODCONA6	ODCONA5	ODCONA4	ODCONA3	ODCONA2	ODCONA1	ODCONA0	00000000
8DH	EECON1	EEPGD	----	----	----	WRERR	WREN	WR	RD	0---0000
8EH	EECON2	EEPROM control register 2 (not a physical register)								--------
8FH	EEDAT	EEDAT7	EEDAT6	EEDAT5	EEDAT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0	xxxxxxxx
90H	EEDATH	EEDATH7	EEDATH6	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0	xxxxxxxx
91H	EEADR	EEADR7	EEADR6	EEADR5	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0	00000000
92H	EEADRH	----	----	----	----	EEADRH3	EEADRH2	EEADRH1	EEADRH0	----0000
93H	ANSEL0	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0	00000000
94H	ANSEL1	ANS15	ANS14	ANS13	ANS12	ANS11	ANS10	ANS9	ANS8	00000000
95H	ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/ $\overline{\text{DONE}}$	ADON	00000000
96H	ADCON1	ADFM	CHS4	----	----	----	LDO_EN	LDO_SEL1	LDO_SEL0	00---000
98H	ADRESL	Low byte of the A/D result register								xxxxxxxx
99H	ADRESH	High byte of the A/D result register								xxxxxxxx
9AH	CMP2CON0	CMP2EN	CMP2PS	CMP2NS2	CMP2NS1	CMP2NS0	CMP2NV	CMP2OUT	CMP2OEN	00000000
9BH	PWMD2L	PWM2 duty cycle low 8 bits								00000000
9CH	PWMD3L	PWM3 duty cycle low 8 bits								00000000
9DH	PWM23DT	----	----	PWM23 dead zone delay time						--000000
9EH	PWMD23H	----	----	PWMD3<9:8>		----	----	PWMD2<9:8>		--00--00
9FH	CMP2CON1	CMP2IM	AN2_EN	RBIAS2_H	RBIAS2_L	LVDS2<3:0>				00000000

SC8F073 special function register summary Bank2

Addr.	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
100H	INDF	Addressing this unit will address data memory (not a physical register) using the contents of the FSR.								xxxxxxxx
101H	OPTION_REG	T0LSE_EN	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	01111011
102H	PCL	Program counter low byte								00000000
103H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
104H	FSR	Indirect data memory address pointer								xxxxxxxx
105H	TRISC	----	----	----	----	----	----	TRISC1	TRISC0	------11
106H	PORTC	----	----	----	----	----	----	RC1	RC0	------xx
108H	WPUC	----	----	----	----	----	----	WPUC1	WPUC0	------00
109H	ANSEL2	----	----	----	----	----	----	ANS17	ANS16	------00
10AH	PCLATH	----	----	----	----	Write buffer for the high 4 bits of the program counter				----0000
10BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
10CH	TMR1L	Data register of the 16-bit TIMER1 register low bytes								xxxxxxxx
10DH	TMR1H	Data register of the 16-bit TIMER1 register high bytes								xxxxxxxx
10EH	T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	00000000
10FH	PIR2	----	----	CMP2IF	----	----	----	----	----	--0-----
110H	PIE2	----	----	CMP2IE	----	----	----	----	----	--0-----
118H	TXSTA	CSRC	TX9EN	TXEN	SYNC	SCKP	----	TRMT	TX9D	00000-10
119H	RCSTA	SPEN	RX9EN	SREN	CREN	RCIDL	FERR	OERR	RX9D	00001000
11AH	SPBRG	USART baud rate data register								00000000
11BH	TXREG	USART transmit data register								00000000
11CH	RCREG	USART receive data register								00000000

SC8F073 special function register summary Bank3

Addr.	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
180H	INDF	Addressing this unit will address data memory (not a physical register) using the contents of the FSR.								xxxxxxxx
181H	TMR0	TIMER0 data register								xxxxxxxx
182H	PCL	Program counter low byte								00000000
183H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
184H	FSR	Indirect data memory address pointer								xxxxxxxx
18AH	PCLATH	----	----	----	----	Write buffer for the high 4 bits of the program counter				----0000
18BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000

2.2 Addressing mode

2.2.1 Direct addressing

It operates the RAM through the operation register (ACC).

Example: load the value in ACC to 30H register

```
LD      30H,A
```

Example: load the value in 30H register to ACC

```
LD      A,30H
```

2.2.2 Immediate addressing

Load the immediate value to accumulator (ACC)

Example: load the immediate value 12H to ACC

```
LDIA    12H
```

2.2.3 Indirect addressing

The data memory can be addressed directly or indirectly. Direct addressing can be achieved through the INDF register, and the INDF is not a physical register. When the INDF is accessed, it is addressed according to the value in the FSR register, and points to the register at that address. Therefore, after setting the FSR register, the INDF register can be regarded as a target register. Reading the INDF (FSR=0) indirectly will produce a 00H. Write to the INDF register indirectly will cause a null operation. The following example shows how indirect addressing works.

Example: application of FSR and INDF

```
        LDIA    30H
        LD      FSR,A       ;point to 30H for indirect addressing
        CLR     INDF        ;clearing INDF actually clears the RAM at address 30H
                            pointed to by FSR.
```

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

```
        LDIA    1FH
        LD      FSR,A       ;point to 1FH for indirect addressing
LOOP:
        INCR    FSR         ;address add 1, and initial address is 20H
        CLR     INDF        ;clearing the address pointed to by FSR.
        LDIA    7FH
        SUBA    FSR
        SNZB    STATUS,C    ;clear until the address of FSR is 7FH
        JP      LOOP
```

2.3 Stack

The stack buffer of the chip has 8 levels. The stack buffer is not part of data memory nor program memory, and it cannot be written nor read. It is operated by the stack pointer (SP) and cannot be read out or written in, the stack pointer will point to the top of the stack after system reset. When a subroutine call or an interrupt occurs, values in the program counter (PC) will be transferred to the stack buffer. When returning from an interruption or subroutine, the values are returned to the program counter (PC). Figure 2-2 illustrates how this works.

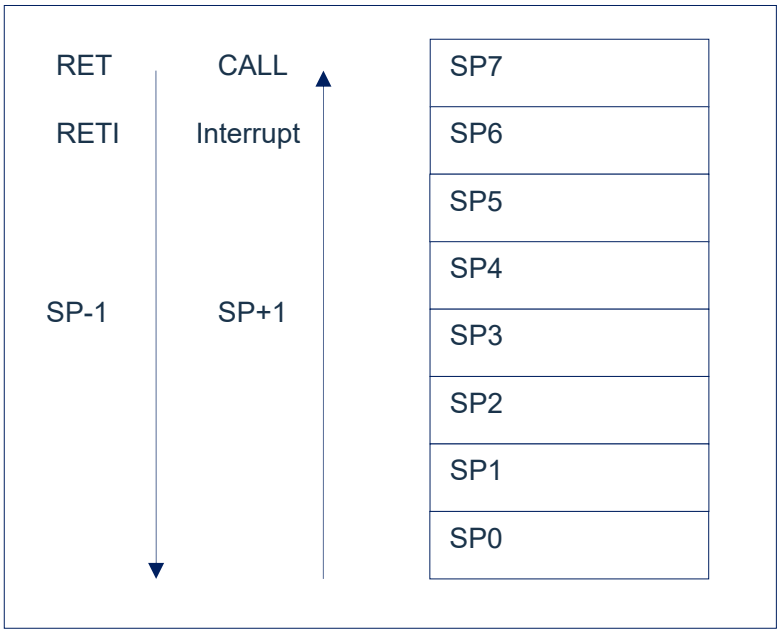


Figure 2-2: How the stack buffer works

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

Note: Stack buffer has only 8 levels, if the stack is full and an interrupt happens which is non-maskable, then only the flag bit of the interrupt will be logged. The response for the interrupt will be suppressed until the pointer of stack starts to decrease. This feature can prevent overflow of the stack caused by the interrupt. Similarly, when stack is full and subroutine happens, then stack will overflow and the contents which enter the stack first will be lost, only the last 8 return addresses will be saved. Therefore, users should pay attention to this point when writing programs to avoid program loops.

2.4 Accumulator (ACC)

2.4.1 Overview

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

The ACC register is an 8-bit register where the ALU's operation results can be stored. It is not part of the data storage but is located in the CPU for the ALU to use in its operation, so it cannot be addressed and can only be used by the instructions provided.

2.4.2 ACC application

Example: use ACC for data transferring

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

Example: use ACC for immediate addressing

```
        LDIA    30H         ;load 30H to ACC
        ANDIA   30H         ;perform 'AND' on ACC and 30H
                            ;save the result to ACC
        XORIA   30H         ;perform 'XOR' on ACC and 30H
                            ;save the result to ACC
```

Example: use ACC as the first operand of a dual operand instruction

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

Example: use ACC as the second operand of a dual operand instruction

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

2.5 Program status register (STATUS)

STATUS register includes:

- ALU arithmetic status
- Reset status

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

For example, CLRSTATUS will clear the high 3 bits and set the Z bit to 1. Hence the value of STATUS will be 000uu1uu (u will not change). It is recommended to only use CLRB, SETB, SWAPA and SWAPR instructions to change STATUS register because these will not affect any status bits.

Program status register STATUS (03H)

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

Bit7	IRP:	Register memory select bit (indirect addressing)
	1=	Bank2 and Bank3 (100h~1FFh)
	0=	Bank0 and Bank1 (00h~FFh)
Bit6~Bit5	RP<1:0>:	Memory select bit
	00=	Select Bank0
	01=	Select Bank1
	10=	Select Bank2
	11=	Select Bank3
Bit4	TO:	Time out bit
	1=	Power on or CLRWDT instruction or STOP instruction
	0=	WDT time out
Bit3	PD:	Power down bit
	1=	Power on or CLRWDT instruction
	0=	Execute STOP instruction
Bit2	Z:	Result bit
	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 bit/borrow bit
	1=	Carry happens from the lower 4 bits to the higher bits, or no borrow from the lower 4 bits.
	0=	No carry from the lower 4 bits to the higher bits, or borrow from the lower 4 bits to the higher bits.
Bit0	C:	Carry/borrow bit
	1=	A carry from the highest bit, or no borrow.
	0=	No carry from the highest bit, or a borrow hanppens.

The TO and PD flag bits can reflect the reason for chip reset. The following lists the events that affect the TO and PD and the status of the TO and PD after various resets.

Event	TO	PD
Power on	1	1
WDT overflow	0	X
STOP instruction	1	0
CLRWDT instruction	1	1
Sleep	1	0

Table of events affecting PD and TO

TO	PD	Reset reason
0	0	WDT overflow in sleep state
0	1	WDT overflow in non-sleep state
1	1	Power on
---	---	-----

Status of TO/PD after reset

2.6 Pre-scaler (OPTION_REG)

The OPTION_REG register is a readable/writable register that contains various control bits for configuration.

- WDT pre-scaler
- External interrupt trigger edge

Prescaler control register OPTION_REG (01H)

01H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION_REG	T0LSE_EN	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	1	1	1	1	0	1	1

Bit7 T0LSE_EN: TIMER0 clock source select F_{LSE} enable bit
0= TIMER0 clock source is determined by T0CS
1= TIMER0 clock source select F_{LSE}
Bit6 INTEDG: Trigger interrupt edge selection bit
0= INT pin falling edge trigger interrupt
1= INT pin rising edge trigger interrupt
Bit5 T0CS: TIMER0 clock source select bit
0= Internal instruction period clock (F_{CPU})
1= Transition edge on the T0CKI pin
Bit4 T0SE: TIMER0 clock source edge select bit
0= Incremented when the signal at the T0CKI pin jumps from a low level to a high level
1= Incremented when the signal at the T0CKI pin jumps from a high level to a low level
Bit3 PSA: Pre-scaler allocation bit
0= Allocate pre-scaler to TIMER0 module
1= Allocate pre-scaler to WDT
Bit2~Bit0 PS2~PS0: Pre-allocation parameter configure bit

PS2	PS1	PS0	TMR0 frequency division ratio	WDT frequency division ratio (WDT_DIV=DISABLE)	WDT frequency division ratio (WDT_DIV=ENABLE)
0	0	0	1:2	1:1	1:3
0	0	1	1:4	1:2	1:6
0	1	0	1:8	1:4	1:12
0	1	1	1:16	1:8	1:24
1	0	0	1:32	1:16	1:48
1	0	1	1:64	1:32	1:96
1	1	0	1:128	1:64	1:192
1	1	1	1:256	1:128	1:384

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

When used for WDT, the CLRWDT instruction will clear pre-scaler and WDT timer.

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

2.7 Program counter (PC)

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

When encountering a condition jump instruction and the condition is met, the next instruction to be read during current instruction execution will be discarded and an empty instruction period will be inserted. After this, the correct instruction can be obtained. If not, the next instruction is executed in sequence.

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

Note: When the programmer uses PCL to make a short jump, the programmer must first load a value to the PC high bit buffer register PCLATH.

The PC values for several special cases are given below

Reset	PC=0000;
Interrupt	PC=0004 (original PC+1 will be add to stack automatically);
CALL	PC=Program specified address (original PC+1 will be add to stack automatically);
RET, RETI, RETI	PC=Value from stack;
PCL operation	PC[11:8] unchanged, PC[7:0]=user defined value;
JP	PC=Program specified value;
Other instructions	PC=PC+1;

2.8 Watchdog timer (WDT)

The Watch Dog Timer (WDT) is an on-chip self-oscillating RC oscillator timer, without any peripheral components. Even if the chip's main clock stops working, the WDT can also keep time. The WDT overflow will generate a reset.

2.8.1 WDT period

The WDT uses an 8-bit prescaler. After all resets, the WDT overflow period is 128ms, and the WDT overflow period is calculated as 16ms*dividing factor. Setting the OPTION_REG register will change the WDT period, and the WDT overflow period will be affected by the ambient temperature, power supply voltage and other parameters.

The "CLRWDT" and "STOP" instructions clear the WDT timer and the count value in the prescaler (when the prescaler is assigned to the WDT). WDT generally is used to prevent the system and MCU program from being out of control. Under normal circumstances, the WDT should be cleared by the "CLRWDT" instruction before it overflows to prevent a reset. If program is out of control for some reason such that "CLRWDT" instruction is not able to execute before overflow, WDT overflow will then generate a reset to make sure the system restarts. If a reset is generated by the WDT overflow, then 'TO' bit of STATUS will be cleared to 0. Users can judge whether the reset is caused by WDT overflow according to this.

Note:

1. If WDT is used, 'CLRWDT' instruction must be placed somewhere is the program to make sure it is cleared before WDT overflow. If not, chip will keep resetting and the system cannot be operated normally.
2. It is not allowed to clear WDT during interrupt so that the main program 'run away' can be detected.
3. The program should have one WDT clearing operation in the main program, and try not to clear the WDT in multiple branches, this architecture can maximize the protection function of the watchdog counter.
4. The overflow time of the watchdog counter varies from chip to chip, so when setting the clear WDT time, there should be a greater redundancy with the WDT overflow time to avoid an unnecessary WDT reset.

2.8.2 Registers related to watchdog control

Oscillation control register OSCCON (14H)

14H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OSCCON	---	IRCF2	IRCF1	IRCF0	---	---	SWDTEN	---
R/W	---	R/W	R/W	R/W	---	---	R/W	---
Reset value	---	1	0	1	---	---	1	---

Bit7	Unused.	
Bit6~Bit4	IRCF<2:0>:	Internal oscillator frequency selection bit
	111=	$F_{SYS} = F_{HSI}/1$
	110=	$F_{SYS} = F_{HSI}/2$
	101=	$F_{SYS} = F_{HSI}/4$ (default)
	100=	$F_{SYS} = F_{HSI}/8$
	011=	$F_{SYS} = F_{HSI}/16$
	010=	$F_{SYS} = F_{HSI}/32$
	001=	$F_{SYS} = F_{HSI}/64$
	000=	F_{SYS} = 32KHz (F_{LSI})
Bit3~Bit2	Unused	
Bit1	SWDTEN:	Software enable or disable watchdog timer bit
	1=	Enable WDT
	0=	Disable WDT
Bit0	Unused	

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

3. System Clock

3.1 Overview

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

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

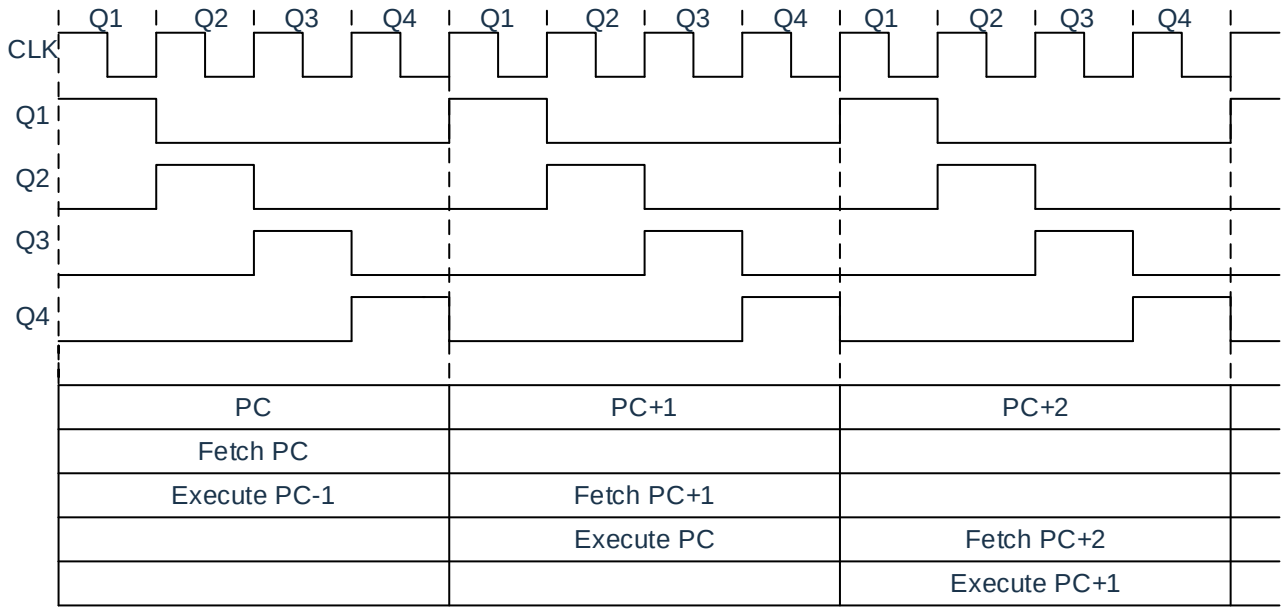
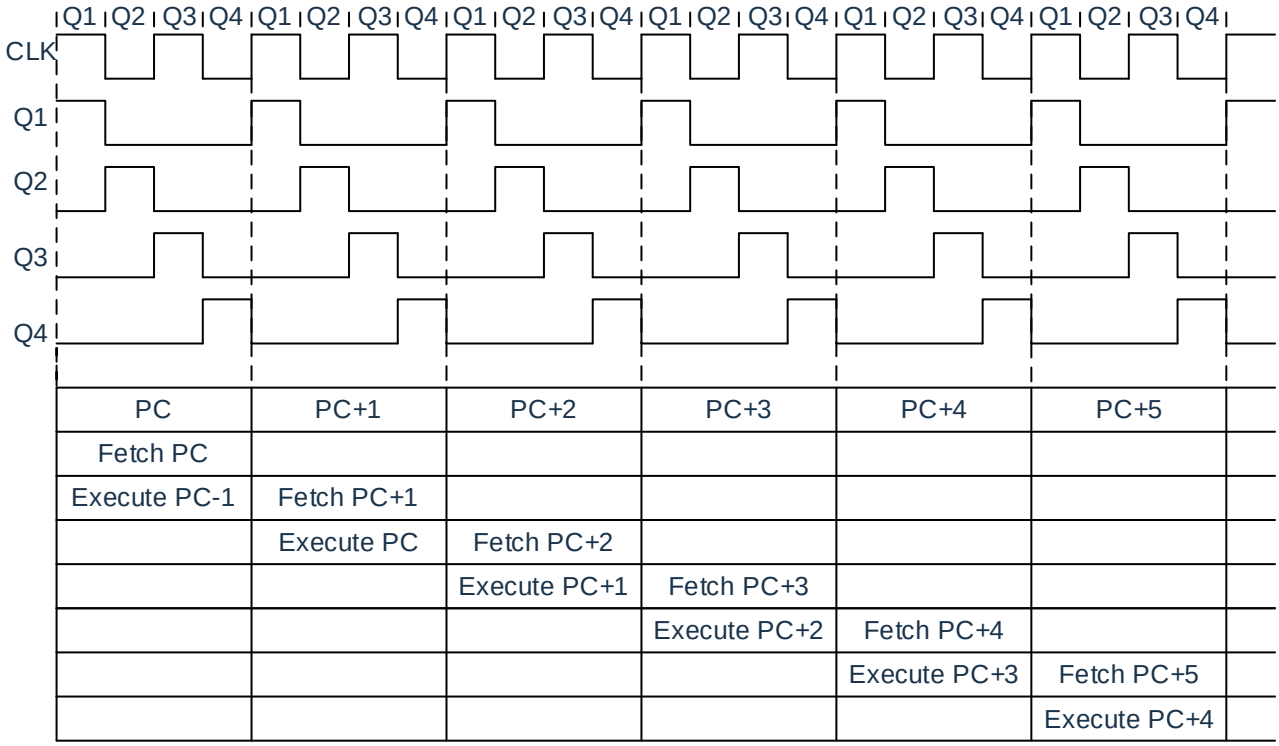


Figure 3-1: Clock and instruction cycle timing chart (F_{CPU}_DIV=4T)

Following is the relationship between working frequency of system and the speed of instructions when F_{CPU}_DIV=4T:

System frequency (F_{SYS})	Dual instruction period	Single instruction period
1MHz	8μs	4μs
2MHz	4μs	2μs
4MHz	2μs	1μs
8MHz	1μs	500ns
16MHz	500ns	250ns



PC	PC+1	PC+2	PC+3	PC+4	PC+5	
Fetch PC						
Execute PC-1	Fetch PC+1					
	Execute PC	Fetch PC+2				
		Execute PC+1	Fetch PC+3			
			Execute PC+2	Fetch PC+4		
				Execute PC+3	Fetch PC+5	
					Execute PC+4	

Figure 3-2: Clock and instruction cycle timing chart (F_{CPU}_DIV=2T)

Following is the relationship between working frequency of system and the speed of instructions when F_{CPU}_DIV=2T:

System frequency (F_{SYS})	Dual instruction period	Single instruction period
1MHz	4μs	2μs
2MHz	2μs	1μs
4MHz	1μs	500ns
8MHz	500ns	250ns
16MHz	250ns	125ns

3.2 System oscillator

The chip has one type of oscillation: internal RC oscillation.

3.2.1 Internal RC oscillation

The default oscillation mode of the chip is internal RC oscillation, and the oscillation frequency is fixed at 16MHz. On this basis, the operating frequency of the chip can be set through the OSCCON register.

3.3 Reset time

The reset time is the time from the chip reset to the chip oscillation stabilization, its design value is about 16ms.

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

3.4 Oscillator control register

The Oscillator Control (OSCCON) register controls the system clock and frequency selection.

Oscillator control register OSCCON (14H)

14H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OSCCON	---	IRCF2	IRCF1	IRCF0	---	---	SWDTEN	---
R/W	---	R/W	R/W	R/W	---	---	R/W	---
Reset value	---	1	0	1	---	---	1	---

Bit7	Unused	
Bit6~Bit4	IRCF<2:0>:	Internal oscillator frequency selection bit
	111=	F_{SYS} = F_{HSI}/1
	110=	F_{SYS} = F_{HSI}/2
	101=	F_{SYS} = F_{HSI}/4 (default)
	100=	F_{SYS} = F_{HSI}/8
	011=	F_{SYS} = F_{HSI}/16
	010=	F_{SYS} = F_{HSI}/32
	001=	F_{SYS} = F_{HSI}/64
	000=	F_{SYS} = 32KHz (LFINTOSC)
Bit3~Bit2	Unused.	
Bit1	SWDTEN:	Software enable or disable watchdog timer bit
	1=	Enable WDT
	0=	Disable WDT
Bit0	Unused	

3.5 Clock block diagram

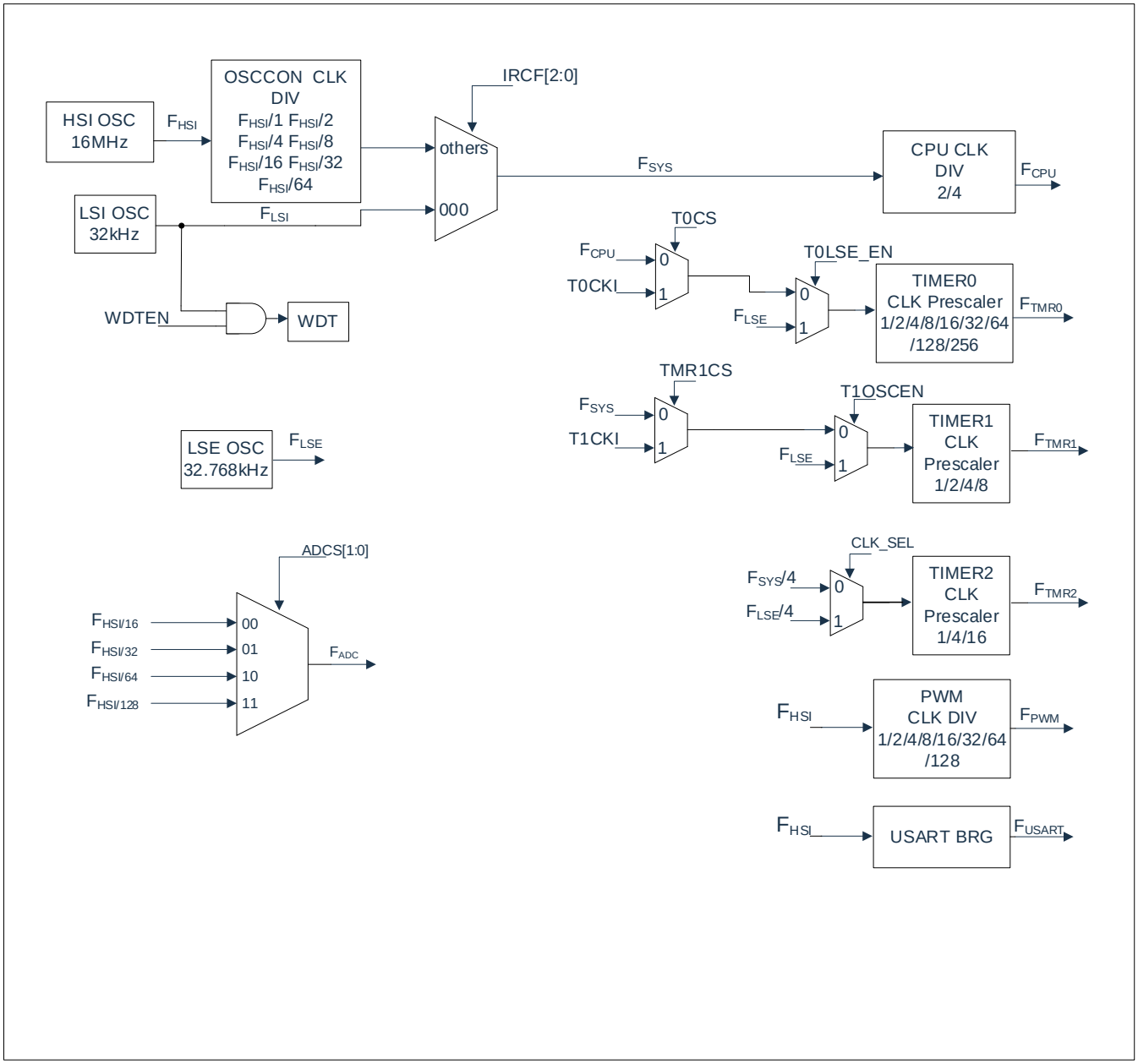


Figure 3-3: Clock block diagram

4. Reset

The chip can be reset in the following 4 ways:

- Power on reset
- External reset
- LVR reset
- Watchdog overflow reset during normal operation

When any of the above reset occurs, all system registers will be restored to their default state, the program will stop running, and the program counter (PC) will be cleared to zero. At the same time, the program will start running from reset vector 0000H after the reset. The TO and PD flags of STATUS can give information about the reset state of the system (see the description of STATUS for details), and the user can control the program execution path according to the state of PD and TO.

