CMS79F11x User Manual

8-bit CMOS microcontroller with enhanced flash memory

Rev. 1.30

Manual

1. PRODUCT DESCRIPTION.. 6
1.1 FEATURES 6
1.2 SYSTEM STRUCTURE 7
1.3 PIN CONFIGURATION 8
1.3.1 CMS79F111 8
1.3.2 CMS79F112 8
1.3.3 CMS79F113 8
1.3.4 CMS79F116 9
1.4 SYSTEM CONFIGURATION REGISTER 11
1.5 ONLINE SERIAL PROGRAMMING 12
2. CENTRAL PROCESSING UNIT (CPU).. 13
2.1 MEMORY 13
2.1.1 Program Memory 13
2.1.2 Data Memory..18
2.2 ADDRESSING MODE 23
2.2.1 Direct Addressing 23
2.2.2 Immediate Addressing..23
2.2.3 Indirect Addressing..23
2.3 STACK.. 24
2.4 ACCUMULATOR (ACC).. 25
2.4.1 General 25
2.4.2 ACC Applications 25
2.5 PROGRAM STATUS REGISTER (STATUS) 26
2.6 PRE-SCALER (OPTION_REG) 28
2.7 PROGRAM COUNTER (PC) 30
2.8 WATCHDOG TIMER (WDT) 31
2.8.1 WDT Period..31
2.8.2 Watchdog Timer Control Register WDTCON 31
3. SYSTEM CLOCK 32
3.1 GENERAL 32
3.2 SYSTEM OSCILLATOR 33
3.2.1 Internal RC Oscillation 33
3.2.2 External XT Oscillation..33
3.3 RESET TIME.. 33
3.4 OSCILLATOR CONTROL REGISTER.. 34
3.5 CLOCK BLOCK DIAGRAM 35
4. RESET.. 36
4.1 POWER ON RESET.. 36
4.2 POWER OFF RESET 37
4.2.1 General 37
4.2.2 Improvements for Power off Reset..38
4.3 WATCHDOG RESET.. 38
5. SLEEP MODE 39
5.1 ENTER SLEEP MODE 39
5.2 AWAKEN FROM SLEEP MODE.. 39
5.3 INTERRUPT AWAKENING 40
5.4 SLEEP MODE APPLICATION.. 40
5.5 SLEEP MODE AWAKEN TIME 41

6. I/O PORT 42
6.1 I/O PORT STRUCTURE 43
6.2 PORTA 45
6.2.1 PORTA Data and Direction Control 45
6.2.2 PORTA Pull Up Resistance 46
6.2.3 PORTA Pull Down Resistance 46
6.2.4 PORTA Analog Control Selection 47
6.2.5 PORTA Level Chage Interrupt 47
6.3 PORTB 49
6.3.1 PORTB Data and Direction 49
6.3.2 PORTB Pull up Resistance 50
6.3.3 PORTB Pull Down Resistance 50
6.3.4 PORTB Analog Selection Control 51
6.4 PORTC 52
6.4.1 PORTC Data and Direction 52
6.4.2 PORTC pull up resistance 53
6.4.3 PORTC Pull Down Resistance 53
6.4.4 PORTC Analog Selection Control 54
6.5 I/O USAGE 55
6.5.1 Write I/O port 55
6.5.2 Read I/O Port 55
6.6 PRECAUTIONS FOR I/O PORT USAGE 56
7. INTERRUPT 57
7.1 INTERRUPT GENERAL 57
7.2 INTERRUPTCONTROL REGISTER 58
7.2.1 Interrupt Control Register 58
7.2.2 Peripherals Interrupt Enable Register 59
7.2.3 Peripherals Interrupt Request Register 60
7.3 PROTECTION METHODS FOR INTERRUPT 61
7.4 INTERRUPT PRIORITY AND MULTI-INTERRUPT NESTING 61
8. TIMER0 62
8.1 TIMER0 GENERAL 62
8.2 WORKING PRINCIPLE FOR TIMER0 63
8.2.1 8-bit Timer Mode 63
8.2.2 8-bit Counter Mode 63
8.2.3 Software Programmable Pre-scaler 63
8.2.4 Switch Prescaler Between TIMER0 and WDT Module 63
8.2.5 TIMER0 Interrupt 64
8.3 TIMER0 RELATED REGISTER 65
9. TIMER1 66
9.1 TIMER1 GENERAL 66
9.2 WORKING PRINCIPLE FOR TIMER1 67
9.3 CLOCK SOURCE SELECTION 67
9.3.1 Internal Clock Source 67
9.3.2 External Clock Source 68
9.4 TIMER1 PRE-SCALER 68
9.5 TIMER1 OSCILLATOR 68
9.6 TIMER1 WORKING PRINCIPLE UNDER AASYNCHRONOUS COUNTER MODE 69
9.6.1 Read and Write Operations to TIMER1 In Asynchronous Counter Mode 69
9.7 TIMER1 GATE CONTROL 70

9.8 TIMER1 INTERRUPT 70
9.9 TIMER1 WORKING PRINCIPLE DURING SLEEP 70
9.10 TIMER1 CONTROL REGISTER 71

10. TIMER2 72

10.1 TIMER2 GENERAL 72
10.2 WORKING PRINCIPLE OF TIMER2 73
10.3 TIMER2 RELATED REGISTER 74

11. ANALOG TO DIGITAL CONVERSION (ADC) 75

11.1 ADC GENERAL 75
11.2 ADC CONFIGURATION 76
11.2.1 Port configuration 76
11.2.2 Channel selection 76
11.2.3 ADC internal reference voltage 76
11.2.4 ADC reference voltage 76
11.2.5 Converter clock 77
11.2.6 ADC Interrupt 77
11.2.7 Output Formatting 77
11.3 ADC WORKING PRINCIPLE 78
11.3.1 Start conversion 78
11.3.2 Complete conversion 78
11.3.3 Stop conversion 78
11.3.4 Working principle of ADC in sleep mode 78
11.3.5 A/D conversion procedure 79
11.4 ADC RELATED REGISTER 80

12. LCD DRIVER MOD 83

12.1 LCD FUNCTION ENABLE 83
12.2 LCD RELATED REGISTER 83
12.3 LCD OPERATING INSTRUCTIONS 87
12.3.1 1/3 Bias register operation 87
12.3.2 1/3 Bias timing diagram 87
12.3.3 1/2 Bias register operation 88
12.3.4 1/2 Bias timing diagram 88

13. PWM MODE 89

13.1 PIN CONFIGURATION 89
13.2 DESCRIPTION OF RELATED REGISTERS 89
13.3 PWM REGISTER WRITE OPERATION SEQUENCE 94
13.4 PWM PERIOD 94
13.5 PWM DUTY CYCLE 94
13.6 SYSTEM CLOCK FREQUENCY CHANGES 94
13.7 PROGRAMMABLE DEAD TIME DELAY MODE 95
13.8 PWM SETTINGS 95

14. UNIVERSAL SYNCHRONOUS/ASYNCHRONOUS TRANSMITTER (USART) 96

14.1 USART ASYNCHRONOUS MODE 98
14.1.1 USART Asynchronous Generator 98
14.1.2 USART Asynchronous Receiver 101
14.2 CLOCK PRECISION FOR ASYNCHRONOUS OPERATIONS 104
14.3 USART RELATED REGISTER 104
14.4 USART BAUD RATE GENERATOR (BRG) 106
14.5 USART SYNCHRONOUS MODE 107

14.5.1 Synchronous Master Control Mode 107
14.5.2 Synchronous Slave Mode 111

15. PROGRAM EEPROM AND PROGRAM MEMORY CONTROL 113

15.1 GENERAL 113
15.2 RELATED REGISTER 114
15.2.1 EEADR and EEADRH Register....114
15.2.2 EECON1 and EECON2 Register 114
15.3 READ PROGRAM EEPROM 116
15.4 WRITE PROGRAM EEPROM....117
15.5 READ PROGRAM MEMORY 119
15.6 WRITE PROGRAM MEMORY 119
15.7 PRECAUTIONS ON PROGRAM EEPROM.... 120
15.7.1 Programming Time for Program EEPROM 120
15.7.2 The times of programming EEPROM....120
15.7.3 Write Verification 120
15.7.4 Protection to Avoid Writing Wrongly 121

16. LOW VOLTAGE DETECTION (LVD)....122

16.1 LVD MOD GENERAL 122
16.2 LVD RELATED REGISTER 122
16.3 LVD OPERATION 122

17. ELECTRICAL PARAMETERS 123

17.1 LIMIT PARAMETERS 123
17.2 DC ELECTRICAL CHARACTERISTIC 124
17.3 ADC ELECTRICAL CHARACTERISTIC.... 125
17.4 ADC INTERNAL LDO REFERENCE VOLTAGE CHARACTERISTICS 125
17.5 POWER-ON RESET FEATURES 125
17.6 LVD ELECTRICAL CHARACTERISTIC 125
17.7 ELECTRICAL CHARACTERISTICS 126

18. INSTRUCTIONS....127

18.1 INSTRUCTIONS TABLE 127
18.2 INSTRUCTIONS ILLUSTRATION 130

19. PACKAGING 146

19.1 SOP8.... 146
19.2 SOP14.... 147
19.3 SOP16.... 148
19.4 SOP20.... 149

20. REVISION HISTORY 150

1. Product Description

1.1 Features

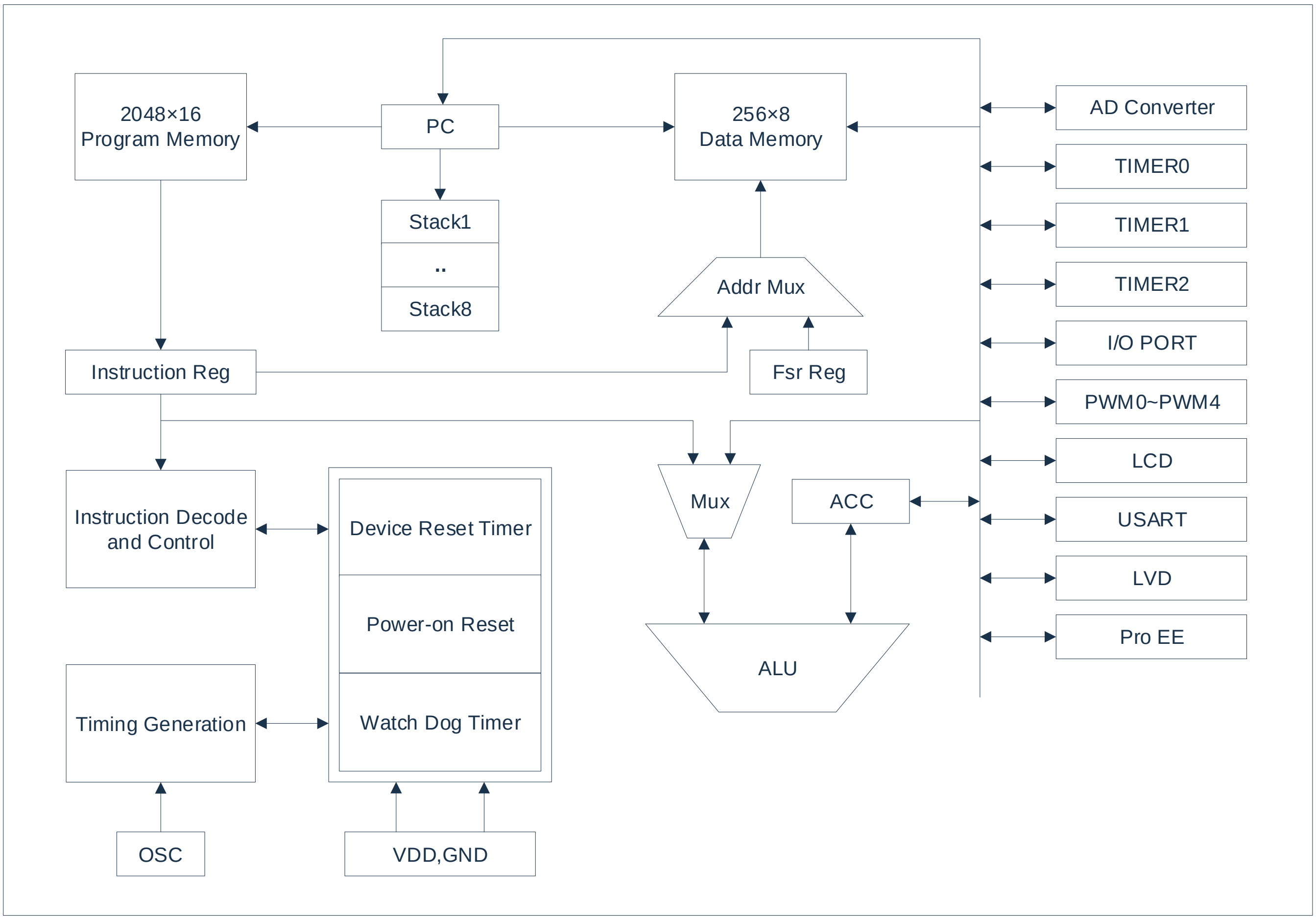
- Memory
  - ROM: 2K×16Bit
  - RAM: 256×8Bit
- 8-level stack buffer
- Short and clear command system (68 commands)
- Lookup table function
- Built in WDTtimer
- Interrupt
  - 3 timer Interrupt
  - RA Interrupt on Change in logic level
  - Other external hardware Interrupt
- Timer:
  - 8-bit Timer: TIMER0, TIMER2
  - 16-bit Timer: TIMER1
- Built in LVD module
  - Support multiple voltage: 2.2V/2.4V/2.7V/ 3.0V/ 3.3V/3.7V/4.0V/4.3V
- Built in 128-byte EEPROM
  - Rewritable up to 100,000 times
- Voltage range: 2.6V—5.5V@16MHz
  1.8V—5.5V@8MHz
  Temperature range: -40°C—85°C
- Multiple ways of oscillation
  - Internal RC: design frequency of 8MHz/16MHz
  - External XT oscillation: up to 8MHz
- Instructions period (single instruction or double instructions)
- Built-in low voltage detection circuit
- USART communication module
- PWM module with complementary output
  - 5 channels of PWM, can be set to 2 channels of complementary output
  - 4ch PWM with shared period and separated duty cycle
  - 1ch PWM with separated period and separated duty cycle
- Built-in LCD driver module
  - All I/O ports have optional SEG/ COM output
  - All I/O ports can choose 1/2BIAS or 1/3BIAS output
  - All I/O port drive currents are optional
- High precision 12-bit ADC
  - Built in high presicion 1.2V reference voltage
  - ±1.5% @VDD=2.5V~5.5V T_A=25°C
  - ±2% @VDD=2.5V~5.5V T_A=-40°C~85°C

Product comparison:

PRODUCT	ROM	RAM	Pro EE	I/O	LCD	ADC	USART	PACKAGE
CMS79F111	2Kx16	256x8	128x8	6	1/2Bias1/3Bias	12Bitx6	1	SOP8
CMS79F112	2Kx16	256x8	128x8	12	1/2Bias1/3Bias	12Bitx12	1	SOP14
CMS79F113	2Kx16	256x8	128x8	14	1/2Bias1/3Bias	12Bitx14	1	SOP16
CMS79F116	2Kx16	256x8	128x8	18	1/2Bias1/3Bias	12Bitx18	1	SOP20

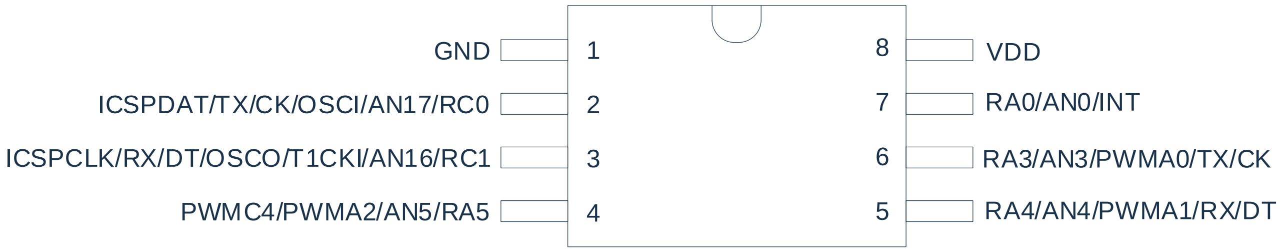
Note: ROM---- Read Only Memory Pro EE---- program EEPROM

1.2 System structure



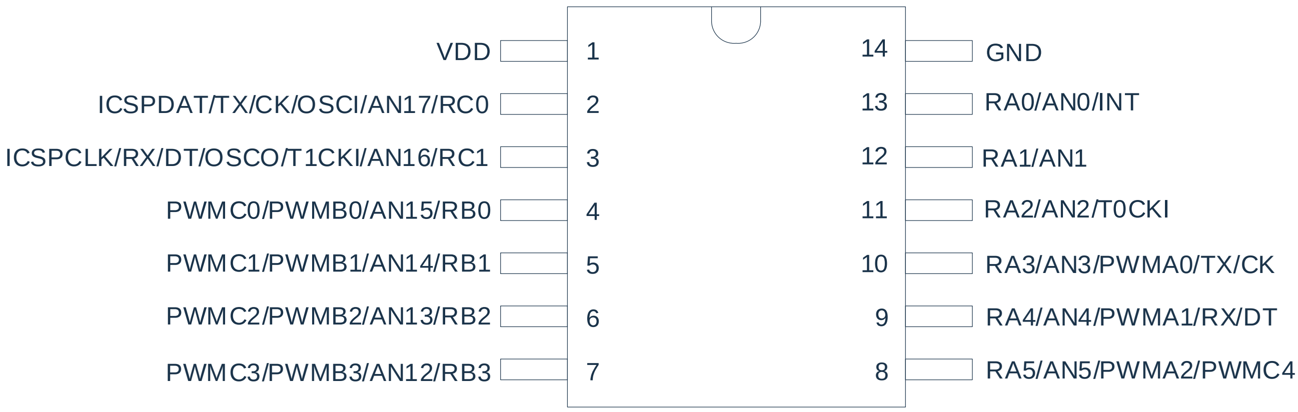
1.3 Pin configuration

1.3.1 CMS79F111



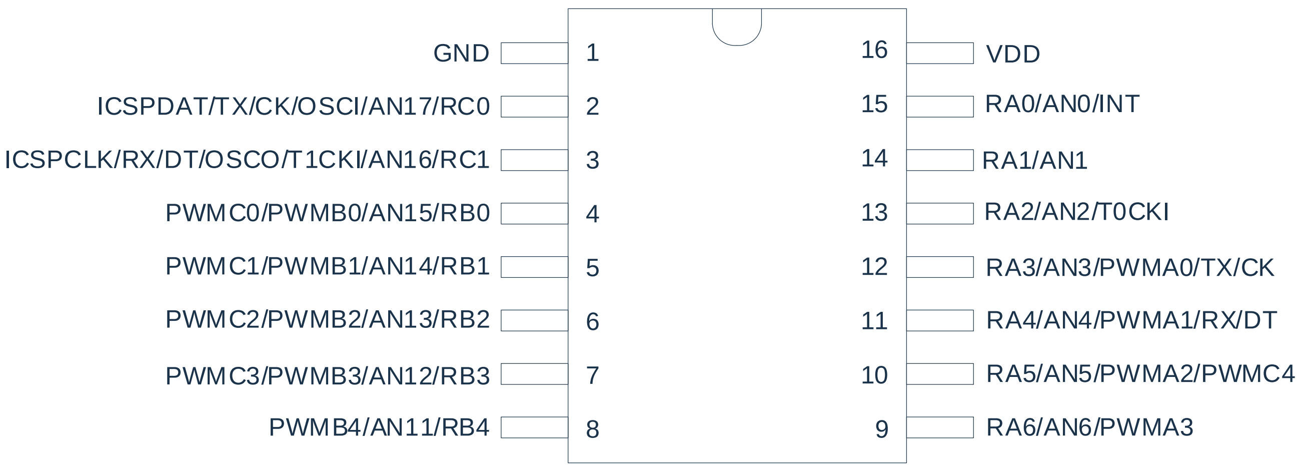
CMS79F111

1.3.2 CMS79F112



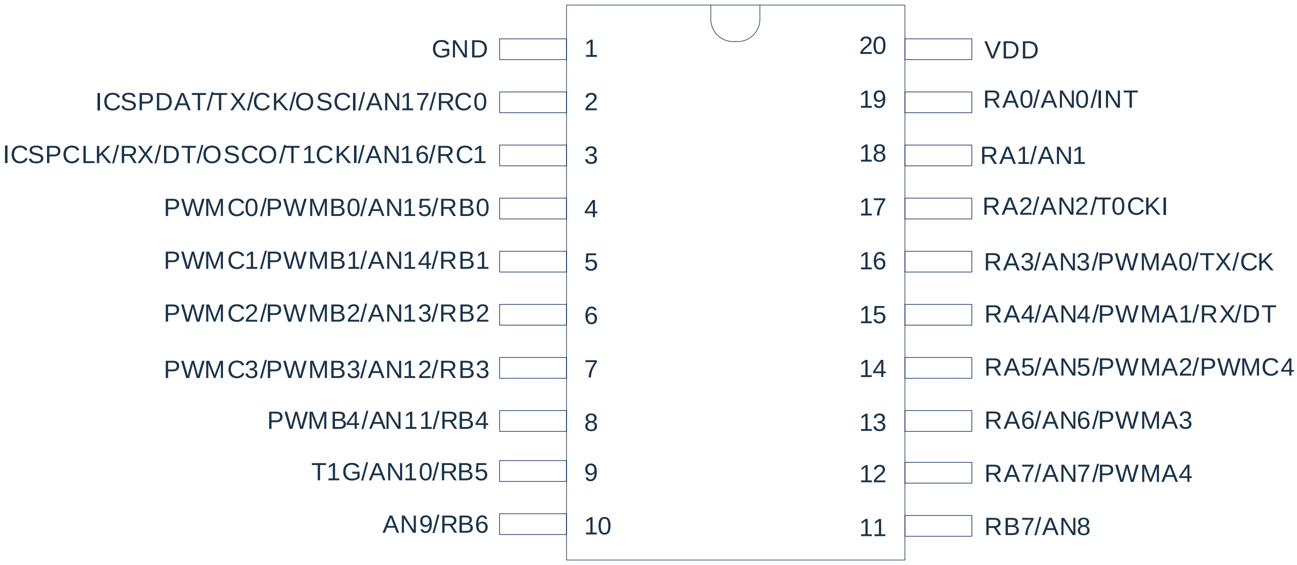
CMS79F112

1.3.3 CMS79F113



CMS79F113

1.3.4 CMS79F116



CMS79F116

Note:
1) The serial port function of RA4 and RA3 and the serial port function of RC0 and RC1 are set by CONFIG.
2) The PWMx function is set by the PWMCON1 register.

CMS79F11x Pin description and function:

Pin	IO	Pin description
VDD, GND	P	Power supply voltage input pin, ground pin
OSCIN/OSCOUT	I/O	Crystal Oscillator Input/Output Pin
RA0-RA7	I/O	Programmed as input pin or push-pull output pin, with pull-up resistor function and interrupt on change in logic level function
RB0-RB7	I/O	Programmed as input pin or push-pull output pin, with pull-up resistor function
RC0-RC1	I/O	Programmed as input pin or push-pull output pin, with pull-up resistor function
ICSPCLK/ICSKCLK	I/O	Programming clock/data pin
AN0-AN17	I	12-bit ADC input pin
T0CKI	I	TIMER0 external clock input pin
T1CKI	I	TIMER1 external clock input pin
T1G	I	TIMER1 gate control input pin
TX/CK	O	Asynchronous serial port sending pin/synchronous serial port clock input and output pin (configurable in different IO ports)
RX/DT	I	Asynchronous serial port receiving pin/synchronous serial port data input and output pins (configurable in different IO ports)
PWMx0-4	O	PWM0-4 output function (configurable in different IO ports)
INT	I	External interrupt input pin

1.4 System Configuration Register

System configuration registor (CONFIG) is MCU's initial condition's ROM choice. It can only be programmed by CMS burner. User cannot visit it or place any action on it. It includes the following.

