SC8F6770_6771 User Manual

8-bit CMOS microcontroller with enhanced flash memory

Rev. 1.4

Manual

1. Product Description

1.1 Features

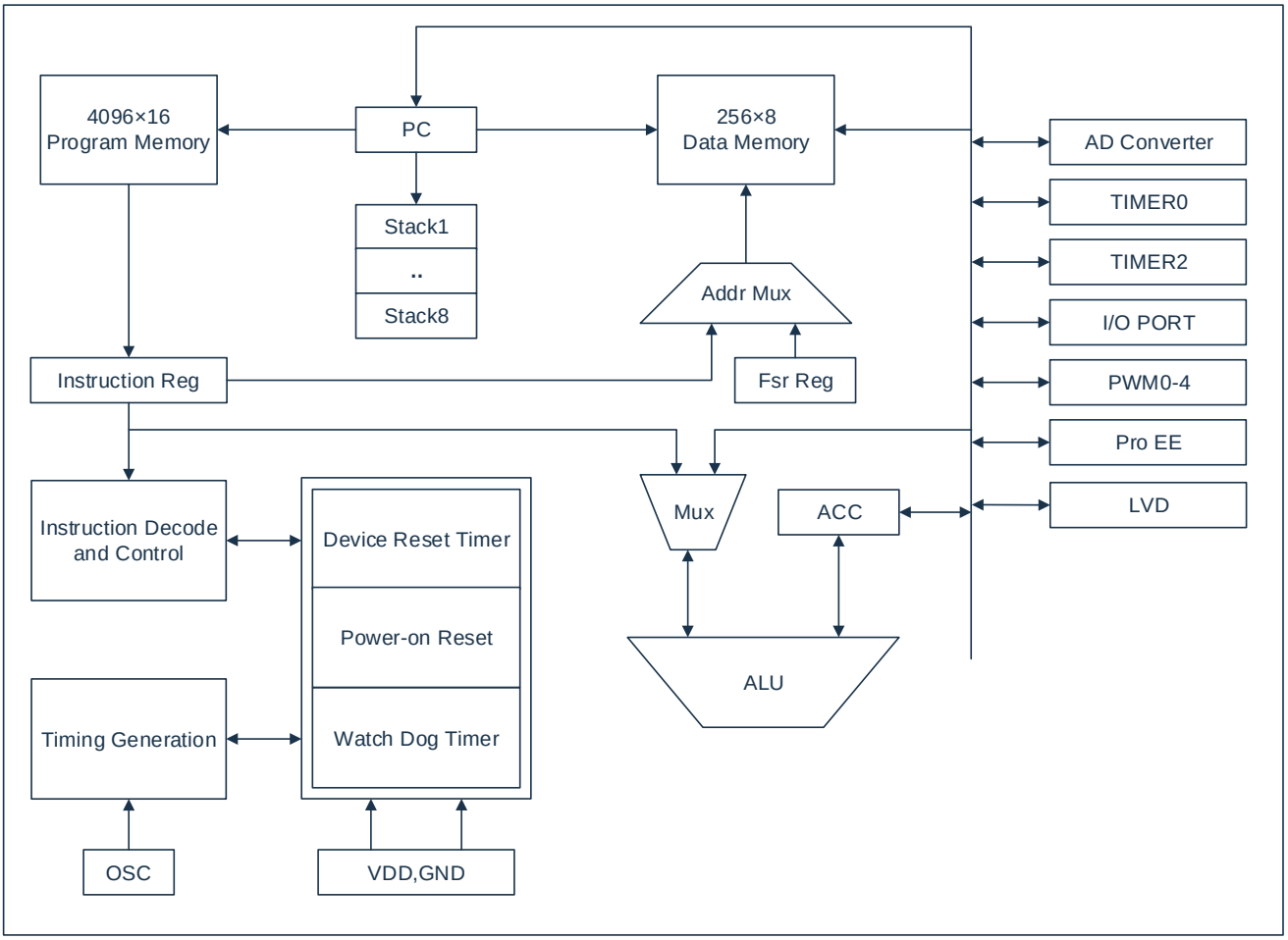
- Memory
  - ROM: 4K×16Bit
  - RAM: 256×8Bit
- 8-level stack buffer
- Short and clear command system(68 commands)
- Command period(Single period and daul period)
- Built in Low potential detector circuit
- Built in WDTtimer
- Interrupt
  - 2 timer Interrupt
  - RA Interrupt on Change in logic level
  - RB Interrupt on Change in logic level
  - Other external hardware Interrupt
- Timer
  - 8-bit Timer: TIMER0, TIMER2
  - TIMER2 can choose external 32.768KHz crystal oscillator as the source of clock
- Built in LVD module
  - Support multiple voltage: 2.2V/2.4V/2.7V/3.0V/ 3.3V/3.7V/4.0V/4.3V
- Voltage range: 1.8V~5.5V@16M/4T
  Temperature range:-20℃~85℃
- RC oscillator: 8MHz/16MHz
- 5ch PWM module
  - 4ch PWM with shared period and separated duty cycle
  - 1ch PWM with separated period and separated duty cycle
- High precision 12-bit ADC
  - Built in high presicion 1.2V reference voltage
  - ±1.5% @VDD=1.8V~5.5V TA=25℃
  - ±2% @VDD=1.8V~5.5V TA=-20℃~85℃
  - internal reference Voltage:2.0V/2.4V
- Built in 128-byte program EEPROM
  - Rewritable up to 10,000 times