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

4.1 Power on reset

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

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

4.2 External reset

The SC8F073 supports external reset function. RB2 can be configured as a reset port via CONFIG, at which point RB2 automatically enables the internal weak pull-up. If RB2 is pulled down, the chip will be reset.

4.3 Brown-out reset

4.3.1 Overview

Brown-out Reset is designed to handle situations where external factors cause a drop in system voltage (e.g., interference or changes in external load). A voltage drop may cause the system to enter a dead zone, where the power supply can no longer meet the minimum operating voltage requirements of the system.

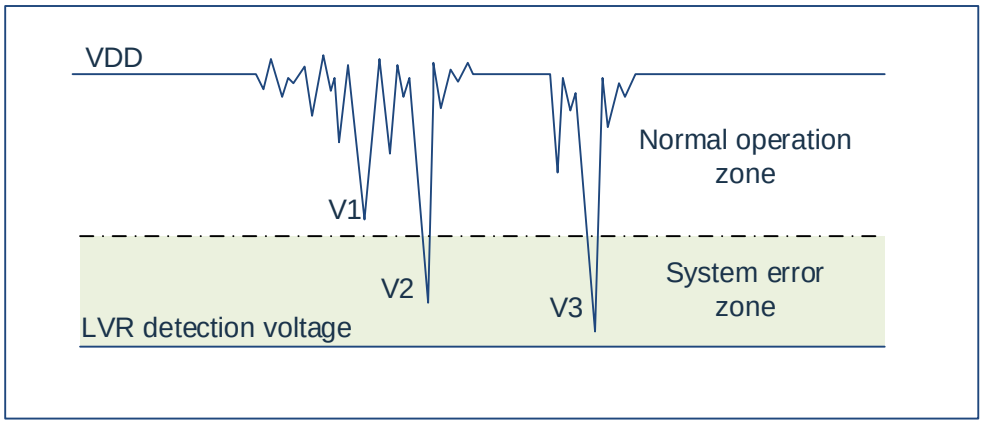


Figure 4-1: Brown-out reset

The diagram above illustrates a typical Brown-out Reset (BOR) scenario. In this diagram, VDD is subjected to severe interference, causing the voltage to drop significantly. The region above the dashed line represents the normal operating state of the system, while the region below the dashed line indicates an unstable or unknown working state, referred to as the dead zone. When VDD drops to V1, the system remains in a normal state. However, if VDD drops to V2 or V3, the system enters the dead zone, which can lead to errors.

The system may enter the dead zone under the following circumstances:

- DC applications:
  - In DC-powered applications, the system is generally powered by a battery. When the battery voltage is too low, or the microcontroller is driving a load, the system voltage may drop and enter the dead zone. In this case, the power supply will not fall further to the Low Voltage Detection (LVD) threshold, so the system stays in the dead zone.
- AC applications:
  - In AC-powered systems, the DC voltage is affected by noise from the AC power supply. When the external load is too high, such as when driving a motor, the interference generated by the load also impacts the DC power supply. If VDD drops below the minimum operating voltage due to interference, the system may enter an unstable operating state.
  - In AC applications, the system startup and power-down durations are typically longer. During power-up, the sequence protection ensures normal power-up, but during the power-down process, the situation is similar to that in DC applications. After the AC power is turned off, the VDD voltage slowly decreases, potentially entering the dead zone.

As shown in the diagram, the normal operating voltage range is typically higher than the system reset voltage. The reset voltage is determined by the Low Voltage Reset (LVR) threshold. When the system's operating speed increases, the minimum operating voltage also increases. However, since the reset voltage is fixed, a voltage range exists between the minimum operating voltage and the reset voltage where the system cannot function properly, nor can it reset. This voltage range is called the dead zone.

4.3.2 Improvements for brown-out reset

Here are some suggestions for improving the system's Brown-out Reset (BOR) performance:

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

Watchdog timer

The watchdog timer ensures the program runs correctly. If the system enters a dead zone or the program encounters an error, the watchdog timer will overflow and trigger a system reset.

Lower the system's operating frequency

The faster the system operates, the higher the minimum operating voltage. This increases the likelihood of the system operating within the dead zone. By reducing the system's operating frequency, the minimum operating voltage is lowered, reducing the chances of the system running in the dead zone.

Increase the voltage descent slope

This method is useful for systems powered by AC. In typical AC-powered systems, the voltage during a power-down process decreases slowly, causing the chip to operate in the dead zone for a longer time. If the system is powered on again under such conditions, the chip may enter an incorrect state. To address this, a discharge resistor can be added between the chip's power and ground pins, allowing the MCU to quickly pass through the dead zone and enter the reset region, minimizing the chances of errors when the chip is powered on.

4.4 Watchdog reset

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

The timing of the watchdog reset is as follows:

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

For application of watchdog timer, please refer to Section 2.8.

5. Sleep Mode

5.1 Enter sleep mode

When the STOP instruction is executed to enter sleep mode, if the Watchdog Timer (WDT) is enabled, then:

- The WDT will be cleared and continue running.
- The PD bit in the STATUS register will be cleared.
- The TO bit will be set to 1.
- The oscillator driver will be turned off.
- The I/O ports will maintain the state they were in before the STOP instruction was executed (whether driven high, low, or in high-impedance state).

In sleep mode, to minimize current consumption, all I/O pins should be held at either VDD or GND, with no external circuits drawing current from the I/O pins. To avoid introducing switching currents due to floating input pins, external pull-up or pull-down resistors should be used to ensure high-impedance input I/O pins are pulled to either a high or low level. Additionally, the impact of the internal pull-up resistors of the chip should be considered to further reduce current consumption.

5.2 Wake up from sleep mode

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

1. Watchdog timer wakeup;
2. INT interrupt;
3. PORTB interrupt on change;
4. PORTA interrupt on change or peripheral interrupt.

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

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

5.3 Interrupt wakeup

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

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

5.4 Sleep mode application

Before the system enters the sleep mode, if the user needs to get a smaller sleep current, please confirm the status of all I/O ports, if there are floating I/O ports in the user's program, set all floating ports as output ports to make sure that each input port has a fixed state to avoid that when the I/O is an input state, the port level is in an unstable state which increases the sleep current; turn off the other peripheral modules, such as the AD module. According to the actual functional requirements of the program, the WDT function can be disabled to reduce the sleep current.

Example: procedures for entering sleep mode

```
SLEEP_MODE:
        CLR     INTCON      ;disable interrupts
        LDIA    B'00000000'
        LD      TRISB,A     ;all I/Os set as output ports
        ...                 ;disable other functions
        LDIA    0A5H
        LD      SP_FLAG,A   ;set sleep status memory register (user-defined)
        CLRWDT              ;clear WDT
        STOP                ;execute STOP instruction
        NOP
        NOP
```

5.5 Sleep mode wake-up time

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

System main clock source	System clock frequency (IRCF<2:0>)	Sleep wakeup wait time T_{WAIT}
Internal high-speed RC oscillation (F_{HSI})	$F_{SYS}=F_{HSI}$	$T_{WAIT}=264*1/F_{HSI}+16*1/F_{HSI}$
	$F_{SYS}= F_{HSI} /2$	$T_{WAIT}=264*2/F_{HSI}+16*1/F_{HSI}$
	…	…
	$F_{SYS}= F_{HSI} /64$	$T_{WAIT}=264*64/F_{HSI}+16*1/F_{HSI}$
Internal low-speed RC oscillation ($F_{LFINTOSC}$)	----	$T_{WAIT}=11/F_{LSI}$

6. I/O Ports

The chip has three I/O ports: PORTA, PORTB and PORTC (up to 18 I/Os). These ports can be accessed directly from reading from/writing to the port data registers.

Port	Bit	Pin description	I/O
PORTA	0	Schmitt trigger input, push-pull or open-drain output, AN0, PWMA0, PWMB0, INT, CMP2_O	I/O
	1	Schmitt trigger input, push-pull or open-drain output, AN1, PWMA1, PWMB1, CMP2_+, CMP2_3-	I/O
	2	Schmitt trigger input, push-pull or open-drain output, AN2, PWMA2, PWMB2, PWMD2, CMP2_0-	I/O
	3	Schmitt trigger input, push-pull or open-drain output, AN3, PWMA3, PWMD3, TX/CK	I/O
	4	Schmitt trigger input, push-pull or open-drain output, AN4, PWMA4, RX/DT	I/O
	5	Schmitt trigger input, push-pull or open-drain output, AN5, PWMC0	I/O
	6	Schmitt trigger input, push-pull or open-drain output, AN6	I/O
	7	Schmitt trigger input, push-pull or open-drain output, AN7	I/O
PORTB	0	Schmitt trigger input, push-pull or open-drain output, programming data input/output, AN8, PWMD0, TX/CK, INT, CMP2_2-, CMP1_O, OSCIN	I/O
	1	Schmitt trigger input, push-pull or open-drain output, programming clock input, AN9, PWMB4, PWMD1, RX/DT, T1CKI, CMP2_1-, CMP1_3-, CMP1_+, OSCOUT	I/O
	2	Schmitt trigger input, push-pull or open-drain output, AN10, PWMB3, PWMD4, T0CKI, CMP1_0-, MCLR	I/O
	3	Schmitt trigger input, push-pull or open-drain output, AN11, PWMD2	I/O
	4	Schmitt trigger input, push-pull or open-drain output, AN12, PWMC4, PWMD3, CMP1_1-	I/O
	5	Schmitt trigger input, push-pull or open-drain output, AN13, PWMC3, PWMC3, CMP1_2-	I/O
	6	Schmitt trigger input, push-pull or open-drain output, AN14, PWMC2	I/O
	7	Schmitt trigger input, push-pull or open-drain output, AN15, PWMC1, T1G	I/O
PORTC	0	Schmitt trigger input, push-pull output, programming clock input, AN16	I/O
	1	Schmitt trigger input, push-pull output, programming data input/output, AN17	I/O

<Table 6-1: Port configuration overview>

6.1 I/O port structure

6.1.1 PORTA I/O port

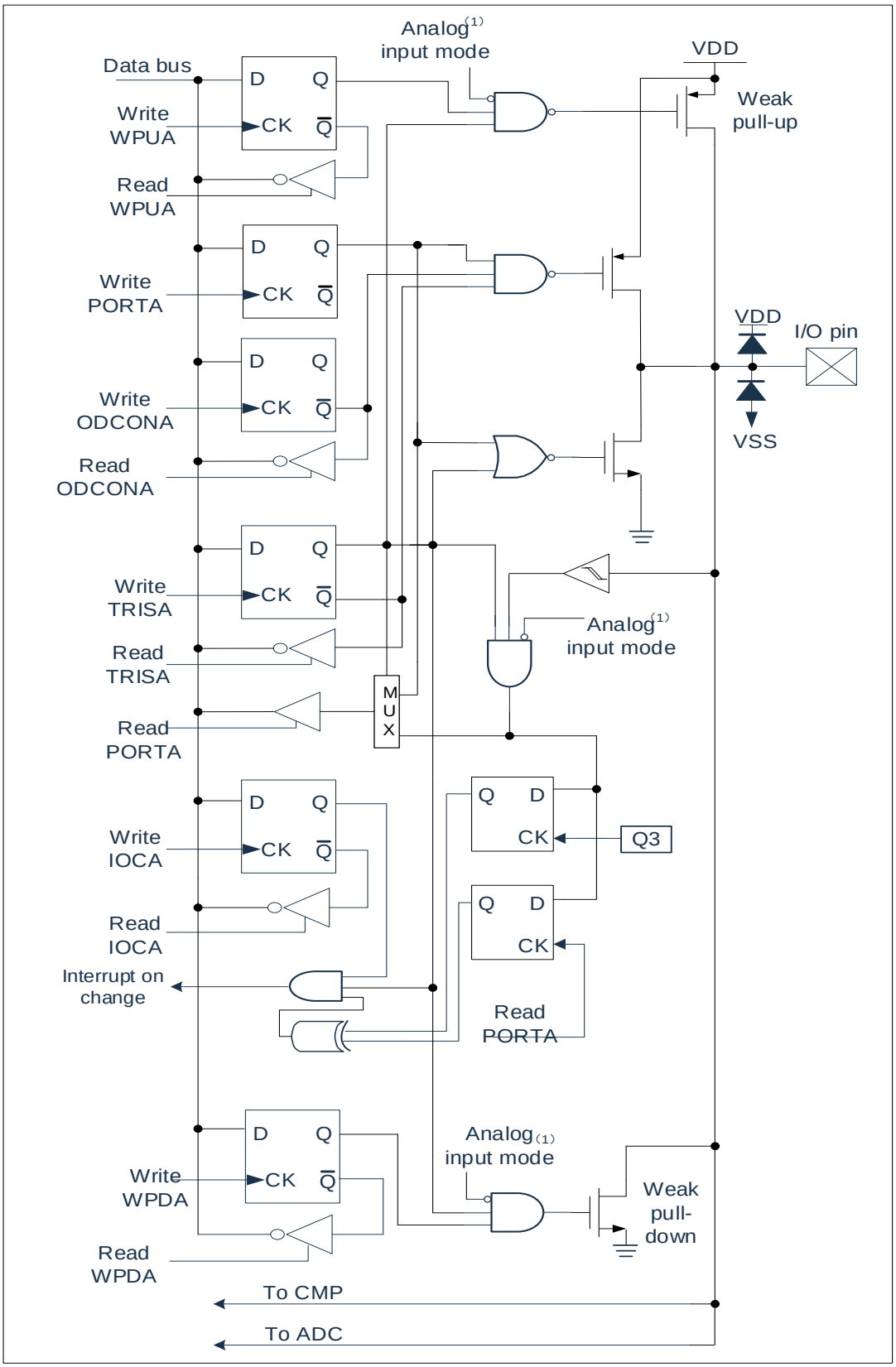


Figure 6-1: Diagram of PORTA I/O port stucture

6.1.2 PORTB I/O port

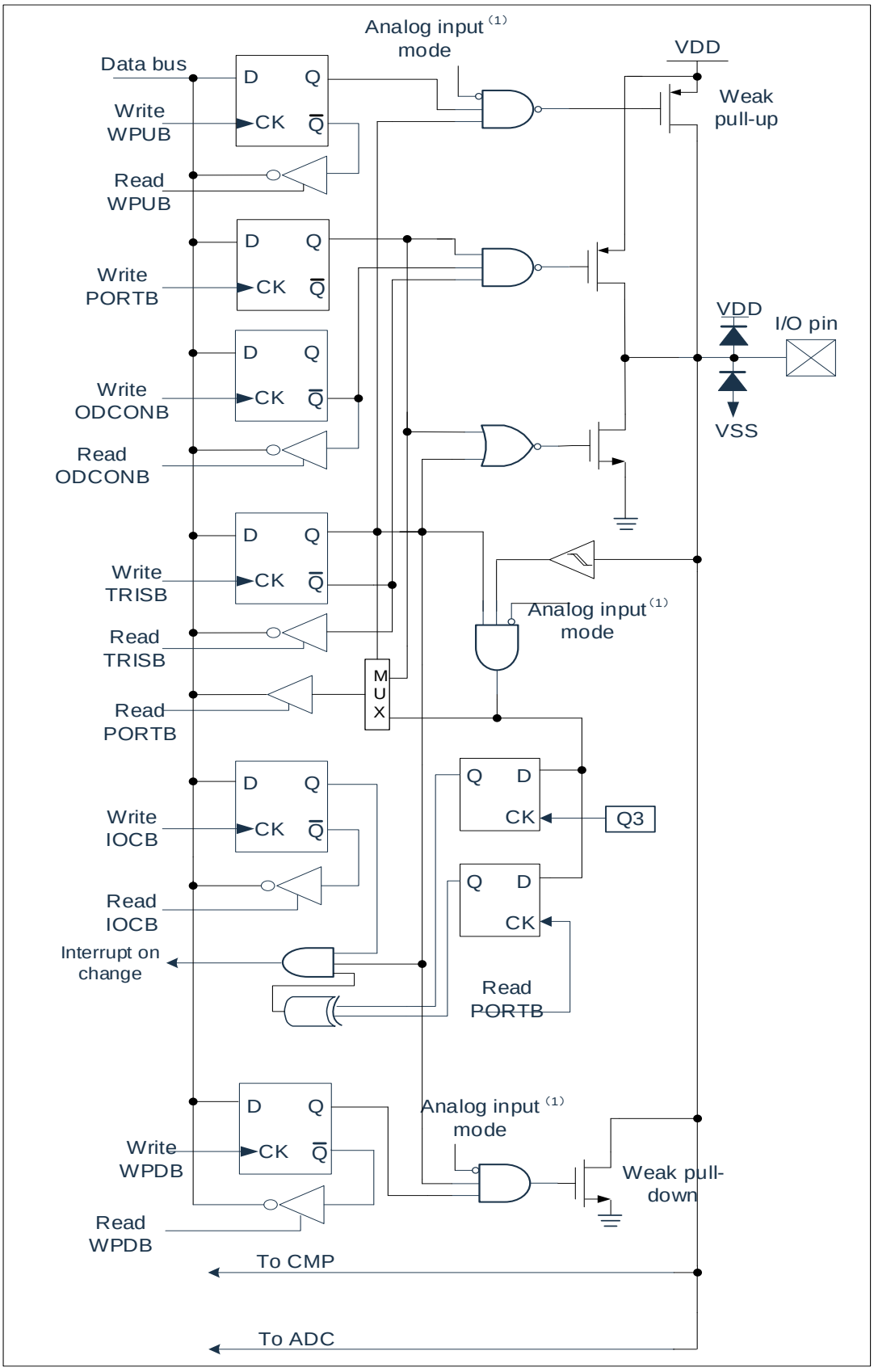


Figure 6-2: PORTB I/O port stucture

6.1.3 PORTC I/O port

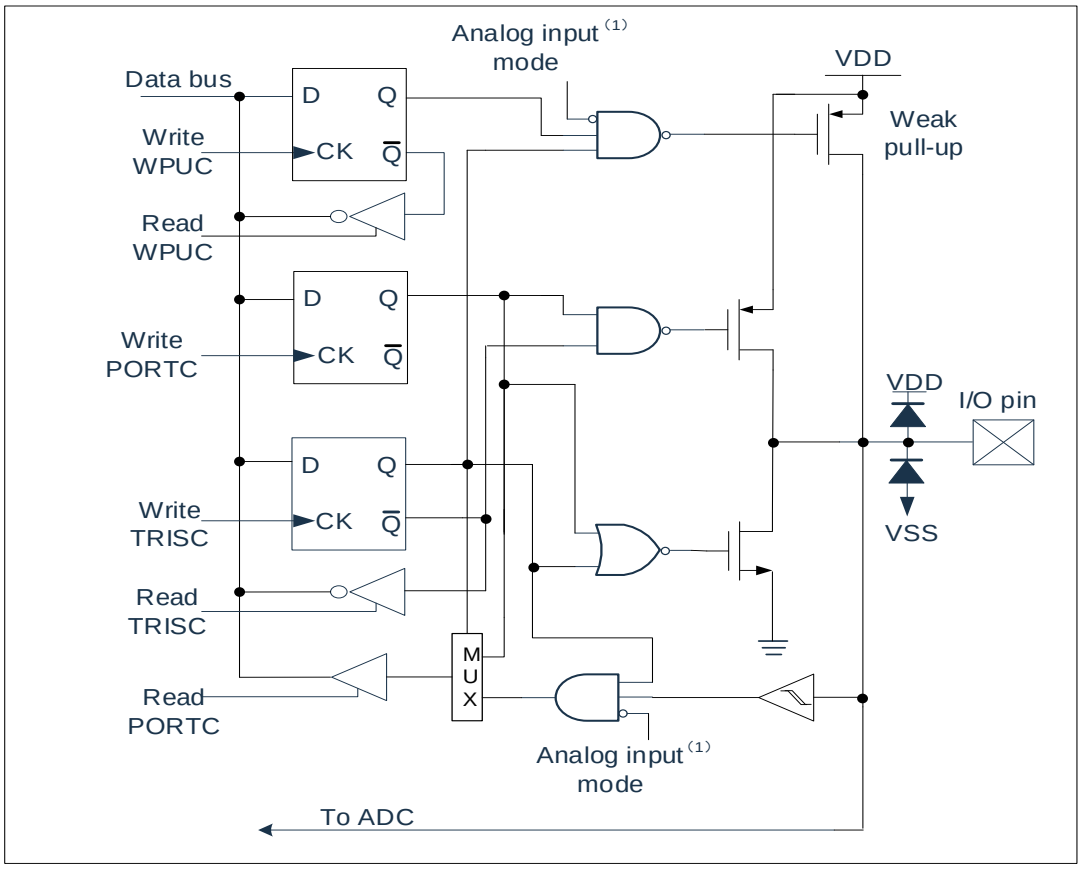


Figure 6-3: PORTC I/O port stucture

Note: ANx_EN or ANSx determines the analog input mode.

6.2 PORTA

6.2.1 PORTA data and direction

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

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

The registers related to the PORTA port are PORTA, TRISA, WPUA, WPDA, ODCONA, IOCA, ANSEL0 and so on.

PORTA data register PORTA(86H)

86H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	X	X	X	X	X	X	X	X

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

PORTA direction register TRISA(85H)

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

Bit7~Bit0	TRISA<7:0>:	PORTA tristate control bit
	1=	PORTA pin is configured as an input (tristate)
	0=	PORTA pin is configured as an output

Example: PORTA procedure

```
LDIA    B'00110000'    ;set PORTA<3:0> as an output port, PORTA<5:4>as an input port
LD      TRISA,A
LDIA    03H            ;PORTA<1:0>output high level, PORTA<3:2>output low level
LD      PORTA,A        ;since PORTA<5:4> is an input port, loading 0 or 1 has no effect.
```

6.2.2 PORTA open-drain output control

Each PORTA pin has an individually configurable open-drain output enable control bit.

PORTA open-drain output enable register ODCONA(8CH)

8CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ODCONA	ODCONA7	ODCONA6	ODCONA5	ODCONA4	ODCONA3	ODCONA2	ODCONA1	ODCONA0
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 ODCONA<7:0>: PORTA open-drain output enable
1= Enable open-drain output
0= Disable open-drain output

6.2.3 PORTA analog selection control

The ANSEL0 register is used to configure the input mode of the I/O pins as analog mode. Setting the appropriate bits in ANSEL0 to 1 will disable digital read operations on the corresponding pins (they will always return 0) and enable the analog functionality of the pins. The state of the ANSEL0 bits does not affect the digital output functionality. Pins with TRIS cleared and ANSEL0 set to 1 will still function as digital outputs, but their input mode will switch to analog. This can lead to unpredictable results when performing read-modify-write operations on the affected ports.

PORTA analog selection register ANSEL0(93H)

93H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL0	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 ANS<7:0>: Analogue select bit, selects the analog or digital function of pins AN<7:0> respectively
1= Analog input: The pin is configured as an analog input.
0= Digital I/O, pins are assigned to ports or special functions

6.2.4 PORTA pull-up resistor

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

PORTA pull-up resistor register WPUA(88H)

88H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUA	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

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

Note: If the pin is configured as an output or analogue input, the weak pull-up is automatically disabled.

6.2.5 PORTA pull-down resistor

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

PORTA pull-down resistor register WPDA (87H)

87H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPDA	WPDA7	WPDA6	WPDA5	WPDA4	WPDA3	WPDA2	WPDA1	WPDA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 WPDA<7:0>: Weak pull-down register bit
1= Enable pull-down
0= Disable pull-down

Note: If the pin is configured as an output or analogue input, the weak pull-down is automatically disabled.

6.2.6 PORTA interrupt on change

All PORTA pins can be individually configured as interrupt on change pins. The control bit IOCA<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 allowed level change interrupt, compare the value on the pin with the old value latched when PORTA was read last time. Perform a logical OR operation with the output “mismatch” of the last read operation to set the PORTA level change interrupt flag (RAIF) of the PIR1 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 by the following ways:

- Read from or write to PORTA. This will end the mismatch state of the pin level.
- Clear the flag bit RAIF.

The mismatch status will continuously set the RAIF flag bit as 1. Reading or writing PORTA will end the mismatch state and allow the RAIF flag to be cleared.

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

PORTA interrupt-on-change register IOCA (89H)

89H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOCA	IOCA7	IOCA6	IOCA5	IOCA4	IOCA3	IOCA2	IOCA1	IOCA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 IOCA<7:0> PORTA interrupt-on-change control bit.
1= Enable interrupt on change.
0= Disable interrupt on change.