1. OSC (choice of oscillation)
   - INTRC — Internal RC oscillation
   - XT — External crystal oscillation
2. INTRC_SEL (internal oscillation frequency)
   - INTRC8M — F_{HSI} choose internal 8MHz RC oscillation
   - INTRC16M — F_{HSI} choose internal 16MHz RC oscillation
3. WDT (watchdog choice)
   - ENABLE — Enable watchdog timer
   - DISABLE — Disable watchdog timer
4. PROTECT (encyption)
   - DISABLE — Disable FLASH code encyption
   - ENABLE — Enable FLASH code encryption, after which the read value from burning the simulator is uncertain.
5. LVR_SEL (low voltage detection selection)
   - 1.8V — To select this reset voltage point, F_{OSC} needs to select 8MHz
   - 2.0V
   - 2.6V
6. USART_SEL（TX/RX）（USART port selection）
   - RC0/RC1 — Select RC0 as TX, RC1 as RX
   - RA3/RA4 — Select RA3 as TX 口, RA4 as RX
7. ICSPPORT_SEL (simulation port selection)
   - ICSP — ICSPCLK, DAT port keep as similation port, all functions disabled
   - NORMAL — ICSPCLK, DAT port as normal port

1.5 Online Serial Programming

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

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

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

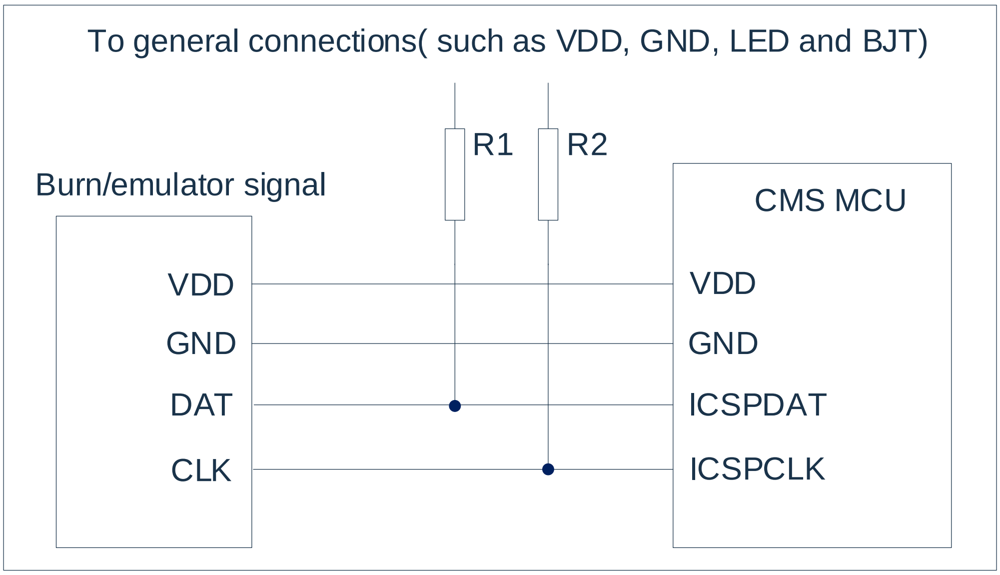


Fig 1-1：Typical connection for online serial programming

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

2. Central Processing Unit (CPU)

2.1 Memory

2.1.1 Program Memory

CMS79F11x program memory space

FLASH:2K

0000H	Reset Vector	Program start, jump to user program
0001H		
0002H		
0003H		
0004H	Interrupt vector	Interrupt entry, user interrupt program
...		User program area
...		
...		
7FDH		
7FEH		
7FFH	Jump to Reset Vector 0000H	End of program

2.1.1.1 Reset Vector (0000H)

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

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

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

Example: define reset vector

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

2.1.1.2 Interrupt Vector

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

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

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

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

Example: interrupt-in protection

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

Example: interrupt-out restore

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

2.1.1.3 Look-up Table

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

Related instructions:

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

related register:

- TABLE_SPH(9AH) Read/write register to indicate higher 3 bits in the table.
- TABLE_SPL(9BH) Read/write register to indicate lower 8 bits in the table.
- TABLE_DATAH(94H) Read only register to save higer bit information in the table

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

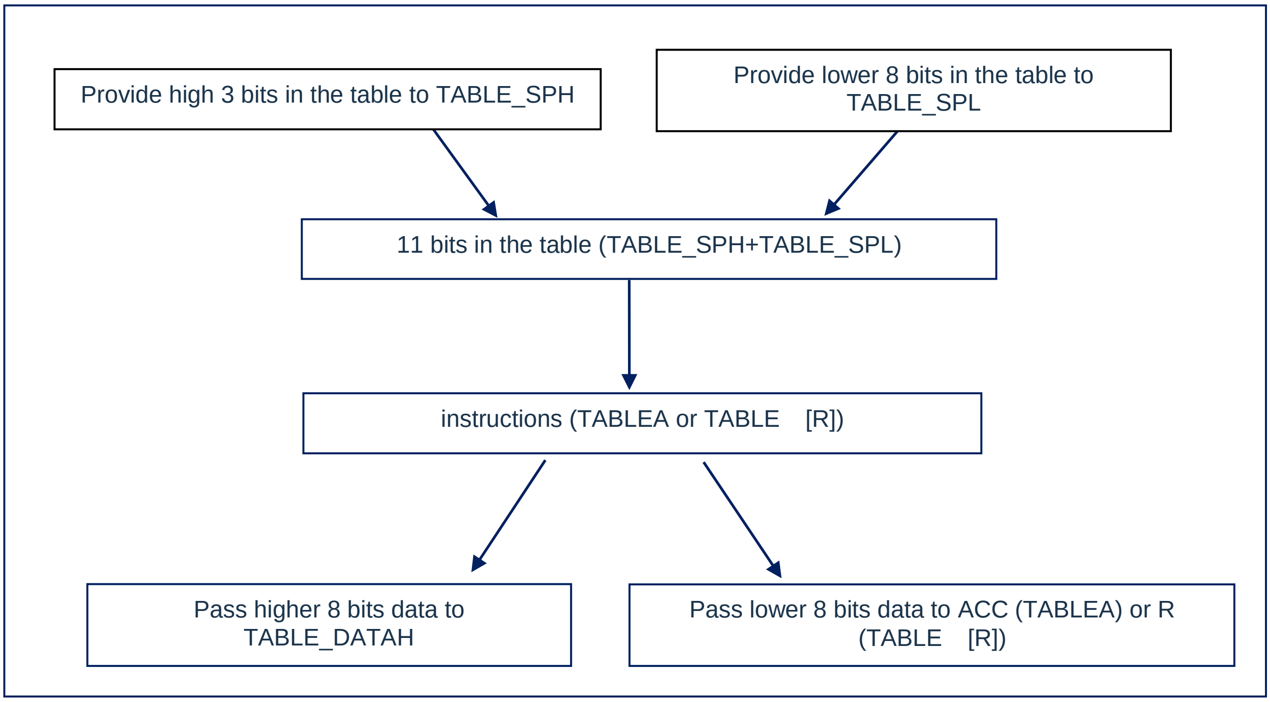


Fig 2-1: Flow chart for table usage

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

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

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

2.1.1.4 Jump Table

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

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

Example: correct illustration of multi-address jump

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

Example: wrong illustration of multi-address jump

```
FLASH address
                CLR         PCLATH
                …
   00FCH:       ADDR        PCL          ;ACC+PCL
   00FDH:       JP          LOOP1        ;ACC=0, jump to LOOP1
   00FEH:       JP          LOOP2        ;ACC=1, jump to LOOP2
   00FFH:       JP          LOOP3        ;ACC=2, jump to LOOP3
   0100H:       JP          LOOP4        ;ACC=3, jump to 0000H address
   0101H:       JP          LOOP5        ;ACC=4, jump to 0001H address
   0102H:       JP          LOOP6        ;ACC=5, jump to 0002H address
```

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

2.1.2 Data Memory

List of data memory of CMS79F11x

BANK0	address	BANK1	address	BANK2	address	BANK3	address
INDF	00H	INDF	80H	INDF	100H	INDF	180H
TMR0	01H	OPTION_REG	81H	----	101H	----	181H
PCL	02H	PCL	82H	PCL	102H	PCL	182H
STATUS	03H	STATUS	83H	STATUS	103H	STATUS	183H
FSR	04H	FSR	84H	FSR	104H	FSR	184H
PORTA	05H	TRISA	85H	PIR1	105H	----	185H
PORTB	06H	TRISB	86H	PIE1	106H	----	186H
WPUA	07H	WPDB	87H	PIR2	107H	----	187H
WPUB	08H	OSCCON	88H	PIE2	108H	----	188H
----	09H	WDTCON	89H	----	109H	----	189H
PCLATH	0AH	PCLATH	8AH	PCLATH	10AH	PCLATH	18AH
INTCON	0BH	INTCON	8BH	INTCON	10BH	INTCON	18BH
----	0CH	EECON1	8CH	TMR1L	10CH	----	18CH
----	0DH	EECON2	8DH	TMR1H	10DH	----	18DH
PWMD23H	0EH	EEDATAL	8EH	T1CON	10EH	----	18EH
PWM01DT	0FH	EEDATAH	8FH	----	10FH	----	18FH
PWM23DT	10H	EEADRL	90H	ANSEL0	110H	----	190H
TMR2	11H	PR2	91H	ANSEL1	111H	----	191H
T2CON	12H	PORTC	92H	ANSEL2	112H	----	192H
PWMCON0	13H	TRISC	93H	LCDCON	113H	----	193H
PWMCON1	14H	TABLE_DATAH	94H	CSSEL0	114H	----	194H
PWMTL	15H	IOCA	95H	CSSEL1	115H	----	195H
PWMTH	16H	EEADRH	96H	CSSEL2	116H	----	196H
PWMD0L	17H	WPDA	97H	TXSTA	117H	----	197H
PWMD1L	18H	WPDC	98H	RCSTA	118H	----	198H
PWMD2L	19H	WPUC	99H	SPBRG	119H	----	199H
PWMD3L	1AH	TABLE_SPH	9AH	TXREG	11AH	----	19AH
PWMD4L	1BH	TABLE_SPL	9BH	RCREG	11BH	----	19BH
PWMD01H	1CH	ADCON1	9CH	CSEN0	11CH	----	19CH
PWMCON2	1DH	ADCON0	9DH	CSEN1	11DH	----	19DH
PWM4TL	1EH	ADRESH	9EH	CSEN2	11EH	----	19EH
----	1FH	ADRESL	9FH	LVDCON	11FH	----	19FH
Universal register 96 byte	20H 6FH 70H -- 7FH	Universal register 80byte	A0H EFH	Universal register 80byte	120H 16FH	----	1A0H 1EFH
		Fast memory spce 70H-7FH	F0H -- FFH	Fast memory space 70H-7FH	170H -- 17FH	Fast memory space 70H-7FH	1F0H -- 1FFH
BANK0		BANK1		BANK2		BANK3	

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

Summary of special registers in CMS79F11x Bank0

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
00H	INDF	Look-up for this unit will use FSR, not physical register.								xxxxxxxx
01H	TMR0	TIMER0 data register								xxxxxxxx
02H	PCL	Lower bit of program counter								00000000
03H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
04H	FSR	memory pointers for indirect addressing of data memory								xxxxxxxx
05H	PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	xxxxxxxx
06H	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxxxxxx
07H	WPUA	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0	00000000
08H	WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0	00000000
0AH	PCLATH	----	----	----	----	----	Write buffer of higher 3 bits of program counter			-----000
0BH	INTCON	GIE	PEIE	T0IE	INTE	----	T0IF	INTF	----	0000-00-
0EH	PWMD23H	----	----	PWMD3[9:8]		----	----	PWMD[9:8]		--00--00
0FH	PWM01DT	----	----	PWM01 dead zone delay time						--000000
10H	PWM23DT	----	----	PWM23 dead zone delay time						--000000
11H	TMR2	TIMER2 mod register								00000000
12H	T2CON	----	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-0000000
13H	PWMCON0	CLKDIV[2:0]			PWM4EN	PWM3EN	PWM2EN	PWM1EN	PWM0EN	00000000
14H	PWMCON1	PWMIO_SEL[1:0]		PWM2DTEN	PWM0DTEN	----	----	DT_DIV[1:0]		0000--00
15H	PWMTL	PWM period low register								00000000
16H	PWMTH	----	----	PWMD4[9:8]		PWM4T9	PWM4T8	PWMT9	PWMT8	--000000
17H	PWMD0L	PWM0 duty cycle low register								00000000
18H	PWMD1L	PWM1 duty cycle low register								00000000
19H	PWMD2L	PWM2 duty cycle low register								00000000
1AH	PWMD3L	PWM3 duty cycle low register								00000000
1BH	PWMD4L	PWM4 duty cycle low register								00000000
1CH	PWMD01H	----	----	PWMD1[9:8]		----	----	PWMD0[9:8]		--00--00
1DH	PWMCON2	----	----	----	PWM4DIR	PWM3DIR	PWM2DIR	PWM1DIR	PWM0DIR	---00000
1EH	PWM4TL	PWM4 period low register								00000000

Summary of special registers in CMS79F11x Bank1

address	name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
80H	INDF	Look-up for this unit will use FSR, not physical register.								xxxxxxxx
81H	OPTION_REG	----	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	-1111011
82H	PCL	Lower bit of program counter								00000000
83H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
84H	FSR	memory pointers for indirect addressing of data memory								xxxxxxxx
85H	TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	11111111
86H	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	11111111
87H	WPDB	WPDB7	WPDB6	WPDB5	WPDB4	WPDB3	WPDB2	WPDB1	WPDB0	00000000
88H	OSCCON	----	IRCF2	IRCF1	IRCF0	----	----	----	SCS	-110---0
89H	WDTCON	----	----	----	----	----	----	----	SWDTEN	-------0
8AH	PCLATH	----	----	----	----	----	Write buffer of higher 3 bits of program counter			-----000
8BH	INTCON	GIE	PEIE	T01E	INTE	----	T0IF	INTF	----	0000-00-
8CH	EECON1	EEPGD	----	EETIME1	EETIME0	WRERR	WREN	WR	RD	0-00x000
8DH	EECON2	EEPROM control register 2 (not physical address)								--------
8EH	EEDAT	EEDAT7	EEDAT6	EEDAT5	EEDAT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0	xxxxxxxx
8FH	EEDATH	EEDATH7	EEDATH6	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0	xxxxxxxx
90H	EEADR	EEADR7	EEADR6	EEADR5	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0	00000000
91H	PR2	TIMER2 period register								00000000
92H	PORTC	----	----	----	----	----	----	RC1	RC0	------xx
93H	TRISC	----	----	----	----	----	----	TRISC1	TRISC0	------11
94H	TABLE_DATAH	Table high data								xxxxxxxx
95H	IOCA	IOCA7	IOCA6	IOCA5	IOCA4	IOCA3	IOCA2	IOCA1	IOCA0	00000000
96H	EEADRH	----	----	----	----	----	EEADRH2	EEADRH1	EEADRH0	-----000
97H	WPDA	WPDA7	WPDA6	WPDA5	WPDA4	WPDA3	WPDA2	WPDA1	WPDA0	00000000
98H	WPDC	----	----	----	----	----	----	WPDC1	WPDC0	------00
99H	WPUC	----	----	----	----	----	----	WPUC1	WPUC0	------00
9AH	TABLE_SPH	----	----	----	----	----	Table high 3 address			-----xxx
9BH	TABLE_SPL	Table low pointer								xxxxxxxx
9CH	ADCON1	ADFM	CHS4	----	----	----	LDO_EN	LDO_SEL[1:0]		00---000
9DH	ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/ $\overline{DONE}$	ADON	00000000
9EH	ADRESH	Higher bit of A/D result register								xxxxxxxx
9FH	ADRESL	Lower bit of A/D result register								xxxxxxxx

Summary of special registers in CMS79F11x Bank2

address	name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
100H	INDF	Look-up for this unit will use FSR , not physical register.								xxxxxxxx
102H	PCL	Lower bit of program counter								00000000
103H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
104H	FSR	memory pointers for indirect addressing of data memory								xxxxxxxx
105H	PIR1	----	ADIF	RCIF	TXIF	----	PWMIF	TMR2IF	TMR1IF	-000-000
106H	PIE1	----	ADIE	RCIE	TXIE	----	PWMIE	TMR2IE	TMR1IE	-000-000
107H	PIR2	----	----	----	EEIF	----	----	RACIF	LVDIF	---0--00
108H	PIE2	----	----	----	EEIE	----	----	RACIE	LVDIE	---0--00
10AH	PCLATH	----	----	----	----	----	Write buffer of higher 3 bits of program counte			-----000
10BH	INTCON	GIE	PEIE	T0IE	INTE	----	T0IF	INTF	----	0000-00-
10CH	TMR1L	16-bit TIMER1 register low byte data register								xxxxxxxx
10DH	TMR1H	16-bit TIMER1 register high byte data register								xxxxxxxx
10EH	T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	T0OSCEN	T1SYNC	TMR1CS	TMR1ON	00000000
110H	ANSEL0	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0	00000000
111H	ANSEL1	ANS15	ANS14	ANS13	ANS12	ANS11	ANS10	ANS9	ANS8	00000000
112H	ANSEL2	----	----	----	----	----	----	ANS17	ANS16	------00
113H	LCDCON	LCDEN	FRAME	BIAS	----	----	----	LCDISLE[1:0]		000---00
114H	CSSEL0	CS7SEL	CS6SEL	CS5SEL	CS4SEL	CS3SEL	CS2SEL	CS1SEL	CS0SEL	00000000
115H	CSSEL1	CS15SEL	CS14SEL	CS13SEL	CS12SEL	CS11SEL	CS10SEL	CS9SEL	CS8SEL	00000000
116H	CSSEL2	----	----	----	----	----	----	CS17SEL	CS16SEL	------00
117H	TXSTA	CSRC	TX9EN	TXEN	SYNC	SCKP	STOPBIT	TRMT	TX9D	00000010
118H	RCSTA	SPEN	RX9EN	SREN	CREN	RCIDL	FERR	OERR	RX9D	00001000
119H	SPBRG	USART baud rate 8-bit register								00000000
11AH	TXREG	USART transmit data register								00000000
11BH	RCREG	USART receive data register								xxxxxxxx
11CH	CSEN0	CS7EN	CS6EN	CS5EN	CS4EN	CS3EN	CS2EN	CS1EN	CS0EN	00000000
11DH	CSEN1	CS15EN	CS14EN	CS13EN	CS12EN	CS11EN	CS10EN	CS9EN	CS8EN	00000000
11EH	CSEN2	----	----	----	----	----	----	CS17EN	CS16EN	------00
11FH	LVDCON	LVD_RES	----	----	----	LVD_SEL2	LVD_SEL1	LVD_SEL0	LVDEN	00---0000

Summary of special registers in CMS79F11x Bank3

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
180H	INDF	Look-up for this unit will use FSR , not physical register.								xxxxxxxx
182H	PCL	Lower bit of program counter (PC)								00000000
183H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
184H	FSR	memory pointers for indirect addressing of data memory								xxxxxxxx
18AH	PCLATH	----	----	----	Write buffer of higher 5 bits of program counter					---00000
18BH	INTCON	GIE	PEIE	T01E	INTE	----	T01F	INTF	----	0000-00-

2.2 Addressing Mode

2.2.1 Direct Addressing

Operate on RAM through accumulator (ACC)

Example: pass the value in ACC to 30H register

```
LD        30H,A
```

Example: pass the value in 30H register to ACC

```
LD        A,30H
```

2.2.2 Immediate Addressing

Pass the immediate value to accumulator (ACC).

Example: pass immediate value 12H to ACC

```
LDIA      12H
```

2.2.3 Indirect Addressing

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

Example: application of FSR and INDF

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

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

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

2.3 Stack

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

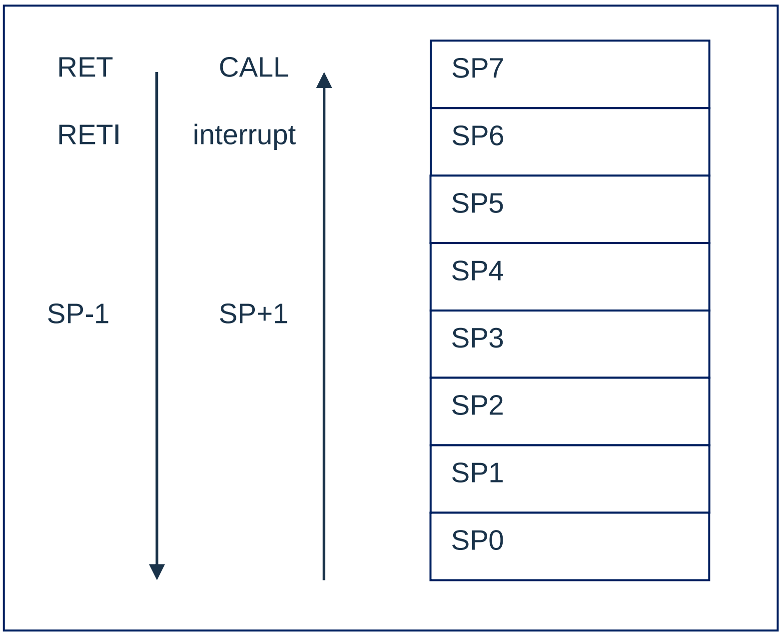


Fig 2-2: stack buffer working principle

Stack bufferwill follow one principle: 'first in last out'.

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

2.4 Accumulator (ACC)

2.4.1 General

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

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

2.4.2 ACC Applications

Example: use ACC for data transfer

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

Example: use ACC for immediate addressing

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

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

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

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

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

2.5 Program Status Register (STATUS)

STATUS register includes:

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

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

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

program status register STATUS (03H)

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

Bit7	IRP:	Selection bit of register memory (for indirect addressing)
	1=	Bank2 and Bank3 (100h-1FFh);
	0=	Bank0 and Bank1 (00h-FFh).
Bit6~Bit5	RP[1:0]:	Selection bit of memory;
	00:	Select Bank 0;
	01:	Select Bank 1;
	10:	Select Bank 2;
	11:	Select Bank 3.
Bit4	TO:	Time out bit;
	1=	Power on or CLRWDT instructions or STOP instructions;
	0=	WDT time out.
Bit3	PD:	Power down;
	1=	Power on or CLRWDT instructions;
	0=	STOP instructions.
Bit2	Z:	Bit for result inzero;
	1=	The result of arithmetic or logical operation is zero;
	0=	The result of an arithmetic or logical operation is not zero.
Bit1	DC:	Half Carry/Borrow Bit;
	1=	The fourth low bit of the result is carried to the high bit;
	0=	The fourth low bit of the result did not carry to the high bit.
Bit0	C:	Carry/borrow bit;
	1=	The highest bit of the result has a carry;
	0=	The highest bit of the result is not carried.

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

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

Events which affect TO/PD

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

TO/PD status after reset

2.6 Pre-scaler (OPTION_REG)

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

- TIMER0/WDT pre-scaler.
- TIMER0.

pre-scaler OPTION_REG(81H)

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

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

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

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

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

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

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

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

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

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

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

2.7 Program Counter (PC)

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

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

Program counter (PC)is 11 Bit, user can access lower 8 bits through PCL (02H). The higher 3 bits cannot be accessed. It can hold address for 2K×16Bit program. Passing a value to PCL will cause a short jump which range until the 256 addressof the current page.