6.3 PORTB

6.3.1 PORTB data and direction

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

Reading the PORTB register reads the pin status and writing to the register will write the port latch. All write operations are read-modify-write operations. Therefore, writing a port means to read the pin level of the port first, modify the read value, and then write the modified value into the port data latch. Even when the PORTB pin is used as an analog input, the TRISB register still controls the direction of the PORTB pin. When using the PORTB pin as an analog input, the user must ensure that the bits in the TRISB register remain set as 1. I/O pin is always read 0 when configured as analog input.

The registers related to the PORTB port are PORTB, TRISB, WUPB, WDPB, IOCB, ODCONB, ANSEL1 and so on.

PORTB data register PORTB (06H)

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

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

PORTB direction register TRISB (05H)

05H	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 tristate control bit
1= PORTB pin configured as an input (tristate)
0= PORTB pin configured as an output

Example: PORTB procedure

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

6.3.2 PORTB open-drain output control

Each PORTB pin has an individually configurable open-drain output enable control bit.

PORTB open drain output enable register ODCONB(0CH)

0CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ODCONB	ODCONB7	ODCONB6	ODCONB5	ODCONB4	ODCONB3	ODCONB2	ODCONB1	ODCONB0
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 ODCONB<7:0>: PORTB open-drain output enable
1= Enable open-drain output
0= Disable open-drain output

6.3.3 PORTB analog selection control

The ANSEL1 register is used to configure the input mode of the I/O pins as analog mode. Setting the appropriate bits in ANSEL1 to 1 will disable digital read operations on the corresponding pins (they will always return 0) and enable the analog functionality of the pins. The state of the ANSEL1 bits does not affect the digital output functionality. Pins with TRIS cleared and ANSEL1 set to 1 will still function as digital outputs, but their input mode will switch to analog. This can lead to unpredictable results when performing read-modify-write operations on the affected ports.

PORTB analog selection register ANSEL1 (94H)

94H	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 ANS<15:8>: Analogue select bit, selects the analog or digital function of pins AN<15:8> respectively
1= Analog input: The pin is configured as an analog input.
0= Digital I/O, pins are assigned to ports or special functions.

6.3.4 PORTB pull-up resistor

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

PORTB pull-up resistor register WPUB(08H)

08H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 WPUB<7:0>: PORTB weak pull-up enable bit
1= Enable pull-up
0= Disable pull-up

Note: If pins are configured as outputs or analog inputs, the weak pull-ups will be disabled automatically.

6.3.5 PORTB pull-down resistor

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

PORTB pull-down resistor register WPDB(07H)

07H	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>: PORTB weak pull-down enable bit
1= Enable pull-down
0= Disable pull-down

Note: If pins are configured as outputs or analog inputs, the weak pull-down is automatically disabled.

6.3.6 PORTB interrupt on change

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

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

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

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

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

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

PORTB interrupt-on-change register IOCB(09H)

09H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOCB	IOCB7	IOCB6	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	IOCB0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 IOCB<7:0> PORTB interrupt-on-change control bit.
1= Enable interrupt on change.
0= Disable interrupt on change.

6.4 PORTC

6.4.1 PORTC data and direction

PORTC is a 2-bit wide bi-directional port. The corresponding data direction register is TRISC. Setting one bit of TRISC to 1 (=1) can configure the corresponding PORTC pin to be input. Setting one bit of TRISC to 0 (=0) can configure the corresponding PORTC pin to be output.

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 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 PORTC pin is used as an analog input, the TRISC register still controls the direction of the PORTC pin. When using the PORTC pin as an analog input, the user must ensure that the bits in the TRISB register remain set as 1. I/O pin is always read 0 when configured as analog input.

The registers related to the PORTC port are PORTC, TRISC, WUPC and so on.

PORTC data register PORTC (106H)

106H	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>:	PORTB I/O pin bit
	1=	Port pin level>V_{IH}
	0=	Port pin level<V_{IL}

PORTC direction register TRISC (105H)

105H	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 tristate control bit
	1=	PORTC pin configured as an input (tristate)
	0=	PORTC pin configured as an output

Example: PORTC procedure

```
CLR     PORTC           ;clear data register
LDIA    B'00000001'     ;set PORTC<0> as the input port and PORTC<1> as the output port.
LD      TRISC,A
```

6.4.2 PORTC analog selection control

The ANSEL2 register is used to configure the input mode of the I/O pins as analog mode. Setting the appropriate bits in ANSEL2 to 1 will disable digital read operations on the corresponding pins (they will always return 0) and enable the analog functionality of the pins. The state of the ANSEL2 bits does not affect the digital output functionality. Pins with TRIS cleared and ANSEL2 set to 1 will still function as digital outputs, but their input mode will switch to analog. This can lead to unpredictable results when performing read-modify-write operations on the affected ports.

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 ANS<17:16>: Analogue select bit, selects the analog or digital function of pins AN<17:16> respectively

1= Analog input: The pin is configured as an analog input.

0= Digital I/O, pins are assigned to ports or special functions

6.4.3 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 a port pin is configured as an output, its weak pull-up is 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<1:0>: PORTC weak pull-up enable bit

1= Enable pull-up

0= Disable pull-up

Note: If pins are configured as outputs or analog inputs, the weak pull-ups will be disabled automatically.

6.5 I/O usage

6.5.1 Write to I/O port

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

Example: write to I/O port program

```
LD      PORTB,A     ;load ACC to PORTB
CLRB    PORTB,1     ;clear PORTB.1
SET     PORTB       ;set all output ports of PORTB to 1
SETB    PORTB,1     ;set PORTB.1 to 1
```

6.5.2 Read from I/O port

Example: read from I/O port program

```
LD      A,PORTB     ;load PORTB to ACC
SNZB    PORTB,1     ;check if PORTB,1 is 1, if it is 1, skip the next statement
SZB     PORTB,1     ;check if PORTB,1 is 0, if it is 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 Cautions on I/O port usage

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

1. When I/O is converted from output to input, it is necessary to wait for several instruction periods for the I/O port to stabilize.
2. If the internal pull up resistor is used, when the I/O is converted from output to input, the stable time of the internal level is related to the capacitance connected to the I/O port. The user should set the waiting time according to the actual situation. Prevent the I/O port from scanning the level by mistake.
3. When the I/O port is an input port, its input level should be between “VDD+0.3V” and “GND-0.3V”. If the input port voltage is not within this range, the method shown in the figure below can be used.

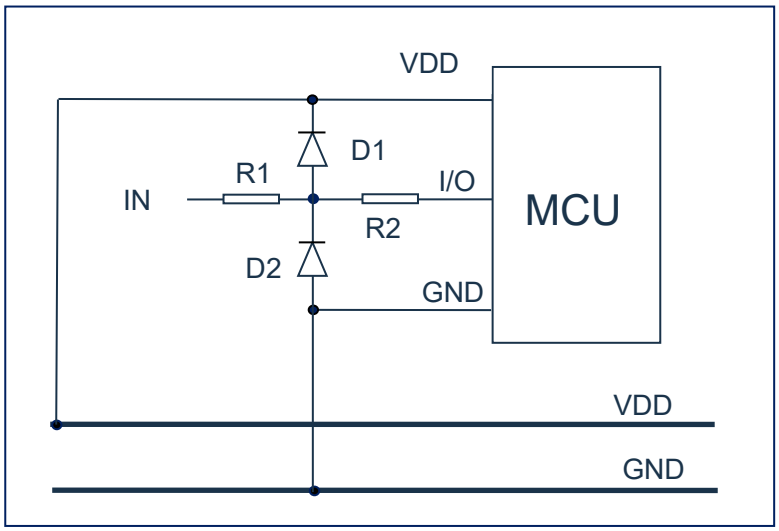


Figure 6-3: I/O port caution connection diagram

4. If long wires are connected to the I/O ports, add current limiting resistors near the I/O ports of the chip to enhance the MCU’s EMC resistance.

7. Interrupt

7.1 Overview

The chip has the following interrupt sources:

- PORTA interrupt on change
- PORTB interrupt on change
- TIMER0 overflow interrupt
- TIMER1 overflow interrupt
- TIMER2 match interrupt
- A/D interrupt
- PWM interrupt
- INT interrupt
- CMP1 interrupt
- CMP2 interrupt
- USART receive interrupt
- USART transmit interrupt

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

The global interrupt enables bit GIE (INTCON<7>) allows all unmasked interrupts when set to 1, and prohibits all interrupts when cleared. Each interrupt can be prohibited through the corresponding enable bits in the INTCON, PIE1, and PIR2 registers. GIE is cleared when reset.

Executing the “return from interrupt” instructions, RETI, will exit the interrupt service program and set the GIE bit to 1, thereby re-allowing unmasked interrupts.

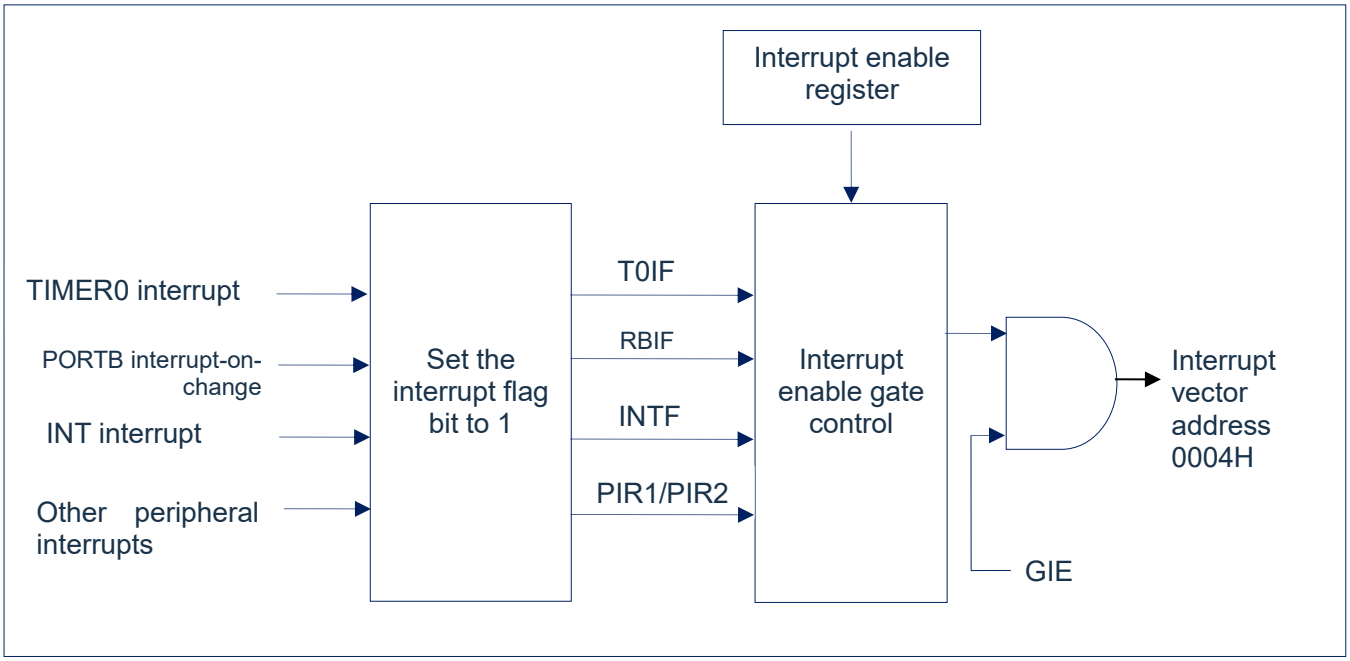


Figure 7-1: Schematic diagram of interrupt

7.2 Interrupt control register

7.2.1 Interrupt control register

Interrupt control register (INTCON) is a readable and writable register, including the enable and flag bits of INT interrupt, TIMER0 overflow interrupt and PORTB port interrupt on change.

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

Interrupt control register INTCON (0BH)

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

Bit7 GIE: Global interrupt enable bit
1= Enable all unmask interrupts
0= Disable all interrupts

Bit6 PEIE: Peripheral interrupt enable bit
1= Enable all unmask peripheral interrupts
0= Disable all peripheral interrupts

Bit5 T0IE: TIMER0 overflow interrupt enable bit
1= Enable TIMER0 interrupt
0= Disable TIMER0 interrupt

Bit4 INTE: INT external interrupt enable bit
1= Enable INT external interrupt
0= Disable INT external interrupt

Bit3 RBIE: PORTB level change interrupt enable bit (1)
1= Enable PORTB level change interrupt
0= Disable PORTB level change interrupt

Bit2 T0IF: TIMER0 overflow interrupt enable bit (2)
1= TMR0 register overflowed (cleared by software)
0= No TMR0 register overflow

Bit1 INTF: INT external interrupt flag bit
1= An INT external interrupt occurs (cleared by software)
0= No INT external interrupt occured

Bit0 RBIF: PORTB level change interrupt flag bit
1= At least one pin level status in the PORTB port has changed (cleared by software)
0= None of the PORTB universal I/O pin status has changed

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

7.2.2 Peripheral interrupt enable register

The peripheral interrupt enable registers are PIE1 and PIE2, before allowing any peripheral interrupt, set the PEIE bit of the INTCON register to 1.

Peripheral interrupt enable register PIE1 (0EH)

0EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE1	RCIE	TXIE	CMP1IE	PWMIE	RAIE	TMR1IE	TMR2IE	ADIE
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 RCIE: USART receive interrupt enable bit
1= Enable USART receive interrupt
0= Disable USART receive interrupt

Bit6 TXIE: USART transmit interrupt enable bit
1= Enable USART transmit interrupt
0= Disable USART transmit interrupt

Bit5 CMP1IE: Comparator 1 interrupt enable bit
1= Enable Comparator 1 interrupt
0= Disable Comparator 1 interrupt

Bit4 PWMIE: PWM interrupt enable bit (PWM0/1/2/3)
1= Enable PWM interrupt
0= Disable PWM interrupt

Bit3 RAIE: PORTA interrupt-on-change enable bit
1= Enable PORTA interrupt-on-change
0= Disable PORTA interrupt-on-change

Bit2 TMR1IE: TIMER1 overflow interrupt enable bit
1= Enable TIMER1 overflow interrupt
0= Disable TIMER1 overflow interrupt

Bit1 TMR2IE: TIMER2 and PR2 match interrupt enable bit
1= Enable TMR2 and PR2 match interrupt
0= Disable TMR2 and PR2 match interrupt

Bit0 ADIE: ADC interrupt enable bit
1= Enable ADC interrupt
0= Disable ADC interrupt

Peripheral interrupt enable register PIE2(110H)

110H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE2	----	----	CMP2IE	----	----	----	----	----
R/W	----	----	R/W	----	----	----	----	----
Reset value	----	----	0	----	----	----	----	----

Bit7~Bit6 Unused

Bit5 CMP2IE: Comparator2 interrupt enable bit
1= Enable comparator2 interrupt
0= Disable comparator2 interrupt

Bit4~Bit0 Unused

7.2.3 Peripheral interrupt request register

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

Peripheral interrupt request register PIR1(0DH)

0DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIR1	RCIF	TXIF	CMP1IF	PWMIF	RAIF	TMR1IF	TMR2IF	ADIF
R/W	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7 RCIF: USART receive interrupt flag bit
1= USART receive buffer is non-empty (cleared by reading from RCREG)
0= USART receive buffer is empty

Bit6 TXIF: USART transmit interrupt flag bit
1= USART transmit buffer is empty (cleared by writing to TXREG)
0= USART transmit buffer is non-empty

Bit5 CMP1IF: Comparator 1 interrupt flag bit (cleared by software)
1= A comparator 1 interrupt occurs
0= No comparator 1 interrupt occurred

Bit4 PWMIF: PWM interrupt flag bit (PWM0/1/2/3) (must be cleared by software)
1= A PWM interrupt occurs
0= No PWM interrupt occurred

Bit3 RAIF: PORTA interrupt-on-change flag bit
1= At least one pin level status in PORTA port has changed (cleared by software)
0= None of the PORTA universal I/O pin status has changed

Bit2 TMR1IF: TIMER1 overflow interrupt flag bit
1= TMR1 register is overflowed (must be cleared by software)
0= TMR1 register is not overflowed

Bit1 TMR2IF: TIMER2 and PR2 match interrupt flag bit (cleared by software)
1= A TMR2 and PR2 match interrupt occurs
0= No TMR2 and PR2 match interrupt occured

Bit0 ADIF: AD converter interrupt flag bit
1= AD conversion is completed (cleared by software)
0= AD conversion is not completed or not started

Peripheral interrupt request register PIR2(10FH)

10FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIR2	----	----	CMP2IF	----	----	----	----	----
R/W	----	----	R/W	----	----	----	----	----
Reset value	----	----	0	----	----	----	----	----

Bit7~Bit6 Unused

Bit5 CMP2IF: Comparator 2 interrupt flag bit (cleared by software)

1= A Comparator 2 interrupt occurs

0= No Comparator 2 interrupt occurred

Bit4~Bit0 Unused

7.3 Protection methods for interrupt

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

Example: Stack protection for ACC and STATUS

```
                ORG     0000H
                JP      START           ;user program start address
                ORG     0004H
                JP      INT_SERVICE     ;interrupt service routine
                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 is user defined)
                SWAPA   STATUS
                LD      STATUS_BAK,A    ;save the value of STATUS (STATUS_BAK is user 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. Firstly, the priority of each interrupt must be set in advance; secondly, the interrupt enable bit and the interrupt control bit are used to control whether the system responds to the interrupt. In the program, the interrupt control bit and interrupt request flag must be checked.

8. TIMER0

8.1 TIMER0 overview

TIMER0 is composed of the following functions:

- 8-bit timer/counter register (TMR0)
- 8-bit pre-scaler (shared with watchdog timer)
- Programmable internal or external clock source
- Programmable external clock edge selection
- External 32.768K oscillating clock (F_{LSE}) can be selected
- Overflow interrupt

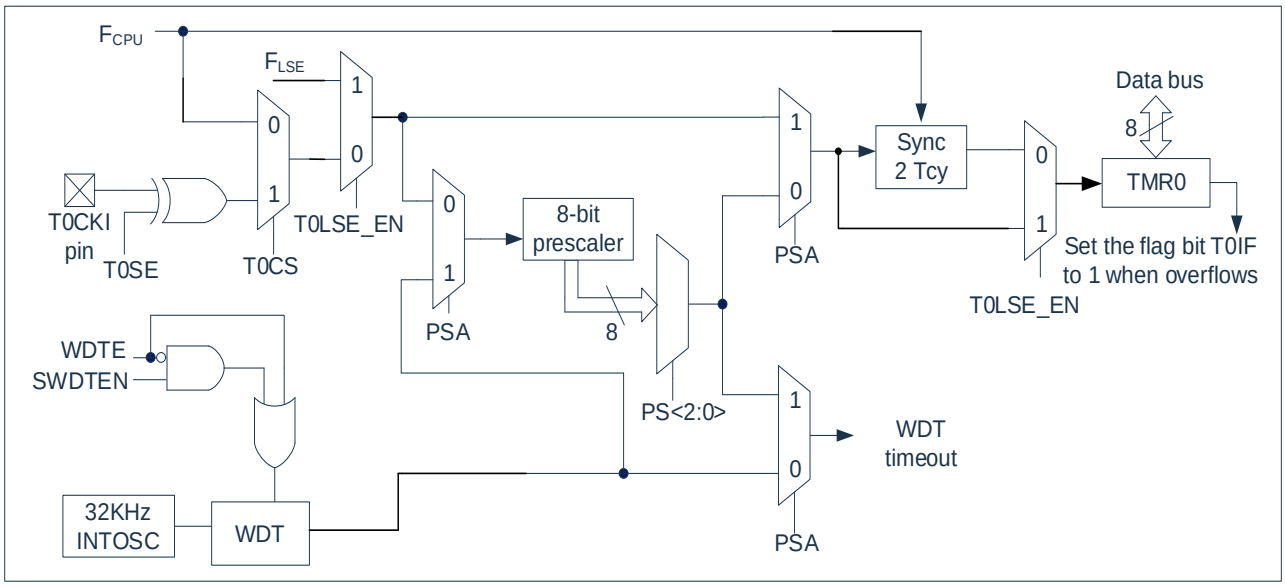


Figure 8-1: TIMER0/WDT structure diagram

Note:
1. T0SE, T0LSE_EN, T0CS, PSA, PS<2:0> are the bits in the OPTION_REG register.
2. SWDTEN is a bit in the OSCCON register.
3. WDTEN bit is in CONFIG.

8.2 How it works

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

8.2.1 8-bit timer mode

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

8.2.2 8-bit counter mode

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

8.2.3 Software programmable pre-scaler

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

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

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

8.2.4 Switch prescaler between TIMER0 and WDT module

Whether to use the prescaler from TIMER0 or WDT is completely controlled by software and can be changed dynamically. In order to avoid undesired chip reset, the following instruction should be executed when switching from TIMER0 to WDT.

```
CLR     TMR0            ;clear TMR0
CLRWDT                  ;clear WDT
LDIA    B'00xx1111'
LD      OPTION_REG,A
LDIA    B'00xx1xxx'     ;set new pre-scaler
LD      OPTION_REG,A
```

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

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

8.2.5 TIMER0 interrupt

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

Note: The TIMER0 interrupt wakes up the processor only if F_{LSE} is selected as the clock source.

8.3 TIMER0 related registers

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

TMR0 is an 8-bit readable and writable timer/counter, OPTION_REG is an 8-bit write-only register, the user can change the value of OPTION_REG to change the working mode of TIMER0, etc. Please refer to Section 2.6 Pre-scaler (OPTION_REG) applications.

8-bit timer/counter TMR0 (81H)

81H	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

Prescaler control register OPTION_REG (01H)

01H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION_REG	T0LSE_EN	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	1	1	1	1	0	1	1

Bit7 T0LSE_EN: TIMER0 clock source selection F_{LSE} enable bit
0= TIMER0 clock source is determined by T0CS
1= TIMER0 clock source selects F_{LSE}

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

Bit5 T0CS: TMR0 clock source selection bit
0= Internal instruction period clock (F_{CPU})
1= Transition edge of T0CKI pin

Bit4 T0SE: TIMER0 clock source edge selection bit
0= Increment when the T0CKI pin signal transitions from low to high
1= Increment when the T0CKI pin signal transitions from high to low

Bit3 PSA: Pre-scaler allocation bit
0= Pre-scaler allocated to TIMER0 mod
1= Pre-scaler allocated to WDT

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

PS2	PS1	PS0	TMR0 frequency division ratio	WDT frequency division ratio (WDT_DIV=DISABLE)	WDT frequency division ratio (WDT_DIV=ENABLE)
0	0	0	1:2	1:1	1:3
0	0	1	1:4	1:2	1:6
0	1	0	1:8	1:4	1:12
0	1	1	1:16	1:8	1:24
1	0	0	1:32	1:16	1:48
1	0	1	1:64	1:32	1:96
1	1	0	1:128	1:64	1:192
1	1	1	1:256	1:128	1:384

9. TIMER1

9.1 TIMER1 overview

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

- 16-bit timer/counter register (TMR1H:TMR1L)
- 3-bit pre-scaler
- Synchronous or asynchronous operation
- Wake-up on overflow (external clock asynchronous mode only)
- Special event trigger function (with ECCP)
- Programmable as internal or external clock sources
- Optional LP oscillator (FLSE)
- Gating TIMER1 via T1G pin (enable counting)
- Overflow interrupt
- Time base for capture/compare function

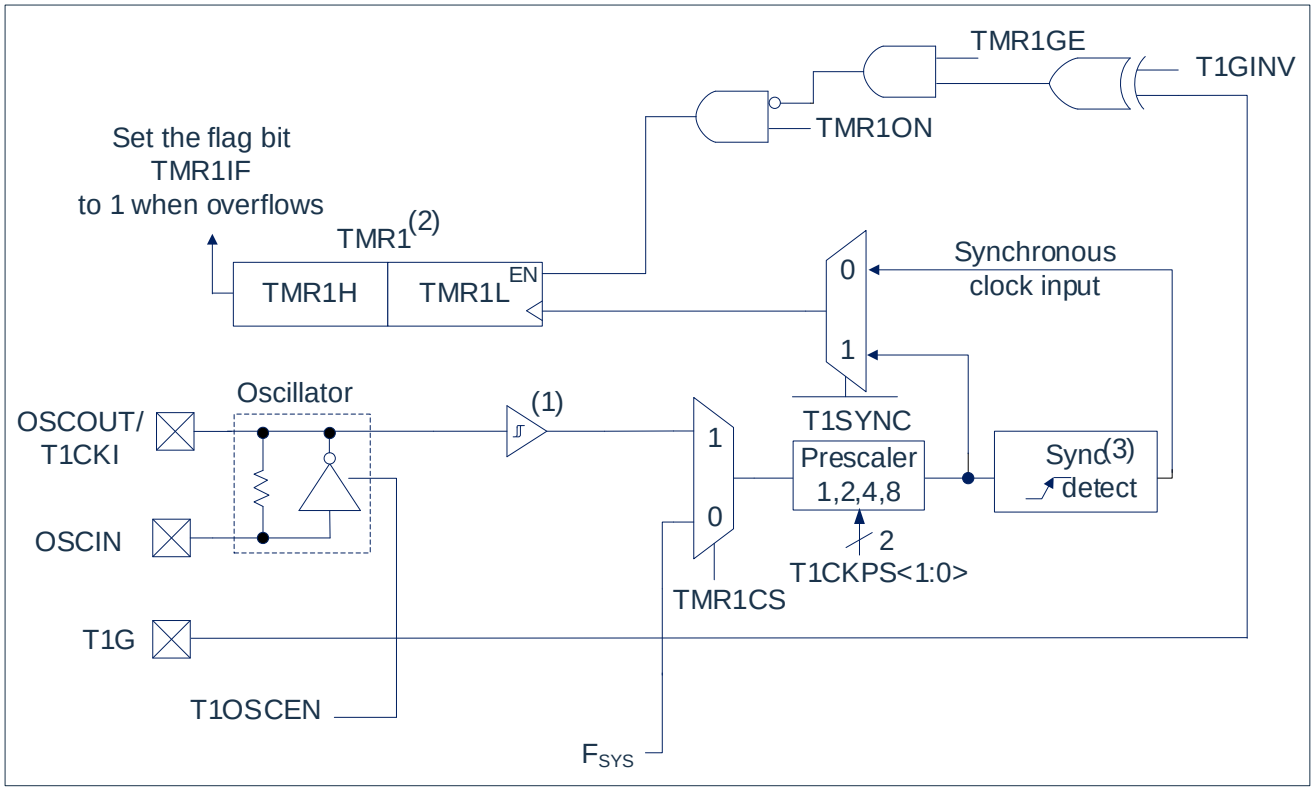


Figure 9-1: Block diagram of TIMER1

Note:

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

9.2 How it works

The TIMER1 module is a 16-bit incremental counter accessed through a pair of registers 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 in the T1CON register is used to select the clock source. When TMR1CS=0, the frequency of the clock source is F_{SYS}. When TMR1CS=1, the clock source is provided externally.

Clock source	TMR1CS
F_{SYS}	0
T1CKI pin	1

9.3.1 Internal clock sources

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 sources

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

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

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

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

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

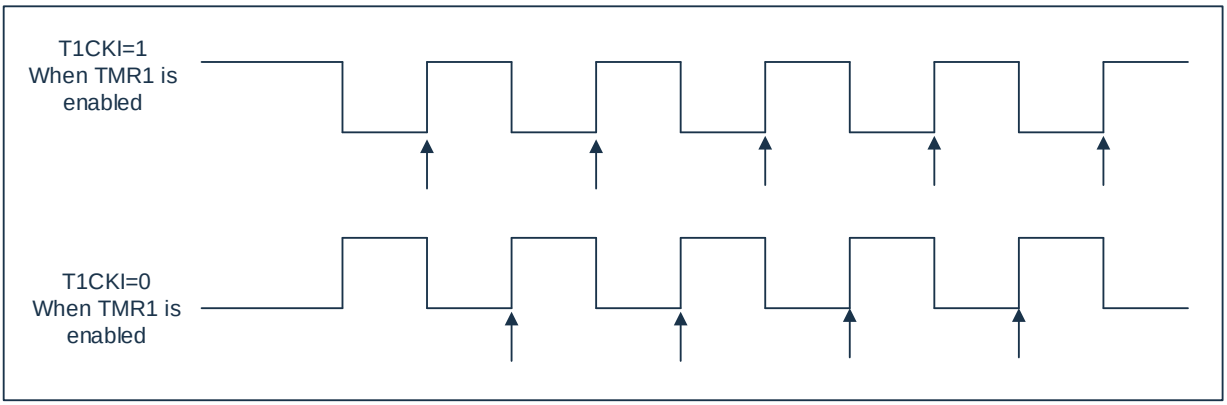


Figure 9-2: Incremental edge of TIMER1

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 prescaler

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

9.5 TIMER1 oscillator

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

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

PORTB0 and PORTB1 are set to analog inputs when the TIMER1 oscillator is enabled.

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

9.6 How it works in 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 9.6.1 Read and write operations to TIMER1 in asynchronous counter mode).

Note:

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

9.6.1 Read/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 unpredictable values in the register pair TMR1H:TMR1L.

9.7 TIMER1 gate control

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

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

The T1GINV bit of the T1CON register can be used to set the polarity of the TIMER1 gate control signal. The gate control signal can come from T1G pin. This bit can configure TIMER1 to time the high-level time or low-level time between events.

9.8 TIMER1 interrupt

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

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

In the interrupt service routine, clearing the TMR1IF bit will clear the interrupt.

Note: Before re-enabling the interrupt, the register pair TMR1H:TMR1L and the TMR1IF bit should be cleared.

9.9 How it works in sleep mode

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

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

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

9.10 TIMER1 related registers

There are three registers associated with TIMER1, namely, data register TMR1L, TMR1H, and control register T1CON.

TIMER1 data low bit register TMR1L(10CH)

10CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR1L	TMR1<7:0>							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	X	X	X	X	X	X	X	X

Bit7~Bit0 TMR1<7:0>: TIMER1 data low 8 bits

TIMER1 data high bit register TMR1H(10DH)

10DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR1H	TMR1<15:8>							
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 TMR1<15:8>: TIMER1 data high 8 bits

TIMER1 control register T1CON(101H)

101H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7 T1GINV: TIMER1 gate control signal polarity bit;
1= TIMER1 gate control signal is active high (TIMER1 counts when the gate control signal is high level);
0= The TIMER1 gate control signal is active low (TIMER1 counts when the gate control signal is low).

Bit6 TMR1GE: TIMER1 gate control enable bit.
If TMR1ON=0, ignore this bit.
If TMR1ON=1: 1= TIMER1 counting is controlled by TIMER1gate control function;
0=TIMER1always counts.

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

Bit3 T1OSCEN: LP oscillator enable control bit;
1= Enable LP oscillators the clock source of TIMER1;
0= Disable LP oscillator.

Bit2 T1SYNC: TIMER1 external clock input synchronous control bit.
TMR1CS=1: 1= Not synchronous with external clock input;
0= Synchronous with external clock input.
TMR1CS=0: ignore this bit, TIMER1 uses internal clock.

Bit1 TMR1CS: TIMER1 clock source selection bit;

1= From LP oscillator clock source or clock source from T1CKI pin (rising edge trigger);

0= Internal clock source F_{SYS}.

Bit0 TMR1ON: TIMER1enable bit;

1= Enable TIMER1;

0= Disable TIMER1.

10. TIMER2

10.1 TIMER2 overview

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

1. 8-bit timer register (TMR2)
2. 8-bit period register (PR2)
3. Interrupt when TMR2 matches PR2
4. Software programmable prescaler ratio (1:1, 1:4 and 1:16)
5. Software programmable postscaler ratio (1:1 to 1:16)
6. External 32.768KHz oscillating clock (F_{LSE}) can be selected

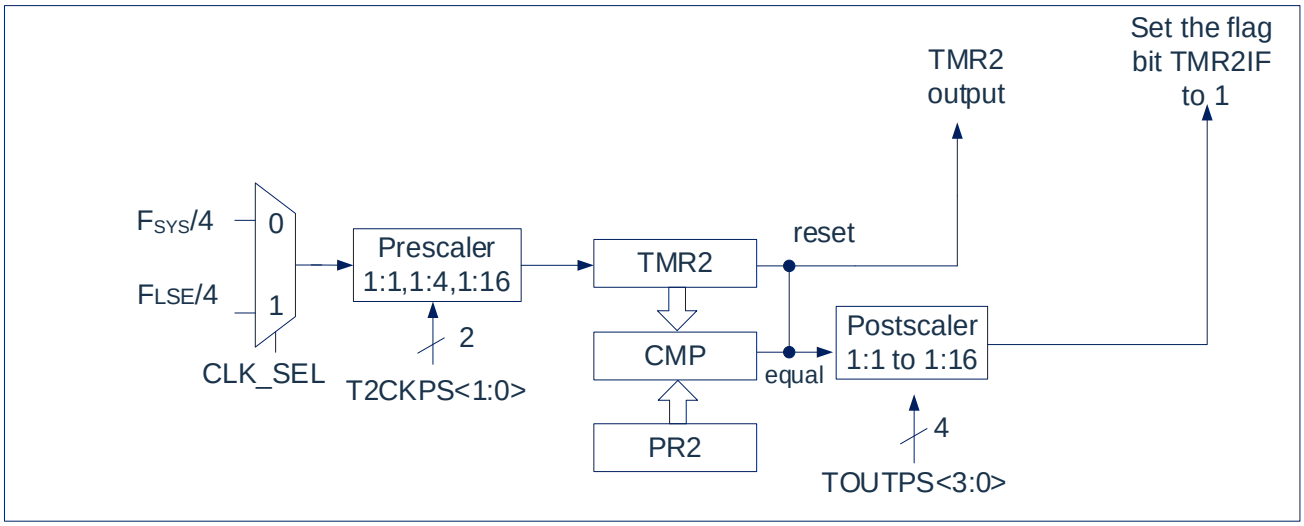


Figure 10-1: Block diagram of TIMER2

10.2 How it works

The input clock to the TIMER2 module is the system command clock (F_{SYS}/4) or an external 32.768 kHz oscillation (F_{LSE}). The clock is input to the TIMER2 prescaler, which is available in the following ratios: 1:1, 1:4 or 1:16. The output of the prescaler is then used to increment the TMR2 register.

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

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

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:

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

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

10.3 TIMER2 related registers

There are 3 registers related to TIMER2, namely data register TMR2, period register PR2 and control register T2CON.

TIMER2 data register TMR2 (12H)

12H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR2								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

TIMER2 period register PR2 (11H)

11H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PR2								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	1	1	1	1	1	1	1	1

TIMER2 control register T2CON(13H)

13H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T2CON	CLK_SEL	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
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 CLK_SEL: Clock source selection
1= Select external F_{LSE}/4 as TMR2 clock source (continue counting in sleep state)
0= Select internal F_{SYS}/4 as TMR2 clock source

Bit6~Bit3 TOUTPS<3:0>: TIMER2 output postscaler ratio select bit
0000= 1:1
0001= 1:2
0010= 1:3
0011= 1:4
0100= 1:5
0101= 1:6
0110= 1:7
0111= 1:8
1000= 1:9
1001= 1:10
1010= 1:11
1011= 1:12
1100= 1:13
1101= 1:14
1110= 1:15
1111= 1:16

Bit2 TMR2ON: TIMER2 enable bit
1= Enable TIMER2
0= Disable TIMER2

Bit1~Bit0 T2CKPS<1:0>: TIMER2 clock prescaler ratio select bit
00= 1
01= 4
1x= 16

11. 10-Bit PWM Module

The chip includes a 10-bit PWM module, which can be configured for four channels with a shared period and independent duty cycles, one channel with an independent period and duty cycle, or two pairs of complementary outputs.

11.1 Pin configuration

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

11.2 Relevant registers description

PWM control register PWMCON0(15H)

15H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMCON0	CLKDIV<2:0>			PWM4EN	PWM3EN	PWM2EN	PWM1EN	PWM0EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit5 CLKDIV<2:0>: PWM clock frequency division
111= F_{HSI}/128
110= F_{HSI}/64
101= F_{HSI}/32
100= F_{HSI}/16
011= F_{HSI}/8
010= F_{HSI}/4
001= F_{HSI}/2
000= F_{HSI}/1

Bit4~Bit0 PWM0/1/2/3/4EN: PWM0/1/2/3/4 enable bit
1= Enable PWM0/1/2/3/4
0= Disable PWM0/1/2/3/4

PWM control register PWMCON1(16H)

16H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMCON1	PWMIO_SEL<1:0>		PWM2DTEN	PWM0DTEN	---	---	DT_DIV<1:0>	
R/W	R/W	R/W	R/W	R/W	---	---	R/W	R/W
Reset value	0	0	0	0	---	---	0	0

Bit7~Bit6 PWMIO_SEL: PWM IO group selection
11= PWM in group A, PWM0-RA0,PWM1-RA1,PWM2-RA2,PWM3-RA3,PWM4-RA4
10= PWM in group B, PWM0-RA0,PWM1-RA1,PWM2-RA2,PWM3-RB2,PWM4-RB1
01= PWM in group C, PWM0-RA5,PWM1-RB7,PWM2-RB6,PWM3-RB5,PWM4-RB4
00= PWM in group D, PWM0-RB0,PWM1-RB1,PWM2-RB3,PWM3-RB4,PWM4-RB2

Bit5 PWM2DTEN: PWM2 dead zone enable bit
1= Enable the dead zone function for PWM2. PWM2 and PWM3 form a complementary output pair.
0= Disable the dead zone function for PWM2.

Bit4 PWM0DTEN: PWM0 dead zone enable bit.
1= Enable the dead zone function for PWM0. PWM0 and PWM1 form a complementary output pair.
0= Disable the dead zone function for PWM0.

Bit3~Bit2 Unused.

Bit1~Bit0 DT_DIV<1:0>: Dead zone clock source frequency division
11= $F_{HSI}/8$
10= $F_{HSI}/4$
01= $F_{HSI}/2$
00= $F_{HSI}/1$

Note: When selecting PWMD group, PWM2, PWM3 can be configured on RB3, RB4 or RA2, RA3 via config.

PWM control register PWMCON2(1DH)

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

Bit7~Bit5 Unused
Bit4~Bit0 PWM0/1/2/3/4DIR PWM output reverse control bit
1= PWM0/1/2/3/4 reverse outputs
0= PWM0/1/2/3/4 normal outputs

PWM0~PWM3 period low bit register PWMTL (17H)

17H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMTL	PWMT<7:0>							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMT<7:0>: PWM0~PWM3 period low 8 bits

PWM4 period low bit register PWMT4L(1CH)

1CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMT4L	PWM4T<7:0>							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWM4T<7:0>: PWM4 period low 8 bits

Period high bit register PWMTH (18H)

18H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMTH	---	---	PWMD4<9:8>		PWM4T<9:8>		PWMT<9:8>	
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	---	0	0	0	0	0	0

Bit7~Bit6 Unused.
Bit5~Bit4 PWMD4<9:8>: High 2 bits of PWM4 duty cycle
Bit3~Bit2 PWM4T<9:8>: High 2 bits of PWM4 period
Bit1~Bit0 PWMT<9:8>: High 2 bits of PWM0~PWM3 period

Note: Writing PWMD4<9:8> does not take effect immediately, it needs to be written to PWMD4L to take effect.

PWM0 duty cycle low bit register PWMD0L (19H)

19H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD0L	PWMD0<7:0>							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD0<7:0>: PWM0 duty cycle low 8 bits

PWM1 duty cycle low bit register PWMD1L (1AH)

1AH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD1L	PWMD1<7:0>							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD1<7:0>: PWM1 duty cycle low 8 bits

PWM2 duty cycle low bit register PWMD2L (9BH)

9BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD2L	PWMD2<7:0>							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD2<7:0>: PWM0 duty cycle low 8 bits

PWM3 duty cycle low bit register PWMD3L (9CH)

9CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD3L	PWMD3<7:0>							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD3<7:0>: PWM3 duty cycle low 8 bits

PWM4 duty cycle low bit register PWMD4L (1BH)

1BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD4L	PWMD4<7:0>							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD4<7:0>: PWM4 duty cycle low 8 bits

PWM0 and PWM1 duty cycle high bit register PWMD01H (1EH)

1EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD01H	---	---	PWMD1<9:8>		---	---	PWMD0<9:8>	
R/W	---	---	R/W	R/W	---	----	R/W	R/W
Reset value	---	---	0	0	---	----	0	0

Bit7~Bit6 Unused
Bit5~Bit4 PWMD1<9:8>: PWM1 duty cycle high 2 bits
Bit3~Bit2 Unused.
Bit1~Bit0 PWMD0<9:8>: PWM0 duty cycle high 2 bits

Note: Writing PWMD0<9:8> does not take effect immediately, it needs to write PWMD0L to take effect. Writing PWMD1<9:8> does not take effect immediately, it needs to write PWMD1L to take effect.

PWM2 and PWM3 duty cycle high bit register PWMD23H (9EH)

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

Bit7~Bit6 Unused.
Bit5~Bit4 PWMD3<9:8>: PWM3 duty cycle high 2 bits
Bit3~Bit2 Unused.
Bit1~Bit0 PWMD2<9:8>: PWM2 duty cycle high 2 bits

Note: Writing PWMD2<9:8> does not take effect immediately, it needs to write PWMD2L to take effect.
Writing PWMD3<9:8> does not take effect immediately, it needs to write PWMD3L to take effect.

PWM0 and PWM1 dead time register PWM01DT(1FH)

1FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM01DT	---	---	PWM01DT<5:0>					
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	---	0	0	0	0	0	0

Bit7~Bit6 Unused.
Bit5~Bit0 PWM01DT<5:0>: PWM0 and PWM1 dead time

PWM2 and PWM3 dead time register PWM23DT(9DH)

9DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM23DT	---	---	PWM23DT<5:0>					
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	---	0	0	0	0	0	0

Bit7~Bit6 Unused
Bit5~Bit0 PWM23DT<5:0>: PWM2 and PWM3 dead time

11.3 10-bit PWM register write sequence

Since the 10-bit PWM duty cycle value is allocated in two registers, when modifying the duty cycle, the program always modifies these two registers successively. In order to ensure the correctness of the duty cycle value, the chip is designed with an internal cache loading function. To operate the 10-digit duty cycle value, the following sequence should be strictly followed.

1) Write the higher 2 bits, the high 2-bit value is just written to the internal cache;

2) Write the lower 8 bits, then the full 10-bit duty cycle value is latched;

3) The above operations are only for PWM0, PWM1, PWM2, PWM3, PWM4 duty cycle registers.

11.4 10-bit PWM period

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

Formula 1: PWM cycle calculation formula.

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

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

When the PWM cycle counter is equal to PWMT, the following three events occur during the next incremental counting cycle:

- PWM period counter is cleared;
- PWMx pin is set to 1;
- New period of PWM is latched;
- New duty cycle of PWM is latched;
- Generate a PWM interrupt flag bit (No interrupt for PWM4).

11.5 10-bit PWM duty cycle

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

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

Formula 2: Pulse width calculation formula

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

Formula 3: PWM duty cycle calculation formula

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

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

11.6 System clock frequency change

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

11.7 Programmable dead time delay mode

The complementary output mode can be enabled by setting PWMxDT_EN, and the dead time delay function is automatically enabled after the complementary output is enabled.

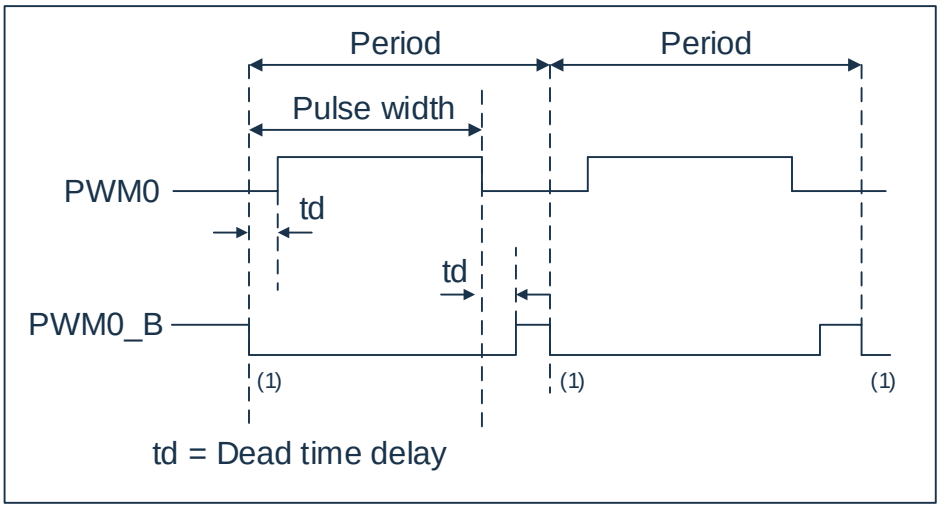


Figure 11-1: Example of PWM dead time delay output

The dead time calculation formula is:

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

11.8 10-bit PWM configuration

The following steps should be performed when using the PWM module.

1. Set the corresponding TRIS bit to 1 to make it as an input pin.
2. Set the PWM period by loading the PWMTH and PWMTL registers.
3. Set the PWM duty cycle by loading the PWMDxL and PWMDxxH registers.
4. Clear the PWMIF flag bit.
5. Set the PWMENx bit to enable the corresponding PWM outputs.
6. After the new PWM period starts, enable PWM output:
   - Wait for PWMIF bit set to 1.
   - Enable the PWM pin output driver by clearing the corresponding TRIS bit.

12. Universal Synchronous/Asynchronous Transmitter (USART)

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

The USART mod includes the following functions:

- Full-duplex asynchronous transmission and reception
- Single-character output buffer
- Double-character input buffer
- Frame error detection for received characters
- Half-duplex synchronous slave mode
- Programmable character length of 8 or 9 bits
- Input buffer overflow error detection
- Half-duplex synchronous master mode
- Programmable clock polarity in synchronous mode

Figure 12-1 and Figure 12-2 below are the block diagrams of the USART transmitter.

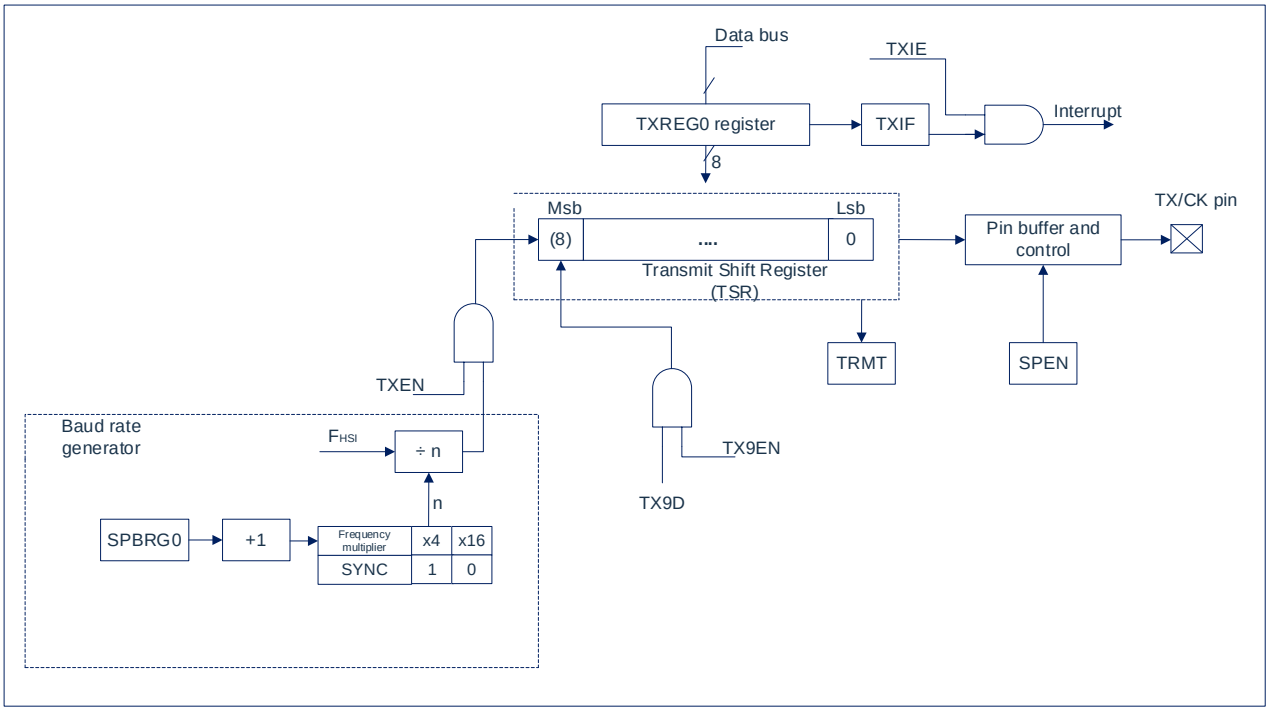


Figure 12-1: USART transmit block diagram

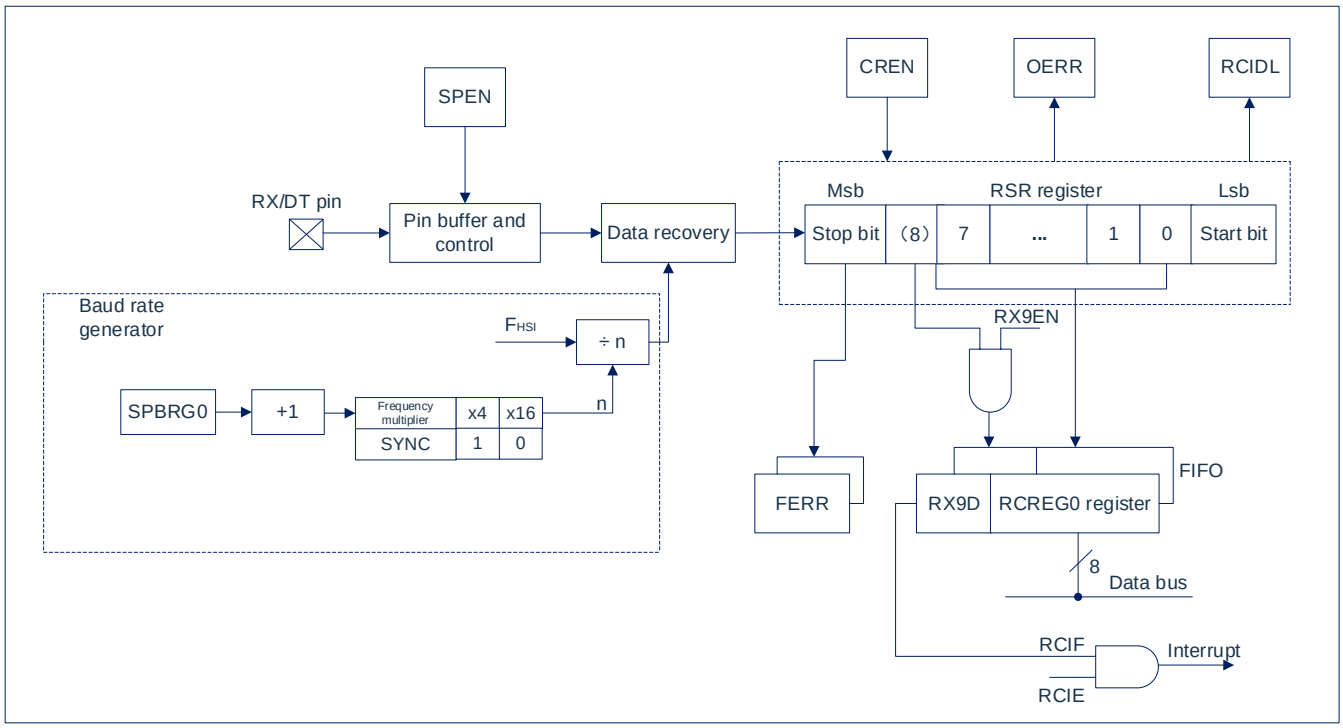


Figure 12-2: USART receive block diagram

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

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

12.1 USART asynchronous mode

The USART uses the Non-Return-to-Zero (NRZ) format for sending and receiving data. It employs two voltage levels to implement NRZ: the VOH (mark state) representing the logical '1' bit and the VOL (space state) representing the logical '0' bit. When sending a sequence of the same data bits in NRZ format, the output level will maintain the level of the current bit and will not return to the middle voltage level after transmitting each bit. The NRZ transmitter is idle in the mark state. Each transmitted character includes a start bit, followed by 8 or 9 data bits and one or more stop bits. The start bit is always in the space state, and the stop bit is always in the mark state. The most common data format is 8 bits. The duration of each transmitted bit is 1/(baud rate). An on-chip 8-bit/16-bit baud rate generator can be used to generate standard baud rate frequencies from the system oscillator.