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

The following are the value of PC under special conditions.

reset	PC=0000;
interrupt	PC=0004 (original PC+1will be add to stack automatically);
CALL	PC=program defined address (original PC+1will be add to stack automatically);
RET、RETI、RET i	PC=value coming out from stack;
Operating on PCL	PC[11:8] unchange, PC[7:0]=user defined value;
JP	PC=program defined value;
Other instructions	PC=PC+1;

2.8 Watchdog Timer (WDT)

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

2.8.1 WDT Period

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

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

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

2.8.2 Watchdog Timer Control Register WDTCON

Watchdog Timer Control Register WDTCON (89H)

89H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WDTCON	---	---	---	---	---	---	---	SWDTEN
R/W	---	---	---	---	---	---	---	R/W
Reset value	---	---	---	---	---	---	---	0

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

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

3. System Clock

3.1 General

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

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

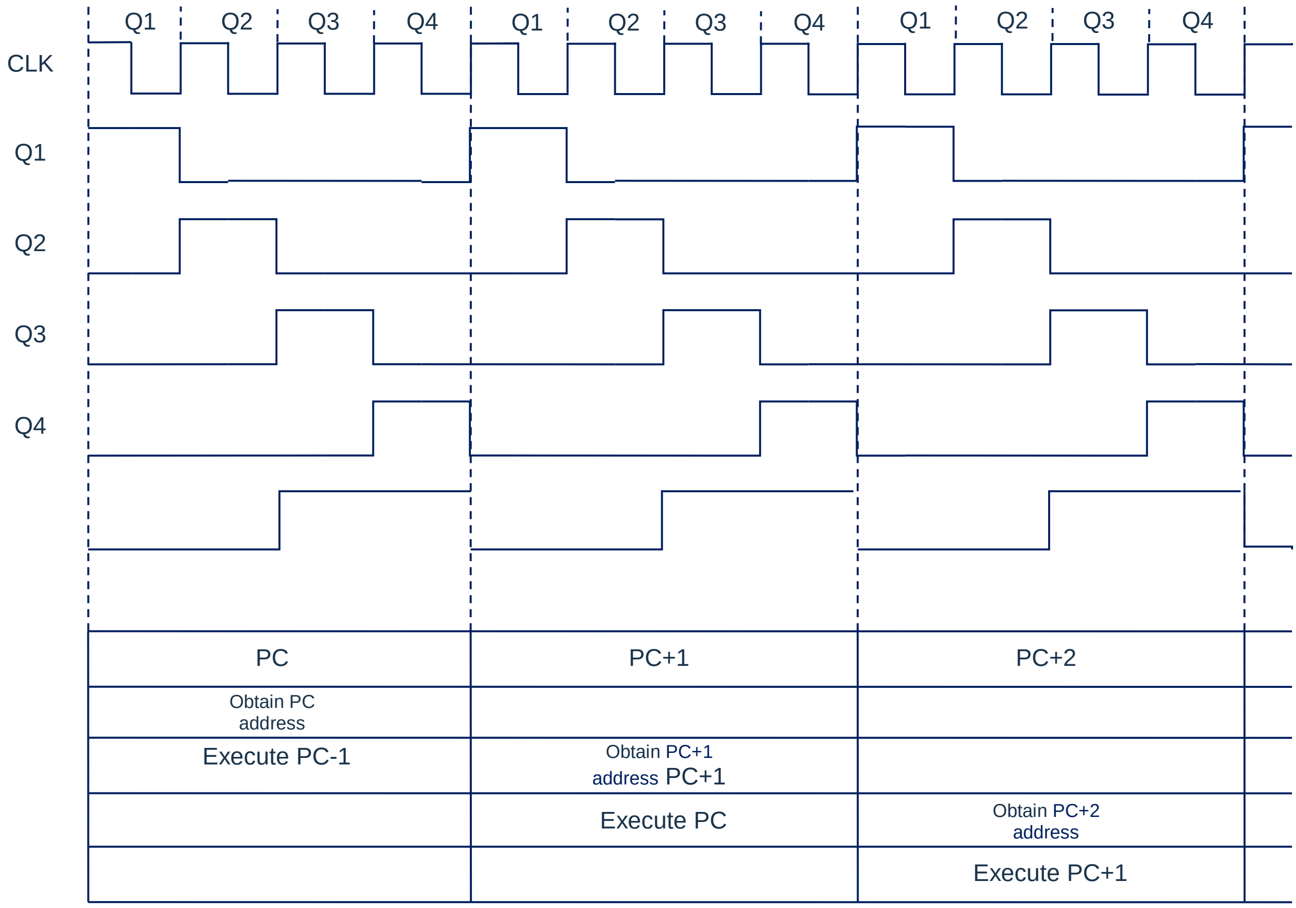


Fig 3-1: time series for clock and instruction period

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

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

3.2 System Oscillator

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

3.2.1 Internal RC Oscillation

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

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

3.2.2 External XT Oscillation

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

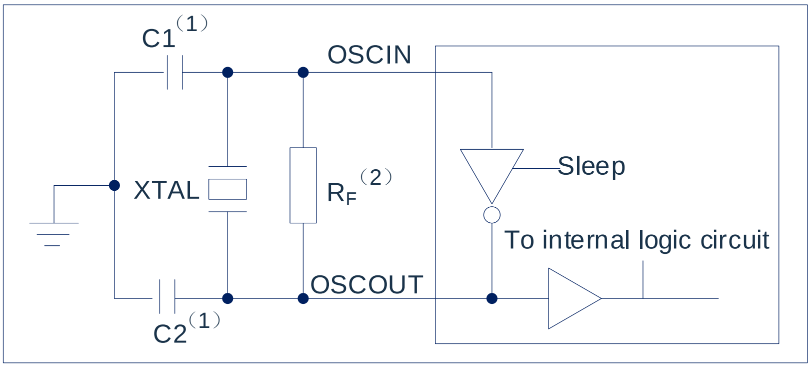


Fig 3-2: Typical XT oscillation

Recommend parameters:

Type	Frequency	Recommend R_F	Recommend C1 ~ C2
XT	2MHz	1MΩ	10pF～47pF
XT	4MHz	1MΩ	10pF～47pF
XT	8MHz	1MΩ	10pF～47pF
XT	16MHz	1MΩ	10pF～47pF

3.3 Reset Time

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

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

3.4 Oscillator Control Register

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

Oscillator control register OSCCON (88H)

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

Bit7 Not used, read 0

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

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

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

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

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

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

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

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

000= 32KHz（LFINTOSC）.

Bit3~Bit1 Not used

Bit0 SCS: System clock selection bit.

1= The internal oscillator is used as the system clock.

0= The clock source is defined by CONFIG.

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

3.5 Clock Block Diagram

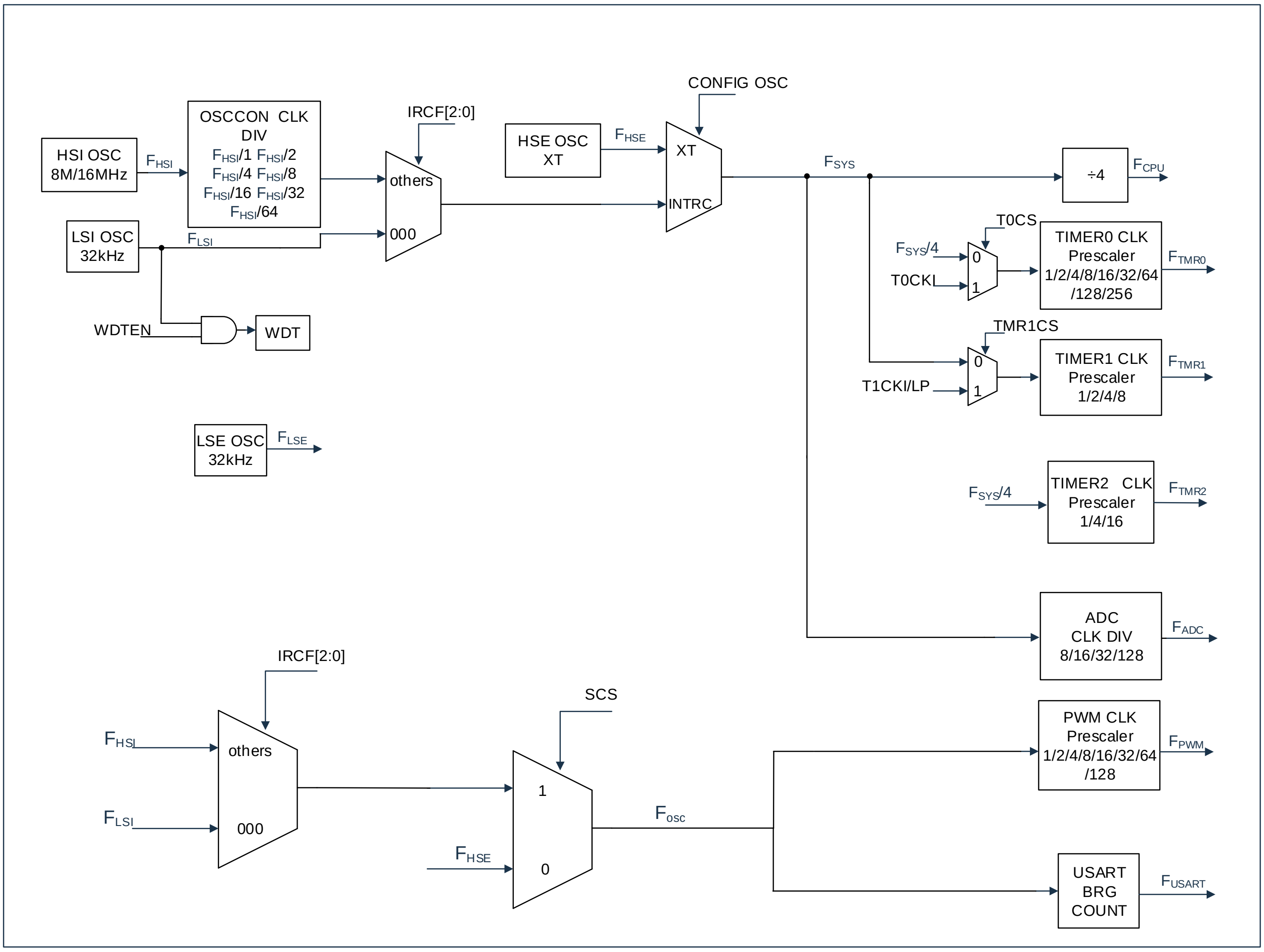


Fig 3-2：clock block diagram

4. Reset

Chip has 3 ways of reset:

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

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

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

4.1 Power on Reset

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

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

4.2 Power off Reset

4.2.1 General

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

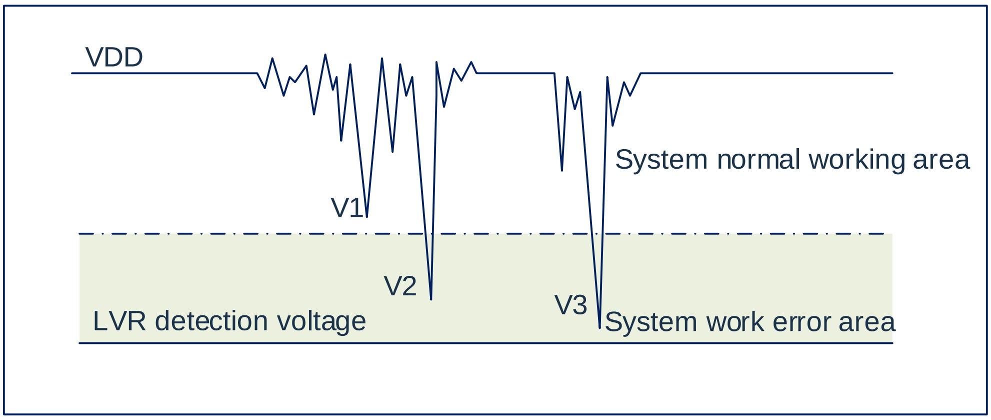


Fig 4-1: power off reset

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

System will enter the dead zone under the following situation:

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

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

4.2.2 Improvements for Power off Reset

Suggestions to improve the power off reset:

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

Watchdog timer

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

Lower the working speed of the system

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

Increase the gradient of the voltage drop

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

4.3 Watchdog Reset

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

Time series for watchdog reset:

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

For applications of watchdog timer, see chapters at 2.8.

5. Sleep Mode

5.1 Enter Sleep Mode

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

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

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

5.2 Awaken from Sleep Mode

Awaken through any of the following events:

1. Watchdog timer awake (WDT force enable)
2. PORTA electrical level interrupt
3. Peripheral interrupt

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

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

5.3 Interrupt Awakening

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

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

5.4 Sleep Mode Application

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

Example: procedures for entering sleep mode.

```
SLEEP_MODE:
        CLR         INTCON          ; disable interrupt
        LDIA        B'00000000'
        LD          TRISA,A
        LD          TRISB,A         ;all I/O set as output
        LD          TRISC,A
        …                           ;turn off other functions
        LDIA        0A5H
        LD          SP_FLAG,A       ; Set sleep state memory register (user-defined)
        CLRWDT                      ;clear WDT
        STOP                        ; Execute the STOP instruction
```

5.5 Sleep Mode Awaken Time

When the MCU is awakened from the sleep state, it needs to wait for an oscillation stabilization time (Reset Time). This time is 128 T_{SYS} clock cycles in the internal high-speed oscillation mode, and 12 T_{SYS} clock cycles in the internal low-speed oscillation mode. In the mode, it is 128 F_{SYS} clocks. The specific relationship is shown in the following table.

System main clock source	System clock frequency (IRCF<2:0>)	Sleep wake-up waiting time T_{WAIT}
Internal high-speed RC oscillation (F_{OSC})	F_{SYS}=F_{OSC}	T_{WAIT}=128*1/F_{OSC}
	F_{SYS}= F_{OSC} /2	T_{WAIT}=128*2/F_{OSC}
	…	…
	F_{SYS}= F_{OSC} /64	T_{WAIT}=128*64/F_{OSC}
Internal low-speed RC oscillation ($F_{LFINTOSC}$)	----	T_{WAIT}=12/$F_{LFINTOSC}$
XT oscillation (F_{XT})	----	T_{WAIT}=128/F_{XT}

6. I/O Port

The chip has 3 I/O ports: PORTA, PORTB, PORTC (up to 18 I/O). The readable and writable port data registers can directly access these ports.

Port	Bit	Pin Description	I/O
PORTA	0	Schmitt trigger input, push-pull output, AN0, LCD drive port, external interrupt input	I/O
	1	Schmitt trigger input, push-pull output, AN1, LCD drive port	I/O
	2	Schmitt trigger input, push-pull output, AN2, LCD drive port, TMR0 clock input	I/O
	3	Schmitt trigger input, push-pull output, AN3, LCD drive port, PWM output, asynchronous serial port output, synchronous serial port clock	I/O
	4	Schmitt trigger input, push-pull output, AN4, LCD drive port, PWM output, asynchronous serial port input, synchronous serial port data	I/O
	5	Schmitt trigger input, push-pull output, AN5, LCD drive port, PWM output, TMR1 gate control input	I/O
	6	Schmitt trigger input, push-pull output, AN6, LCD drive port, PWM output	I/O
	7	Schmitt trigger input, push-pull output, AN7, LCD drive port, PWM output	I/O
PORTB	0	Schmitt trigger input, push-pull output, AN15, LCD drive port, PWM output	I/O
	1	Schmitt trigger input, push-pull output, AN14, LCD drive port, PWM output	I/O
	2	Schmitt trigger input, push-pull output, AN13, LCD drive port, PWM output	I/O
	3	Schmitt trigger input, push-pull output, AN12, LCD drive port, PWM output	I/O
	4	Schmitt trigger input, push-pull output, AN11, LCD drive port, PWM output	I/O
	5	Schmitt trigger input, push-pull output, AN10, LCD drive port	I/O
	6	Schmitt trigger input, push-pull output, AN9, LCD drive port	I/O
	7	Schmitt trigger input, push-pull output, AN8, LCD drive port	I/O
PORTC	0	Schmitt trigger input, push-pull output, AN17, LCD drive port, programming data input/output, oscillation input port, asynchronous serial port output, synchronous serial port clock	I/O
	1	Schmitt trigger input, push-pull output, AN18, LCD drive port, programming clock input, oscillator output port, TMR1 clock input, asynchronous serial port input, synchronous serial port data	I/O

< Table 6-1: port configuration summary >

6.1 I/O Port Structure

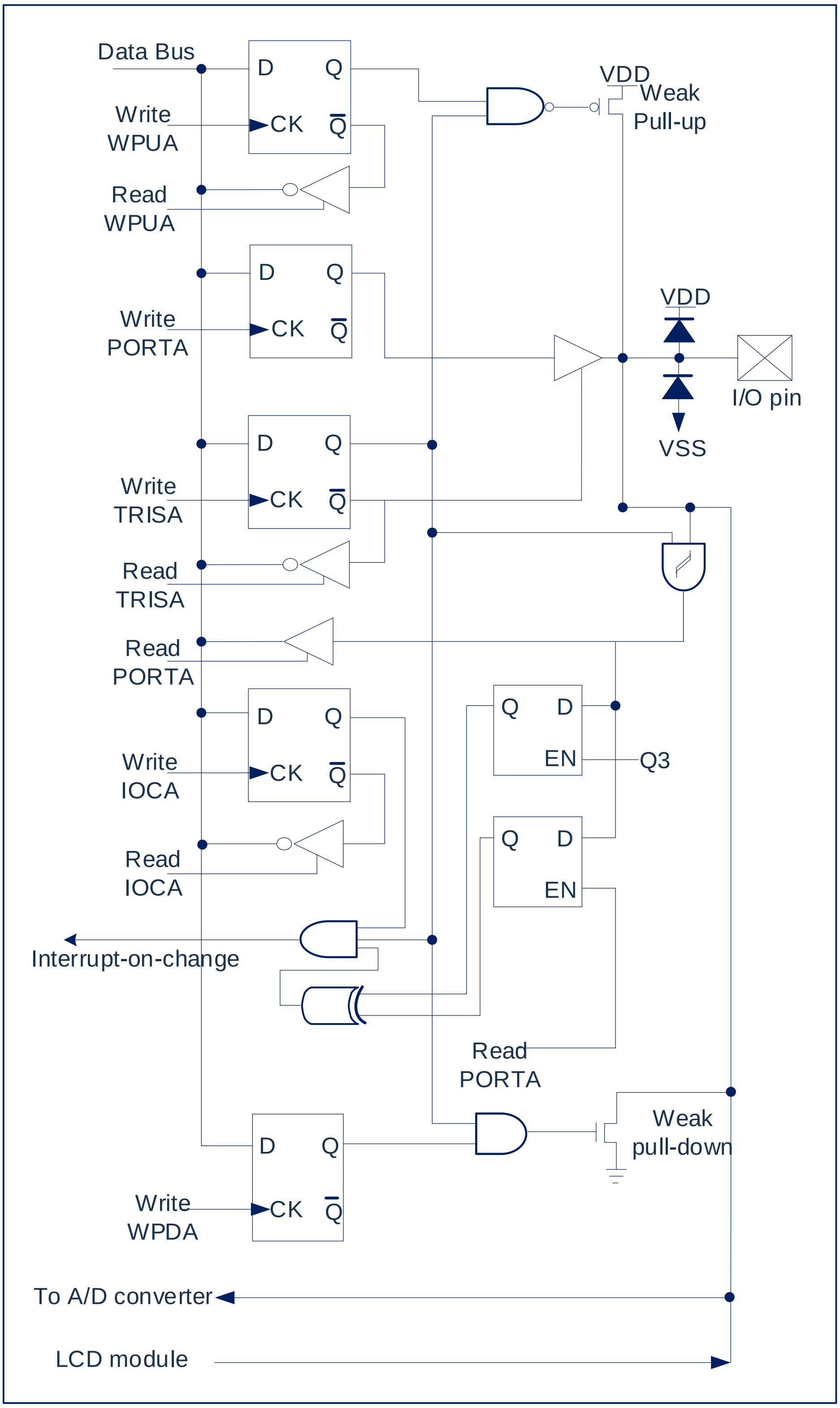


Fig 6-1: I/O port structure—PORTA

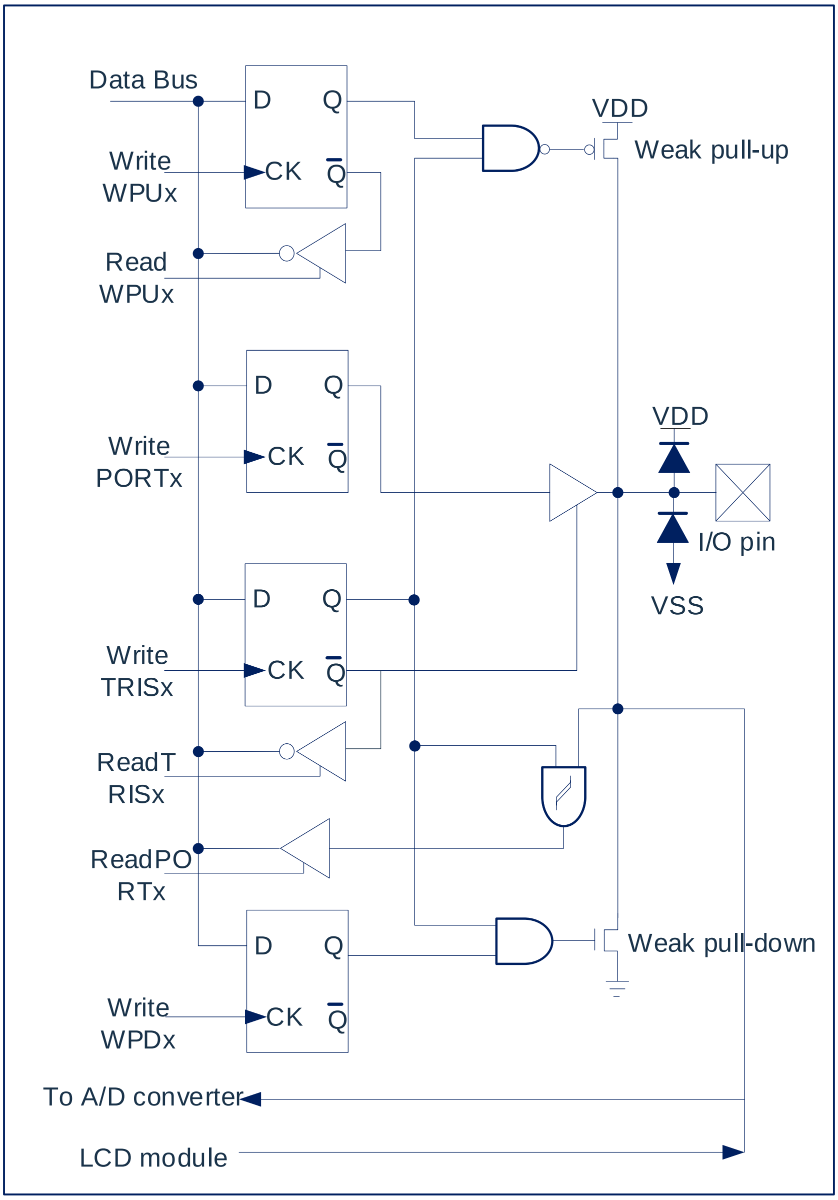


Fig 6-2: I/O port structure—PORTB/C

6.2 PORTA

6.2.1 PORTA Data and Direction Control

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

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

Registes related to PORTA ports are PORTA, TRISA, WPUA, WPDA, IOCA, ANSEL0 etc.

PORTA data register PORTA (05H)

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

Bit7~Bit0	PORTA<7:0>:	PORTAI/O 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	TRISA6	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
reset value	1	1	1	1	1	1	1	1

Bit7~Bit0	TRISA<7:0>:	PORTA three-state control bit;
	1=	The PORTA pin is configured as an input (three-state);
	0=	PORTA pin set to be output

Example: procedure for PORTA

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

6.2.2 PORTA Pull Up Resistance

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

PORTA pull up resistance register WPUA(07H)

07H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUA	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

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

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

6.2.3 PORTA Pull Down Resistance

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

PORTA pull down resistance register WPDA (97H)

97H	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 output, weak pull-down will be automatically disabled.

6.2.4 PORTA Analog Control Selection

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

PORTA analog selection register ANSEL0(110H)

110H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL0	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

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

6.2.5 PORTA Level Chage Interrupt

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

For the pin that has allowed level change interrupt, compare the value on the pin with the old value latched when PORTA was read last time. Perform a logical OR operation with the output "mismatch" of the last read operation to set the PORTA level change interrupt flag (RACIF) in the PIR2 register as 1.

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

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

-Clear the flag bit RACIF.

The mismatch status will continuously set the RACIF flag bit as 1. Reading or writing PORTA will end the mismatch state and allow the RACIF flag to be cleared. The latch will keep the last read value from the undervoltage reset. After reset, if the mismatch still exists, the RACIF 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 RACIF interrupt flag bit will not be set as 1. In addition, since reading or writing to a port affects all bits of the port, special care must be taken when using multiple pins in interrupt-on-change mode. When dealing with the level change of one pin, you may not notice the level change on the other pin.

PORTA level change interrupt register IOCA(95H)

95H	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>	Control bit of level change interrupt of PORTA
	1=	Enable level change interrupt
	0=	Disable level change interrupt

6.3 PORTB

6.3.1 PORTB Data and Direction

PORTB is a 8Bit wide bi-directional port. The corresponding data direction register is TRISB. Set a bit in TRISB to 1 (=1) to make the corresponding PORTBpin as the input pin. Clearing a bit in TRISB (=0) will make the corresponding PORTB pin as the output pin.

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

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

PORTB data register PORTB (06H)

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

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

PORTB direction register TRISB (86H)

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

Bit7~Bit0 TRISB<7:0>: PORTB tri-state control bit
1= PORTB pin configured as input (tri-state)
0= PORTB pin configured as output

example: PORTB port procedure

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

6.3.2 PORTB Pull up Resistance

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

PORTB pull up resistance register WPUB (08H)

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

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

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

6.3.3 PORTB Pull Down Resistance

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

PORTB pull down resistance register WPDB(87H)

87H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPDB	WPDB7	WPDB6	WPDB5	WPDB4	WPDB3	WPDB2	WPDB1	WPDB0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

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

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

6.3.4 PORTB Analog Selection Control

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

PORTB analog selection register ANSEL1(111H)

111H	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>: Analog selection bits, select the analog or digital functions of pin AN<15:8>.
1= Pin set as analog input. The pins are assigned as analog inputs.
0= Digital I/O. Assign ports or special functions.

6.4 PORTC

6.4.1 PORTC Data and Direction

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

Reading the PORTC register reads the pin status and writing to the register will write the port latch. All write operations are read-modify-write operations. Therefore, writing a port means reading the pin level of the port first, modifying the read value, and then writing the modified value to the port data latch. 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 TRISC register remain set. I/O pins configured as analog inputs always read 0.