The USART transmits and receives the Least Significant Bit (LSB) first. The transmitter and receiver are functionally independent but operate with the same data format and baud rate. The hardware does not support parity checking, but it can be implemented in software (with the parity bit being the 9th data bit).

12.1.1 USART asynchronous transmitter

Figure 12-1 shows the block diagram of the USART transmitter. The core of the transmitter is the serial transmit shift register (TSR), which cannot be directly accessed by software. The TSR obtains data from the TXREG transmit buffer register.

12.1.1.1 Enable the transmitter

The USART transmitter for asynchronous operation is enabled by configuring the following three control bits:

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

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

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

Note:

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

12.1.1.2 Transmit data

Writing a character to the TXREG register starts the transmission. If this is the first character or if the previous character has been completely removed from the TSR, the data in TXREG is immediately transferred to the TSR register for transmission. If the TSR still contains all or part of the previous character, the new character data will remain in TXREG until the stop bit of the previous character has been fully transmitted. After the stop bit is sent and a TCY (instruction cycle) has elapsed, the data waiting in TXREG will be transferred to TSR. Once the data has been transferred from TXREG to TSR, the transmission of the start bit, data bit, and stop bit sequence will begin immediately.

12.1.1.3 Transmit interrupts

If 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.

12.1.1.4 TSR status

The TRMT bit in the TXSTA register indicates the status of the TSR register. TRMT is a read-only bit. When the TSR register is empty, the TRMT bit is set to 1. When a character is transferred from TXREG to the TSR register, TRMT is cleared to 0. The TRMT bit remains cleared until all bits have been transmitted and removed from the TSR register. There is no interrupt logic associated with this bit, so the user must poll this bit to determine the status of the TSR.

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

12.1.1.5 Transmit 9-bit characters

USART supports 9-bit character transmission. When the TX9EN bit in the TXSTA register is set to 1, the USART will transmit each character as a 9-bit word. The TX9D bit in the TXSTA register represents the 9th bit, which is the most significant data bit (MSB). When transmitting 9-bit data, the TX9D bit must be written before writing the 8 least significant bits to TXREG. Once the TXREG register is written, the 9 data bits will immediately be transferred to the TSR shift register for transmission.

12.1.1.6 Configure asynchronous transmission

1. Initialize the SPBRG register to obtain the required baud rate (see Chapter USART Baud Rate Generator (BRG)).
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit to 1.
3. If 9-bit transmission is required, set the TX9EN control bit to 1. When the receiver is configured for address detection, set the 9th data bit to 1 to indicate that the 8 lowest significant data bits represent an address.
4. Set the TXEN control bit to 1 to enable transmission; this will cause the TXIF interrupt flag to be set to 1.
5. If interrupts are needed, set the TXIE interrupt enable bit in the PIE1 register to 1. If the GIE and PEIE bits in the INTCON register are also set to 1, an interrupt will be generated immediately.
6. If 9-bit data transmission is selected, the 9th bit should be loaded into the TX9D data bit.
7. Load the 8-bit data into the TXREG register to start the transmission of data.

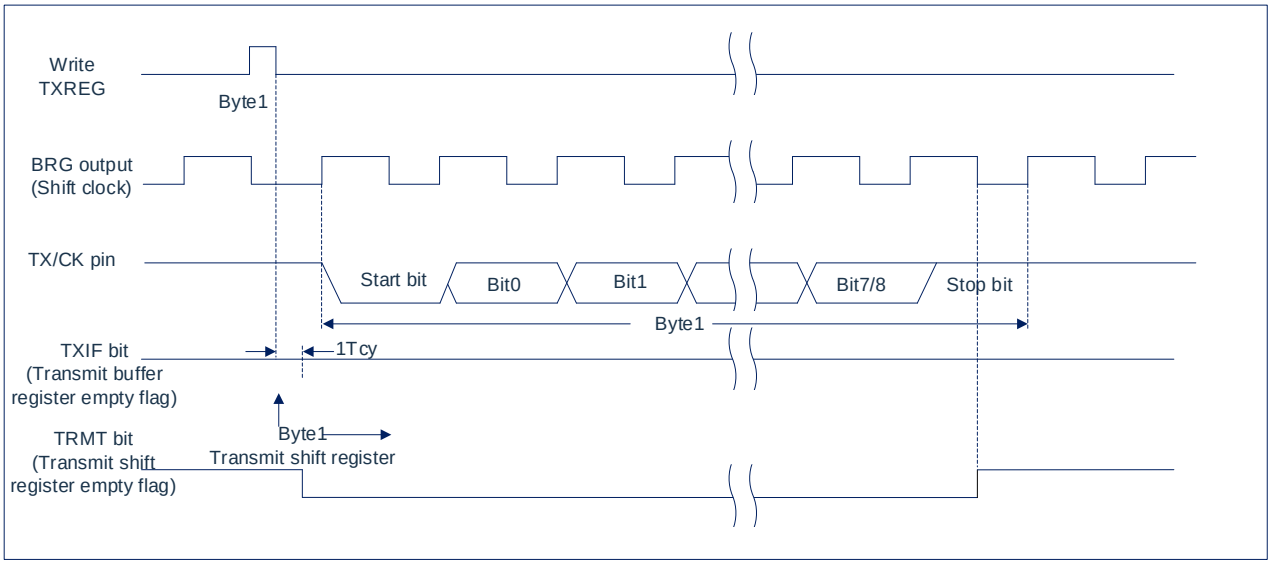


Figure 12-3: Asynchronous transmission

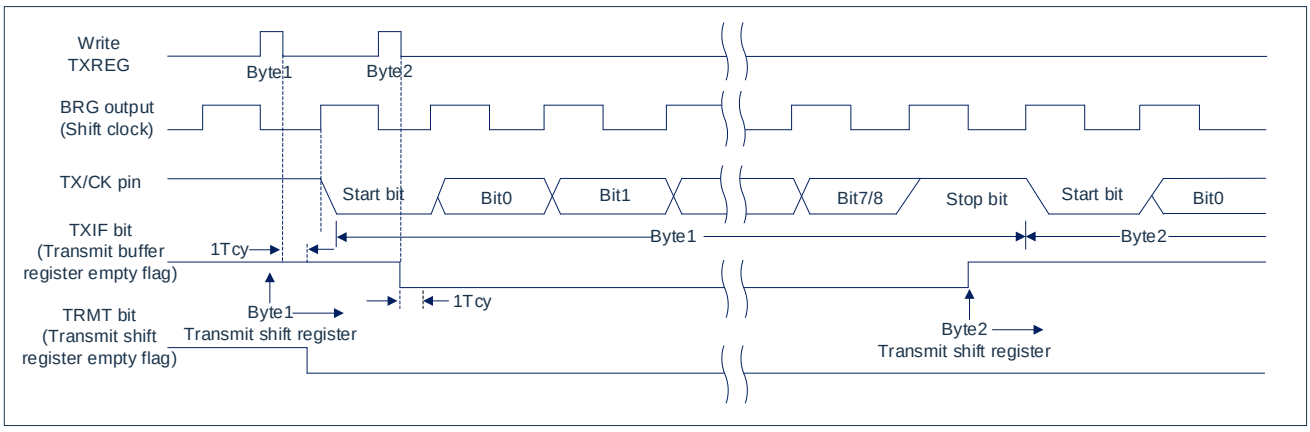


Figure 12-4: Asynchronous transmit (back to back)

Note: This timing diagram shows two consecutive transmissions.
In asynchronous send-back-to-back mode, when using the TXIF flag to determine whether data can be written to TXREG, it is necessary to first write to TXREG and then check the TRMT bit. If the TRMT bit is 1, a second write to TXREG should be performed.

Example：Asynchronous transmission (back to back)

```
INTERRUPT_USART_SEND:                                   Interrupted Asynchronous Transmission Subroutine
        SNZB    PIR1,TXIF                               Check if the TXIF flag is set to 1. If it is 0, exit the
                                                        interrupt function.
        RETI
        LDIA    040H                                    The data to be transmitted is 0x40H
        LD      TXREG,A                                 Write the data to be transmitted into TXREG
        SNZB    TXSTA,TRMT                              Check if the TRMT flag is set to 1. If it is 0, exit the
                                                        interrupt function.
        RETI
                                                        Reload 0x40H into TXREG and perform a second write
        LD      TXREG,A                                 operation.
        RETI
```

12.1.2 USART asynchronous receiver

The asynchronous mode is commonly used in RS-232 systems. Figure 12-2 shows a block diagram of the receiver. Receive data and drive data recovery circuits on the RX/DT pins. The data recovery circuit is essentially a high-speed shift register operating at 16 times the baud rate, while the serial receive shift register (RSR) operates at the bit rate. Once all 8 or 9 data bits of a character are shifted in, they are immediately transferred to a 2-character FIFO (First In, First Out) buffer. The FIFO buffer allows the receiver to store 2 complete characters along with the start bit of a 3rd character, after which the received data must be retrieved by software and provided to the USART receiver. The FIFO and RSR registers cannot be directly accessed by software; the received data is accessed through the RCREG register.

12.1.2.1 Enable the receiver

To enable the USART receiver for asynchronous operation, configure the following three control bits:

- 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/CK 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/DT pin is automatically configured as an input pin, regardless of the state of the corresponding TRIS bit.

12.1.2.2 Receive data

The receiver's data recovery circuit begins receiving a character at the falling edge of the first bit. The first bit, typically referred to as the start bit, is always 0. The data recovery circuit counts half a bit time to reach the center of the start bit and checks if the bit is still 0. If the bit is not 0, the data recovery circuit abandons the character and does not generate an error, instead continuing to search for the next falling edge of a start bit. If the start bit passes the zero check, the data recovery circuit counts one full bit time to reach the center of the next bit. The sample and hold circuit samples this bit, and the corresponding sampled value (0 or 1) is shifted into the Receive Shift Register (RSR). This process is repeated for each data bit, which are all shifted into the RSR register. Once all the data bits have been sampled and shifted into the RSR, the timing of the last bit is measured, and its level is sampled. This bit is the stop bit and is always 1. If the data recovery circuit samples a 0 at the stop bit position, the frame error flag for that character will be set to 1. Otherwise, if the stop bit is correctly sampled as 1, the frame error flag is cleared.

After all the data and stop bits have been received, the character in the RSR is immediately transferred to the USART receive FIFO, and the RCIF interrupt flag in the PIR1 register is set to 1. The character at the top of the FIFO can be retrieved by reading the RCREG register, which moves it out of the FIFO.

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

12.1.2.3 Receive interrupts

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 to 1:

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

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

12.1.2.4 Receive frame errors

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.

A frame 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. Clearing the CREN bit of the RCSTA register forces the FERR bit to be cleared. Framing error itself will not cause an interrupt.

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

12.1.2.5 Receive overflow errors

The receive FIFO buffer can hold 2 characters. However, if a complete third character is received before accessing the FIFO, an overflow error will occur. At this point, the OERR bit in the RCSTA register will be set to 1. Characters in the FIFO buffer can be read, but no additional characters can be received until the error is cleared. The error can be cleared by clearing the CREN bit in the RCSTA register or by resetting the USART by clearing the SPEN bit in the RCSTA register.

12.1.2.6 Receive 9-bit characters

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

12.1.2.7 Configure asynchronous reception

1. Initialize the SPBRG register to obtain the required baud rate (see Chapter USART Baud Rate Generator (BRG)).
2. Set the SPEN bit to enable the serial port. The SYNC bit must be cleared to operate in asynchronous mode.
3. If interrupts are required, set the RCIE bit in the PIE1 register and the GIE and PEIE bits in the INTCON register to 1.
4. If 9-bit data reception is needed, set the RX9EN bit to 1.
5. Set the CREN bit to enable reception.
6. When a character is transferred from the RSR to the receive buffer, the RCIF interrupt flag will be set to 1. If the RCIE interrupt enable bit is also set to 1, an interrupt will be generated.
7. Read the RCREG register to obtain the 8 low data bits received from the receive buffer.
8. Read the RCSTA register to get the error flags and the 9th data bit (if 9-bit data reception is enabled).
9. If an overflow occurs, clear the OERR flag by clearing the CREN receiver enable bit.

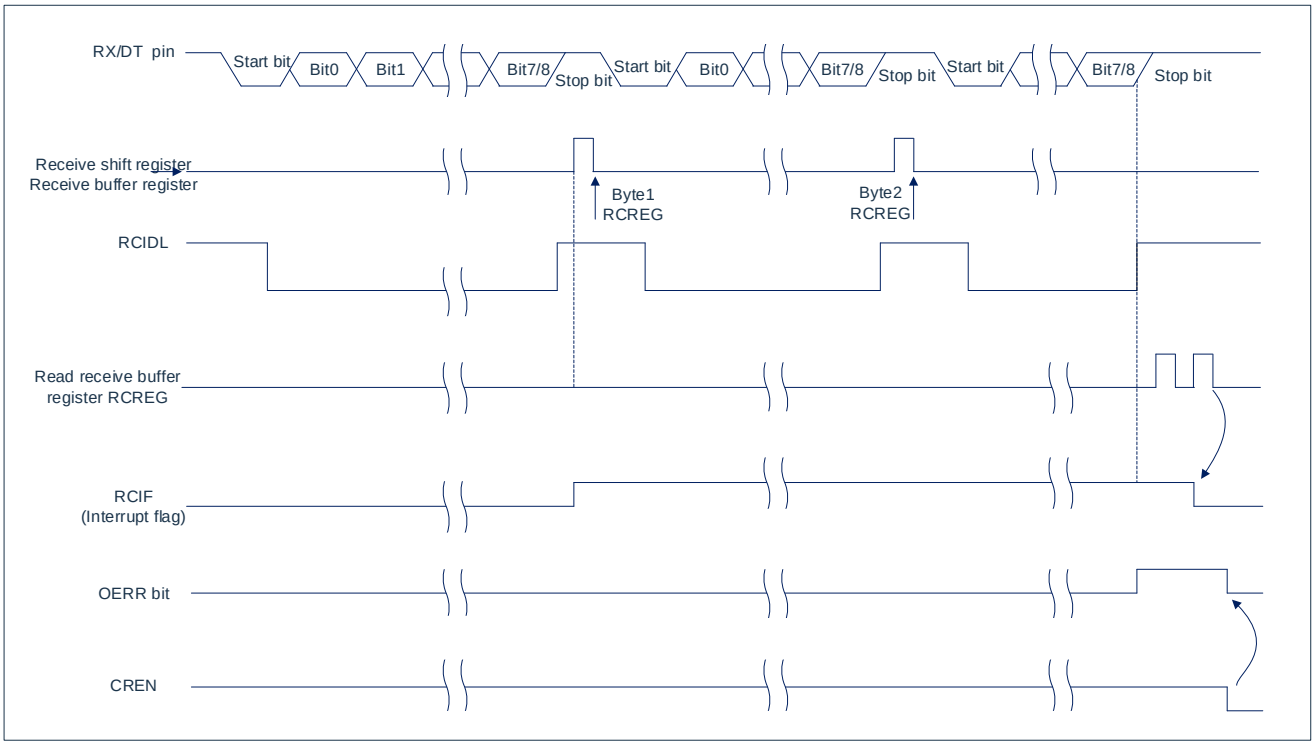


Figure 12-5: Asynchronous reception

Note: This timing diagram illustrates the scenario of receiving three bytes at the RX input pin. Reading RCREG (the receive buffer) after the third byte results in the OERR (overflow) bit being set to 1.

12.2 Clock accuracy during asynchronous operation

The output of the internal oscillator circuit (INTOSC) is calibrated by the manufacturer. However, INTOSC will drift in frequency during VDD or temperature changes, which will directly affect the asynchronous baud rate.

12.3 USART related registers

Transmit status and control register TXSTA(118H)

118H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TXSTA	CSRC	TX9EN	TXEN	SYNC	SCKP	----	TRMT	TX9D
R/W	R/W	R/W	R/W	R/W	R/W	----	R	R/W
Reset value	0	0	0	0	0	----	1	0

Bit7 CSRC: Clock source selection bit
Asynchronous mode: Any value
Synchronous mode:
1=Master mode (Clock signal generated by internal BRG)
0=Slave mode (Clock signal generated by external clock source)

Bit6 TX9EN: 9-bit transmission enable bit
1= Enable 9-bit transmission
0= Select 8-bit transmission

Bit5 TXEN: Transmission enable bit (1)
1= Enable transmission
0= Disable transmission

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

Bit3 SCKP: Synchronous cock polarity selection bit
Asynchronous mode:
1=Invert the data character level and send to TX/CK pin
0=Send data character directly to TX/CK pin
Synchronous mode:
0=Data transmitted on clock rising edge
1=Data transmitted on clock falling edge

Bit2 Unused

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

Bit0 TX9D: Transmit 9th bit data
Can be an address/data bit or a parity bit

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

Receive status and control register RCSTA(119H)

119H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RCSTA	SPEN	RX9EN	SREN	CREN	RCIDL	FERR	OERR	RX9D
R/W	R/W	R/W	R/W	R/W	R	R	R	R
Reset value	0	0	0	0	1	0	0	0

Bit7 SPEN: Serial port enable bit
1= Enable serial port (configure RX/DT and TX/CK pins as serial port pins)
0= Disable serial port (hold in reset)

Bit6 RX9EN: 9-bit reception enable bit
1= Select 9-bit reception
0= Select 8-bit reception

Bit5 SREN: Single-byte reception enable bit
Asynchronous mode: Any value
Synchronous master mode:
1= Enable single-byte reception
0= Disable single-byte reception
Clear this bit after reception is completed
Synchronous slave mode: Any value

Bit4 CREN: Continuous receive enable bit
Asynchronous mode:
1= Enable reception
0= Disable reception
Synchronous mode:
1= Enable continuous reception until CREN is cleared (CREN overrides SREN)
0= Disable continuous reception

Bit3 RCIDL: Receive idle flag bit
Asynchronous mode:
1= Receiver is idle
0= Start bit received, receiver is currently receiving data.
Synchronous mode: Any value

Bit2 FERR: Frame error bit
1= Frame error (can be updated by reading RCREG register to receive the next valid byte).
0= No frame error

Bit1 OERR: Overflow error bit
1= Overflow error (can be cleared by setting CREN to zero)
0= No overflow error

Bit0 RX9D: Received 9th bit data
This bit can be an address/data bit or a parity bit, must be calculated by user firmware.

Transmit data register TXREG(11BH)

11BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TXREG								
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

Receive data register RCREG(11CH)

11CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RCREG								
R/W	R	R	R	R	R	R	R	R
Reset value	0	0	0	0	0	0	0	0

Baud rate data register SPBRG(11AH)

11AH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SPBRG								
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

12.4 USART baud rate generator (BRG)

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

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

Table 12-1 contains formulas for calculating the baud rate. Formula 1 provides an example of calculating the baud rate and baud rate error.

Table 12-1 also includes typical baud rates and baud rate error values for various asynchronous modes, which can be convenient for your use.

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

If the system clock changes during valid reception, it may result in reception errors or data loss. To avoid this issue, the status of the RCIDL bit should be checked to ensure that the reception operation is idle before changing the system clock.

Formula 1: Calculate baud rate error

When F_{HSI} is 16MHz, the target baud rate: 9600 bps, using an 8-bit BRG device in asynchronous mode.

$$\text{Target baud rate}=\frac{F_{HSI}}{16([SPBRG]+1)}$$

SPBRG:

$$X=\frac{\frac{F_{HSI}}{\text{Target baud rate}}}{16}-1=\frac{\frac{160000000}{9600}}{16}-1=[103.16]=103$$

$$\text{Calculate baud rate}=\frac{16000000}{16\ (103+1)}=9615$$

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

Table 12-1: Baud rate formula

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

Note: n= value of SPBRG register.

Table 12-2: Baud rate in asynchronous mode

Target baud rate	SYNC=0		
	F_{HSI}=16.00MHz		
	Real baud rate	Error (%)	SPBRG value
2400	----	----	----
9600	9615	0.16	103
10417	10417	0	95
14400	14492	0.63	68
19200	19230	0.16	51

12.5 USART synchronous mode

Synchronous serial communication is typically used in systems with one master device and one or more slave devices. The master device contains the circuitry necessary to generate the baud rate clock and provides this clock to all devices in the system. Slave devices can use the master clock, eliminating the need for internal clock generation circuitry.

In synchronous mode, there are two signal lines: a bidirectional data line and a clock line. Slave devices use the external clock provided by the master device to serially shift data into or out of the respective receive and transmit shift registers. Because a bidirectional data line is used, synchronous operation can only be conducted in a half-duplex manner. Half-duplex means that both the master and slave devices can receive and send data, but cannot do so simultaneously. The USART can function as either a master device or a slave device.

Synchronous transmission does not require the use of start bits and stop bits.

12.5.1 Synchronous master mode

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

- SYNC=1
- CSRC=1
- SREN=0 (for transmission); SREN=1 (for reception)
- CREN=0 (for transmission); CREN=1 (for reception)
- SPEN=1

Setting the SYNC bit in the TXSTA register to 1 configures the USART for synchronous operation. Setting the CSRC bit in the TXSTA register to 1 configures the device as the master device. Clearing the SREN and CREN bits in the RCSTA register ensures the device is in transmission mode; otherwise, it is configured for reception. Setting the SPEN bit in the RCSTA register to 1 enables the USART.

12.5.1.1 Master clock

Synchronous data transmission uses a separate clock line to synchronize data transfer. The device configured as the master provides the clock signal on the TX/CK pin. When the USART is configured for synchronous transmission or reception, the TX/CK output driver is automatically enabled. Serial data bits change on the rising edge of each clock cycle to ensure they are valid on the falling edge. The duration of each data bit is one clock cycle, so the number of data bits determines the number of clock cycles produced.

12.5.1.2 Clock polarity

The device provides a clock polarity option for compatibility with Microwire. The clock polarity is selected by the SCKP bit in the TXSTA register. Setting the SCKP bit to 1 configures the clock idle state to high. When SCKP is set to 1, data changes on the falling edge of each clock. Clearing the SCKP bit sets the clock idle state to low. When the SCKP bit is cleared, data changes on the rising edge of each clock.

12.5.1.3 Synchronous master transmission

Data is output from the device's RX/DT pin. When the USART is configured for synchronous master transmission, the RX/DT and TX/CK output pins are automatically enabled.

To start sending data, write a character to the TXREG register. If the TSR still holds all or part of the previous character, the new character data is stored in TXREG until the stop bit of the previous character has been sent. 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. Once the character is transferred from TXREG to TSR, data transmission begins immediately. Each data bit changes on the rising edge of the master clock and remains valid until the next rising edge.

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

12.5.1.4 Synchronous master transmission configuration

1. Initialize the SPBRG register to achieve the desired baud rate. (see Chapter USART Baud Rate Generator (BRG))
2. Set the SYNC, SPEN, and CSRC bits to 1 to enable the synchronous master serial port.
3. Clear the SREN and CREN bits to disable reception mode.
4. Set the TXEN bit to 1 to enable transmission mode.
5. If you need to send 9-bit characters, set the TX9EN bit to 1.
6. If interrupts are required, set the TXIE bit in the PIE1 register, and the GIE and PEIE bits in the INTCON register to 1.
7. If choosing to send 9-bit characters, load the 9th bit of data into the TX9D bit.
8. Start transmission by loading data into the TXREG register.

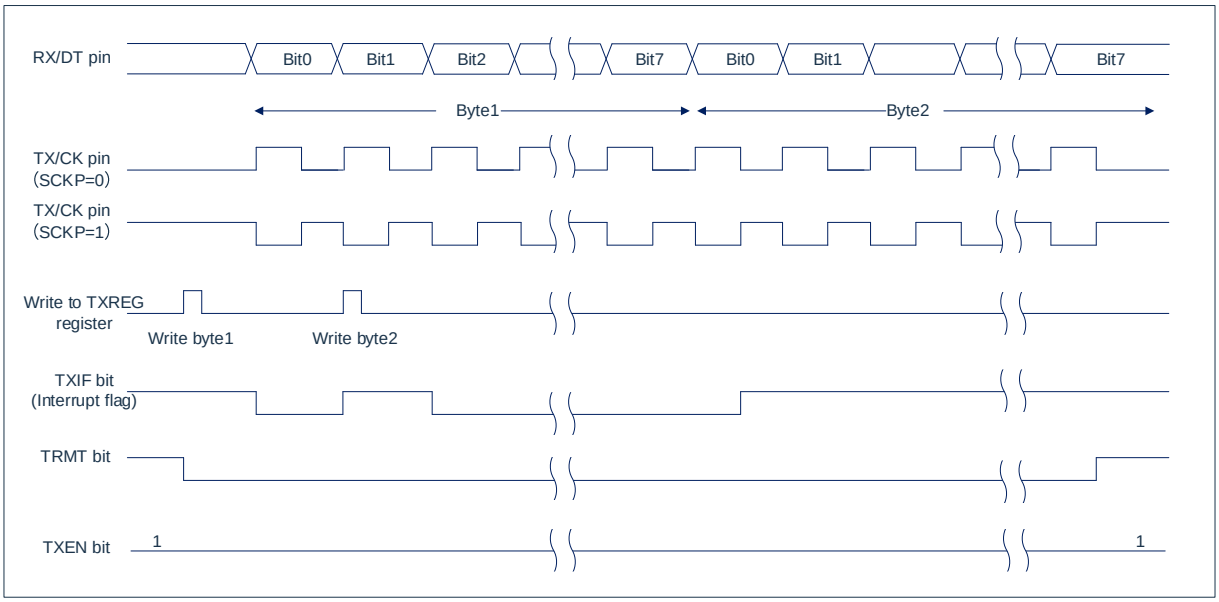


Figure 12-6: Synchronous transmission

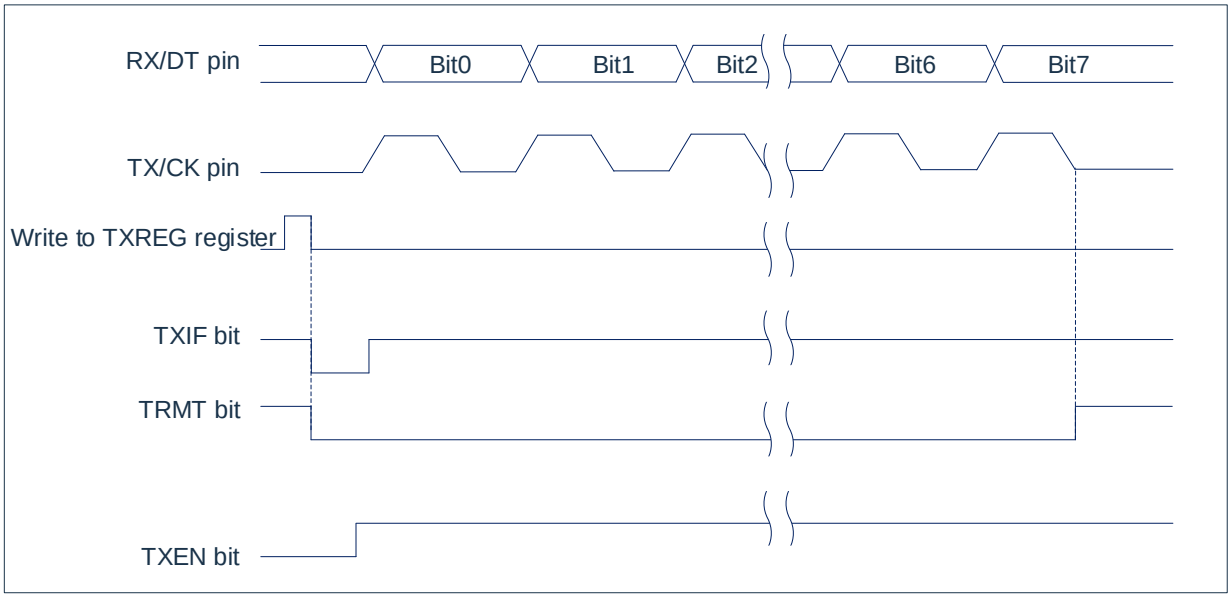


Figure 12-7: Synchronous transmit (through TXEN)

12.5.1.5 Synchronous master reception

Data is received on the RX/DT pin. When the USART is configured for synchronous master reception, the output driver for the RX/DT pin is automatically disabled.