The registers related to the PORTC port include PORTC, TRISC, WPUC, WPDC, ANSEL2, etc.

PORTC data register PORTC (92H)

92H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTC	-	-	-	-	-	-	RC1	RC0
R/W	-	-	-	-	-	-	R/W	R/W
Reset value	-	-	-	-	-	-	X	X

Bit7~Bit2 Not Used.

Bit1~Bit0 PORTC<1:0> PORTC I/O pin bit.

1= Port pin level $>V_{IH}$;

0= Port pin level $<V_{IL}$.

PORTC direction register TRISC (93H)

93H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISC	-	-	-	-	-	-	TRISC1	TRISC0
R/W	-	-	-	-	-	-	R/W	R/W
Reset value	-	-	-	-	-	-	1	1

Bit7~Bit2 Not Used.

Bit1~Bit0 TRISC<1:0>: Control bit of PORTC tri-state

1= PORTC pin configured as input (tri-state).

0= PORTC pin configured as output.

Example: procedure for PORTC

```
CLR     PORTC         ;clear data register
LDIA    B'00000001'   ; set PORTC<0> as intput, PORTC<1> as output,
LD      TRISC,A
```

6.4.2 PORTC pull up resistance

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

PORTC pull up resistance register WPUC (99H)

99H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUC	-	-	-	-	-	-	WPUC1	WPUC0
R/W	-	-	-	-	-	-	R/W	R/W
Reset value	-	-	-	-	-	-	0	0

Bit7~Bit2	Not Used.	
Bit1~Bit0	TRISC<1:0>:	Weak pull up register bit
	1=	Enable pull up
	0=	Disable pull up

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

6.4.3 PORTC Pull Down Resistance

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

PORTC pull down resistance register WPDC (98H)

98H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPDC	-	-	-	-	-	-	WPDC1	WPDC0
R/W	-	-	-	-	-	-	R/W	R/W
Reset value	-	-	-	-	-	-	0	0

Bit7~Bit2	Not used.	
Bit1~Bit0	WPDC<1:0>:	Weak pull-down register bit
	1=	Enable pull down
	0=	Disable pull down

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

6.4.4 PORTC Analog Selection Control

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

PORTC analog selection register ANSEL2(112H)

112H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL2	-	-	-	-	-	-	ANS17	ANS16
R/W	-	-	-	-	-	-	R/W	R/W
Reset value	-	-	-	-	-	-	0	0

Bit7~Bit2 Not Used.

Bit1~Bit0 ANS<17:16>: Analog selection bits, select the analog or digital functions of pin AN<17:16>.

1= Pin set as analog input. The pins are assigned as analog inputs.

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

6.5 I/O Usage

6.5.1 Write I/O port

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

Example: write I/O port program

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

6.5.2 Read I/O Port

Example: write I/O port program

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

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

6.6 Precautions for I/O Port Usage

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

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

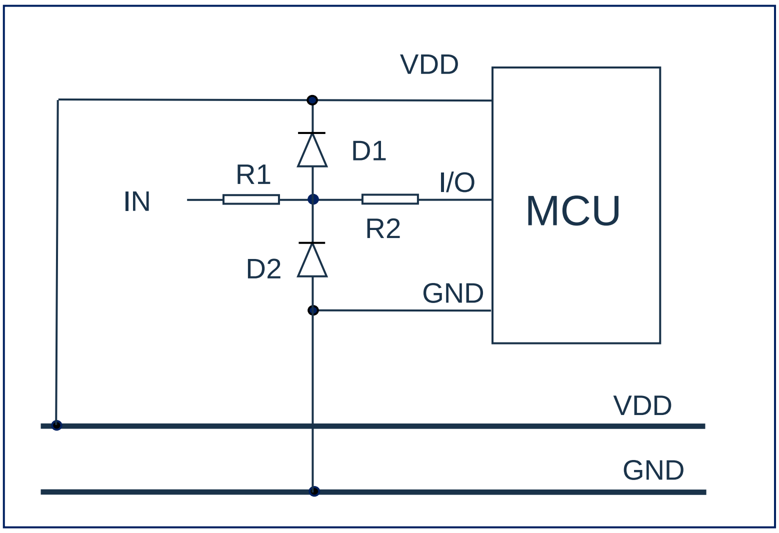


Fig 6-3: The input voltage is not within the specified range

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

7. Interrupt

7.1 Interrupt General

The chip has the following interrupt source:

- TIMER0 overflow interrupt
- TIMER1 overflow interrupt
- TIMER2 match interrupt
- INT interrupt
- PORTA level change interrupt
- A/D interrupt
- PWM interrupt
- LVD interrupt
- USART receive/transmit interrupt
- Program EEPROM write interrupt

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

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

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

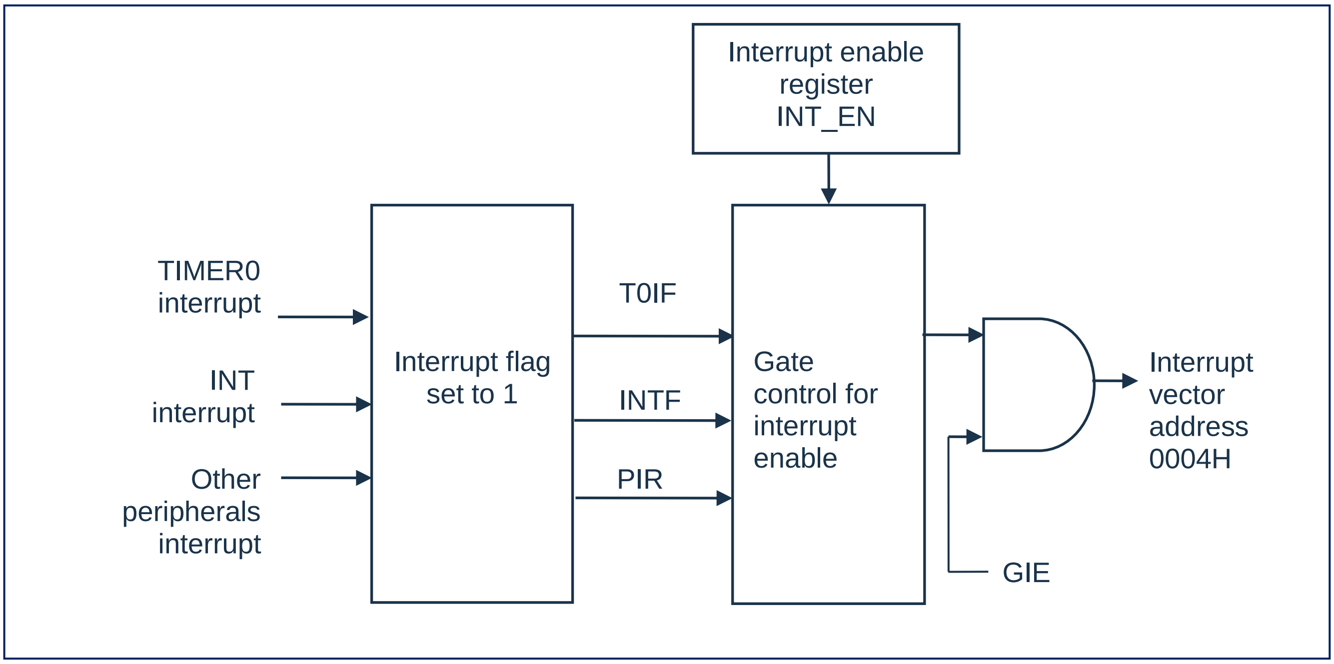


Fig 7-1: interrupt theory

7.2 Interruptcontrol Register

7.2.1 Interrupt Control Register

The interrupt control register INTCON is a readable and writable register, including the permission and flag bits for TMR0 register overflow and external interruption.

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

Interrupt control register INTCON（0BH）

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

Bit7	GIE:	Global interrupt enable bit;
	1=	Enable all unshielded interrupt;
	0=	Disable all interrupt
Bit6	PEIE:	Peripherals interrupt enable bit;
	1=	Enable all unshielded peripherals interrupt;
	0=	Disable all peripherals interrupt.
Bit5	T0IE:	TIMER0 overflow interrupt enable bit;
	1=	Enable TIMER0 interrupt;
	0=	Disable TIMER0 interrupt
Bit4	INTE:	INT external interrupt enable bit;
	1=	Enable INT external interrupt;
	0=	Disable INT external interrupt
Bit3	Not Used.	
Bit2	T0IF:	TIMER0 overflow interrupt enable bit (1);
	1=	TMR0 register overflow already (must clear through software);
	0=	TMR0 register not overflow
Bit1	INTF:	INT external interrupt flag bit;
	1=	INT external interrupt happens (must clear through software);
	0=	INT external interrupt not happen
Bit0	Not Used.	

Note: The T0IF bit is set as 1when TMR0 rolls over to 0. Reset will not change TMR0 and should be initialized before clearing the T0IF bit.

7.2.2 Peripherals Interrupt Enable Register

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

Peripherals interrupt enable register PIE1(106H)

106H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE1	-	ADIE	RCIE	TXIE	-	PWMIE	TMR2IE	TMR1IE
R/W	-	R/W	R/W	R/W	-	R/W	R/W	R/W
Reset value	-	0	0	0	-	0	0	0

Bit7 Not used.
Bit6 ADIE: A/D converter (ADC)interrupt enable bit;
1= enable ADC interrupt;
0= disable ADC interrupt
Bit5 RCIE: USART receive interrupt enable bit;
1= enable USART receive interrupt;
0= disable USART receive interrupt.
Bit4 TXIE: USART transmit interrupt enable bit;
1= enable USART transmit interrupt;
0= disable USART transmit interrupt.
Bit3 Not used.
Bit2 PWMIE: PWM interrupt enable bit;
1= enable PWM interrupt;
0= disable PWM interrupt.
Bit1 TMR2IE: TIMER2 and PR2 match interrupt enable bit;
1= enable TMR2 and PR2 match interrupt;
0= disable TMR2 and PR2 match interrupt.
Bit0 TMR1IE: TIMER1 overflow interrupt enable bit;
1= enable TIMER1 overflow interrupt;
0= disable TIMER1 overflow interrupt.

Peripherals interrupt enable register PIE2 (108H)

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

Bit7~Bit5 Not used.
Bit4 EEIE: Program EEPROM write operation interrupt enable bit;
1= Enable program EEPROM write operationinterrupt;
0= Disable program EEPROM write operation interrupt.
Bit3~Bit2 Not used.
Bit1 RACIE: PORTA interrupt-on-change enable bit
1= enable PORTA level change interrupt;
0= disable PORTA level change interrupt.
Bit0 LVDIE: LVDenable bit;
1= Enable LVD interrupt;
0= Disable LVD interrupt.

7.2.3 Peripherals Interrupt Request Register

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

Peripherals interrupt request register PIR1(105H)

105H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIR1	-	ADIF	RCIF	TXIF	-	PWMIF	TMR2IF	TMR1IF
R/W	-	R/W	R	R	-	R/W	R/W	R/W
Reset value	-	0	0	0	-	0	0	0

Bit7 Not used.
Bit6 ADIF: A/D converter interrupt flag bit;
1= A/D conversion complete (must clear through software);
0= A/D conversion not complete or not start.
Bit5 RCIF: USART0 receive interrupt flag bit;
1= USART0 receive buffer full (clear through reading RCREG);
0= USART0 receive buffer empty.
Bit4 TXIF: USART0 transmit interrupt flag bit;
1= USART0 transmit buffer empty (clear through TXREG);
0= USART0 transmit buffer full.
Bit3 Not used.
Bit2 PWMIF: PWM interrupt flag bit.
1= PWM interrupt occurred (must clear through software)
0= No PWM interrupt occurred
Bit1 TMR2IF: TIMER2 and PR2 match interrupt flag bit.
1= TIMER2 and PR2 match heppens (must clear through software);
0= TIMER2 and PR2 not match.
Bit0 TMR1IF: TIMER1 overflow interrupt flag bit.
1= TMR1 register overflow (must clear through software);
0= TMR1 register not overflow.

Peripherals interrupt request register PIR2(107H)

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

Bit7~Bit5 Not used.
Bit4 EEIF: EE write operation interrupt flag bit;
1= The write operation is complete (must be cleared by software);
0= The write operation has not been completed or has not been started yet.
Bit3~ Bit2 Not used.
Bit1 RACIF: PORTA interrupt-on-change flag.
1= Generate PORTA interrupt-on-change flag bit (must be cleared by software);
0= The PORTA interrupt-on-change flag is not generated.
Bit0 LVDIF: LVD interrupt flag
1= The power supply voltage is lower than the voltage point set by LVD (must be cleared by software);
0= The power supply voltage is higher than the voltage point set by the LVD.

7.3 Protection Methods for Interrupt

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

Example: Stack protection for ACC and STATUS

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

7.4 Interrupt Priority and Multi-interrupt Nesting

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

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

8. TIMER0

8.1 TIMER0 General

TIMER0 is composed of the following functions:

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

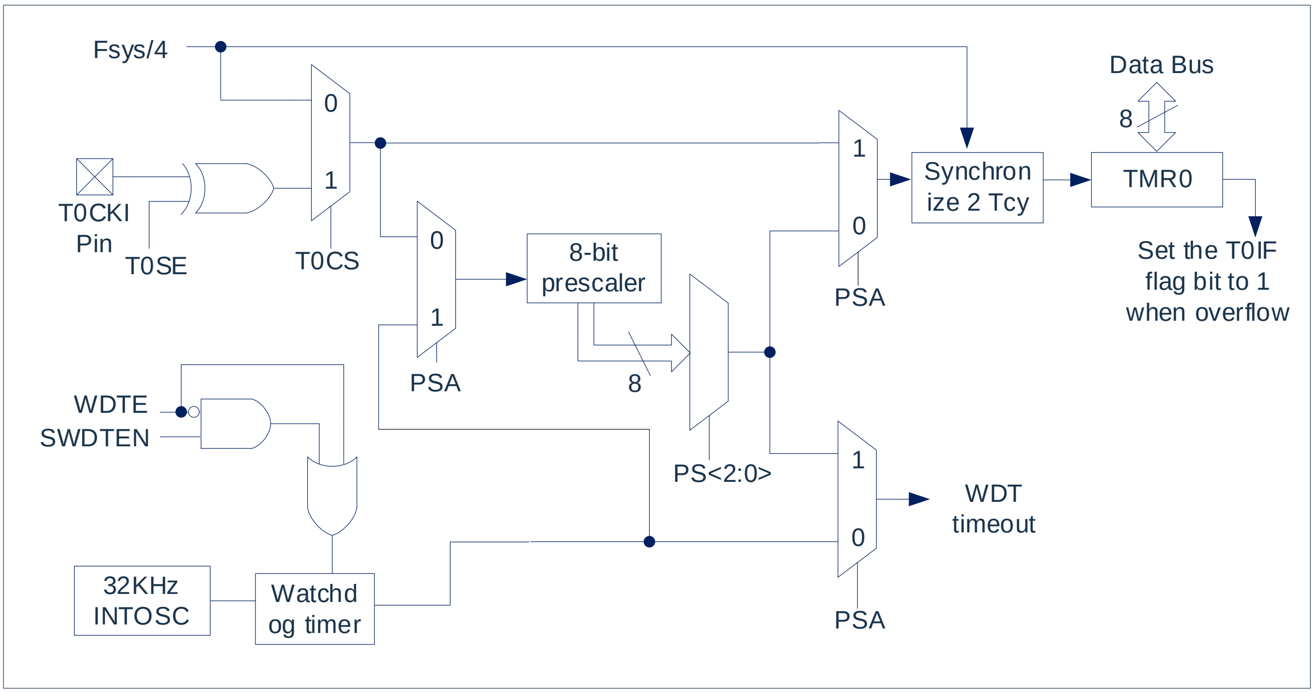


Fig 8-1: TIMER0/WDT mod structure

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

8.2 Working Principle for TIMER0

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

8.2.1 8-bit Timer Mode

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

8.2.2 8-bit Counter Mode

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

8.2.3 Software Programmable Pre-scaler

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

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

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

8.2.4 Switch Prescaler Between TIMER0 and WDT Module

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

Modify pre-scaler (TMR0-WDT)

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

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

Modify pre-scaler (WDT-TMR0)

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

8.2.5 TIMER0 Interrupt

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

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

8.3 TIMER0 Related Register

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

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

8-bit timer/counter TMR0 (01H)

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

OPTION_REG register (81H)

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

Bit7 Not Used.
Bit6 INTEDG: Interrupt edge selection bit.
1= The rising edge of the INT pin triggers interrupt.
0= The falling edge of the INT pin triggers interrupt.
Bit5 T0CS: TMR0 clock source selection bit.
1= Transition edge of T0CKI pin.
0= Interna linstruction period clock (F_{SYS}/4).
Bit4 T0SE: TIMER0 clock source edge selection bit.
1= Increment when the T0CKI pin signal transitions from high to low.
0= Increment when the T0CKI pin signal transitions from low to high.
Bit3 PSA: pre-scaler allocation bit.
1= pre-scaler allocated to WDT.
0= pre-scaler allocated toTIMER0 mod.
Bit2~Bit0 PS2~PS0: Pre-allocated parameter configuration bits.

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

9. TIMER1

9.1 TIMER1 General

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

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

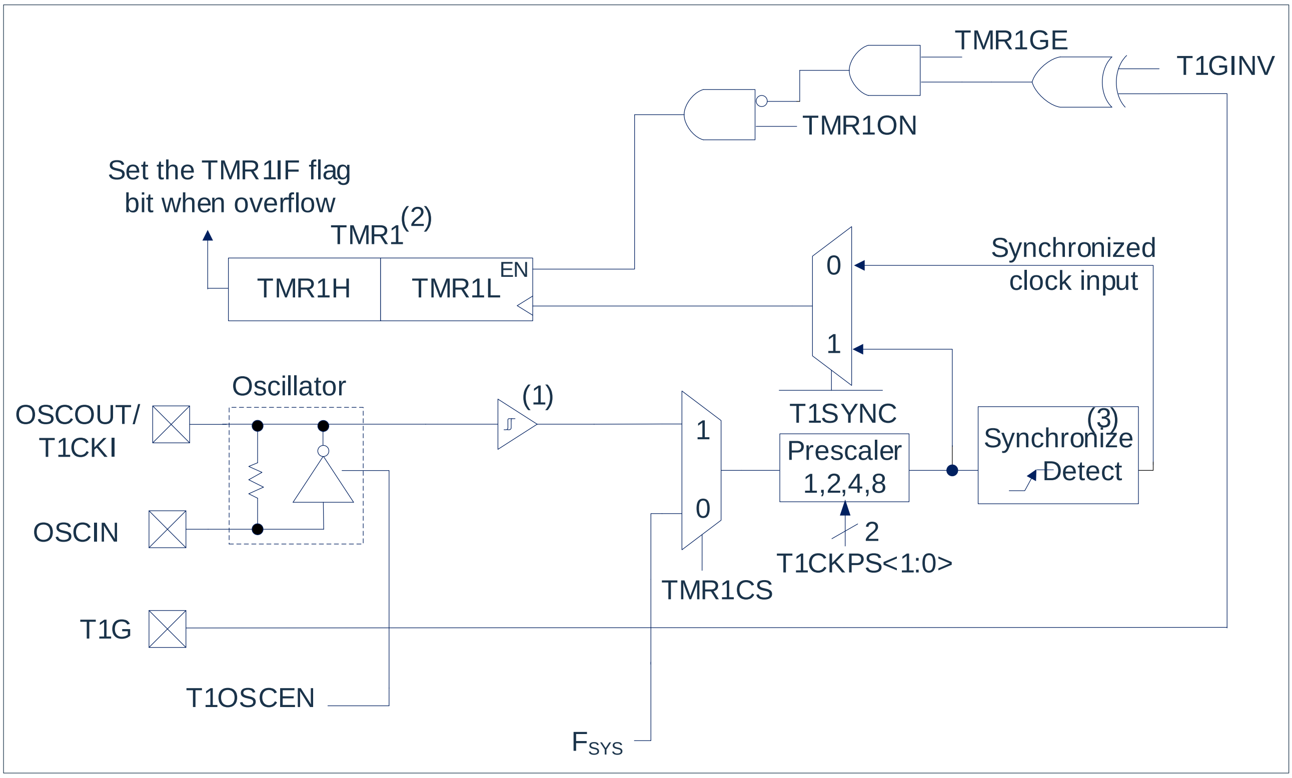


Fig 9-1: TIMER1 structure

Note:

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

9.2 Working Principle for TIMER1

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

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

9.3 Clock Source Selection

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

Clock source	TMR1CS
F_{SYS}	0
T1CKI pin	1

9.3.1 Internal Clock Source

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

9.3.2 External Clock Source

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

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

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

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

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

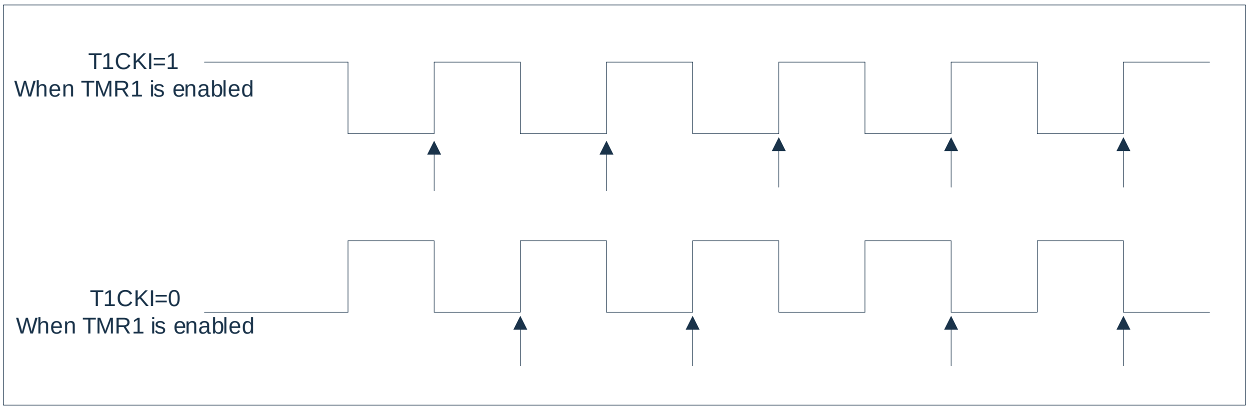


Fig 9-2: incremental edge of TIMER1

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

9.4 TIMER1 Pre-scaler

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

9.5 TIMER1 Oscillator

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

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

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

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

9.6 TIMER1 Working Principle Under Aasynchronous Counter Mode

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

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

9.6.1 Read and Write Operations to TIMER1 In Asynchronous Counter Mode

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

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

9.7 TIMER1 Gate Control

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

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

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

9.8 TIMER1 Interrupt

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

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

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

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

9.9 TIMER1 Working Principle During Sleep

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

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

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

9.10 TIMER1 Control Register

TIMER1 control register T1CON (10EH)

10EH	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: If TMR1ON=1:
Bit5~Bit4	T1CKPS<1:0>:	TIMER1 input clock frequency ratio selection bit;
	11=	1:8;
	10=	1:4;
	01=	1:2;
	00=	1:1.
Bit3	T1OSCEN:	LP oscillator enable controlbit;
	1=	Enable LP oscillatoras the clock source of TIMER1;
	0=	Disable LP oscillator.
Bit2	T1SYNC:	TIMER1 external clock input synchronous control bit. TMR1CS=1: TMR1CS=1: TMR1CS=0: ignore this bit, TIMER1 uses internal clock.
Bit1	TMR1CS:	TIMER1 clock source selection bit;
	1=	From LP oscillator clock source or clock source from T1CKI pin (rising edge trigger);
	0=	Internal clock source F_{SYS}.
Bit0	TMR1ON:	TIMER1enable bit;
	1=	Enable TIMER1;
	0=	Disable TIMER1.