In synchronous mode, enable reception by setting either the single character receive enable bit (SREN bit in the RCSTA register) or the continuous receive enable bit (CREN bit in the RCSTA register) to 1. When SREN is set to 1 and CREN is cleared, the number of clock cycles generated corresponds to the number of data bits in a single character. After the transmission of one character is completed, the SREN bit is automatically cleared. When CREN is set to 1, continuous clock pulses are generated until CREN is cleared. If CREN is cleared during the transmission of a character, the CK clock stops immediately, causing the incomplete character to be discarded. If both SREN and CREN are set to 1, SREN is cleared after the first character transmission is completed, with CREN taking priority.

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

12.5.1.6 Slave clock

Synchronous data transmission uses a separate clock line that is synchronized with the data line. For devices configured as slaves, the clock signal from the TX/CK line is received. When the device is set up for synchronous slave transmission or reception, the output driver for the TX/CK pin is automatically disabled. Serial data bits change on the leading edge of the clock signal, ensuring validity on the trailing edge of each clock pulse. Only one data bit can be transmitted per clock cycle, so the number of clock cycles must match the number of data bits to be transmitted.

12.5.1.7 Receive overflow error

The receive FIFO buffer can hold 2 characters. If a complete third character is received before reading RCREG to access the FIFO, an overflow error occurs. In this case, the OERR bit in the RCSTA register is set to 1. The previously stored data in the FIFO will not be overwritten. You can read the two characters in the FIFO, but no additional characters can be received until the error is cleared. To clear the overflow condition, you must reset the OERR bit. If the overflow occurs while the SREN bit is set to 1 and the CREN bit is cleared, the error can be cleared by reading the RCREG register. If the overflow occurs while CREN is set to 1, you can either clear the CREN bit in the RCSTA register or clear the SPEN bit to reset the USART and clear the error.

12.5.1.8 Receive 9-bit characters

The USART supports receiving 9-bit characters. When the RX9EN bit in the RCSTA register is set to 1, the USART will move the 9-bit data of each received character into the RSR. When reading 9-bit data from the receive FIFO, you must read the 8 low bits from RCREG first, followed by the RX9D data bit.

12.5.1.9 Synchronous master reception configuration

1. Initialize the SPBRG register to achieve the desired baud rate. (Note: SPBRG>05H)
2. Set the SYNC, SPEN, and CSRC bits to 1 to enable the synchronous master serial port.
3. Ensure that the CREN and SREN bits are cleared to 0.
4. If using interrupts, set the GIE and PEIE bits in the INTCON register, and the RCIE bit in the PIE1 register to 1.
5. If 9-bit character reception is required, set the RX9EN bit to 1.
6. Set the SREN bit to 1 to start reception, or set the CREN bit to 1 to enable continuous reception.
7. After a character is received, the RCIF interrupt flag is set to 1. If the RCIE bit is also set to 1, an interrupt will be generated.
8. Read the RCREG register to obtain the received 8-bit data.
9. Read the RCSTA register to get the 9th data bit (when 9-bit reception is enabled) and check for any errors during reception.
10. If an overflow error occurs, clear the CREN bit in the RCSTA register or clear the SPEN bit to reset the USART and clear the error.

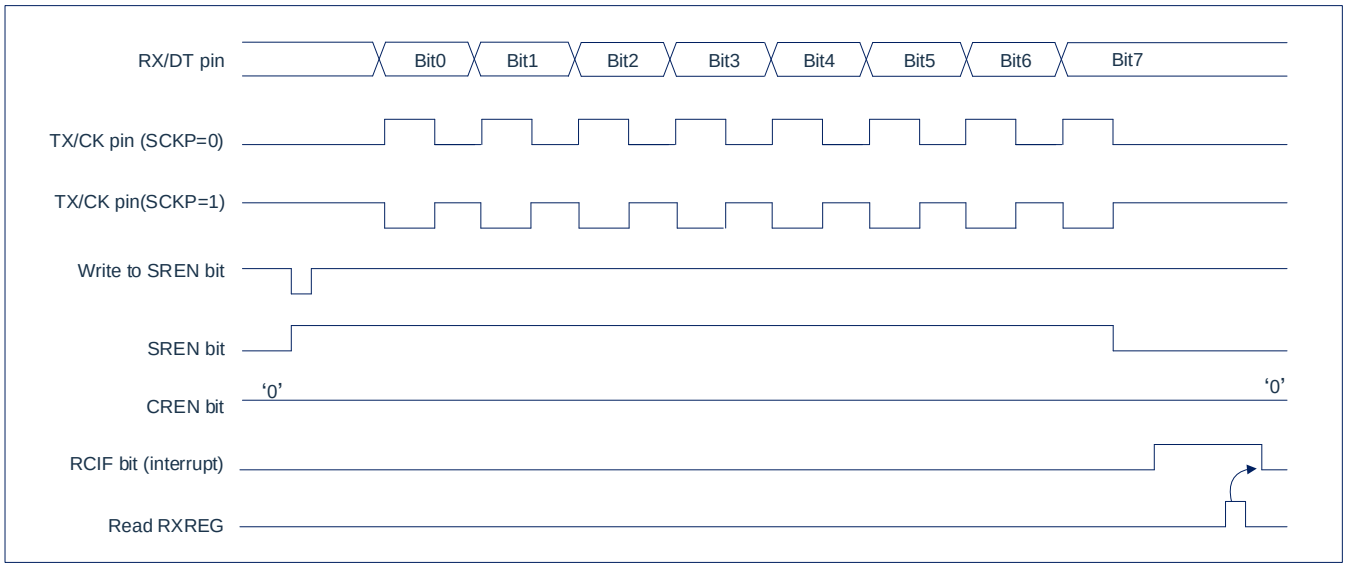


Figure 12-8: Synchronous reception (master mode, SREN)

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

12.5.2 Synchronous slave mode

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

- SYNC=1
- CSRC=0
- SREN=0 (for transmission); SREN=1 (for reception)
- CREN=0 (for transmission); CREN=1 (for reception)
- SPEN=1

Setting the SYNC bit in the TXSTA register to 1 configures the device for synchronous operation. Setting the CSRC bit in the TXSTA register to 0 configures the device as a slave. Clearing the SREN and CREN bits in the RCSTA register ensures the device is in transmission mode; otherwise, it will be configured for reception. Setting the SPEN bit in the RCSTA register enables the USART.

12.5.2.1 USART synchronous slave transmission

The operation of synchronous master and slave modes is similar (see Section Synchronous Master Transmission).

12.5.2.2 Synchronous slave transmission configuration

1. Set SYNC and SPEN bits to 1, and clear the CSRC bit.
2. Clear the CREN and SREN bits.
3. If using interrupts, set the GIE and PEIE bits in the INTCON register, and the TXIE bit in the PIE1 register to 1.
4. If sending 9-bit data, set the TX9EN bit to 1.
5. Set the TXEN bit to 1 to enable transmission.
6. If choosing to send 9-bit data, write the highest bit to the TX9D bit.
7. Write the low 8 bits of data to the TXREG register to start transmission.

12.5.2.3 USART synchronous slave reception

Apart from the following differences, the operation of synchronous master and slave modes is similar:

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

12.5.2.4 Synchronous slave reception configuration

1. Set the SYNC and SPEN bits to 1, and clear the CSRC bit.
2. If using interrupts, set the GIE and PEIE bits in the INTCON register, and the RCIE bit in the PIE1 register to 1.
3. If receiving 9-bit characters, set the RX9EN bit to 1.
4. Set the CREN bit to 1 to enable reception.
5. When reception is completed, the RCIF flag will be set to 1. If RCIE is also set to 1, an interrupt will be generated.
6. Read the RCREG register to obtain the received 8 low data bits from the receive FIFO buffer.
7. If 9-bit mode is enabled, retrieve the highest bit from the RX9D bit in the RCSTA register.
8. If an overflow error occurs, clear the CREN bit or the SPEN bit of the RCSTA register to reset the USART and clear the error.

13. Comparators (CMP1 and CMP2)

The chip has two built-in comparators, CMP1 and CMP2, the function and performance of the two comparators are the same, the following value of x is 1 and 2.

13.1 Block diagram of CMPx

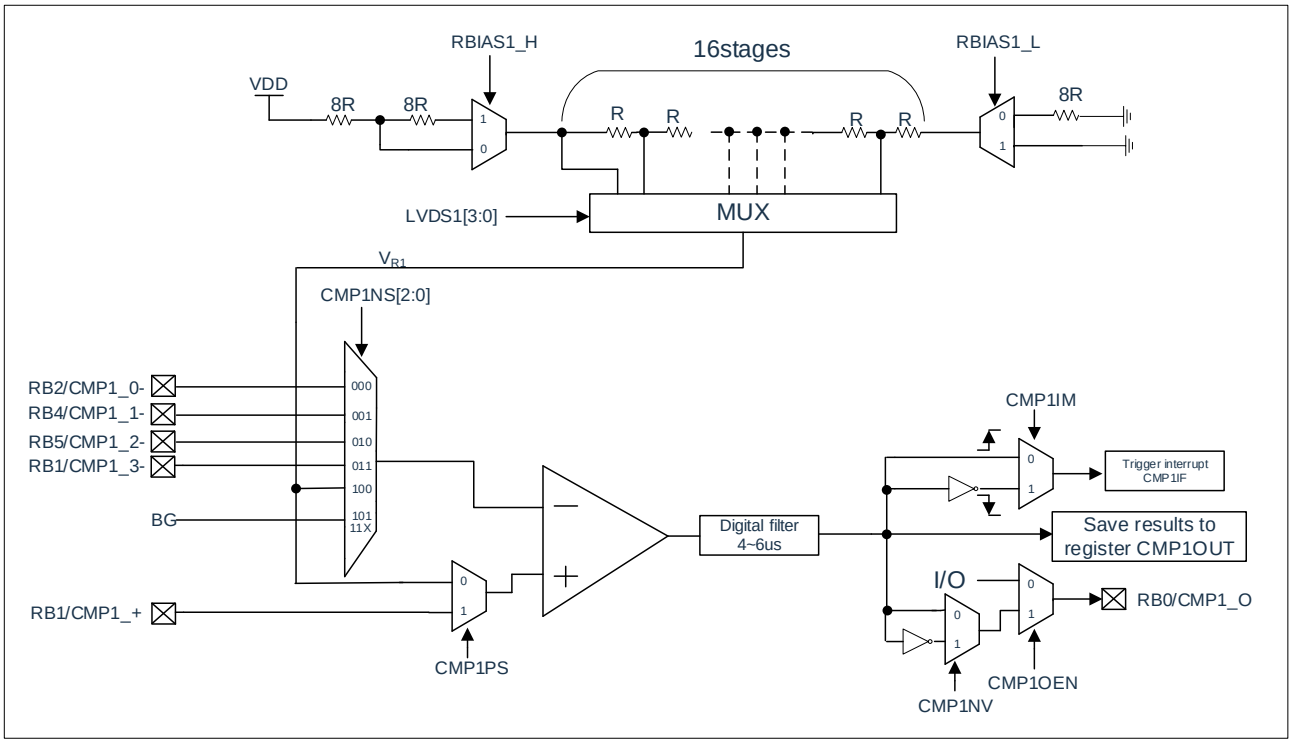


Figure 13-1: Functional block diagram of CMP1

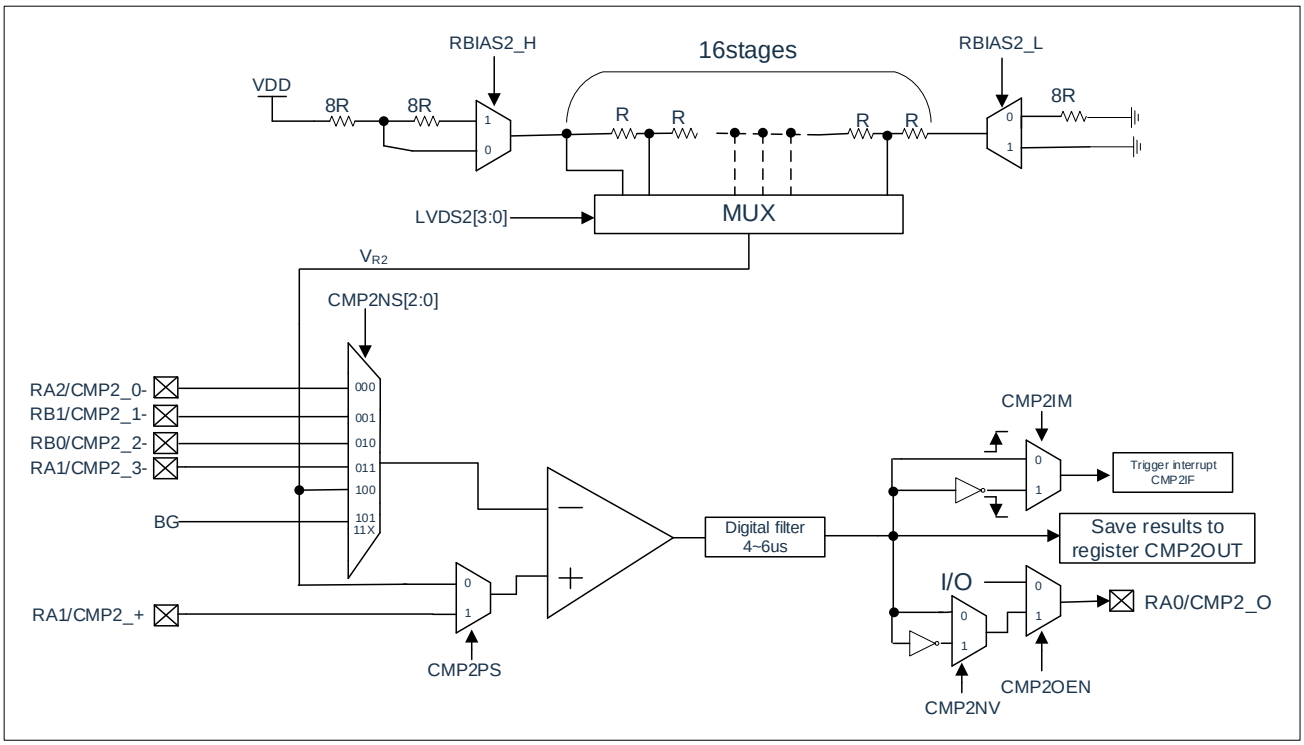


Figure 13-2: Functional block diagram of CMP2

13.2 Features of CMPx

- Comparator offset voltage is≤±13mV.
- Input common-mode voltage range: 0V~VDD-1.3V.
- Built-in resistor divider module with VDD as the reference voltage.
- The comparator result can be triggered by either rising or falling edge and generate an interrupt.
- The comparator result can be output from RA0 port and supports inverted output.

13.3 CMPx related functions

13.3.1 CMPx functions description

The positive input of CMPx can be selected by configuring the CMPxPS bits in the CMPxCON0 register to choose either the CMPx_+ pin or the internal voltage reference signal V_{Rx}. The negative input can be selected by configuring the CMPxNS<2:0> bits in the CMPxCON0 register to choose between the CMPx_m- pin, the internal voltage reference signal V_{Rx}, or the 1.2V BG voltage. When the voltage at the positive input of CMPx is greater than the voltage at the negative input, the output of CMPx will be 1 after digital filtering. Conversely, when the voltage at the positive input is less than the voltage at the negative input, the output of CMPx will be 0 after digital filtering.

Note: CMPx_m- denotes CMPx_0-, CMPx_1-, CMPx_2-, CMPx_3-.

13.3.2 Internal resistor voltage divider output of CMPx

The CMPx integrates an internal resistor voltage divider module with a reference voltage of VDD. Different resistor voltage divider outputs V_{Rx} can be obtained by configuring the values of the control bits RBIASx_H, RBIASx_L, LVDSx<3:0> of the CMPxCON1 register. Depending on the value of the RBIASx_H, RBIASx_L bits, the four calculation formulas for V_{Rx} are as follows:

RBIASx_H	RBIASx_L	V_{Rx} calculation formula
0	0	$V_{Rx}=\frac{1}{4}*VDD+\frac{n+1}{32}*VDD$
0	1	$V_{Rx}=\frac{n+1}{24}*VDD$
1	0	$V_{Rx}=\frac{1}{5}*VDD+\frac{n+1}{40}*VDD$
1	1	$V_{Rx}=\frac{n+1}{32}*VDD$

Note: n is the value of LVDSx<3:0>, i.e., n=0, 1, 2……14, 15.

13.3.3 Monitoring supply voltage

According to the CMPx structure block diagram and the formulas in section 13.3.2, when the negative end of the CMPx selects BG 1.2V, and the positive end selects the internal resistor voltage divider output V_{Rx}, the power supply voltage can be monitored by the CMPx. When the power supply voltage is lower than the set value, the CMPx outputs 0, and when the power supply voltage is higher than the set value, the CMPx outputs 1. By configuring the values of RBIASx_H, RBIASx_L, LVDSx<3:0>, different voltage monitoring points can be set as follows.

RBIASx_H	RBIASx_L	LVDSx[3:0]	Monitor value (V)	RBIASx_H	RBIASx_L	LVDSx[3:0]	Monitor value (V)	RBIASx_H	RBIASx_L	LVDSx[3:0]	Monitor value (V)
0	1	0101	4.80	0	0	0100	2.95	1	0	1101	2.18
1	0	0010	4.36	0	1	1001	2.88	0	0	1001	2.13
0	0	0000	4.27	1	0	1000	2.82	1	0	1110	2.09
0	1	0110	4.11	0	0	0101	2.74	0	1	1101	2.06
1	0	0011	4.00	1	0	1001	2.67	0	0	1010	2.02
0	0	0001	3.84	0	1	1010	2.62	1	0	1111	2.00
1	0	0100	3.69	0	0	0110	2.56	-	-	-	-
0	1	0111	3.60	1	0	1010	2.53	-	-	-	-
0	0	0010	3.49	0	0	0111	2.40	-	-	-	-
1	0	0101	3.43	1	0	1100	2.29	-	-	-	-
0	0	0011	3.20	0	0	1000	2.26	-	-	-	-
1	0	0111	3.00	0	1	1100	2.22	-	-	-	-

13.3.4 Interrupt usage of CMPx

To use the interrupt function of the CMPx, the comparator interrupt can be enabled through the following configuration steps:

- Configure the CMPxPS bit of the CMPxCON0 register to select the positive input.
- Configure the CMPxNS<2:0> bits of the CMPxCON0 register to select the negative input.
- Configure the CMPxIM bit of the CMPxCON1 register to select the rising or falling edge trigger for the interrupt.
- Set the CMPxEN bit of the CMPxCON0 register to enable the comparator.
- Delay for 10us.
- Clear the CMPxIF bit of the PIR1/PIR2 register.
- Set the CMPxIE bit of the PIE1/PIE2 register to 1 to enable the comparator interrupt.
- Set the PEIE and GIE bits of the INTCON register to 1 to enable peripheral and global interrupts.

13.3.5 Interrupt sleep wake-up of CMPx

The CMPx interrupt can wake up the chip from sleep mode. The specific configuration can be found in the following program routine:

Example: CMP1 interrupt sleep wake-up program

```
SLEEP_MODE:
        LDIA    B'00000110'
        LD      TRISB,A         ;configure RB1/CMP1_+ and RB2/CMP1_0- as input ports
        ...                     ;disable other functions
        LDIA    00H
        LD      CMP1CON0,A      ;CMP1NS<2:0>=000, select RB2/CMP1_0- as negative inputs
        SETB    CMP1CON0,6      ;CMP1PS=1, select RB1/CMP1_+ as positive inputs
        SETB    CMP1CON1,6      ;AN1_EN=1, set CMP1_+, CMP1_0- as analog ports to reduce sleep power consumption
        SETB    CMP1CON1,7      ;CMP1IM=1, select falling edge triggered interrupt
        SETB    CMP1CON0,7      ;enable CMP1
        CALL    DELAY10US       ;delay after enable to wait for comparator output to stabilize
        CLRB    PIR1,5          ;clear CMP1IF (required)
        SETB    PIE1,5          ;enable CMP1 interrupt
        SETB    INTCON,6        ;enable peripheral interrupt
        SETB    INTCON,7        ;enable global interrupts, the program will jump to the interrupt vector address 0004H after waking up
        CLRWDT                  ;clear the WDT
        STOP                    ;execute the STOP instruction
        NOP
        NOP
```

13.3.6 Result output pin configuration of CMPx

After digital filtering, the result of the CMPx can be obtained by reading the CMPxOUT bit of the CMPxCON0 register. It can also be output to the CMPx_O pin through the following configuration steps:

- Set CMPx_O pin as an output pin.
- Configure the CMPxNV bit of the CMPxCON0 register to select normal or inverted output.
- Set the CMPxOEN bit of the CMPxCON0 register to 1 to enable the output of CMPxOUT to the CMPx_O pin.

13.4 CMPx related registers

CMPx control register CMPxCON0

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CMPxCON0	CMPxEN	CMPxPS	CMPxNS2	CMPxNS1	CMPxNS0	CMPxNV	CMPxOUT	CMPxOEN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7 CMPxEN: CMPx enable bit
1= Enable CMPx
0= Disable CMPx
Bit6 CMPxPS: CMPx positive input select bit
1= CMPx_+ port voltage
0= VDD voltage after dividing by internal resistor
Bit5~Bit3 CMPxNS<2:0>: CMPx negative input select bit
000= CMPx_0- port voltage
001= CMPx_1- port voltage
010= CMPx_2- port voltage
011= CMPx_3- port voltage
100= VDD voltage after dividing by internal resistor
101= BG
11x= BG
Bit2 CMPxNV: CMPx_O port output inverse control bit
1= Invert CMPxOUT output at CMPx_O port
0= Normal CMPxOUT output at CMPx_O port
Bit1 CMPxOUT: CMPx result bit
Bit0 CMPxOEN: CMPx_O port output enable bit
1= Enable CMPxOUT output at CMPx_O port
0= Dsiable CMPxOUT output at CMPx_O port

CMPx control register CMPxCON1

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CMPxCON1	CMPxIM	ANx_EN	RBIASx_H	RBIASx_L	LVDSx<3:0>			
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7 CMPxIM: CMPx interrupt trigger edge selection
1= Falling edge of CMPx output triggers an interrupt
0= Rising edge of CMPx output triggers an interrupt
Bit6 ANx_EN: Analog port enable bit, enables analog functionality for CMPx_+ and CMPx_m-
1= Analog port
0= Digital port
Bit5 RBIASx_H: Specific usage refers to the CMPx block diagram
Bit4 RBIASx_L: Specific usage refers to the CMPx block diagram
Bit3~Bit0 LVDSx<3:0>: Internal resistor divider ratio selection bit

Note: The ANx_EN bit is only valid for I/O pins selected for comparator functionality. Additionally, the ANSEL0 and ANSEL1 registers can be used to enable the analog functions for CMPx_+ and CMPx_m- pins.

14. Analog to Digital Conversion (ADC)

14.1 ADC overview

An analog-to-digital converter (ADC) converts an analog input signal into a 12-bit binary number that represents that signal. The analog input channels used by the device share a sample circuit. The output of the sample circuit is connected to the input of the analog-to-digital converter. The analog-to-digital converter uses successive approximation to produce a 12-bit binary result, which is stored in the ADC result registers (ADRESL and ADRESH).

The ADC reference voltage can be selected from the internal LDO or VDD The ADC can generate an interrupt after the conversion is completed.