10. TIMER2

10.1 TIMER2 General

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

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

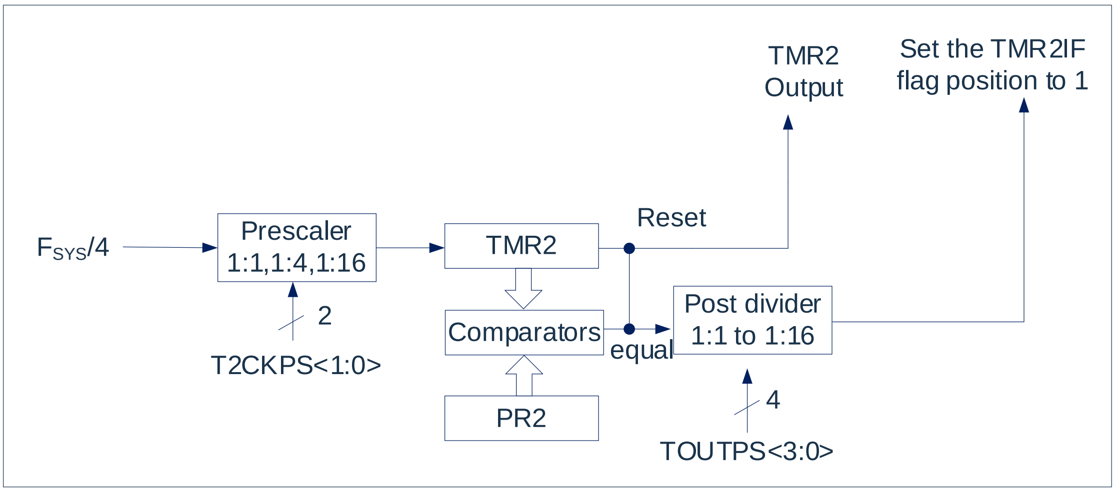


Fig 10-1: TIMER2 structure

10.2 Working Principle of TIMER2

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

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

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

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

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

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

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

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

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

Note: Writing T2CON will not clear TMR2. When TMR2ON=0, TMR2 will be cleared; you need to set TMR2ON to 1 to write to TMR2.

10.3 TIMER2 related register

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

TIMER2 data register TMR2 (11H)

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

TIMER2 Period registerPR2 (91H)

91H	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	0	0	0	0	0	0	0	0

TIMER2 control register T2CON (12H)

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

Bit7 Not used, read 0.

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

Bit2 TMR2ON: TIMER2 enable bit;
- 1= Enable TIMER2;
- 0= Disable TIMER2.

Bit1~Bit0 T2CKPS<1:0>: TIMER2 clock frequency division ratio selection bit;
- 00= 1;
- 01= 4;
- 1x= 16.

11. Analog to Digital Conversion (ADC)

11.1 ADC general

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

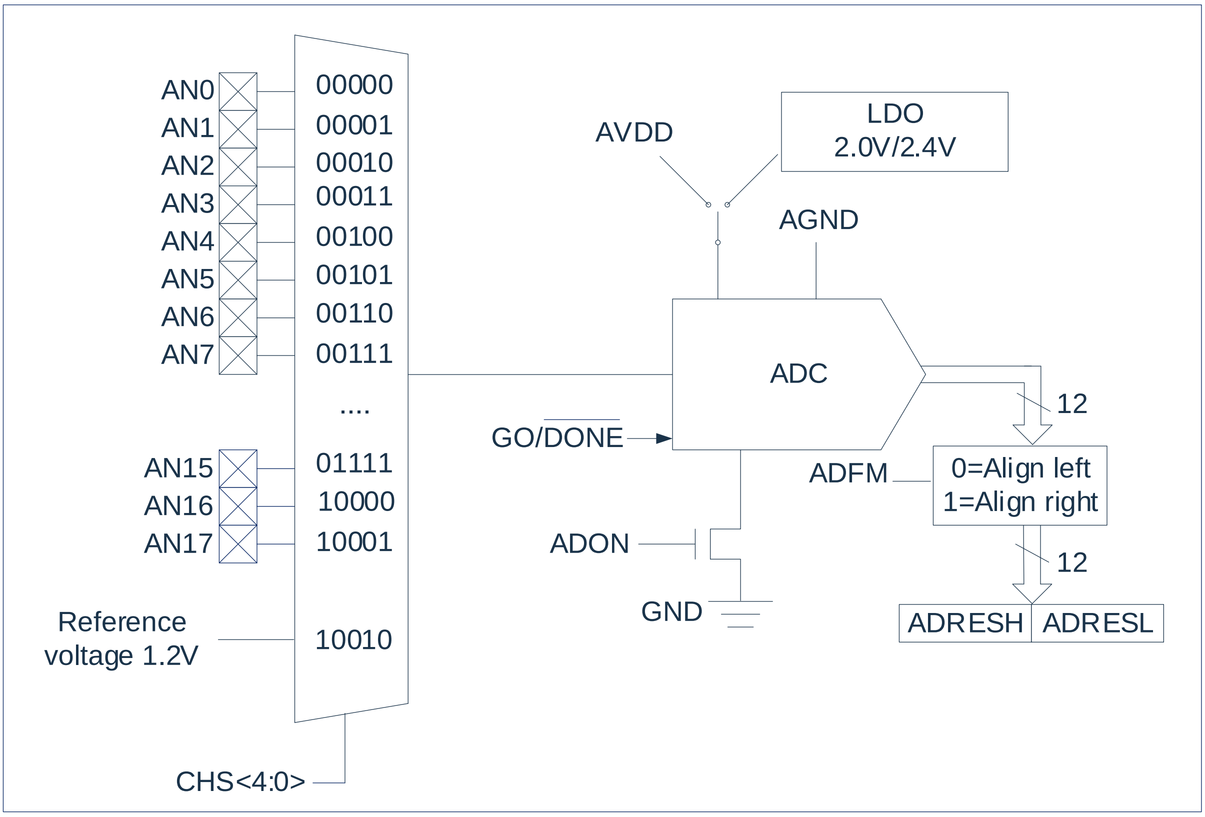


Fig 11-1: ADC structure

11.2 ADC configuration

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

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

11.2.1 Port configuration

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

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

11.2.2 Channel selection

The CHS bit of the ADCON0/ ADCON1register determines which channel is connected to the sample and hold circuit.

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

11.2.3 ADC internal reference voltage

The chip has a built-in 1.2V reference voltage. To detect the reference voltage, set the CHS[4:0] bits to 10010.

11.2.4 ADC reference voltage

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

When the internal reference voltage is selected, the conversion clock frequency division needs to select $F_{SYS}/32$ or slower frequency division.

11.2.5 Converter clock

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

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

The time to complete one-bit conversion is defined as TAD. A complete 12-bit conversion requires 16 TAD periods.

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

The relationship between ADC clock period (TAD) and device operating frequency (VDD=5.0V)

ADC clock selection		AD conversion time	
ADC clock source	ADCS<1:0>	F_{SYS} = 16MHz	F_{SYS} = 8MHz
$F_{SYS}/8$	00	8μs	16μs
$F_{SYS}/16$	01	16μs	32μs
$F_{SYS}/32$	10	32μs	64μs
$F_{SYS}/128$	11	128μs	256μs

Note: It is recommended not to use the values in the shaded cells.

11.2.6 ADC Interrupt

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

11.2.7 Output Formatting

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

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

11.3 ADC working principle

11.3.1 Start conversion

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

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

11.3.2 Complete conversion

When the conversion is complete, the ADC mod will:

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

11.3.3 Stop conversion

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

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

11.3.4 Working principle of ADC in sleep mode

ADC module cannot work in sleep mode.

11.3.5 A/D conversion procedure

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

1. port configuration:
   - Configure pin as input pin (see TRIS register).
2. configuration ADC mod:
   - Select the ADC reference voltage, if it is switched from VDD to the internal 2.0V/2.4V voltage, wait for at least 200us to start detecting AD;
   - Select ADC conversion clock;
   - Select ADC input channel;
   - Choose the format of the result;
   - Start the ADC mod.
3. configuration ADC interrupt (optional):
   - Clear ADC interrupt flag bit;
   - Allow ADC interrupt;
   - Allow peripherals interrupt;
   - Allow global interrupt.
4. Wait for the required acquisition time.
5. Set GO/$\overline{\text{DONE}}$ to 1 to start conversion.
6. Wait for the ADC conversion to end by one of the following methods:
   - Query GO/ $\overline{\text{DONE}}$ bit
   - Wait for ADC interrupt (allow interrupt).
7. Read ADC results.

5. Clear the ADC interrupt flag bit (if interrupt is allowed, this operation is required).

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

Example: AD conversion

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

11.4 ADC Related Register

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

AD control register ADCON0(9DH)

9DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit6	ADCS<1:0>:	A/D conversion clock selection bit.
	00=	FSYS/8
	01=	FSYS/16
	10=	FSYS/32
	11=	FSYS/128
Bit5~Bit2	CHS<3:0>:	The analog channel selection bit
	CHS<4:0>	CHS4在ADCON1 register
	00000=	AN0
	00001=	AN1
	00010=	AN2
	00011=	AN3
	…	…
	10000=	AN16
	10001=	AN17
	10010=	1.2V (fixed reference voltage)
	Other:	Reserve.
Bit1	GO/DONE:	A/D conversion status bit.
	1=	A/D conversion is in progress. Set this bit to 1 to start A/D conversion. When A/D conversion is completed, this bit is automatically cleared by hardware.
	0=	A/D conversion complete or not in progress.
Bit0	ADON:	ADC enable bit.
	1=	Enable ADC;
	0=	Disable ADC.

AD data register high bit ADCON1(9CH)

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

Bit7	ADFM:	A/D conversion result format selection bit
	1=	Right alignment
	0=	left alignment
Bit6	CHS4:	Channel selection bit
Bit5~Bit3	Not used, read 0.	
Bit2	LDOEN	ADC internal reference LDO enable bit
	1=	Enable, VREF input of ADC is LDO
	0=	Disable, VREF input of ADC is VDD
Bit1	Not used, read 0.	
Bit0	LDOSEL:	AD reference voltage selection bit
	1=	2.0V
	0=	2.4V

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

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

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

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

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

Bit7~Bit4 ADRES<3:0>: ADC result register bit.
The lower 4 bits of the 12-bit conversion result.
Bit3~Bit0 Not used.

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

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

Bit7~Bit2 Not used.

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

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

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

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

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

The 2-9 bits of the 12-bit conversion result.

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

12. LCD Driver Mod

The chip has a built-in LCD driver module. CMS79F11x can drive 1/2Bias or 1/3Bias LCD. After the LCD control bit is enabled, the chip needs to control the output to drive the LCD.

12.1 LCD Function Enable

Set the 7th bit LCDEN of LCDCON (113H) to 1 to allow LCD drive function, and enable the corresponding pin LCD function by controlling CSxEN;

Set LCDEN to 0 to turn off the LCD module.

12.2 LCD Related Register

The LCD drive function related registers are: LCDCON, CSSEL0, CSSEL1, CSSEL2, CSEN0, CSEN1, CSEN2.

LCD control registerLCDCON(113H)

113H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDCON	LCDEN	FRAME	BIAS	----	----	----	LCD_ISLE[1:0]	
R/W	R/W	R/W	R/W	----	----	----	R/W	R/W
Reset value	0	0	0	----	----	----	0	0

Bit7	LCDEN:	LCD mod enable bit;
	0=	Disable LCD mod;
	1=	Enable LCD mod.
Bit6	FRAME:	COM/SEG output FRAME selection. (Only valid for 1/3 Bias)
	0=	FRAME0; The COM signal output can be VDD, or VBIAS = (1/3) × VDD. The program needs to write 1 to the PORT value of the corresponding COM port and 0 to the TRIS value. SEG signal output can be VSS, or VBIAS = (2/3) × VDD. The program needs to write 0 for the PORT value of the corresponding SEG port and 0 for the TRIS value.
	1=	FRAME1. The COM signal output can be VSS, or VBIAS =(2/3) × VDD. The program needs to write 0 to the PORT value of the corresponding COM port and 0 to the TRIS value. SEG signal output can be VDD, or VBIAS = (1/3) × VDD The program needs to write 1 for the PORT value of the corresponding SEG port and 0 for the TRIS value.
Bit5	BIAS:	Bias selection.
	0=	1/2 BIAS;
	1=	1/3 BAIS.
Bit4~Bit2	Not Used.	
Bit1~Bit0	LCD_ISLE [1:0]:	LCD output current selection bits:

	1/2 BIAS	1/3 BIAS
00=	100uA@5V;	10uA@5V;
01=	200uA@5V;	20A@5V;
10=	400uA@5V;	50uA@5V;
11=	800uA@5V.	100uA@5V.

LCD COM/SEG selection register CSSEL0 (corresponding to PORTA) (114H)

114H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CSSEL0	CS7SEL	CS6SEL	CS5SEL	CS4SEL	CS3SEL	CS2SEL	CS1SEL	CS0SEL
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 CS7SEL~CS0SEL: LCD COM/SEGx function selection bit (only valid for 1/3 Bias mode)
0= The corresponding pins of PORTA are COM functions;
1= The corresponding pin of PORTA is SEG function.

The corresponding relationship is as follows:

CS7SEL-> PORTA7
CS6SEL-> PORTA6
CS5SEL-> PORTA5
CS4SEL-> PORTA4
CS3SEL-> PORTA3
CS2SEL-> PORTA2
CS1SEL-> PORTA1
CS0SEL-> PORTA0

LCD COM/SEG selection register CSSEL1 (corresponding to PORTB) (115H)

115H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CSSEL1	CS15SEL	CS14SEL	CS13SEL	CS12SEL	CS11SEL	CS10SEL	CS9SEL	CS8SEL
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 CS15SEL~CS8SEL: LCD COM/SEGx function selection bit (only valid for 1/3 Bias mode)
0= The corresponding pins of PORTB are COM functions;
1= The corresponding pin of PORTB is SEG function.

The corresponding relationship is as follows:

CS15SEL-> PORTB7
CS14SEL-> PORTB6
CS13SEL-> PORTB5
CS12SEL-> PORTB4
CS11SEL-> PORTB3
CS10SEL-> PORTB2
CS9SEL-> PORTB1
CS8SEL-> PORTB0

LCD COM/SEG selection register CSSEL2 (corresponding to PORTC) (116H)

116H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CSSEL2	--	--	--	--	--	--	CS17SEL	CS16SEL
R/W	--	--	--	--	--	--	R/W	R/W
Reset value	--	--	--	--	--	--	0	0

Bit7~Bit2 Not Used.

Bit1~Bit0 CS17SEL~CS16SEL: LCD COM/SEGx function selection bit (only valid for 1/3 Bias mode)

0= The corresponding pins of PORTC are COM functions;

1= The corresponding pin of PORTC is SEG function.

The corresponding relationship is as follows:

CS17SEL-> PORTC1

CS16SEL-> PORTC0

LCD COM/SEG enable register CSEN0 (corresponding to PORTA) (11CH)

11CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CSEN0	CS7EN	CS6EN	CS5EN	CS4EN	CS3EN	CS2EN	CS1EN	CS0EN
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 CSEN7~CSEN0 LCD COM/SEGx enable bit, priority is higher than TRIS bit

0: Prohibit the corresponding LCD function of PORTA;

1: Enable the corresponding pins of PORTA as LCD COM or SEG functions.

Correspondence is the same as CSSEL0

LCD COM/SEG selection register CSEN1 (corresponding to PORTB) (11DH)

11DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CSEN1	CS15EN	CS14EN	CS13EN	CS12EN	CS11EN	CS10EN	CS9EN	CS8EN
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 CSEN15~CSEN8 LCD COM/SEGx enable bit, priority is higher than TRIS bit

0: Prohibit the corresponding LCD function of PORTB;

1: Enable the corresponding pins of PORTB as LCD COM or SEG functions.

Correspondence is the same as CSSEL1

LCD COM/SEG selection register CSEN2 (corresponding to PORTC) (11EH)

11EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CSEN2	--	--	--	--	--	--	CS17EN	CS16EN
R/W	--	--	--	--	--	--	R/W	R/W
Reset value	--	--	--	--	--	--	0	0

Bit7~Bit2 Not Used.

Bit1~Bit0 CSEN17~CSEN16 LCD COM/SEGx enable bit, priority is higher than TRIS bit

0: Prohibit the corresponding LCD function of PORTC;

1: Enable the corresponding pins of PORTC as LCD COM or SEG functions.

Correspondence is the same as CSSEL2

CSxEN = 1 This pin is LCD function, according to CSxSEL value and FRAME value output 1/3 or 2/3 voltage; =0 This pin is normal IO function, need TRIS and PORT register to output high and low level. CSxEN has a higher priority than TRIS.

12.3 LCD operating instructions

A complete LCD waveform cycle contains two frames, namely Frame 0 and Frame 1.

12.3.1 1/3 Bias register operation

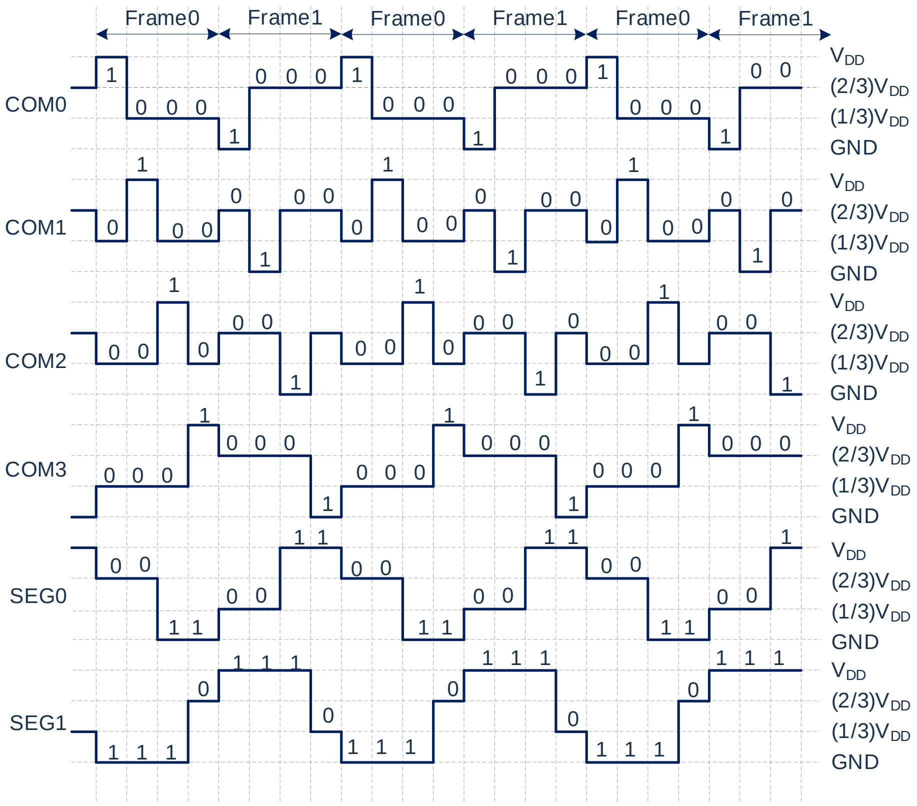
Register operations related to 1/3 Bias are shown in the following table:

Pin	Frame0	Frame1
COM	Pin output VDD: BIAS=1; FRAME=0; CSxEN=0; TRIS=0; PORT=1;	Pin output (2/3) * VDD: BIAS=1; FRAME=1; CSxEN=1, CSxSEL=0;
	Pin output (1/3) * VDD: BIAS=1; FRAME=0; CSxEN=1, CSxSEL=0;	Pin output GND: BIAS=1; FRAME=1; CSxEN=0; TRIS=0; PORT=0;
SEG	Pin output (2/3) * VDD BIAS=1; FRAME=0; CSxEN=1, CSxSEL=1;	Pin output VDD: BIAS=1; FRAME=1; CSxEN=0; TRIS=0; PORT=1;
	Pin output GND: BIAS=1; FRAME=0; CSxEN=0; TRIS=0; PORT=0;	Pin output (1/3) * VDD: BIAS=1; FRAME=1; CSxEN=1, CSxSEL=1;

The shaded part in the table represents the operation of the corresponding SEG/COM of the LCD that needs to be lit in the two frames.

12.3.2 1/3 Bias timing diagram

In the figure, "1" means to light up the LCD pixel.



12.3.3 1/2 Bias register operation

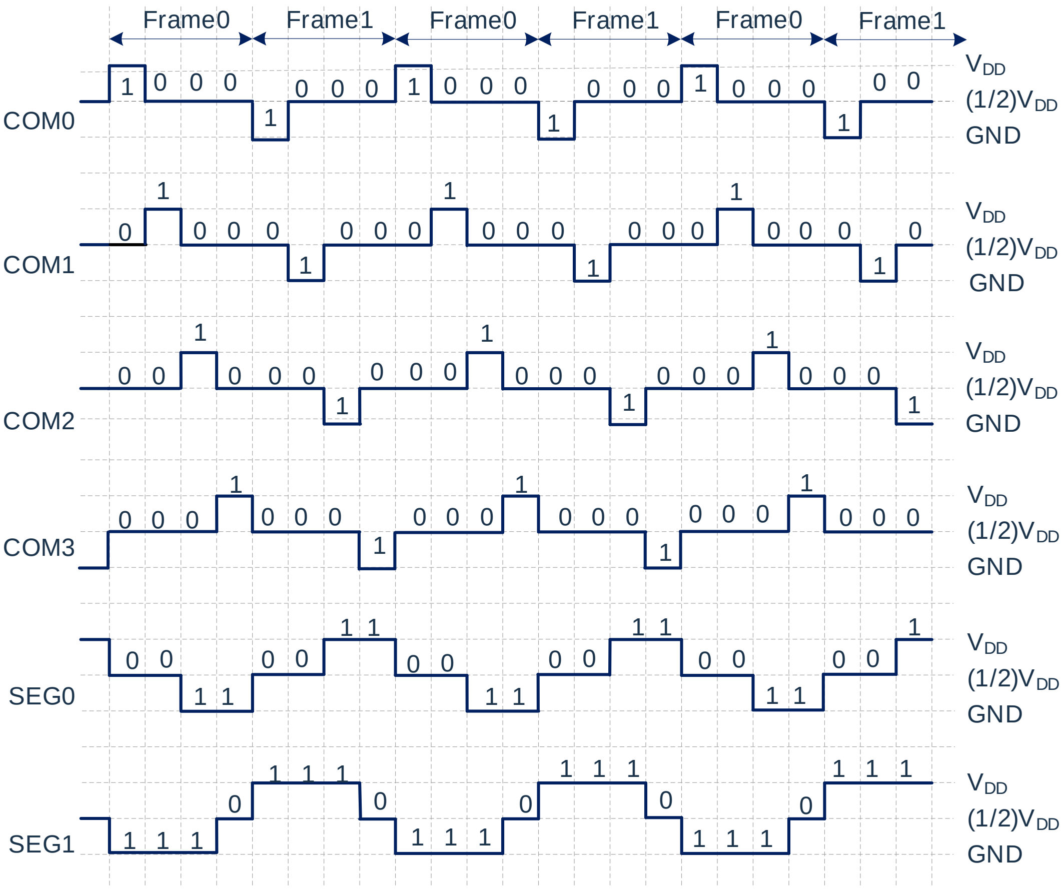
Register operations related to 1/2 Bias are shown in the following table:

Pin	Frame0	Frame1
COM	Pin output VDD: BIAS=0; CSxEN=0; TRIS=0; PORT=1;	Pin output (1/2) * VDD: BIAS=0; CSxEN=1;
	Pin output (1/2) * VDD: BIAS=0; CSxEN=1;	Pin output GND: BIAS=0; CSxEN=0; TRIS=0; PORT=0;
SEG	Pin output (1/2) * VDD: BIAS=0; CSxEN=1;	Pin output VDD: BIAS=0; CSxEN=0; TRIS=0; PORT=1;
	Pin output GND: BIAS=0; CSxEN=0; TRIS=0; PORT=0;	Pin output (1/2) * VDD: BIAS=0; CSxEN=1;

The shaded part in the table represents the operation of the corresponding SEG/COM of the LCD that needs to be lit in the two frames.

12.3.4 1/2 Bias timing diagram

In the figure, "1" means to light up the LCD pixel.



13. PWM Mode

The chip contains a 10-bit PWM module, which can be configured as 4 outputs with shared period and independent duty cycle + 1 independent output, or 2 sets of complementary outputs + 1 independent output.

The PWM output can be selected as RB0-RB4 or RA3-RA7 or RB0-RB3, RA5 through CONFIG. Among them, PWM0/PWM1, PWM2/PWM3 can be configured to have complementary forward and reverse outputs.

13.1 Pin configuration

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

13.2 Description of related registers

PWM control register PWMCON0(13H)

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

Bit7~Bit5	CLKDIV[2:0]:	PWM clock frequency division.
	111=	FOSC/128
	110=	FOSC/64
	101=	FOSC/32
	100=	FOSC/16
	011=	FOSC/8
	010=	FOSC/4
	001=	FOSC/2
	000=	FOSC/1
Bit4~Bit0	PWMxEN:	PWMx enable bit.
	1=	Enable PWMx.
	0=	Disabled PWMx.

PWM control register PWMCON1(14H)

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

Bit7~6 PWMIO_SEL[1:0]: PWM IO selection.
1X= PWM is allocated in group C, PWM0-RB0, PWM1-RB1, PWM2-RB2, PWM3-RB3, PWM4-RA5
01= PWM is allocated in groupB, PWM0-RB0, PWM1-RB1, PWM2-RB2, PWM3-RB3, PWM4-RB4
00= PWM is allocated in group A, PWM0-RA3, PWM1-RA4, PWM2-RA5, PWM3-RA6, PWM4-RA7

Bit5 PWM2DTEN: PWM2 dead zone enable bit.
1= Enable PWM2 dead zone function, PWM2 and PWM3 form a pair of complementary outputs.
0= Disable PWM2 dead zone function.

Bit4 PWM0DTEN: PWM0 dead zone enable bit.
1= Enable PWM0 dead zone function, PWM0 and PWM1 form a pair of complementary outputs.
0= Disable PWM0 dead zone function.

Bit3~Bit2 Not used.

Bit1~Bit0 DT_DIV[1:0] Frequency division of the dead time clock source.
11= FOSC/8
10= FOSC/4
01= FOSC/2
00= FOSC/1

PWM control register PWMCON2(1DH)

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

Bit7~Bit5 Not used.
Bit4~Bit0 PWMxDIR The PWM output inverts the control bit.
1= PWMx output is inverted.
0= PWMx is output normally.

PWM0~PWM3 Period low register PWMTL (15H)

15H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMTL	PWMT[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMT[7:0]: The low 8 bits of the period of PWM0~PWM3.

PWM4Period low registerPWM4TL(1EH)

1EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM4TL	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]: The low 8 bits of the PWM4 period.

PWM Period high register PWMTH (16H)

16H	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 Not used.
Bit5~Bit4 PWMD4[9:8]: PWM4 duty cycle is 2 high bits.
Bit3~Bit2 PWM4T[9:8]: The high 2 bits of the PWM4 period.
Bit1~Bit0 PWMT[9:8]: The high 2 bits of the period of PWM0~PWM3.

> Note: Writing into PWMD4[9:8] does not take effect immediately, it takes effect after writing into PWMD4L.

PWM0 duty cycle low register PWMD0L (17H)

17H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD0L	PWMD0[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD0[7:0]: The low 8 bits of the PWM0 duty cycle.

PWM1 duty cycle low register PWMD1L (18H)

18H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD1L	PWMD1[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD1[7:0]: The low 8 bits of the PWM1 duty cycle.

PWM2 duty cycle low register PWMD2L (19H).

19H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD2L	PWMD2[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD2[7:0]: The low 8 bits of the PWM2 duty cycle.

PWM3 duty cycle low register PWMD3L (1AH)

1AH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD3L	PWMD3[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD3[7:0]: The low 8 bits of the PWM3 duty cycle.

PWM4 duty cycle low 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]: The low 8 bits of the PWM4 duty cycle.

PWM0 and PWM1 duty cycle high register PWMD01H (1CH)

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

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

Note: Writing to PWMD1[9:8] does not take effect immediately, it takes effect after writing to PWMD1L.
Writing to PWMD0[9:8] does not take effect immediately, it takes effect after writing to PWMD0L.

PWM2 and PWM3 duty cycle high register PWMD23H (0EH)

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

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

Note: Writing to PWMD3[9:8] does not take effect immediately, it takes effect after writing to PWMD3L.
Writing to PWMD2[9:8] does not take effect immediately, it takes effect after writing to PWMD2L.

PWM0 and PWM1 Dead Time Register PWM01DT(0FH)

0FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM01DT	---	---	PWM01DT[5:0]					
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	---	0	0	0	0	0	0

Bit7~Bit6 Not used.
Bit5~Bit0 PWM01DT[5:0]: Dead time for PWM0 and PWM1.

PWM2 and PWM3 Dead Time Register PWM23DT (10H)

10H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM23DT	---	---	PWM23DT[5:0]					
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	---	0	0	0	0	0	0

Bit7~Bit6 Not used.
Bit5~Bit0 PWM23DT[5:0]: Dead time for PWM2 and PWM3.

13.3 PWM register write operation sequence

Since the 10-bit PWM duty cycle value is allocated in two registers, when modifying the duty cycle, the program always modifies the two registers one after the other. In order to ensure the correctness of the duty cycle value, the chip has a cache load function designed inside the chip. Operating the 10-bit duty cycle value must strictly follow the following sequence:

1）Write the high 2-bit value, at this time the high 2-bit value is only written into the internal buffer;

2）Write the low 8-bit value, at this time the complete 10-bit duty cycle value is latched.

13.4 PWM Period

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

Formula 1: PWM period:

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

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

When the PWM period counter is equal to PWMT, the following 5 events will occur in the next up-counting period:

- PWM period counte is cleared;
- PWMx pin is set to 1;
- PWM the new period value is latched;
- PWM the new duty cycle value is latched;
- Generate PWM interrupt flag bit.

13.5 PWM Duty Cycle

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

You can write the PWMDxL and PWMDxxH register at any time, but until the PWM period counter is equal to PWMT (that is, the end of the period), the value of the duty cycle is updated to the internal latch.

Formula 2: Pulse width calculation formula:

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

Formula 3: PWM duty cycle calculation formula:

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

Both the PWM period and the PWM duty cycle are double-buffered inside the chip. This double buffering structure is extremely important to avoid glitches during the PWM operation.

13.6 System Clock Frequency Changes

The PWM frequency is only related to the chip oscillation clock, and any change in the system clock frequency will not affect the PWM frequency.

13.7 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.

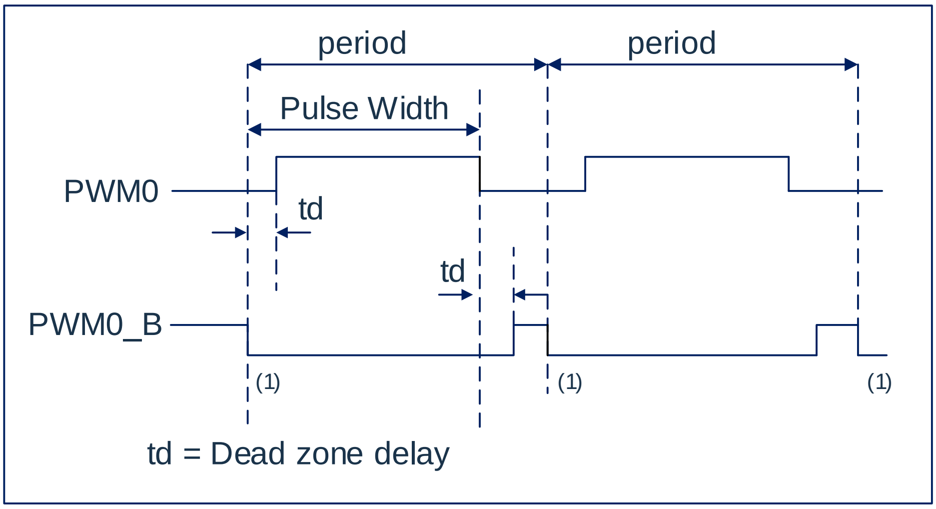


Fig 13-1: Example of PWM dead zone delay output

The dead time calculation formula is:

td=(PWMxxDT[5:0]+1)*TOSC*(DT_DIV *Frequency value*)

13.8 PWM settings

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

1. Set the IO_SEL control bit to select the PWM output IO port.
2. Make it an input pin by setting the corresponding TRIS bit to 1.
3. Set the PWM period by loading the PWMTH, PWMTL registers.
4. Set the PWM duty cycle by loading the PWMDxxH, PWMDxL registers.
5. If you need to use the complementary output mode, you need to set the PWMCON1[6:5] bits, and load the PWMxxDT register to set the dead time.
6. Clear the PWMIF flag bit
7. Set the PWMCN0[4:0] bits to enable the corresponding PWM output.
8. After the new PWM period starts, enable PWM output:
   - Wait for the PWMIF bit to be 1;
   - Enable the TRIS pin output driver by clearing the corresponding PWM bit.

14. Universal Synchronous/Asynchronous Transmitter (USART)

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

The USART mod includes the following functions:

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

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

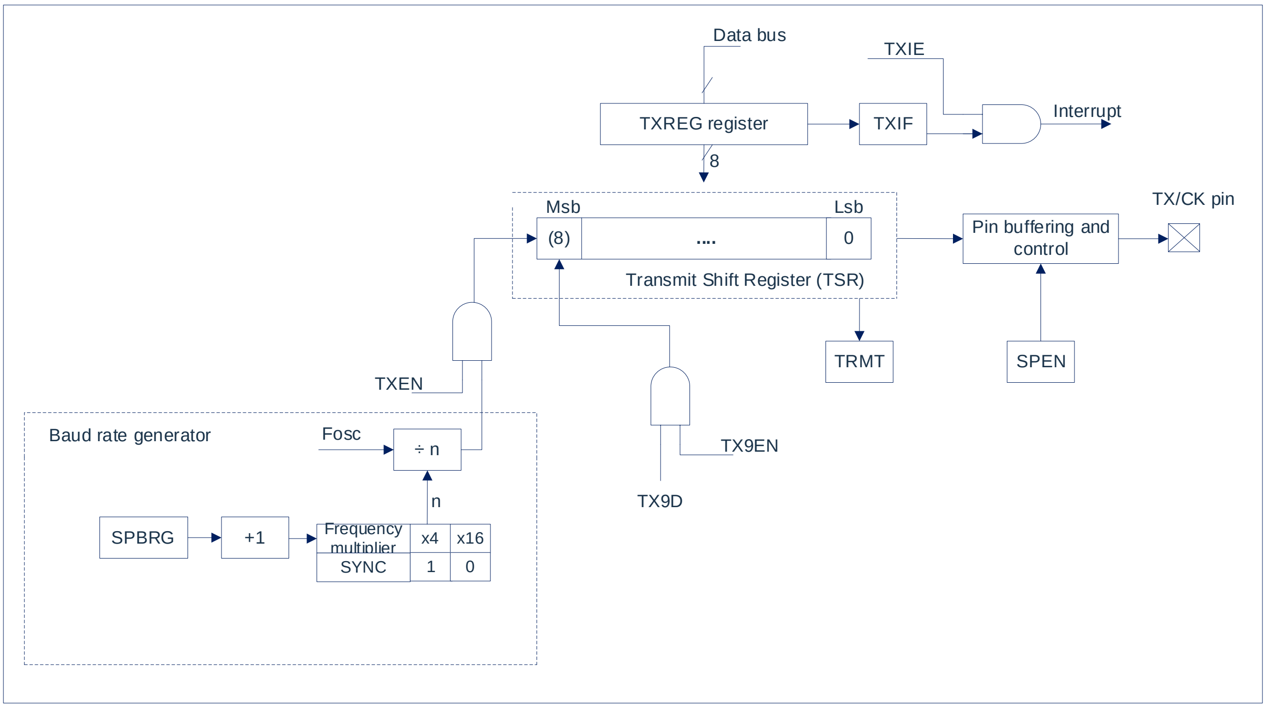


Fig 14-1: USART transmit block diagram

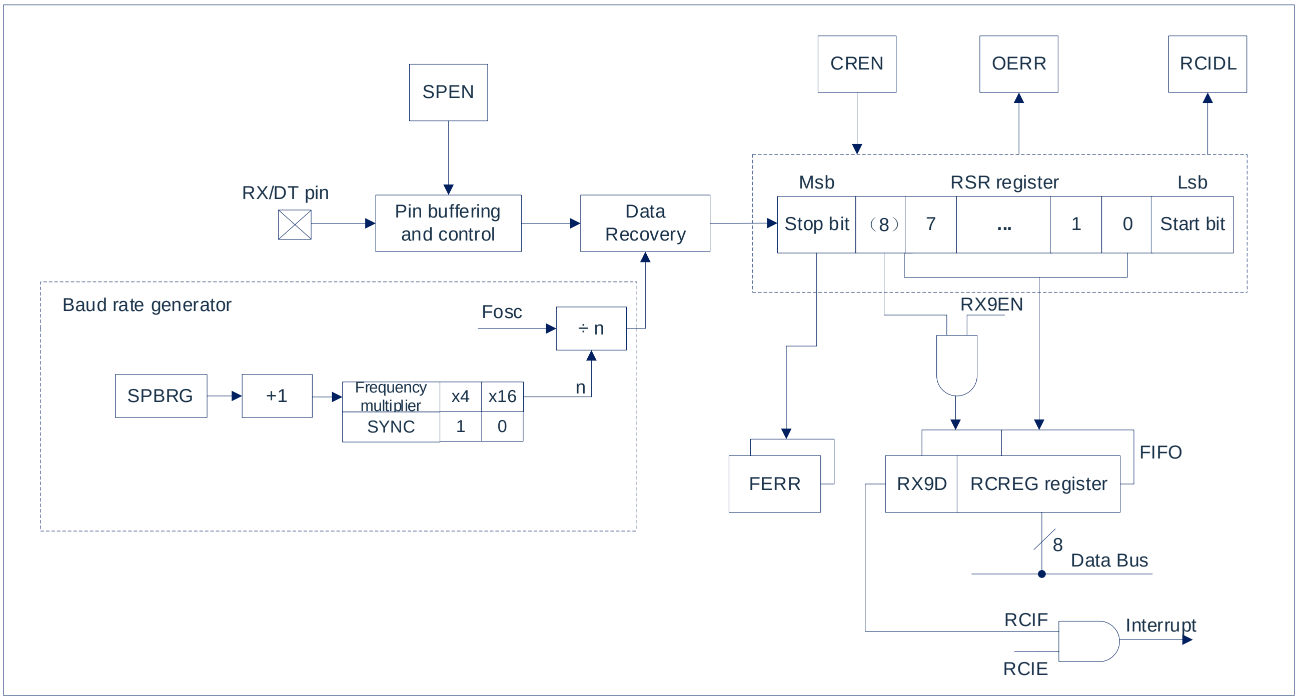


Fig 14-2: USART receive block diagram

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

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

14.1 USART Asynchronous Mode

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

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

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

14.1.1 USART Asynchronous Generator

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

14.1.1.1 Enable Transmit

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

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

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

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

Note:

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

14.1.1.2 Transmit Data

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

14.1.1.3 Transmit Interrupt

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 TXIFflag bit is not cleared immediately. TXIF is cleared at the second instructions period after writing the instructions. Querying TXIF immediately after writing TXREG will return an invalid result. TXIF is a read-only bit and cannot Set or cleared by software.

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

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

14.1.1.4 TSR Status

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

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

14.1.1.5 Transmit 9-bit Character

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

14.1.1.6 Configure Asynchronous Transmit

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

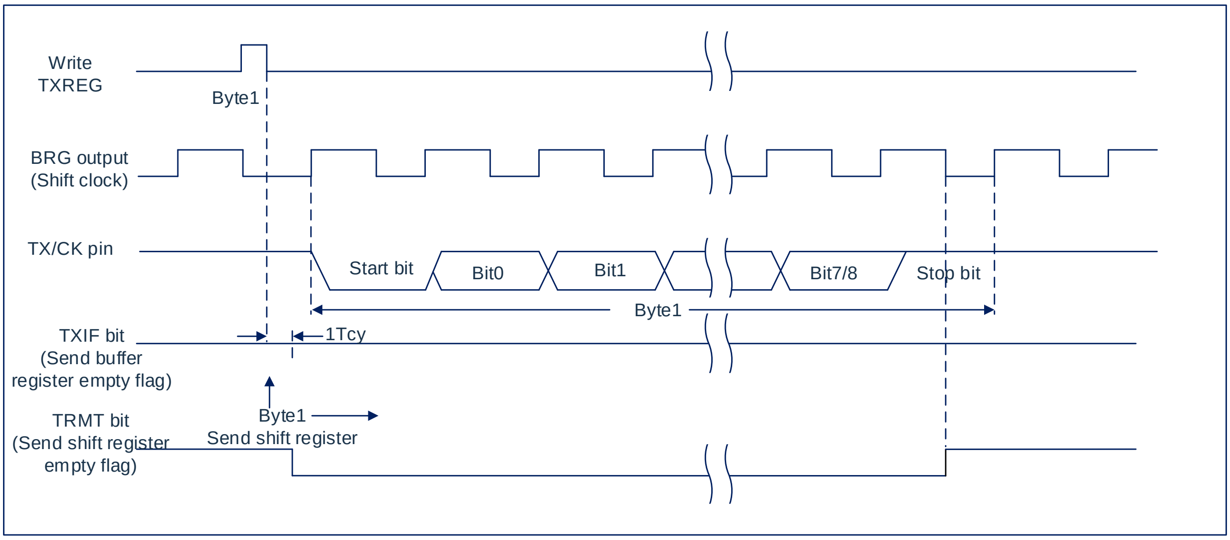


Fig 14-3: asynchronous transmit

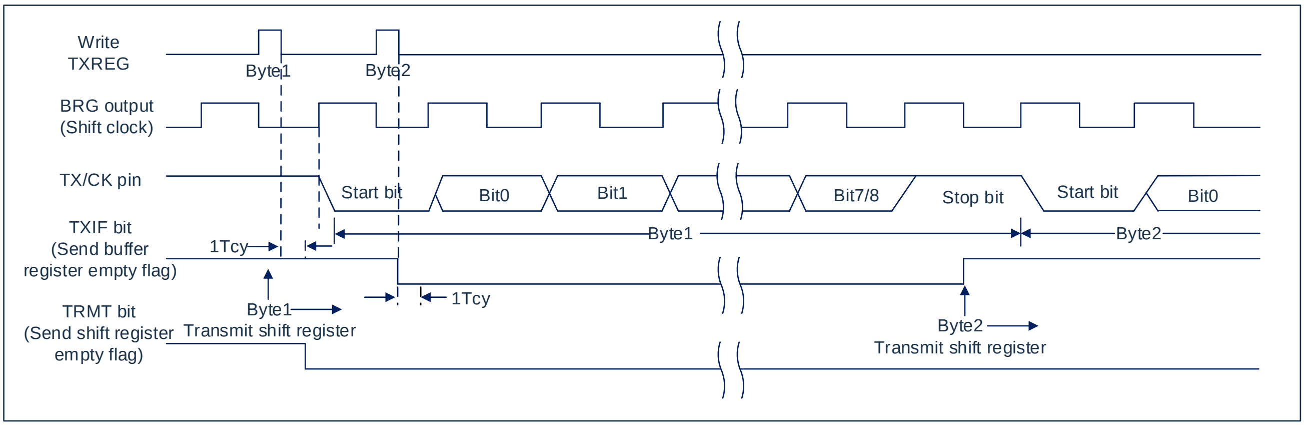


Fig14-4: asynchronous transmit (back to back)

Note: This time series diagram shows two consecutive transmit.

14.1.2 USART Asynchronous Receiver

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

14.1.2.1 Enable Receiver

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

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

Assume 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 USART for asynchronous operation.

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

14.1.2.2 Receive Data

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

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

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

14.1.2.3 Receive Interrupt

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

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

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

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

14.1.2.4 Receive Frame Error

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

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

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

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

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

14.1.2.5 Receive Overflow Error

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

14.1.2.6 Receive 9-bit Character

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

14.1.2.7 Asynchronous Receive Configuration

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

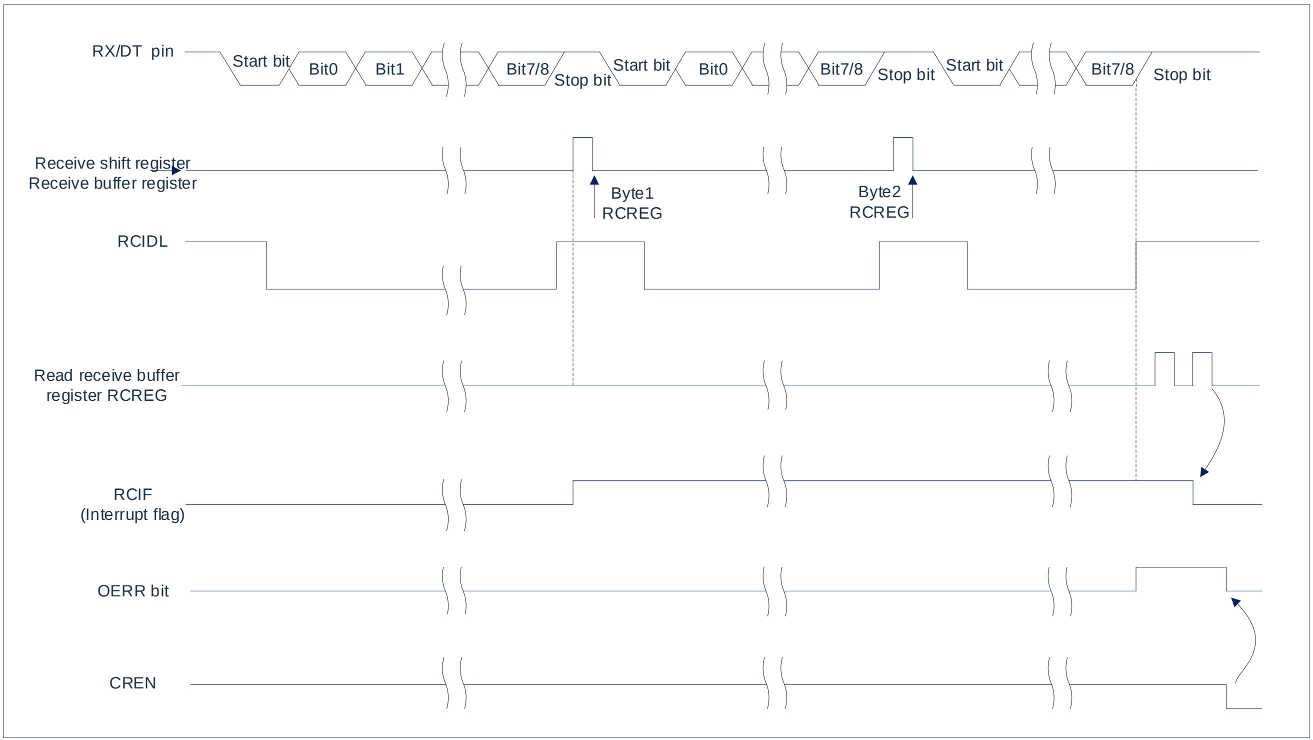


Fig 14-5: Asynchronous receive

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

14.2 Clock Precision for Asynchronous Operations

The output of the internal oscillation circuit (INTOSC) is calibrated by the manufacturer. But when VDD or temperature changes, INTOSC will have a frequency shift, which will directly affect the asynchronous baud rate. There is a way to adjust the value of the baud rate generator. When adjusting the baud rate generator, the adjustment resolution may not be enough to compensate for the gradual change in the peripheral clock frequency.

14.3 USART Related Register

TXSTA: transmit status and control register (117H)

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

Bit7	CSRC:	clock sourceselection bit;
	Asynchronous mode:	Any value;
	Synchronous mode:	
		1=master control mode (internal BRG generate clock signal);
		0=slave mode (external clock source generate clock).
Bit6	TX9:	9-bit transmit enable bit;
	1=	Select 9-bit transmit;
	0=	Select 8-bit transmit.
Bit5	TXEN:	Transmit enable bit (1);
	1=	Enable transmit;
	0=	Disable transmit.
Bit4	SYNC:	USART mode selection bit;
	1=	Synchronous mode;
	0=	Asynchronous mode.
Bit3	SCKP:	Synchronous clock polarity selection bit.
	Asynchronous mode:	
	1=	1= Invert the level of the data character and transmit to the TX/CK pin;
	0=	0= Directly transmit data character to TX/CK pin.
	Synchronous mode:	
	0=	0= Data is transmitted on the rising edge of clock;
	1=	1= Data is transmitted on the falling edge of clock.
Bit2	STOPBIT:	Stop bit selection (only valid for asynchronous transmission)
	1=	1 stop bit;
	0=	2 stop bits. (When the program sends data by judging the TRMT bit, STOPBIT needs to select 2 bits)
Bit1	TRMT:	Transmit shift register status bit;
	1=	TSR empty;
	0=	TSR full.
Bit0	TX9D:	9th bit of Transmit data.
		Can be address/data bit or parity check bit.

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

RCSTA: receive status and control register (118H)

118H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RCSTA	SPEN	RX9EN	SREN	CREN	RCIDL	FERR	OERR	RX9D
read/write	R/W	R/W	R/W	R/W	R	R	R	R
Reset value	0	0	0	0	0	0	0	0

Bit7 SPEN: serialportenable bit;
1= Enable serial port (RX/DT and TX/CK pin configured as serial portp in);
0= Disable serial port (hold on reset).

Bit6 RX9: 9-bit receive enable bit;
1= Select 9-bit receive;
0= Select 8-bit receive;.

Bit5 SREN: Single byte receive enable bit.
Asynchronous mode: any value.
Synchronous master control mode:
1=enable single byte receive;
0=disable single byte receive. Clear after receive completed.
Synchronous slave mode: any value.

Bit4 CREN: Continuous receive enable bit.
Asynchronous mode:
1=enable receive;
0=disable receive.
Synchronous mode:
1=enable continuous receive until clear CREN enable bit (CREN cover SREN);
0=disable continuous receive.

Bit3 RCIDL: Receive idle flag bit.
Asynchronous mode:
1= receiver idle;
0= already receive initial bit，receiving data.
Synchronous mode: any value.

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

Bit1 OERR: Overflow error bit.
1= Overflow error (clear by clearing CREN bit);
0= No overflow error.