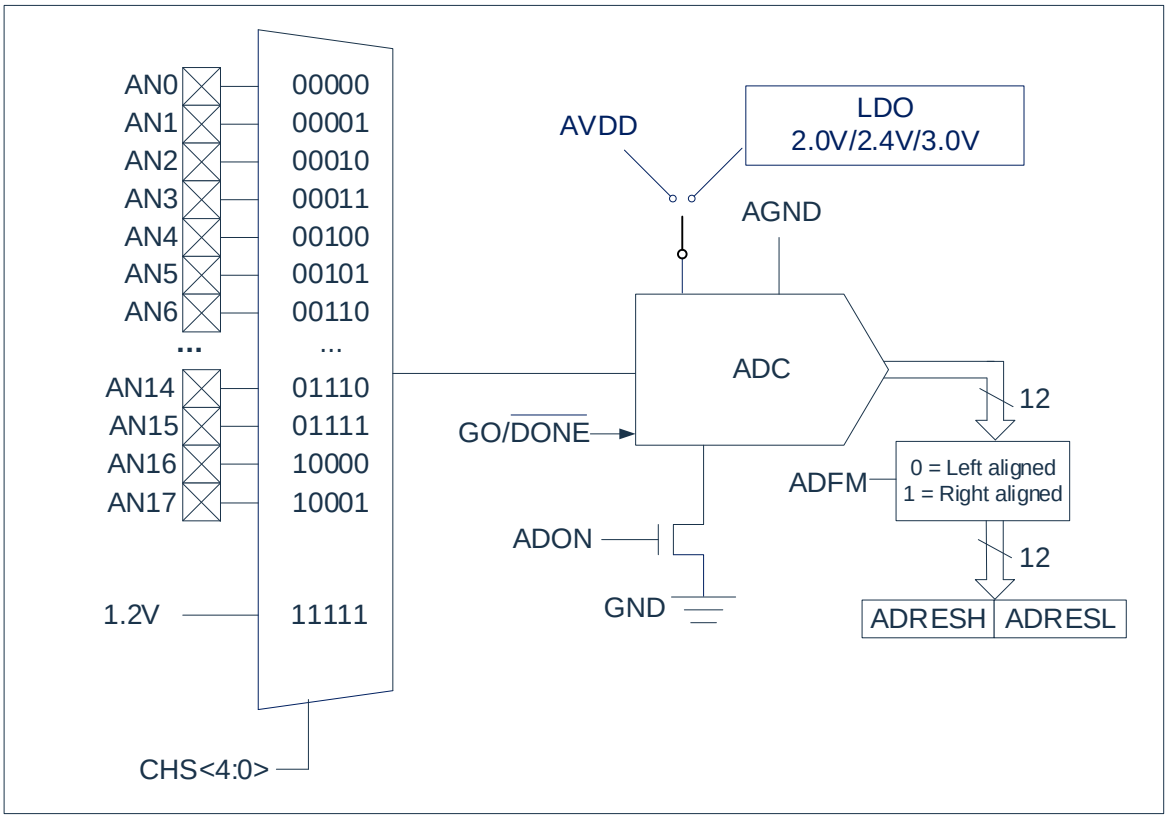


Figure 14-1: Block diagram of ADC

14.2 ADC configuration

The following factors must be considered when configuring and using the ADC:

- Port configuration.
- Reference voltage selection
- Channel selection.
- ADC conversion clock source.
- Interrupt control.
- Storage format of results.

14.2.1 Port configuration

The ADC can convert both analog and digital signals. When converting analog signals, the I/O pins should be configured as analog input pins by setting the corresponding TRIS bit to 1. See the corresponding port section for more information.

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

14.2.2 Channel selection

The CHS bits of the ADCON0 and ADCON1 registers determine which channel is connected to the sample-and-hold circuit.

If the channel is changed, a certain delay will be required before the next conversion starts. For more information, please refer to Section **错误!未找到引用源。** How it works.

14.2.3 ADC internal reference voltage

The chip has a built-in reference voltage. To detect this reference voltage, the CHS<4:0> bits should be set to 11111.

14.2.4 ADC reference voltage

The ADC reference voltage can be selected from the internal LDO output or provided by the chip's VDD and GND. The internal reference voltage options are 2.0V, 2.4V, and 3.0V.

When selecting the internal reference voltage, a slower conversion clock should be chosen. Please refer to the conversion clock section for details.

Note: When the internal LDO is selected as the reference voltage, the ADC accuracy will decrease. The lower the input voltage, the higher the ADC accuracy. It is recommended to set the input voltage to be less than 1V.

14.2.5 Conversion clock

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

- $F_{HSI}/16$
- $F_{HSI}/32$
- $F_{HSI}/64$
- $F_{HSI}/128$

The time to complete a one-bit conversion is defined as the TAD, and a complete 12-bit conversion requires 16 TAD cycles.

The corresponding TAD specification must be complied with in order to obtain the correct conversion result, the following table is an example of the correct ADC clock selection.

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

Reference voltage (V)	Operating voltage (V)	Fastest frequency division	Time for one AD conversion (us)
		F_{HSI}=16M	
VDD	4.0~5.5	$F_{HSI}/16$	16
VDD	2.7~4.0	$F_{HSI}/32$	32
3.0	4.0~5.5	$F_{HSI}/32$	32
3.0	3.3~4.0	$F_{HSI}/64$	64
2.4	4.0~5.5	$F_{HSI}/32$	32
2.4	2.7~4.0	$F_{HSI}/64$	64
2.0	4.0~5.5	$F_{HSI}/64$	64
2.0	2.7~4.0	$F_{HSI}/128$	128

14.2.6 ADC interrupt

The ADC module allows an interrupt to be generated after the completion of an analog-to-digital conversion. The ADC interrupt flag bit is the ADIF bit in the PIR1 register. The ADC interrupt enable bit is the ADIE bit in the PIE1 register. The ADIF bit must be cleared by software. The ADIF bit is set to 1 at the end of each conversion, regardless of whether the ADC interrupt is enabled or not.

14.2.7 Result formatting

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

14.3 How it works

14.3.1 Start conversion

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

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

14.3.2 Complete conversion

When the conversion is completed, the ADC module will:

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

14.3.3 Stop conversion

If the conversion must be terminated before it is complete, the GO/$\overline{\text{DONE}}$ bit can be cleared by software. 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 last conversion. In addition, after the A/D conversion is terminated, a delay of 2 TAD must be passed before the next action can be started. After the delay, the input signal of the selected channel will automatically start to be collected.

Note: Device reset will force all registers to enter the reset state. Therefore, reset will shut down the ADC module and terminate any pending conversions.

14.3.4 How it works in sleep mode

The ADC module cannot operate in sleep mode.

14.3.5 A/D conversion procedure

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

1. Port configuration:
- Configures the pin as an input pin (see TRIS register).
2. ADC mod configuration:
- Select ADC conversion clock;
- Select ADC input channel;
- Choose the format of the result;
- Start the ADC mod.
3. ADC interrupt (optional) configuration:
- Clear ADC interrupt flag bit;
- Enable ADC interrupt;
- Enable peripheral interrupt;
- Enable global interrupt.
4. Wait for the required acquisition time.
5. Set GO/$\overline{\text{DONE}}$ to 1 to start the conversion.
6. Wait for the ADC conversion to finish by one of the following methods:
- Query the GO/$\overline{\text{DONE}}$ bit.
- Wait for ADC interrupt (enable interrupt).
7. Read ADC results.
8. Clear the ADC interrupt flag bit to zero (this operation is required if interrupts are enabled).

Example: AD conversion

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

14.4 ADC related registers

There are four main registers related to ADC conversion: the control registers ADCON0 and ADCON1, and the data registers ADRESH and ADRESL.

AD control register ADCON0(95H)

95H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	ADON
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit6 ADCS<1:0>: A/D conversion clock select bit
00= F_{HSI}/16
01= F_{HSI}/32
10= F_{HSI}/64
11= F_{HSI}/128

Bit5~Bit2 CHS<3:0>: The lower four bits of the analog channel selection and CHS4 form a five-bit channel selection.
CHS<4:0>:
00000= AN0
00001= AN1
00010= AN2
00011= AN3
00100= AN4
00101= AN5
…
01111= AN15
10000= AN16
10001= AN17
11111= 1.2V (fixed reference voltage)
Other= Reserved

Bit1 GO/$\overline{DONE}$: A/D conversion status bit
1= AD conversion is in progress. Set this bit to 1 to start the AD conversion. This bit is automatically cleared by hardware when the AD conversion is completed.
0= AD conversion is completed/or not in progress

Bit0 ADON: ADC enable bit
1= Enable ADC
0= Disable ADC, no operating current consumption.

AD data high bit register ADCON1(96H)

96H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON1	ADFM	CHS4	---	---	---	LDO_EN	LDO_SEL1	LDO_SEL0
R/W	R/W	R/W	---	---	---	R/W	R/W	R/W
Reset value	0	0	---	---	---	0	0	0

Bit7 ADFM: A/D conversion result format selection bit
1= Right aligned
0= Left aligned

Bit6 CHS4: Channel select bit

Bit5~Bit3 Unused

Bit2 LDO_EN: Internal reference voltage enable bit
1= Enable ADC internal LDO reference voltage
Maximum effective ADC precision is 8 bits when internal LDO is selected as reference voltage.
0= VDD is used as the ADC reference voltage

Bit1~Bit0 LDO_SEL<1:0>: Reference voltage selection bit
11= 3.0V
10= 2.4V
0x= 2.0V

AD data high bit register ADRESH(99H), ADFM=0

99H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESH	ADRES11	ADRES10	ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4
R/W	R	R	R	R	R	R	R	R
Reset value	X	X	X	X	X	X	X	X

Bit7~Bit0 ADRES<11:4>: ADC result register bit
High 8 bits of the 12-bit conversion result

AD data low bit register ADRESL(98H), ADFM=0

98H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESL	ADRES3	ADRES2	ADRES1	ADRES0	----	----	----	----
R/W	R	R	R	R	----	----	----	----
Reset value	X	X	X	X	----	----	----	----

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

Bit3~Bit0 Unused

AD data high bit register ADRESH(99H), ADFM=1

99H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESH	----	----	----	----	----	----	ADRES11	ADRES10
R/W	----	----	----	----	----	----	R	R
Reset value	----	----	----	----	----	----	X	X

Bit7~Bit2 Unused.

Bit1~Bit0 ADRES<11:10>: ADC result register bit
High 2 bits of the 12-bit conversion result

AD data low bit register ADRESL(98H), ADFM=1

98H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESL	ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4	ADRES3	ADRES2
R/W	R	R	R	R	R	R	R	R
Reset value	X	X	X	X	X	X	X	X

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

Note: When ADFM = 1, the ADC conversion result only stores the high 10 bits of the 12-bit result. The ADRESH register stores the high 2 bits, and the ADRESL register stores bits 2 to 9.

15. Program EEPROM and Program Memory Control

15.1 Overview

The devices in this series feature 4K words of program memory, with an address range from 0000h to 0FFFh, which is read-only across the entire address range. They also include 128 words of program EEPROM, with an address range from 0h to 07Fh, which is read/write accessible.

These memories are not directly mapped to the register file space but are accessed via Special Function Registers (SFRs) through indirect addressing. There are six SFRs used for accessing these memories:

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

When accessing the program EEPROM, the EEDAT and EEDATH registers form a two-byte word to store the 16-bit read/write data, while the EEADR register stores the address of the accessed program EEPROM cell.

For accessing the device's program memory, the EEDAT and EEDATH registers form a double-byte word for storing the 16-bit data to be read, while the EEADR and EEADRH registers together constitute a double-byte word for the 12-bit Program Memory cell address to be read.

Program memory allows for reading in word units, while program EEPROM supports word read/write operations. Word write operations can automatically erase the target cell before writing new data.

The write time is controlled by an on-chip timer. The write and erase voltages are generated by an on-chip charge pump, which operates within the device's voltage range to facilitate byte or word operations.

When the device is under code protection, the CPU can still continue to read from and write to the program EEPROM and program memory. However, during code protection, the device programmer will no longer be able to access the program EEPROM or program memory.

Note:

1) Program memory refers to ROM space, specifically the space where instruction code is stored, and is read-only.

   Program EEPROM is the space used to store user data and is read/write accessible.

2) The normal write voltage range for program EEPROM is 2.5V to 5.5V, with a write current of 5mA at VDD = 5V

15.2 Related registers

15.2.1 EEADR and EEADRH registers

The EEADR and EEADRH registers can address a maximum of 128 words of program EEPROM or a maximum of 4K words of program memory.

When selecting the address for program memory, the high byte of the address is written to the EEADRH register, while the low byte is written to the EEADR register. In contrast, when selecting the address for program EEPROM, only the low byte of the address is written to the EEADR register.

15.2.2 EECON1 and EECON2 registers

EECON1 is the control register for accessing program EEPROM.

The control bit EEPGD determines whether the access is to program memory or program EEPROM. When this bit is cleared (set to 0), any subsequent operations will target the program EEPROM, similar to a reset condition. When set to 1, subsequent operations will target the program memory, which is read-only.

The control bits RD and WR initiate read and write operations, respectively. These bits can only be set to 1 by software and cannot be cleared. After a read or write operation is completed, hardware clears them automatically. This design prevents accidental premature termination of write operations, as the WR bit cannot be cleared by software.

- When the WREN bit is set to 1, write operations to the program EEPROM are allowed. At power-up, the WREN bit is cleared. If a normal write operation is interrupted by an LVR reset, the WRERR bit will be set to 1. In these cases, the user can check the WRERR bit after a reset and rewrite the corresponding memory cell.

The EECON2 is not a physical register; reading EECON2 returns all zeros.

The EECON2 register is only used during the execution of the program EEPROM write sequence.

EEPROM data register EEDAT(8FH)

8FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEDAT	EEDAT7	EEDAT6	EEDAT5	EEDAT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	X	X	X	X	X	X	X	X

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

EEPROM address register EEADR(91H)

91H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEADR	EEADR7	EEADR6	EEADR5	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

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

EEPROM data register EEDATH(90H)

90H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEDATH	EEDATH7	EEDATH6	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	X	X	X	X	X	X	X	X

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

EEPROM address register EEADRH(92H)

92H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEADRH	---	---	---	---	EEADRH3	EEADRH2	EEADRH1	EEADRH0
R/W	---	---	---	---	R/W	R/W	R/W	R/W
Reset value	---	---	---	---	0	0	0	0

Bit7~Bit4 Unused, read 0.

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

EEPROM control register EECON1(8DH)

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

Bit7 EEPGD: Program/EEPROM selection bit
- 1= Operation on Program Memory
- 0= Operation on EEPROM.

Bit6~Bit4 Unused.

Bit3 WRERR: EEPROM error flag bit
- 1= Write operation error (occurs during any WDT reset or undervoltage reset)
- 0= Write operation completed

Bit2 WREN: EEPROM write enable bit
- 1= Enable write cycle
- 0= Disable writing to memory

Bit1 WR: Write control bit
- 1= Start write cycle (this bit is cleared by hardware once the write operation is completed; software can only set WR to 1, but cannot clear it)
- 0= Write cycle completed

Bit0 RD: Read control bit
- 1= Start memory read operation (cleared by hardware; software can only set RD to 1, but cannot clear it)
- 0= Do not start memory read operation

15.3 Reading from program EEPROM

To read a program EEPROM cell, the user must write the address into the EEADR register, clear the EEPGD control bit in the EECON1 register, and then set the RD control bit to 1. Once the read control bit is set, the program EEPROM controller will use the second instruction cycle to read the data. This will cause the second instruction following the SETB EECON1, RD instruction to be ignored$_{(1)}$. In the next clock cycle, the value at the specified address in the program EEPROM will be latched into the EEDAT and EEDATH registers, allowing the user to read from both registers in subsequent instructions. The EEDAT and EEDATH registers will retain this value until the next read or write operation to that cell occurs.

Note: The two instructions following the program memory read operation must be NOP. This prevents the user from executing a double-cycle instruction immediately after setting RD to 1.

Example: read from program EEPROM

```
EEPDATA_READ:
        LD      A,RADDR         ;load the address to be read into the EEADR register
        LD      EEADR,A
        CLRB    EECON1,EEPGD    ;access data memory
        SETB    EECON1,RD       ;start the read operation
        NOP
        NOP
        LD      A,EEDAT         ;read data into ACC
        LD      RDATAL,A
        LD      A,EEDATH
        LD      RDATAH,A
EEPDATA_READ_BACK:
        RET
```

15.4 Writing to program EEPROM

To write to a program EEPROM memory cell, the user should first write the address of the cell into the EEADR register and the data into the EEDAT and EEDATH registers. The user must then follow a specific sequence to start writing each word.

If the following instruction sequence is not strictly followed (i.e., first writing 55h to EECON2, then writing AAh to EECON2, and finally setting the WR bit to 1), the write operation will not be initiated. Interrupts should be disabled during this code segment.

Additionally, the WREN bit in EECON1 must be set to 1 to enable the write operation. This mechanism prevents accidental writes to the EEPROM due to execution errors (exceptions) (i.e., runaway code). Users should always keep the WREN bit cleared when not updating the EEPROM. The WREN bit cannot be cleared by hardware.

Once a write process is initiated, clearing the WREN bit will not affect that write cycle. The WR bit cannot be set to 1 unless the WREN bit is also set to 1. Upon completion of the write cycle, the WR bit is cleared by hardware.

Note: During the EEPROM write process, the CPU will halt operation. The stop time is T_{EEPROM}.

Example: writing to program EEPROM

```
EEPDATA_WRITE:
        LD      A,WADDR         ;load the address to be written into the EEADR register
        LD      EEADR,A
        LD      A,WDATAL        ;load the low 8-bit data to be written into the EEDAT register
        LD      EEDAT,A
        LD      A,WDATAH        ;load the high 8-bit data to be written into the EEDATH register
        LD      EEDATH,A
        CLR     EECON1
        CLRB    EECON1,EEPGD    ;access data memory
        SETB    EECON1,WREN     ;enable write cycle
        CLRB    F_GIE_ON        ;save the status of interrupt enable
        SZB     INTCON,GIE
        SETB    F_GIE_ON
        CLRB    INTCON,GIE      ;disable interrupts
        SZB     INTCON,GIE      ;ensure interrupts are disabled
        JP      $-2

        LDIA    055H
        LD      EECON2,A
        LDIA    0AAH
        LD      EECON2,A
        SETB    EECON1,WR       ;start write operation
        NOP
        NOP
        CLRWDT
        CLRB    EECON1,WREN     ;write complete, clear write enable bit
```

```
                SZB     F_GIE_ON        ;restore the status of interrupt enable
                SETB    INTCON,GIE

                SNZB    EECON1,WRERR    ;check if EEPROM write operation encountered an error
                JP      EEPDATA_WRITE_BACK
                SZDECR  WERR_C          ;exit on timeout, user-defined
                JP      EEPDATA_WRITE   ;EEPROM write operation is rewritten in case of error
EEPDATA_WRITE_BACK:
                RET
```

15.5 Reading from program memory

To read a program memory, the user must write the high and low address bytes into the EEADRH and EEADR registers, respectively. The EEPGD bit in the EECON1 register must be set to 1, and then the control bit RD must be set to 1. Once the read control bit is set, the program memory controller will use the second instruction cycle to read the data. This causes the second instruction immediately following the "SETB EECON1, RD" instruction to be ignored. In the next clock cycle, the value from the program memory at the specified address will be latched into the EEDAT and EEDATH registers, which the user can read in subsequent instructions. The EEDAT and EEDATH registers will retain this value until the next time the user reads from or writes to the memory.

Note:

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

Example: reading from flash program memory

```
        LD      A,RADDRL        ;load the address to be read into the EEADR register
        LD      EEADR,A
        LD      A,RADDRH        ;load the high byte of the address into the EEADRH register
        LD      EEADRH,A
        SETB    EECON1,EEPGD    ;select the operation for program memory
        SETB    EECON1,RD       ;enable read operation
        NOP
        NOP
        LD      A,EEDAT         ;store the read data
        LD      RDATL,A
        LD      A,EEDATH
        LD      RDATH,A
```

15.6 Writing to program memory

Program memory is read-only, not writeable.

15.7 Cautions on program EEPROM

15.7.1 Writing time for program EEPROM

The program EEPROM writing time is roughly fixed, the time needed to write different data is about 4.6ms. The CPU stops working during the writing period, so the program needs to handle this accordingly.

15.7.2 Write verification

Depending on the specific application, good programming practices generally require verifying the values written to the program EEPROM against the expected values.

15.7.3 Protection against accidental writes

In some cases, users may not want to write data to the program EEPROM. To prevent accidental writes to the EEPROM, the chip includes various protection mechanisms. The WREN bit is reset to zero upon power-up, and a power-up delay timer (with a delay period of 16 ms) prevents write operations to the EEPROM.

The initiation sequence for write operations and the WREN bit work together to prevent accidental writes under the following conditions:

- Undervoltage
- Power glitches
- Software faults

16. Electrical Parameters

16.1 Limit parameters

Supply voltage……GND-0.3V~GND+6.0V
Storage temperature……-50℃~125℃
Working temperature……-40℃~85℃
Port input voltage……GND-0.3V~VDD+0.3V
Maximum positive current for all ports……200mA
Maximum negative current for all ports……-150mA

Note: If the device operating conditions exceed the above “limit parameters”, it may cause permanent damage to the device. The above values are extreme values for the operating conditions, and we do not recommend that the device be operated outside of the range specified in this specification. The stability of the device will be affected if it is operated for a long period of time under extreme conditions.

16.2 DC characteristics

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

Symbol	Parameter	Test condition		Min.	Typ.	Max.	Unit
		VDD	Condition				
VDD	Operating voltage	-	F_{SYS}=16MHz/2T	V_{LVR3}	-	5.5	V
		-	F_{SYS}=16MHz/4T	V_{LVR1}	-	5.5	V
I_{DD}	Operating current	5V	F_{SYS}=16MHz, All analog modules are disabled	-	2.5	-	mA
		5V	F_{SYS}=8MHz, All analog modules are disabled	-	1	-	mA
		3V	F_{SYS}=16MHz, All analog modules are disabled	-	1.5	-	mA
		3V	F_{SYS}=8MHz, All analog modules are disabled	-	0.5	-	mA
		5V	Program EEPROM	-	6	-	mA
I_{STB}	Static current	5V	LVR=DIS WDT=DIS	-	1.5	5	uA
		3V	LVR=DIS WDT=DIS	-	0.8	3	uA
		5V	LVR=DIS WDT=EN	-	4.8	12	uA
		3V	LVR=DIS WDT=EN	-	2.1	5.5	uA
V_{IL}	Low level input voltage	-	----	-	-	0.3VDD	V
V_{IH}	High level input voltage	-	----	0.7VDD	-	-	V
V_{OH}	High level output voltage	-	No load	0.9VDD	-	-	V
V_{OL}	Low level output voltage	-	No load	-	-	0.1VDD	V
V_{EEPROM}	EEPROM module operating voltage	-	----	2.5	-	5.5	V
R_{PH}	Pull-up resistor value	5V	Vo=0.5VDD	-	32	-	KΩ
		3V	Vo=0.5VDD	-	52	-	KΩ
R_{PL}	Pull-down resistor value	5V	Vo=0.5VDD	-	34	-	KΩ
		3V	Vo=0.5VDD	-	56	-	KΩ
I_{OL}	Output port positive current	5V	V_{OL}=0.3VDD	-	37	-	mA
		3V	V_{OL}=0.3VDD	-	17	-	mA
I_{OH}	Output port negative current	5V	V_{OH}=0.7VDD	-	-16	-	mA
		3V	V_{OH}=0.7VDD	-	-7	-	mA
V_{BG}	Internal reference voltage 1.2V	VDD=2.5~5.5V T_A=25°C		-1.5%	1.2	+1.5%	V
		VDD=2.0~5.5V T_A=25°C		-2.5%	1.2	+2.5%	V
		VDD=2.5~5.5V T_A=-40~85°C		-2.0%	1.2	+2.0%	V
		VDD=2.0~5.5V T_A=-40~85°C		-3.0%	1.2	+3.0%	V

16.3 Comparator characteristics

(TA= 25℃, unless otherwise specified)

Symbol	Parameter	Test condition	Min.	Typ.	Max.	Unit
VDD	Operating voltage range	-	2.0	-	5.5	V
Iwork	Operating current	VDD=5V COMP+=2V COMP-=2V	-	34	46	uA
		VDD=3V COMP+=1V COMP-=1V	-	20	26	uA
I_{BG}	BG operating current	VDD=5V	-	35	46	uA
		VDD=3V	-	33	44	uA
V_{IN}	Input common mode voltage range	-	0	-	VDD-1.3	V
Vos	Offset voltage	-	-	-	±13	mV
LSB	Minimum resolution	-	-	10	20	mV
Tr	Response time	-	-	-	6	us
-	Internal resistor divider error	VDD=5V V_R>1V	-1%	-	+1%	-
		VDD=5V V_R<1V	-2%	-	+2%	-

Remark: V_R is the internal resistor voltage divider output value.

16.4 ADC electrical characteristics

(T_A= 25℃, unless otherwise specified)

Symbol	Parameter	Test condition	Min.	Typ.	Max.	Unit
V_{ADC}	ADC operating voltage	V_{ADREF}= VDD, F_{ADCCLK}=1MHz	3.0		5.5	V
		V_{ADREF}= VDD, F_{ADCCLK}=500kHz	2.7		5.5	V
		V_{ADREF}=2.0V, F_{ADCCLK}=250kHz	2.7		5.5	V
		V_{ADREF}=2.4V, F_{ADCCLK}=250kHz	2.7		5.5	V
		V_{ADREF}=3.0V, F_{ADCCLK}=500kHz	3.3		5.5	V
I_{ADC}	ADC conversion current	V_{ADC}=5V, V_{ADREF}= VDD, F_{ADCCLK}=500kHz			500	uA
		V_{ADC} =3V, V_{ADREF}= VDD, F_{ADCCLK}=500kHz			200	uA
V_{AIN}	ADC input voltage	V_{ADC}=5V, V_{ADREF} = VDD, F_{ADCCLK}=500kHz	0		V_{ADC}	V
DNL1	Differential nonlinearity error	V_{ADC}=5V, V_{ADREF} = VDD, F_{ADCCLK}=1MHz		±4		LSB
INL1	Integral nonlinearity error	V_{ADC}=5V, V_{ADREF} = VDD, F_{ADCCLK}=1MHz		±8		LSB
DNL2	Differential nonlinearity error	V_{ADC}=5V, V_{ADREF} = 3.0V, F_{ADCCLK}=500kHz, V_{AIN}<1V		±4		LSB
INL2	Integral nonlinearity error	V_{ADC}=5V, V_{ADREF} = 3.0V, F_{ADCCLK}=500kHz, V_{AIN}<1V		±16		LSB
DNL3	Differential nonlinearity error	V_{ADC}=5V, V_{ADREF} = 2.4V, F_{ADCCLK}=250kHz, V_{AIN}<0.8V		±4		LSB
INL3	Integral nonlinearity error	V_{ADC}=5V, V_{ADREF} = 2.4V, F_{ADCCLK}=250kHz, V_{AIN}<0.8V		±16		LSB
DNL4	Differential nonlinearity error	V_{ADC}=5V, V_{ADREF} = 2.0V, F_{ADCCLK}=250kHz, V_{AIN}<0.7V		±4		LSB
INL4	Integral nonlinearity error	V_{ADC}=5V, V_{ADREF} = 2.0V, F_{ADCCLK}=250kHz, V_{AIN}<0.7V		±16		LSB
T_{ADC}	ADC conversion time			16		T_{ADCCLK}

Remark: Low-temperature specification values are guaranteed by the design, and are not tested in mass production.