Bit0 RX9D: Receive until 9th bit of the data.
This bit can be the address/data bit or the parity check bit, which must be calculated by the user firmware.

14.4 USART Baud Rate Generator (BRG)

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

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

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

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

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

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

formula1: claculate baud rate error

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

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

solve SPBRG:

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

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

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

Table 14-1: baud rateformula

Configuration bit	BRG/USARTmode	baud rateformula
SYNC		
0	8 位/异步	F_{OSC}/[16(n+1)]
1	8 位/同步	F_{OSC}/[4(n+1)]

Note: n= value of SPBRG register.

Table 14-2: baud rate in asynchronous mode

Target baud rate	SYNC=0					
	F_{OSC}=8.00MHz			F_{OSC}=16.00MHz		
	Real baud rate	error (%)	SPBRG value	Real baud rate	error (%)	SPBRG value
2400	2404	0.16	207	----	----	----
9600	9615	0.16	51	9615	0.16	103
10417	10417	0	47	10417	0	95
14400	14286	-0.8%	34	14286	-0.8%	68
19200	19230	0.16	25	19230	0.16	51

14.5 USART Synchronous Mode

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

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

14.5.1 Synchronous Master Control Mode

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

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

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

14.5.1.1 Master Control Clock

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

14.5.1.2 Clock Polarity

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

14.5.1.3 Synchronous Master control Transmit

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

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

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

14.5.1.4 Synchronous Master Control Transmit Configuration

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

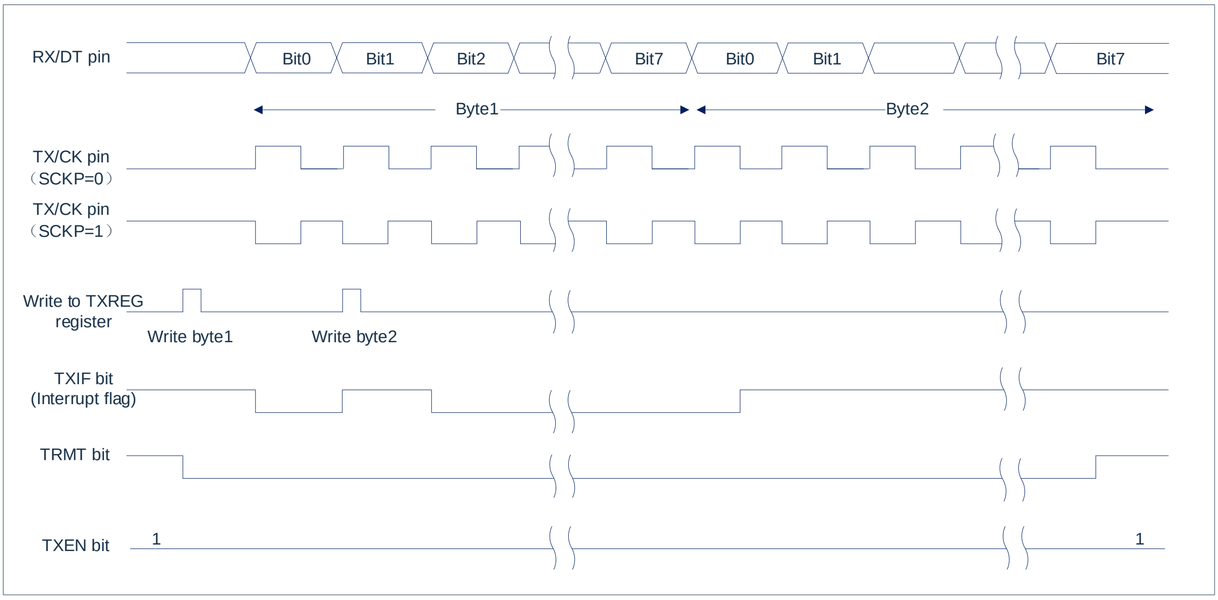


Fig 14-6: synchronous transmit

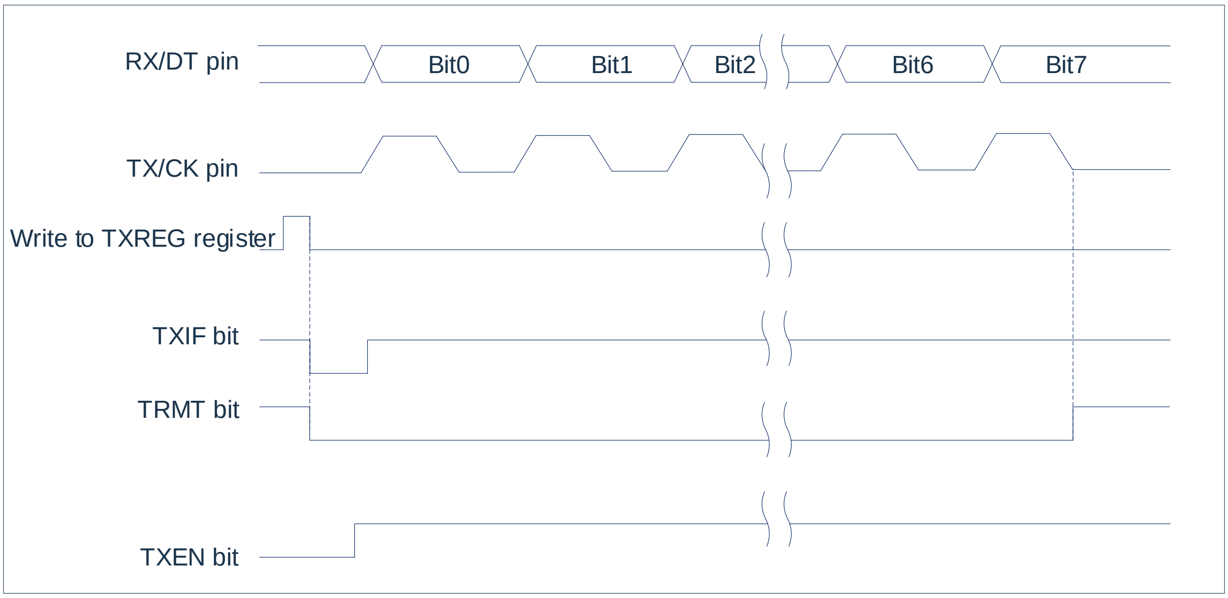


Fig 14-7: synchronous transmit (through TXEN)

14.5.1.5 Synchronous Master Control Receive

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

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

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

14.5.1.6 Slave Clock

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

14.5.1.7 Receive Overflow Error

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

14.5.1.8 Receive 9-bit Character

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

14.5.1.9 Synchronous Master Control Receive Configuration

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

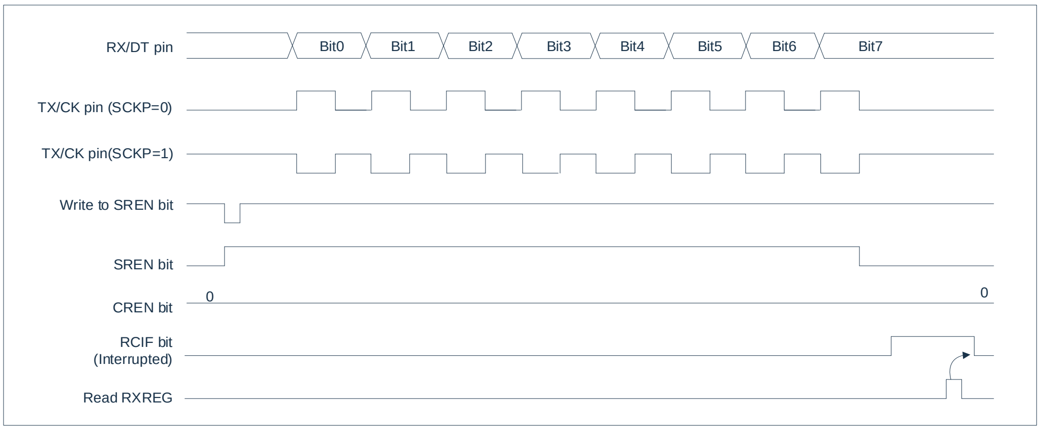


Fig 14-8: synchronous receive (master control mode, SREN)

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

14.5.2 Synchronous Slave Mode

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

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

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

14.5.2.1 USART Synchronous Slave Transmit

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

14.5.2.2 Synchronous Slave Transmit Configuration

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

14.5.2.3 USART Synchronous Slave Receive

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

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

14.5.2.4 Synchronous Slave Receive Configuration

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

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

15. Program EEPROM and Program Memory Control

15.1 General

The devices in this series have 2K words of program memory, the address range is from 000h to FFFh, which is read-only in all address ranges; the device has a 128-byte program EEPROM, and the address range is 0h to 07Fh, which is available in all address ranges. It can be read/write.

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

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

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

When accessing the program memory of the device, the EEDAT and EEDATH register form a double byte word to save the 16-bit data to be read, and the EEADR and EEADRH register form a double byte word to save the 11-bit EEPROM cell address to be read.

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

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

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

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

15.2 Related Register

15.2.1 EEADR and EEADRH Register

The EEADR and EEADRH registers can address up to 128 bytes of program EEPROM or up to 2K bytes of program memory.

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

15.2.2 EECON1 and EECON2 Register

EECON1 is the control register to access the program EEPROM.

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

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

- When WREN is set to 1, the program EEPROM is allowed to be written. When power is on, the WREN bit is cleared. When the normal write operation is LVR reset or WDT timeout reset interrupt, the WRERR bit will be set to 1. In these cases, after reset, the user can check the WRERR bit and rewrite the corresponding unit.
- When the write operation is completed, the interrupt flag bit EEIF in the PIR1 register is set to 1. This flag bit must be cleared by software.

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

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

EEPROM data register EEDAT(8EH)

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

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

EEPROM address register EEADR(90H)

90H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEADR	EEADR7	EEADR6	EEADR5	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

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

EEPROM data register EEDATH(8FH)

8FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEDATH	EEDATH7	EEDATH6	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	X	X	X	X	X	X	X	X

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

EEPROM address register EEADRH(96H)

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

Bit7~Bit3 Not used，read 0.

Bit2~Bit0 EEADRH<2:0>: pecify the upper 3 address of the program memory read operation.

EEPROM control register EECON1(8CH)

8CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EECON1	EEPGD	---	EETIME1	EETIME0	WRERR	WREN	WR	RD
R/W	R/W	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	---	0	0	X	0	0	0

Bit7	EEPGD:	Program/program EEPROMselection bit；
	1=	Operate program memory；
	0=	Operate program EEPROM.
Bit6	Not used.	
Bit5~Bit4	EETIME[1:0]	The longest programming waiting time; (For more EETIME information, please refer to Figure 15-1)
	00=	1.25ms
	01=	2.5ms(VDD=4.0~5.5V, TEMP=0~85°C recommended value)
	10=	5ms
	11=	10ms (Recommended values for other conditions except 2.5ms)
Bit3	WRERR:	EEPROM error flag bit；
	1=	Write operation error (any WDT reset or under-voltage reset during normal operation, or the time set by EETIME has expired but the self-check has not been successful)
	0=	Write complete.
Bit2	WREN:	EEPROM write enable bit；
	1=	Enable write period；
	0=	Disable write memory.
Bit1	WR:	Write control bit；
	1=	Start write period (Once the write operation is completed, this bit is cleared by hardware, and the WR bit can only be set to 1, but not cleared by software)；
	0=	Write period complete.
Bit0	RD:	Read control bit；
	1=	Start the memory read operation (the RD is cleared by hardware, and the RD bit can only be set to 1, but not cleared by software);
	0=	Not start memory read operation.

15.3 Read Program EEPROM

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

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

Example: read program EEPROM

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

15.4 Write Program EEPROM

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

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

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

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

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

Example: write program EEPROM

```
EEPDATA_WRITE:
        LD      A,WADDR             ; Put the address to be written into the EEADR register
        LD      EEADR,A
        LD      A,WDATA             ; Put the lower 8 bits of data to be written into the EEDAT register
        LD      EEDAT,A
        CLRWDT
        CLR     EECON1
        SETB    EECON1,EETIME0
        SETB    EECON1,EETIME1      ; EE programming time 10ms, user can customize
        CLRB    EECON1,EEPGD        ; Access data storage
        SETB    EECON1,WREN         ; Allow write cycles
        CLRB    F_GIE_ON            ; Save interrupt open state
        SZB     INTCON,GIE
        SETB    F_GIE_ON
        CLRB    INTCON,GIE          ;Disable interrupt
        SZB     INTCON,GIE          ;Ensure interrupt is disabled
        JP      $-2

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

```
                NOP
                CLRWDT
                CLRB        EECON1,WREN           ; Write end, turn off write enable bit

                SZB         F_GIE_ON              ; Restoring the interrupted state
                SETB        INTCON,GIE

                SNZB        EECON1,WRERR          ; Determine whether the EEPROM write operation
                                                  is wrong
                JP          EEPDATA_WRITE_BACK
                SZDECR      WERR_C                ; Exit when counting over time, user can
                                                  customize
                JP          EEPDATA_WRITE         ; EEPROM write operation error, rewrite
EEPDATA_WRITE_BACK:
                RET
```

15.5 Read Program Memory

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

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

Example: read flash program memory

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

15.6 Write Program Memory

Program memoryis read only, cannot be written.

15.7 Precautions on Program EEPROM

15.7.1 Programming Time for Program EEPROM

The program EEPROM programming time is not fixed. The time required to program different data is different, ranging from 100us to 10ms. The EETIME bit of the EECON1 register determines the maximum time for program EEPROM programming. The program EEPROM module has a built-in self-checking function, During the programming process, if the self-verification is successful or the time set by EETIME has expired, one of the conditions will be met and the writing operation will end. During the programming, the CPU stops working and the peripheral modules work normally, and the program needs to do the relevant processing.

15.7.2 The times of programming EEPROM

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

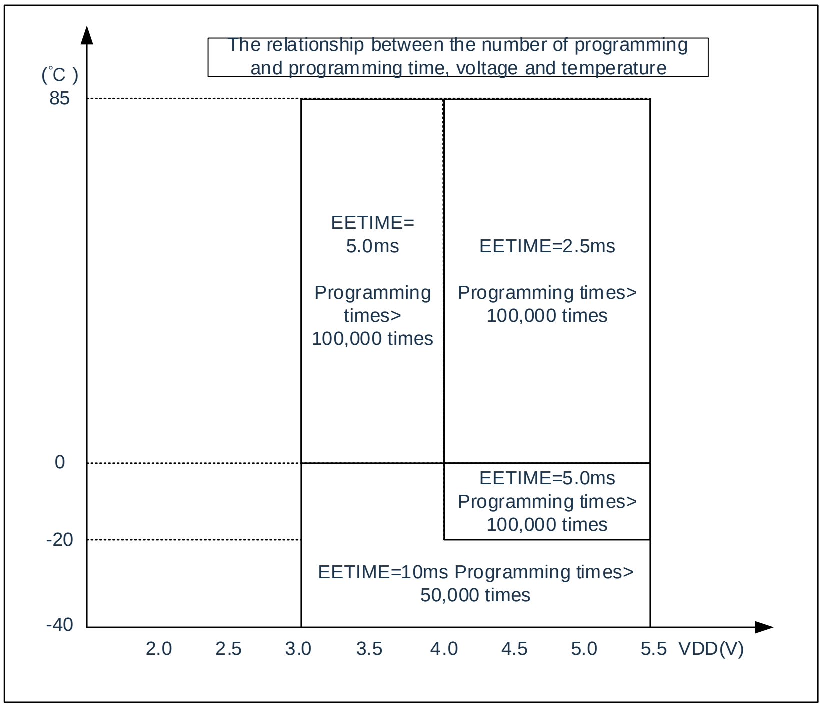


Fig 15-1 The relationship between the number of programming EEPROM programming and programming time, voltage, and temperature

15.7.3 Write Verification

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

15.7.4 Protection to Avoid Writing Wrongly

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

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

- Undervoltage
- Power glitch
- Software failure

16. Low Voltage detection (LVD)

16.1 LVD Mod General

The CMS79F11x series of MCU have a low-voltage detection function, which can be used to monitor the power supply voltage. If the power supply voltage is lower than the set value, an interrupt signal can be generated; the program can read the LVD output flag bit in real time.

16.2 LVD Related Register

LVD control register LVDCON(11FH)

11FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LVDCON	LVD_RES	—	—	—	LVD_SEL[2:0]			LVDEN
R/W	R	—	—	—	R/W	R/W	R/W	R/W
Reset value	X	—	—	—	0	0	0	0

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

16.3 LVD Operation

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

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

17. Electrical parameters

17.1 Limit parameters

Power supply voltage……………………………………………………..…………… GND-0.3V~GND+6V
Storage temperature range………………………………………………………..………… 50°C~125°C
Operating temperature………………………………………………………..………...… - 40°C~85°C
Port input voltage……………………………………………………......………… GND-0.3V~VDD+0.3V
Maximum sink current of all ports…………………...……………………………...…………….. 200mA
Maximum source current of all ports………………………………………………..…………… -150mA

> Note:
> If the device working conditions exceed the above limit parameters, may be caused to damage. The above value is only the maximum of running conditions. We do not recommend that devices run outside the scope in this specification. If the device work for a long time among the limit value condition, the stability will be affected.

17.2 DC Electrical Characteristic

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		VDD	condition				
VDD	Operating voltage		F_{SYS}=16MHz	2.6		5.5	V
			F_{SYS}=8MHz	1.8		5.5	V
I_{DD}	Operating voltage	5V	F_{SYS}=16MHz		3.8		mA
		3V	F_{SYS}=16MHz		2.6		mA
		5V	F_{SYS}=8MHz		2.7		mA
		3V	F_{SYS}=8MHz		2		mA
		5V	Program EEPROM	-	20	-	mA
		3V	Program EEPROM	-	10	-	mA
I_{STB}	Quiescent current	5V	----		0.1	2	μA
		3V	----		0.1	1	μA
V_{IL}	Low-level input voltage		----			0.3VDD	V
V_{IH}	High level input voltage		----	0.7VDD			V
V_{OH}	High level output voltage		Without load	0.9VDD			V
V_{OL}	Low-level output voltage		Without load			0.1VDD	V
V_{EEPROM}	EEPROM module working voltage		----	3.0		5.5	V
R_{PH}	Pull-up resistor value	5V	V_O=0.5VDD		38		kΩ
		3V	V_O=0.5VDD		70		kΩ
R_{PL}	Pull-low resistor value	5V	V_O=0.5VDD		34		kΩ
		3V	V_O=0.5VDD		60		kΩ
I_{OL1}	Output sink current (Ordinary I/O port)	5V	V_{OL}=0.3VDD		60		mA
		3V	V_{OL}=0.3VDD		25		mA
I_{OH1}	Output port sourcing current (Ordinary I/O port)	5V	V_{OH}=0.7VDD		-20		mA
		3V	V_{OH}=0.7VDD		-9		mA
V_{BG}	Internal reference voltage 1.2V	VDD=2.5~5.5V T_A=25℃		-1.5%	1.2	1.5%	V
		VDD=2.5~5.5V T_A=－40~85℃		-2.0%	1.2	2.0%	V

17.3 ADC Electrical Characteristic

(T_A= 25°C, unless otherwise specified)

Symbol	Parameter	Test condition	Min	Typ	Max	Unit
V_{ADC}	Operating voltage range	AD_{VREF}=VDD, F_{ADC}=1MHz	3.0		5.5	V
		AD_{VREF}=VDD, F_{ADC}=500kHz	2.7		5.5	V
		AD_{VREF}=2.4V, F_{ADC}=250kHz	2.7		5.5	V
		AD_{VREF}=2.0V, F_{ADC}=250kHz	2.7		5.5	V
I_{ADC}	ADC Conversion current	V_{ADC}=5V,AD_{VREF}=VDD,F_{ADC}=500kHz			500	uA
		V_{ADC} =3V, AD_{VREF}=VDD,F_{ADC}=500kHz			200	uA
V_{ADI}	Input voltage range	V_{ADC}=5V, AD_{VREF}=VDD,F_{ADC}=250kHz	0		V_{ADC}	V
DNL	Differential non-linearity error 1	V_{ADC}=5V, AD_{VREF}=VDD,F_{ADC}=250kHz			±2	LSB
INL	Integral non-linearity error 1	V_{ADC}=5V, AD_{VREF}=VDD,F_{ADC}=250kHz			±2	LSB
T_{ADC}	Conversion time	-		16		T_{ADCCLK}

17.4 ADC Internal LDO Reference Voltage Characteristics

(T_A= 25°C, unless otherwise specified)

Symbol	Parameter	Test condition	Min	Typ	Max	Unit
AD_{VREF1}	LDO=2.0V Voltage Temperature Characteristics	VDD=5V T_A=25°C	-0.6%	2.0	+0.6%	V
		VDD=2.7~5.5V T_A=25°C	-1.0%	2.0	+1.0%	V
		VDD=2.7~5.5V T_A=-40°C~85°C	-1.5%	2.0	+1.5%	V
AD_{VREF2}	LDO=2.4V Voltage Temperature Characteristics	VDD=5V T_A=25°C	-0.6%	2.4	+0.6%	V
		VDD=2.7~5.5V T_A=25°C	-1.0%	2.4	+1.0%	V
		VDD=2.7~5.5V T_A=-40°C~85°C	-1.5%	2.4	+1.5%	V

17.5 Power-on Reset Features

(T_A= 25°C, unless otherwise specified)

Symbol	Parameter	Test condition	Min	Typ	Max	Unit
t_{VDD}	VDD increase rate	-	0.05			V/ms
V_{LVR1}	LVR setting voltage = 1.8V	VDD=1.6~5.5V	1.7	1.8	1.9	V
V_{LVR2}	LVR setting voltage =2.0V	VDD=1.8~5.5V	1.9	2.0	2.1	V
V_{LVR1}	LVR setting voltage =2.6V	VDD=2.4~5.5V	2.5	2.6	2.7	V

17.6 LVD Electrical Characteristic

(T_A= 25°C, unless otherwise specified)

Symbol	Parameter	Test condition	Min	Typ	Max	Unit
VLVD	Voltage	-	2.0		5.5	V
	Precision	VDD=2.0~5.5V, T_A=－40~85°C	-5%	V_{SET}	+5%	V

17.7 electrical characteristics

(T_A= 25°C, unless otherwise specified)

Symbol	Parameter	Test condition		Min	Typ	Max	Unit
		VDD	condition				
F_{WDT}	WDT clock source stability	VDD=3.0~5.5V	TA=25°C	-10%	32	+10%	KHz
		VDD=3.0~5.5V	TA=—40~85°C	-20%	32	+20%	KHz
T_{EEPROM}	EEPROM programming time	5V	F_{HSI}=8MHz/16MHz			10	ms
		3V	F_{HSI}=8MHz/16MHz			10	ms
F_{RC}	Internal frequency stability	VDD=4.0~5.5V	TA=25°C	-1.5%	8	+1.5%	MHz
		VDD=2.5~5.5V	TA=25°C	-2.0%	8	+2.0%	MHz
		VDD=4.0~5.5V	TA=—40~85°C	-2.5%	8	+2.5%	MHz
		VDD=2.5~5.5V	TA=—40~85°C	-3.5%	8	+3.5%	MHz
		VDD=1.8~5.5V	TA=—40~85°C	-5.0%	8	+5.0%	MHz
		VDD=4.0~5.5V	TA=25°C	-1.5%	16	+1.5%	MHz
		VDD=2.6~5.5V	TA=25°C	-2.0%	16	+2.0%	MHz
		VDD=4.0~5.5V	TA=—40~85°C	-2.5%	16	+2.5%	MHz
		VDD=2.6~5.5V	TA=—40~85°C	-3.5%	16	+3.5%	MHz

18. Instructions

18.1 Instructions Table

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

mnemonic		operation	Instructions period	symbol
RRR	[R]	Data memory does not take carry and rotates to the right by one bit, and the result is stored in R	1	NONE
Increae/decrease-4				
INCA	[R]	Increment data memory R，result stored in ACC	1	Z
INCR	[R]	Increment data memory R，result stored in R	1	Z
DECA	[R]	Decrement data memory R，result stored in ACC	1	Z
DECR	[R]	Decrement data memory R，result stored in R	1	Z
Bit operation-2				
CLRB	[R],b	Clear some bit in data memory R	1	NONE
SETB	[R],b	Set some bit in data memory R 1	1	NONE
look-up table-2				
TABLE	[R]	Read FLASH and save to TABLE_DATAH and R	2	NONE
TABLEA		Read FLASH and save to TABLE_DATAH and ACC	2	NONE
Math operation-16				
ADDA	[R]	ACC+[R]→ACC	1	C,DC,Z,OV
ADDR	[R]	ACC+[R]→R	1	C,DC,Z,OV
ADDCA	[R]	ACC+[R]+C→ACC	1	Z,C,DC,OV
ADDCR	[R]	ACC+[R]+C→R	1	Z,C,DC,OV
ADDIA	i	ACC+i→ACC	1	Z,C,DC,OV
SUBA	[R]	[R]-ACC→ACC	1	C,DC,Z,OV
SUBR	[R]	[R]-ACC→R	1	C,DC,Z,OV
SUBCA	[R]	[R]-ACC-C→ACC	1	Z,C,DC,OV
SUBCR	[R]	[R]-ACC-C→R	1	Z,C,DC,OV
SUBIA	i	i-ACC→ACC	1	Z,C,DC,OV
HSUBA	[R]	ACC-[R]→ACC	1	Z,C,DC,OV
HSUBR	[R]	ACC-[R]→R	1	Z,C,DC,OV
HSUBCA	[R]	ACC-[R]- $\overline{C}$ →ACC	1	Z,C,DC,OV
HSUBCR	[R]	ACC-[R]- $\overline{C}$ →R	1	Z,C,DC,OV
HSUBIA	i	ACC-i→ACC	1	Z,C,DC,OV
Unconditional transfer -5				
RET		Return from subroutine	2	NONE
RET	i	Return from subroutine，save I to ACC	2	NONE
RETI		Return from interrupt	2	NONE
CALL	ADD	Subroutine call	2	NONE
JP	ADD	Unconditional jump	2	NONE
Conditional transfer-8				
SZB	[R],b	If the b bit of data memory R is "0", skip the next instruction	1 or 2	NONE
SNZB	[R],b	If the b bit of data memory R is "1", skip the next instruction	1 or 2	NONE
SZA	[R]	data memory R is sent to ACC, if the content is "0", skip the next instruction	1 or 2	NONE
SZR	[R]	If the content of data memory R is "0", skip the next instruction	1 or 2	NONE

mnemonic	operation	Instructions period	symbol
SZINCA [R]	Add "1" to data memory R and put the result into ACC, if the result is "0", skip the next oneinstructions	1 or 2	NONE
SZINCR [R]	Add "1" to data memory R, put the result into R, if the result is "0", skip the next instruction	1 or 2	NONE
SZDECA [R]	Data memory R minus "1", the result is put into ACC, if the result is "0", skip the next instruction	1 or 2	NONE
SZDECR [R]	Data memory R minus "1", put the result into R, if the result is "0", skip the next oneinstructions	1 or 2	NONE

18.2 Instructions Illustration

ADDA **[R]**

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

period: 1

Affected flag bit: C , DC , Z , OV

example:

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

ADDR **[R]**

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

period: 1

Affected flag bit: C , DC , Z , OV

example:

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

ADDCA **[R]**

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

period: 1

affected flag bit: C , DC , Z , OV

example:

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

ADDCR **[R]**

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

period: 1

affected flag bit: C , DC , Z , OV

example:

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

ADDIA **i**

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

period: 1

affected flag bit: C，DC，Z，OV

example:

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

ANDA **[R]**

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

period: 1

affected flag bit: Z

example:

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

ANDR **[R]**

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

period: 1

affected flag bit: Z

example:

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

ANDIA **i**

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

period: 1

affected flag bit: Z

example:

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

CALL **add**

operation: Call subroutine

period: 2

affected flag bit: none

example:

```
CALL    LOOP    ; Call the subroutine address whose name is defined as "LOOP"
```

CLRA

operation:	ACC clear	
period:	1	
affected flag bit:	Z	
example:		
	CLRA	;execute：ACC=0

CLR **[R]**

operation:	Register R clear		
period:	1		
affected flag bit:	Z		
example:			
	CLR	R01	;execute：R01=0

CLRB **[R],b**

operation:	Clear b bit on register R		
period:	1		
affected flag bit:	none		
example:			
	CLRB	R01,3	;execute：3rd bit of R01 is 0

CLRWDT

operation:	Clear watchdog timer	
period:	1	
affected flag bit:	TO，PD	
example:		
	CLRWDT	;watchdog timer clear

COMA **[R]**

operation:	Reverse register R，save the result to ACC		
period:	1		
affected flag bit:	Z		
example:			
	LDIA	0AH	;load 0AH to ACC
	LD	R01,A	;load ACC (0AH) to R01
	COMA	R01	;execute：ACC=0F5H

COMR **[R]**

operation: Reverse register R，save the result to R

period: 1

affected flag bit: Z

example:

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

DECA **[R]**

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

period: 1

affected flag bit: Z

example:

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

DECR **[R]**

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

period: 1

affected flag bit: Z

example:

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

HSUBA **[R]**

operation: ACC subtract R, save the result to ACC

period: 1

affected flag bit: C,DC,Z,OV

example:

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

HSUBR **[R]**

operation: ACC subtract R, save the result to R

period: 1

affected flag bit: C,DC,Z,OV

example:

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

HSUBCA **[R]**

operation: ACC subtract C, save the result to ACC

period: 1

affected flag bit: C,DC,Z,OV

example:

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

HSUBCR **[R]**

operation: ACC subtract C, save the result to R

period: 1

affected flag bit: C,DC,Z,OV

example:

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

INCA **[R]**

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

period: 1

affected flag bit: Z

example:

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

INCR	**[R]**		
operation:	Register R increment 1, save the result to R		
period:	1		
affected flag bit:	Z		
example:			
	LDIA	0AH	; load 0AH to ACC
	LD	R01,A	; load ACC (0AH) to R01
	INCR	R01	;execute：R01= (0AH+1)=0BH

JP	**add**		
operation:	Jump to add address		
period:	2		
affected flag bit:	none		
example:			
	JP	LOOP	; jump to the subroutine address whose name is defined as "LOOP"

LD	**A,[R]**		
operation:	Load the value of R to ACC		
period:	1		
affected flag bit:	Z		
example:			
	LD	A,R01	;load R01 to ACC
	LD	R02,A	;load ACC to R02, achieve data transfer from R01→R02

LD	**[R],A**		
operation:	Load the value of ACC to R		
period:	1		
affected flag bit:	none		
example:			
	LDIA	09H	;load 09H to ACC
	LD	R01,A	;execute：R01=09H

LDIA	**i**		
operation:	Load in to ACC		
period:	1		
affected flag bit:	none		
example:			
	LDIA	0AH	;load 0AH to ACC

NOP

operation:	Empty instructions
period:	1
affected flag bit:	none
example:	

```
NOP
NOP
```

ORIA **i**

operation:	Perform ‘OR’ on I and ACC, save the result to ACC
period:	1
affected flag bit:	Z
example:	

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

ORA **[R]**

operation:	Perform ‘OR’ on R and ACC, save the result to ACC
period:	1
affected flag bit:	Z
example:	

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

ORR **[R]**

operation:	Perform ‘OR’ on R and ACC, save the result to R
period:	1
affected flag bit:	Z
example:	

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

RET

operation: Return from subroutine

period: 2

affected flag bit: none

example:

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

RET i

operation: Return with parameter from the subroutine, and put the parameter in ACC

period: 2

affected flag bit: none

example:

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

RETI

operation: Interrupt return

period: 2

affected flag bit: none

example:

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

RLCA [R]

operation: Register R rotates to the left with C and savethe result into ACC

period: 1

affected flag bit: C

example:

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

RLCR **[R]**

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

period: 1

affected flag bit: C

example:

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

RLA **[R]**

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

period: 1

affected flag bit: none

example:

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

RLR **[R]**

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

period: 1

affected flag bit: none

example:

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

RRCA **[R]**

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

period: 1

affected flag bit: C

example:

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

RRCR **[R]**

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

period: 1

affected flag bit: C

example:

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

RRA **[R]**

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

period: 1

affected flag bit: none

example:

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

RRR **[R]**

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

period: 1

affected flag bit: none

example:

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

SET **[R]**

operation: Set all bits in register R as 1

period: 1

affected flag bit: none

example:

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

SETB **[R],b**

operation: Set b bit in register R 1

period: 1

affected flag bit: none

example:

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

STOP

operation: Enter sleep

period: 1

affected flag bit: TO，PD

example:

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

SUBIA **i**

operation: ACC minus I, save the result to ACC

period: 1

affected flag bit: C,DC,Z,OV

example:

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

SUBA **[R]**

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

period: 1

affected flag bit: C,DC,Z,OV

example:

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

SUBR **[R]**

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

period: 1

affected flag bit: C,DC,Z,OV

example:

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

SUBCA **[R]**

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

period: 1

affected flag bit: C,DC,Z,OV

example:

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

SUBCR **[R]**

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

period: 1

affected flag bit: C,DC,Z,OV

example:

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

SWAPA **[R]**

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

period: 1

affected flag bit: none

example:

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

SWAPR **[R]**

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

period: 1

affected flag bit: none

example:

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

SZB	**[R],b**		
operation:	Determine the bit b of register R, if it is 0 then jump, otherwise execute in sequence		
period:	1 or 2		
affected flag bit:	none		
example:			
	SZB	R01,3	;determine 3rd bit of R01
	JP	LOOP	;if is 1, execute, jump to LOOP
	JP	LOOP1	; if is 0, jump,execute, jump to LOOP1

SNZB	**[R],b**		
operation:	Determine the bit b of register R, if it is 1 then jump, otherwise execute in sequence		
period:	1 or 2		
affected flag bit:	none		
example:			
	SNZB	R01,3	; determine 3rd bit of R01
	JP	LOOP	; if is 0, execute, jump to LOOP
	JP	LOOP1	; if is 1, jump,execute, jump to LOOP1

SZA	**[R]**		
operation:	Load the value of R to ACC, if it is 0 then jump, otherwise execute in sequence		
period:	1 or 2		
affected flag bit:	none		
example:			
	SZA	R01	;R01→ACC
	JP	LOOP	;if R01 is not 0, execute，jump to LOOP
	JP	LOOP1	;if R01 is 0, jump，execute，jump to LOOP1

SZR	**[R]**		
operation:	Load the value of R to R, if it is 0 then jump, otherwise execute in sequence		
period:	1 or 2		
affected flag bit:	None		
example:			
	SZR	R01	;R01→R01
	JP	LOOP	; if R01 is not 0, execute，jump to LOOP
	JP	LOOP1	; if R01 is 0, jump，execute，jump to LOOP1

SZINCA **[R]**

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

period: 1 or 2

affected flag bit: none

example:

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

SZINCR **[R]**

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

period: 1 or 2

affected flag bit: none

example:

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

SZDECA **[R]**

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

period: 1 or 2

affected flag bit: none

example:

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

SZDECR **[R]**

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

period: 1 or 2

affected flag bit: none

example:

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

TABLE **[R]**

operation: Look-up table, the lower 8 bits of the look-up table result are placed in R, and the high bits are placed in the dedicated register TABLE_DATAH

period: 2

affected flag bit: none

example:

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

TABLEA

operation: Look-up table, the lower 8 bits of the look-up table result are placed in ACC, and the high bits are placed in the dedicated register TABLE_DATAH

period: 2

affected flag bit: none

example:

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

TESTZ **[R]**

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

period: 1

affected flag bit: Z

example:

```
TESTZ   R0            ;
SZB     STATUS,Z      ;check Z flag bit，if it is 0 then jump
JP      Add1          ;if R0 is 0, jump to address Add1
JP      Add2          ;if R0 is not 0, jump to address Add2
```

XORIA **i**

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

period: 1

affected flag bit: Z

example:

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

XORA **[R]**

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

period: 1

affected flag bit: Z

example:

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

XORR **[R]**

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

period: 1

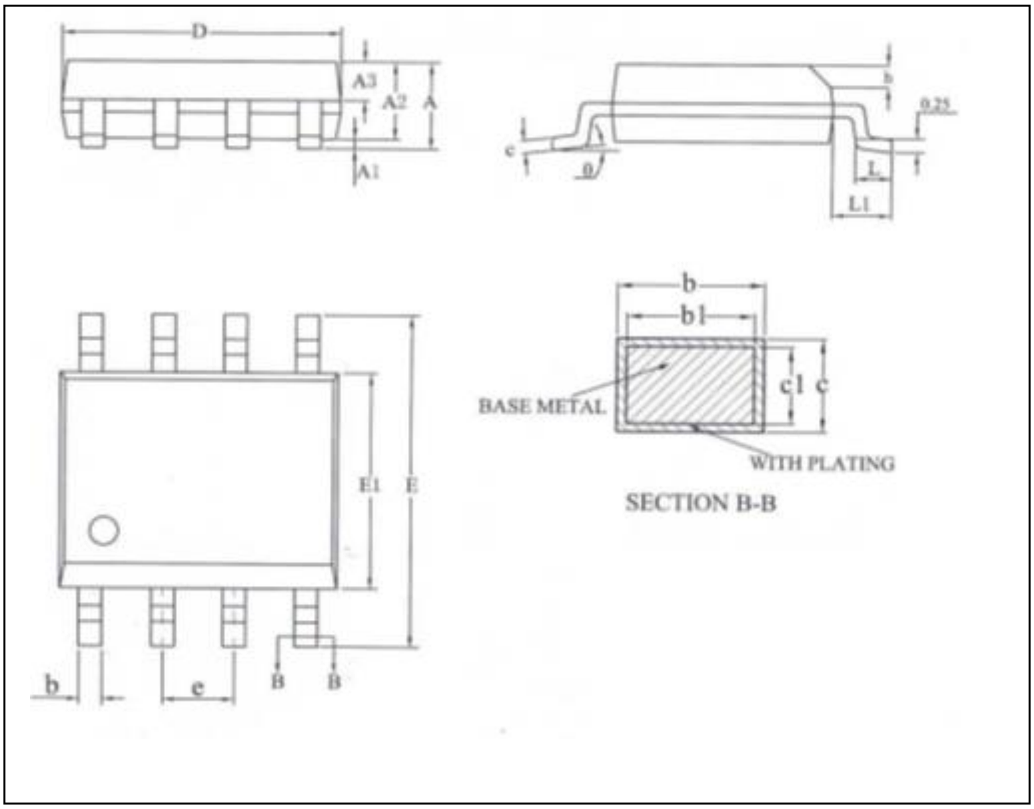
affected flag bit: Z

example:

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

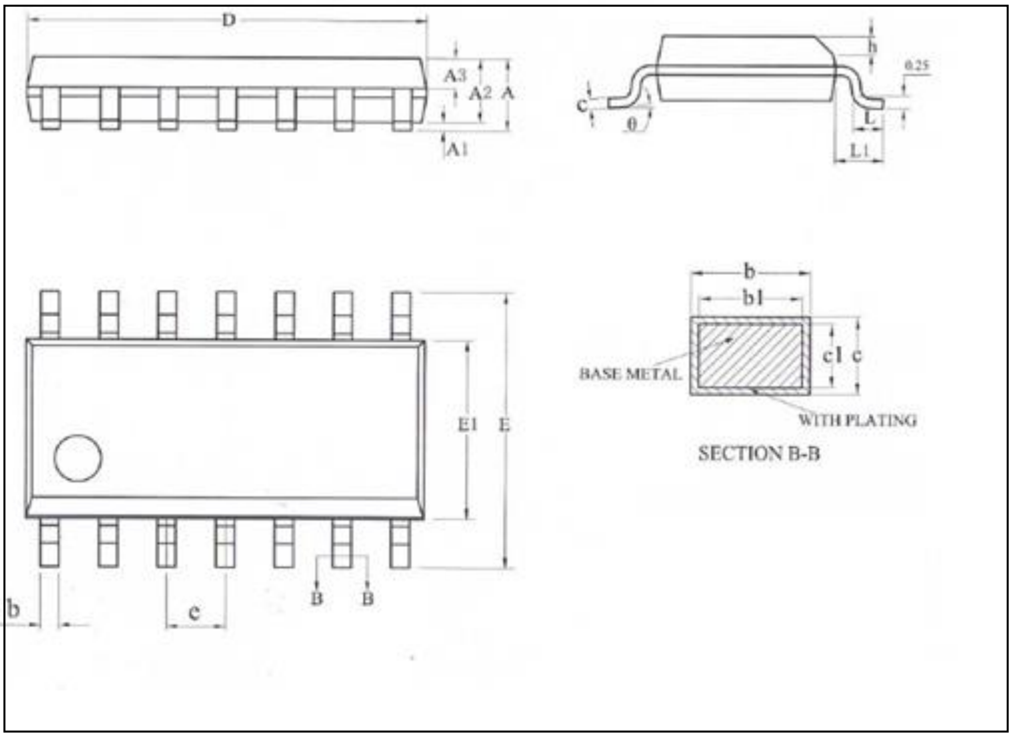
19. Packaging

19.1 SOP8



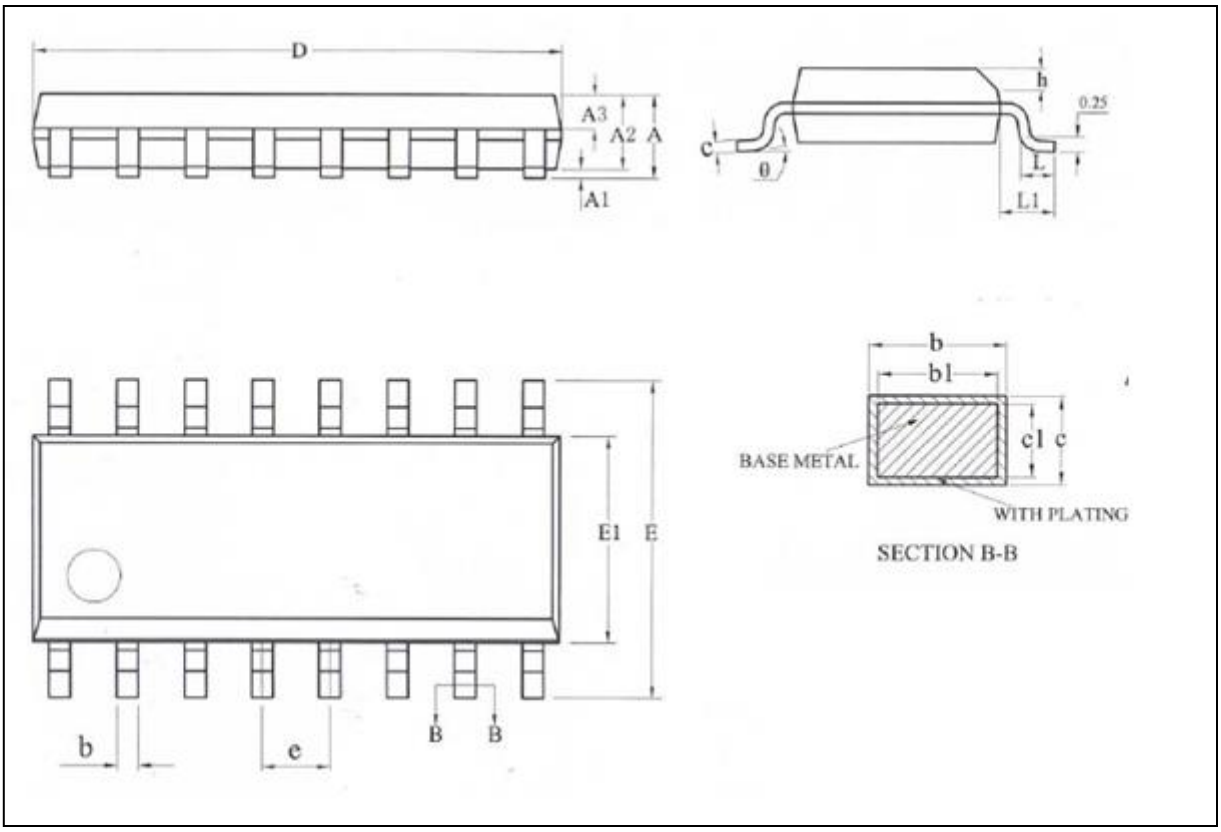
Symbol	Millimeter		
	Min	Nom	Max
A	-	-	1.75
A1	0.10	-	0.225
A2	1.30	1.40	1.50
A3	0.60	0.65	0.70
b	0.39	-	0.47
b1	0.38	0.41	0.44
c	0.20	-	0.24
c1	0.19	0.20	0.21
D	4.80	4.90	5.00
E	5.80	6.00	6.20
E1	3.80	3.90	4.00
e	1.27BSC		
h	0.25	-	0.50
L	0.5	-	0.80
L1	1.05REF		
θ	0	-	8°

19.2 SOP14



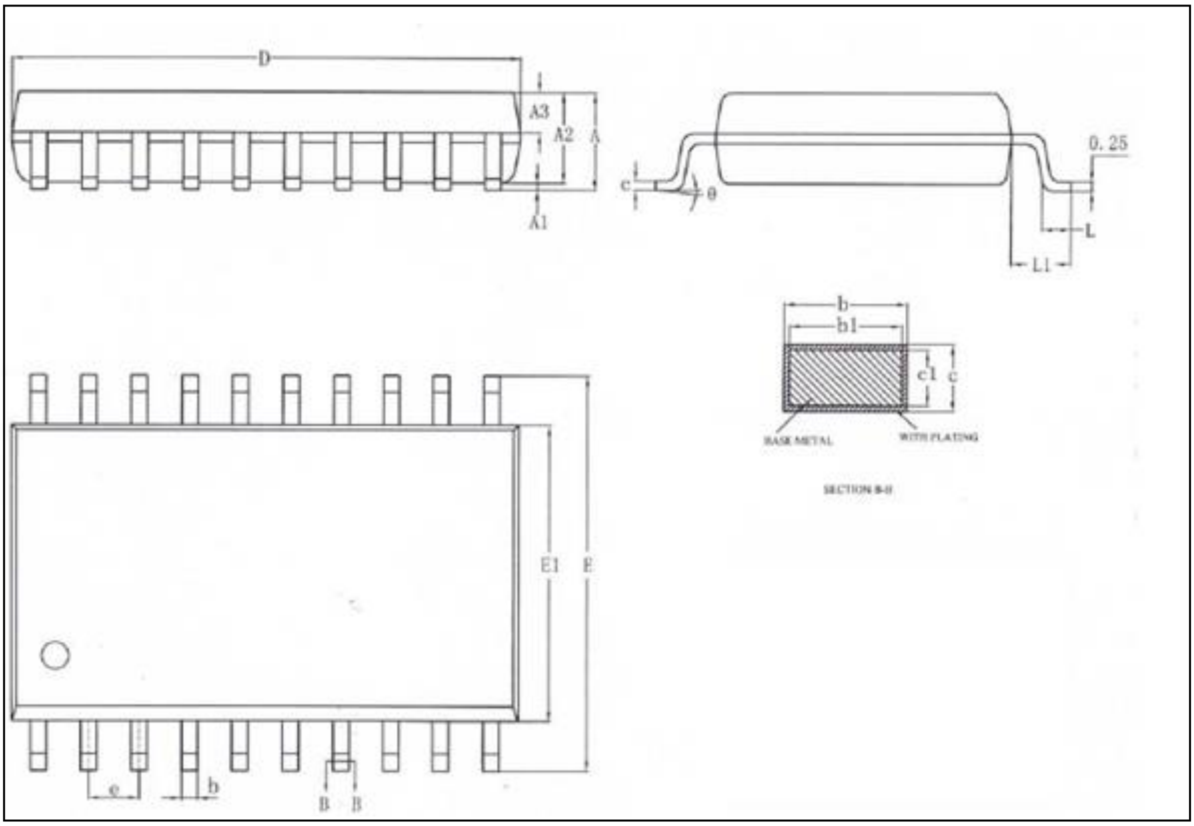
Symbol	Millimeter		
	Min	Nom	Max
A	-	-	1.75
A1	0.05	-	0.225
A2	1.30	1.40	1.50
A3	0.60	0.65	0.70
b	0.39	-	0.47
b1	0.38	0.41	0.44
c	0.20	-	0.24
c1	0.19	0.20	0.21
D	8.55	8.65	8.75
E	5.80	6.00	6.20
E1	3.80	3.90	4.00
e	1.27BSC		
h	0.25	-	0.50
L	0.5	-	0.80
L1	1.05REF		
θ	0	-	8°

19.3 SOP16



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

19.4 SOP20



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

20. Revision History

Revision	Date	Modify content
V1.0	Oct 2019	Initial verison
V1.1	Jun 2020	Update program EEPROM read and write sample program
V1.2	Mar 2021	Correct the pin descriptions corresponding to the CSSELx and CSENx registers in the LCD chapter
V1.30	Mar 2022	1. Corrected that RCIF and TXIF in PIR1 register are read-only 2. Corrected TXIF description in PIR1 register 3. Figs 14-3 and 14-4 corrected for USART asynchronous sending 4. Corrected14.1.2.3 Receive Interrupt description 5. Corrected the FERR frame error bits in RCSTA register are read-only 6. Add clock block diagram 7. Revised the internal high-speed oscillation frequency to F_{HSI}, and revised the clock sources of other modules according to the clock block diagram 8. Revised sleep wake-up waiting time and LDO reference voltage characteristics inside ADC 9. Delete description of sleep wake-up in ADC interrupt