16.5 Power-on reset characteristics

(T_A= 25˚C, unless otherwise specified)

Symbol	Parameter	Test condition	Min.	Typ.	Max.	Unit
t_{VDD}	VDD rising rate	-	0.05	-	-	V/ms
V_{LVR1}	LVR set voltage=1.8V	VDD=1.6~5.5V	1.7	1.8	1.9	V
V_{LVR2}	LVR set voltage=2.0V	VDD=1.8~5.5V	1.9	2.0	2.1	V
V_{LVR3}	LVR set voltage=2.5V	VDD=2.3~5.5V	2.4	2.5	2.6	V
V_{LVR4}	LVR set voltage=3.0V	VDD=2.8~5.5V	2.9	3.0	3.1	V

16.6 AC electrical characteristics

(T_A= 25˚C, unless otherwise specified)

Symbol	Parameter	Test condition		Min.	Typ.	Max.	Unit
		VDD	Condition				
F_{WDT}	WDT clock source	VDD=2.5~5.5V	T_A=25˚C	-20%	32	+20%	KHz
		VDD=1.8~5.5V	T_A=25˚C	-30%	32	+30%	KHz
		VDD=2.5~5.5V	T_A=－40~85˚C	-30%	32	+30%	KHz
		VDD=1.8~5.5V	T_A=－40~85˚C	-50%	32	+50%	KHz
T_{EEPROM}	EEPROM programming time	5V	F_{HSI}=16MHz	-	4.6	-	ms
		3V	F_{HSI}=16MHz	-	4.6	-	ms
F_{INTRC}	Internal frequency 16MHz	VDD=4.0~5.5V	TA=25˚C	-2.0%	16	+2.0%	MHz
		VDD=2.5~5.5V	TA=25˚C	-2.5%	16	+2.5%	MHz
		VDD=1.8~5.5V	TA=25˚C	-3.0%	16	+3.0%	MHz
		VDD=4.0~5.5V	TA=－40~85˚C	-3.0%	16	+3.0%	MHz
		VDD=2.5~5.5V	TA=－40~85˚C	-4.0%	16	+4.0%	MHz
		VDD=1.8~5.5V	TA=－40~85˚C	-5.0%	16	+5.0%	MHz

16.7 LSE characteristics

(TA= 25℃, unless otherwise specified)

Symbol	Parameter	Test condition	Min.	Typ.	Max.	Unit
VDD	Operating voltage range	-	1.8	-	5.5	V
F_{LSE}	LSE oscillator frequency	-	-	32.768	-	KHz
C1	OSCIN pin matching capacitance	-	-	22	-	pF
C2	OSCOUT pin matching capacitance	-	-	22	-	pF
I_{LSE}	LSE operating current	VDD=5V C_1=22pF C_2=22pF	-	20	-	uA
		VDD=3V C_1=22pF C_2=22pF	-	8	-	uA
T_{LSE}	LSE stabilization time	VDD=5V C_1=22pF C_2=22pF	-	260	700	ms
		VDD=3V C_1=22pF C_2=22pF		300	1000	ms

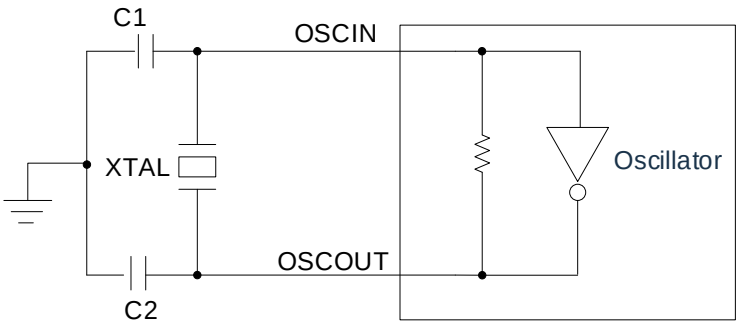


Figure 16-1: Typical application circuit

16.8 EMC characteristics

16.8.1 EFT electrical characteristics

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

Note: Electrical Fast Transient (EFT) immunity performance is closely related to system design (including power supply structure, circuit design, layout, chip configuration, program structure, etc.). The EFT parameters in the above table are measured on CMS' internal test platform and are not intended to be used in all applications and are provided for reference only. All aspects of the system design may affect the EFT performance. In applications with high EFT performance requirements, care should be taken to avoid interference sources affecting the system operation, and it is recommended to analyze the interference paths and optimize the design to achieve the best immunity performance.

16.8.2 ESD electrical characteristics

Symbol	Item	Test condition	Level
V_{ESD}	Electrostatic discharge (Human Body Discharge Mode (HBM))	TA = + 25℃, ANSI/ESDA/JEDEC JS-001-2023	3A
	Electrostatic discharge (Charged-Device Model: CDM)	T_A = + 25℃, ANSI/ESDA/JEDEC JS-002-2022	C3

16.8.3 Latch-up electrical characteristics

Symbol	Item	Test condition	Level
LU	Static latch-up class	JSED 78F	Class I A (T_A = +25℃)

17. Instruction

17.1 Instruction set

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

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

17.2 Instruction description

ADDA **[R]**

operation: Add ACC to R, save the result to ACC

period: 1

affected flag bit: C, DC, Z, OV

example:

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

ADDR **[R]**

operation: Add ACC to R, save the result to R

period: 1

affected flag bit: C, DC, Z, OV

example:

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

ADDCA **[R]**

operation: Add ACC to C, save the result to ACC

period: 1

affected flag bit: C, DC, Z, OV

example:

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

ADDCR **[R]**

operation: Add ACC to C, save the result to R

period: 1

affected flag bit: C, DC, Z, OV

example:

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

ADDIA **i**

operation: Add i to ACC, save the result to ACC

period: 1

affected flag bit: C, DC, Z, OV

example:

```
LDIA    09H     ;load 09H to ACC
ADDIA   077H    ;execute: ACC = ACC(09H) + i(77H)=80H
```

ANDA **[R]**

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

period: 1

affected flag bit: Z

example:

```
LDIA    0FH     ;load 0FH to ACC
LD      R01,A   ;load ACC (0FH) to R01
LDIA    77H     ;load 77H to ACC
ANDA    R01     ;execute: ACC=(0FH and 77H)=07H
```

ANDR **[R]**

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

period: 1

affected flag bit: Z

example:

```
LDIA    0FH     ;load 0FH to ACC
LD      R01,A   ;load ACC (0FH) to R01
LDIA    77H     ;load 77H to ACC
ANDR    R01     ;execute: R01= (0FH and 77H)=07H
```

ANDIA **i**

operation: Perform ‘AND’ on i and ACC, save the result to ACC

period: 1

affected flag bit: Z

example:

```
LDIA    0FH     ;load 0FH to ACC
ANDIA   77H     ;execute: ACC =(0FH and 77H)=07H
```

CALL **add**

operation: Call subroutine

period: 2

affected flag bit: none

example:

```
CALL    LOOP    ;call the subroutine address whose name is defined as “LOOP”
```

CLRA

operation: ACC clear

period: 1

affected flag bit: Z

example:

```
CLRA            ;execute: ACC=0
```

CLR **[R]**

operation: Register R clear

period: 1

affected flag bit: Z

example:

```
CLR     R01     ;execute: R01=0
```

CLRB **[R],b**

operation: Clear b bit on register R

period: 1

affected flag bit: None

example:

```
CLRB    R01,3   ;execute: 3rd bit of R01 is 0
```

CLRWDT

operation: Clear watchdog timer

period: 1

affected flag bit: TO, PD

example:

```
CLRWDT          ;watchdog timer clear
```

COMA **[R]**

operation: Reverse register R, save the result to ACC

period: 1

affected flag bit: Z

example:

```
LDIA    0AH     ;load 0AH to ACC
LD      R01,A   ;load ACC (0AH) to R01
COMA    R01     ;execute: ACC=0F5H
```

COMR **[R]**

operation: Reverse register R, save the result to R

period: 1

affected flag bit: Z

example:

```
LDIA    0AH     ;load 0AH to ACC
LD      R01,A   ;load ACC (0AH) to R01
COMR    R01     ;execute: R01=0F5H
```

DECA **[R]**

operation: Decrement value in register, save the result to ACC

period: 1

affected flag bit: Z

example:

```
LDIA    0AH     ;load 0AH to ACC
LD      R01,A   ;load ACC (0AH) to R01
DECA    R01     ;execute: ACC=(0AH-1)=09H
```

DECR **[R]**

operation: Decrement value in register, save the result to R

period: 1

affected flag bit: Z

example:

```
LDIA    0AH     ;load 0AH to ACC
LD      R01,A   ;load ACC (0AH) to R01
DECR    R01     ;execute: R01=(0AH-1)=09H
```

HSUBA **[R]**

operation: ACC subtract R, save the result to ACC

period: 1

affected flag bit: C,DC,Z,OV

example:

```
LDIA    077H    ;load 077H to ACC
LD      R01,A   ;load ACC (077H) to R01
LDIA    080H    ;load 080H to ACC
HSUBA   R01     ;execute: ACC=(80H-77H)=09H
```

HSUBR **[R]**

operation: ACC subtract R, save the result to R

period: 1

affected flag bit: C,DC,Z,OV

example:

```
LDIA    077H    ;load 077H to ACC
LD      R01,A   ;load ACC (077H) to R01
LDIA    080H    ;load 080H to ACC
HSUBR   R01     ;execute: R01=(80H-77H)=09H
```

HSUBCA **[R]**

operation: ACC subtract $\overline{C}$, 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)= 08H(C=0)
                          ACC=(80H-77H-C)= 09H(C=1)
```

HSUBCR **[R]**

operation: ACC subtract $\overline{C}$, save the result to R

period: 1

affected flag bit: C,DC,Z,OV

example:

```
LDIA    077H    ;load 077H to ACC
LD      R01,A   ;load ACC (077H) to R01
LDIA    080H    ;load 080H to ACC
HSUBCR  R01     ;execute: R01=(80H-77H-C)=08H(C=0)
                          R01=(80H-77H-C)=09H(C=1)
```

INCA **[R]**

operation: Register R increment 1, save the result to ACC

period: 1

affected flag bit: Z

example:

```
LDIA    0AH     ;load 0AH to ACC
LD      R01,A   ;load ACC (0AH) to R01
INCA    R01     ;execute: ACC=(0AH+1)=0BH
```

INCR **[R]**

operation: Register R increment 1, save the result to R

period: 1

affected flag bit: Z

example:

```
LDIA    0AH     ;load 0AH to ACC
LD      R01,A   ;load ACC (0AH) to R01
INCR    R01     ;execute: R01=(0AH+1)=0BH
```

JP **add**

operation: Jump to add address

period: 2

affected flag bit: None

example:

```
JP      LOOP    ;jump to the subroutine address whose name is defined as “LOOP”
```

LD **A,[R]**

operation: Load the value of R to ACC

period: 1

affected flag bit: Z

example:

```
LD      A,R01   ;load R01 to ACC
LD      R02,A   ;load ACC to R02, achieve data transfer from R01→R02
```

LD **[R],A**

operation: Load the value of ACC to R

period: 1

affected flag bit: None

example:

```
LDIA    09H     ;load 09H to ACC
LD      R01,A   ;execute: R01=09H
```

LDIA **i**

operation: Load i to ACC

period: 1

affected flag bit: None

example:

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

NOP

operation: Empty instructions

period: 1

affected flag bit: None

example:

```
NOP
NOP
```

ORIA **i**

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

period: 1

affected flag bit: Z

example:

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

ORA **[R]**

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

period: 1

affected flag bit: Z

example:

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

ORR **[R]**

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

period: 1

affected flag bit: Z

example:

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

RET

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

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

RET i

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

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

RETI

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

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

RLCA [R]

operation: Register R rotates to the left with C and save the result into ACC
period: 1
affected flag bit: C
example:

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

RLCR **[R]**

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

period: 1

affected flag bit: C

example:

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

RLA **[R]**

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

period: 1

affected flag bit: None

example:

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

RLR **[R]**

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

period: 1

affected flag bit: None

example:

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

RRCA **[R]**

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

period: 1

affected flag bit: C

example:

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

RRCR **[R]**

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

period: 1

affected flag bit: C

example:

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

RRA **[R]**

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

period: 1

affected flag bit: None

example:

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

RRR **[R]**

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

period: 1

affected flag bit: None

example:

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

SET **[R]**

operation: Set all bits in register R as 1

period: 1

affected flag bit: None

example:

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

SETB **[R],b**

operation: Set b bit in register R to 1

period: 1

affected flag bit: None

example:

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

STOP

operation: Enter sleep
period: 1
affected flag bit: TO, PD
example:

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

SUBIA **i**

operation: I minus ACC, save the result to ACC
period: 1
affected flag bit: C,DC,Z,OV
example:

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

SUBA **[R]**

operation: Register R minus ACC, save the result to ACC
period: 1
affected flag bit: C,DC,Z,OV
example:

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

SUBR **[R]**

operation: Register R minus ACC, save the result to R
period: 1
affected flag bit: C,DC,Z,OV
example:

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

SUBCA **[R]**

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

period: 1

affected flag bit: C,DC,Z,OV

example:

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

SUBCR **[R]**

operation: Register R minus ACC minus C, the result is put into R

period: 1

affected flag bit: C,DC,Z,OV

example:

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

SWAPA **[R]**

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

period: 1

affected flag bit: None

example:

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

SWAPR **[R]**

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

period: 1

affected flag bit: None

example:

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

SZB **[R],b**

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

period: 1 or 2

affected flag bit: None

example:

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

SNZB **[R],b**

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

period: 1 or 2

affected flag bit: None

example:

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

SZA **[R]**

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

period: 1 or 2

affected flag bit: None

example:

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

SZR **[R]**

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

period: 1 or 2

affected flag bit: None

example:

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

SZINCA **[R]**

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

period: 1 or 2

affected flag bit: None

example:

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

SZINCR **[R]**

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

period: 1 or 2

affected flag bit: None

example:

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

SZDECA **[R]**

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

period: 1 or 2

affected flag bit: None

example:

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

SZDECR **[R]**

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

period: 1 or 2

affected flag bit: None

example:

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

TESTZ **[R]**

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

period: 1

affected flag bit: Z

example:

```
TESTZ   R0          ;load the value of register R0 to R0, which is used to influence the Z flag bit
SZB     STATUS,Z    ;check Z flag bit, if it is 0 then jump
JP      Add1        ;if R0 is 0, jump to address Add1
JP      Add2        ;if R0 is not 0, jump to address Add1
```

XORIA **i**

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

period: 1

affected flag bit: Z

example:

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

XORA **[R]**

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

period: 1

affected flag bit: Z

example:

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

XORR **[R]**

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

period: 1

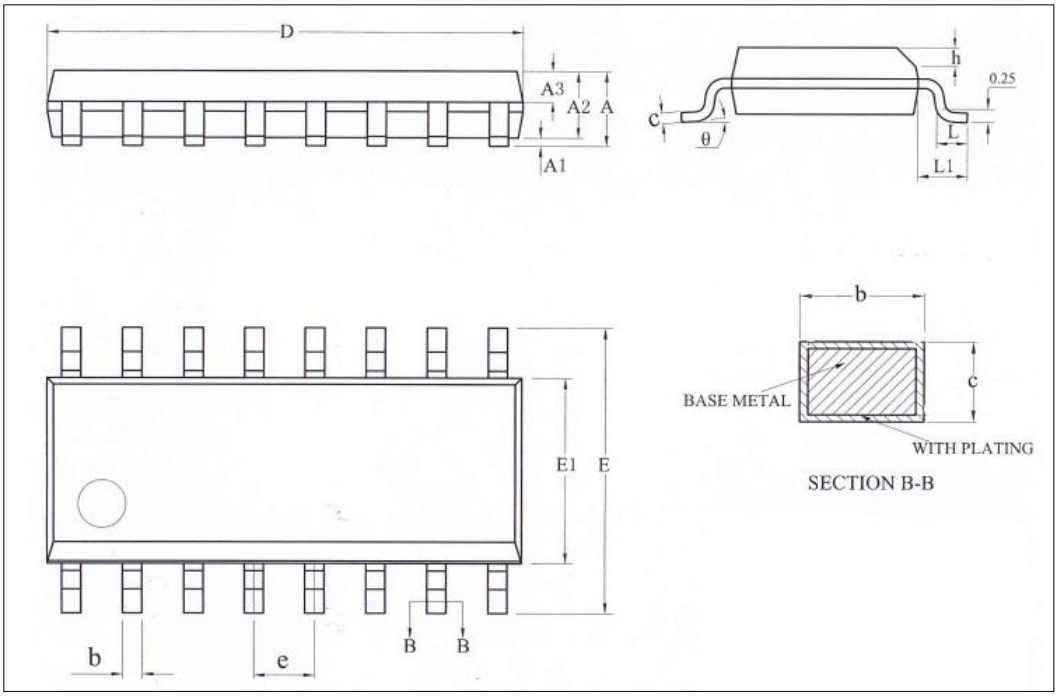
affected flag bit: Z

example:

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

18. Package Dimensions

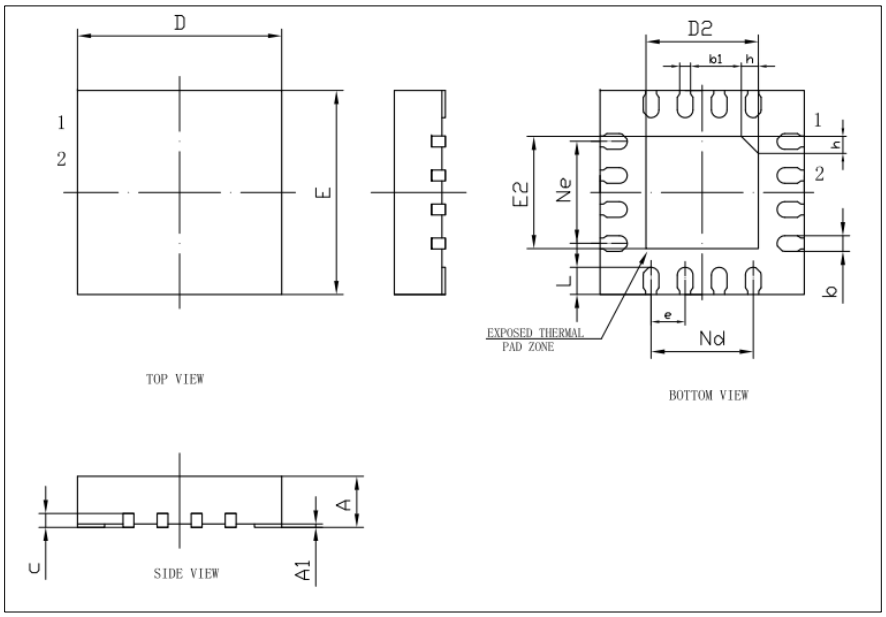
18.1 SOP16



Symbol	Millimeter		
	Min	Nom	Max
A	-	-	1.85
A1	0.05	-	0.25
A2	1.30	-	1.60
A3	0.60	-	0.71
b	0.356	-	0.51
c	0.20	-	0.26
D	9.70	-	10.10
E	5.80	6.00	6.20
E1	3.70	-	4.10
e	1.27BSC		
h	0.25	-	0.50
L	0.40	-	0.80
L1	1.05REF		
θ	0	-	8°

Caution: Package dimensions do not include mold flash or gate burrs.

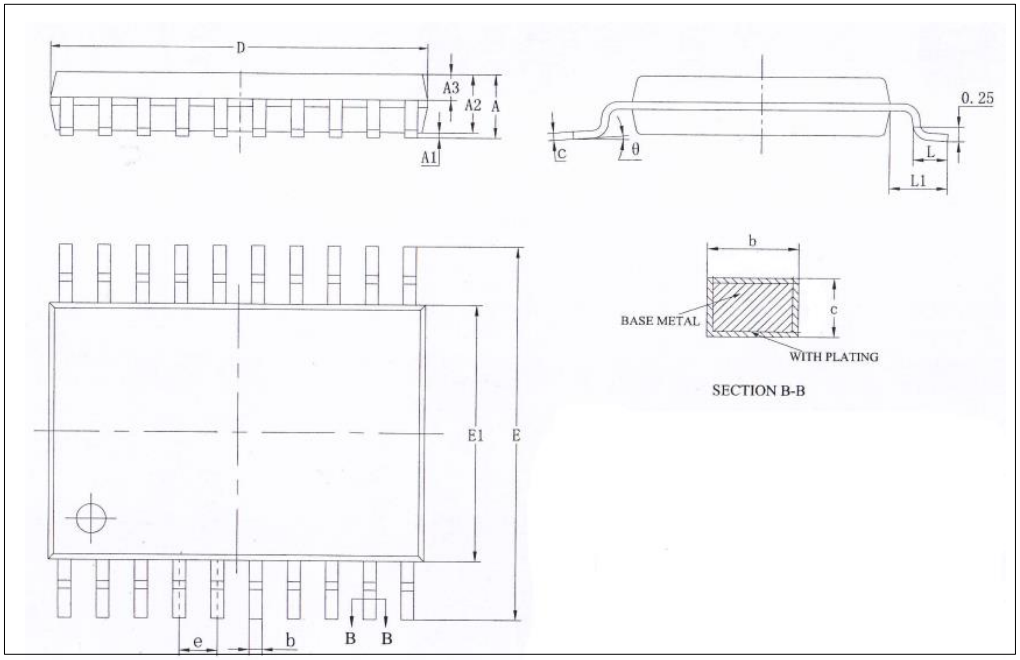
18.2 QFN16 (3x3x0.75-0.50mm)



Symbol	Millimeter		
	Min	Nom	Max
A	0.65	0.75	0.85
A1	0	0.02	0.05
b	0.15	-	0.30
b1		0.16REF	
c	0.18	-	0.30
D	2.90	3.00	3.10
D2	1.55	-	1.80
e		0.50BSC	
Ne		1.50BSC	
Nd		1.50BSC	
E	2.90	3.00	3.10
E2	1.55	-	1.80
L	0.30	0.40	0.50
h	0.20	0.25	0.30

Caution: Package dimensions do not include mold flash or gate burrs.

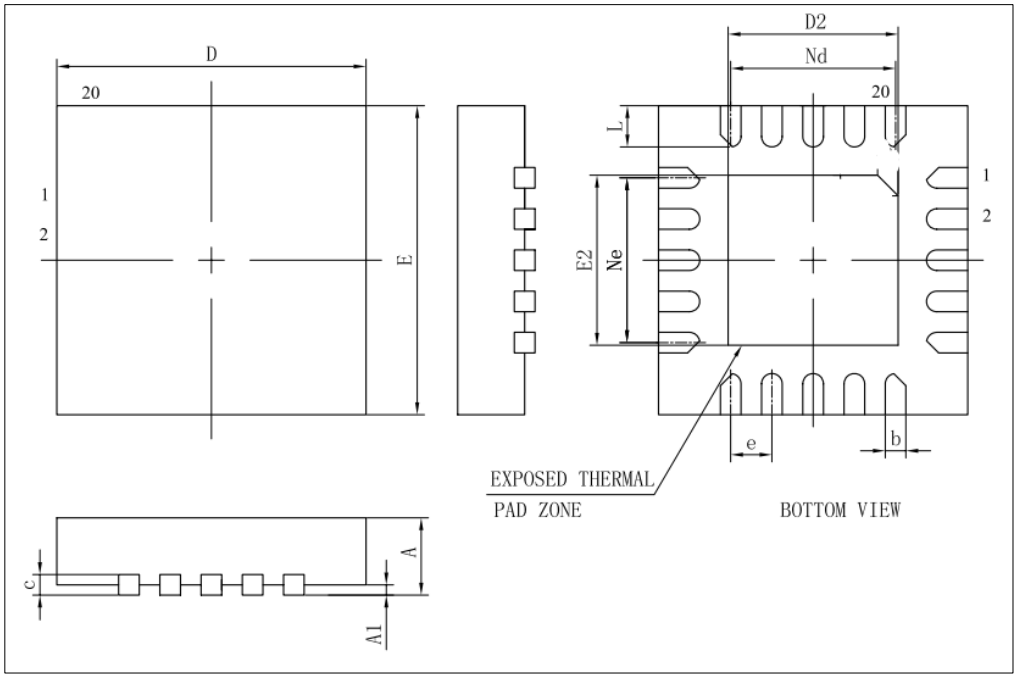
18.3 TSSOP20



Symbol	Millimeter		
	Min	Nom	Max
A	-	-	1.25
A1	0.05	-	0.15
A2	0.80	1.00	1.10
A3	0.34	0.44	0.54
b	0.20	-	0.28
c	0.10	-	0.19
D	6.40	6.50	6.60
E	6.20	6.40	6.60
E1	4.30	4.40	4.50
e	0.65BSC		
L	0.45	0.60	0.75
L1	1.00REF		
θ	0	-	8°

Caution: Package dimensions do not include mold flash or gate burrs.

18.4 QFN20 (3x3x0.75-0.40mm)



Symbol	Millimeter		
	Min	Nom	Max
A	0.65	0.75	0.85
A1	-	0.02	0.05
b	0.15	0.20	0.25
c	0.18	0.20	0.25
D	2.90	3.00	3.10
D2	1.55	-	2.00
e	0.40BSC		
Ne	1.60BSC		
Nd	1.60BSC		
E	2.90	3.00	3.10
E2	1.55	-	2.00
L	0.20	-	0.50

Caution: Package dimensions do not include mold flash or gate burrs.

19. Revision History

Version #	Date	Description of changes
V0.0.1	Aug. 2023	Initial release
V0.5.0	Nov. 2023	1) Modified electrical parameters 2) Corrected the list of special function registers. 3) Deleted SC8F073AD720SS model number and related information.
V1.0.0	Jan. 2024	1) Revised ADC accuracy in 16.4 ADC electrical characteristics. 2) Deleted Section ADC Internal LDO Reference Voltage Characteristics. 3) Revised internal oscillation accuracy in 16.6 AC electrical characteristics.
V1.0.1	Mar. 2024	Deleted SC8F073AD720SA model number and related information
V1.0.2	May 2024	1) Updated the QFN16 pin map format 2) Changed T0CKI from RB2 to RB3 in the pin map
	Oct. 2024	1) Modified SOP16/QFN16/QFN20 package dimensions 2) Added SC8F073AD720SA model number and related information
V1.0.3	Mar. 2025	Modified Baud rate in asynchronous mode
V1.0.4	June 2025	1) Corrected 16MHz/2T operating voltage range 2) Add a chapter on EMC characteristics
V1.0.5	Jul 2025	1) Notes on Adding USART Asynchronous Transmission (Back-to-Back) Mode 2) Correct the TIMER1 increment edge diagram 3) Modified the wording of the operational voltage range