Product comparison:

PRODUCT	ROM	RAM	Pro EE	I/O	PWM	ADC	PACKAGE
SC8F6770	4Kx16	256x8	128x16	6	5	6	SOP8
SC8F6771	4Kx16	256x8	128x16	8	5	8	MSOP10 DFN10

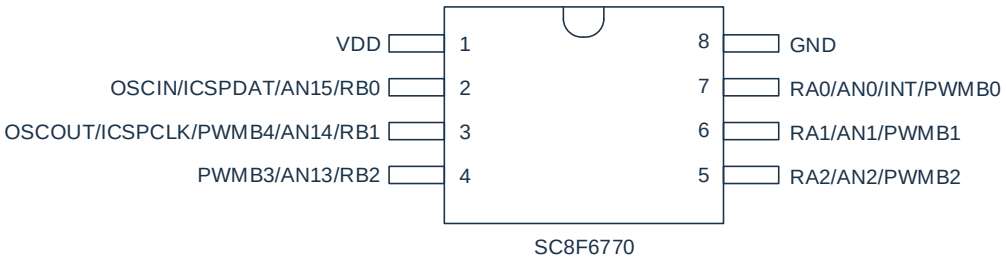
Note: ROM---- Read Only Memory Pro EE---- program EEPROM

1.2 System structure



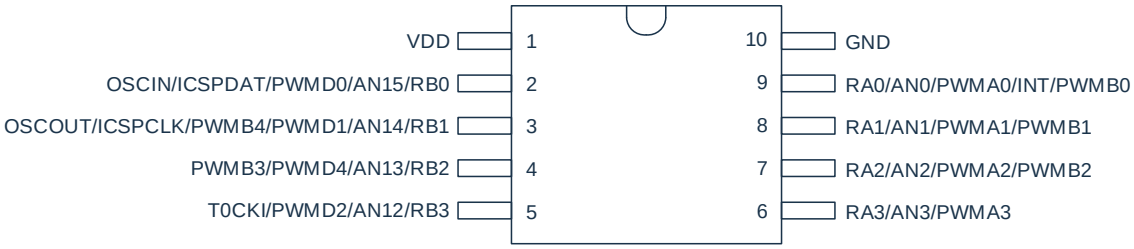
1.3 Pin configuration

1.3.1 SC8F6770



SC8F6770

1.3.2 SC8F6771



SC8F6771

SC8F677x Pin description and function:

Pin	IO	Define
VDD, GND	P	Power VDD/GND
OSCIN/OSCOUT	I/O	32.7632.768KHz Crystal input/output
RA0-RA3	I/O	Programmed as input pin or push-pull output pin,with pull-up resistor function, pull-down resistor function and interrupt on change in logic level function
RB0-RB3	I/O	Programmed as input pin or push-pull output pin,with pull-up resistor function, pull-down resistor function and interrupt on change in logic level function
ICSPCLK	I	Programmable input clock
ICSPDAT	I/O	Programmable data input/output
AN0-AN3 AN12-AN15	I	ADC Input
PWMx0-PWMx4	O	PWM0-4PWM output
INT	I	External Interruput
T0CKI	I	Timer0 External cystal input

1.4 System Configuration Registor

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

1. INTRC_SEL (Selection of internal oscillation frequency)
   - INTRC8M F_{OSC} selects internal 8MHz RC oscillator
   - INTRC16M F_{OSC} selects internal 16MHz RC oscillator
2. WDT (watchdog choice)
   - ENABLE Enable watchdog timer
   - DISABLE Disable watchdog timer
3. PROTECT (encyption)
   - DISABLE Disable FLASH code encyption
   - ENABLE Enable FLASH code encryption, after which the read value from burning the simulator is uncertain.
4. LVR_SEL (low voltage detection selection)
   - 1.8V
   - 2.0V
   - 2.5V
   - 3.0V
5. WDT_DIV (WDT prescaler coefficient control)
   - DISABLE The WDT prescaler can be selected from 1:128 through the OPTION_REG register
   - ENABLE The WDT prescaler can be selected from 3:384 through the OPTION_REG register
6. SLEEP_LVREN (LVR enable bit in sleep state)
   - DISABLE The LVR function is turned off in sleep mode
   - ENABLE The LVR function is turned on in the dormant state
7. ICSPPORT_SEL (Simulation port function selection)
   - ICSP ICSPCLK and DAT ports have been kept as simulation ports, and all functions cannot be used
   - NORMAL ICSPCLK and DAT ports are ordinary function ports

1.5 Online Serial Programming

It is possible to perform in-circuit serial programming on MCU in the final applications of circuits. Programming can be done through the following 4 lines:

- VDD
- GND
- DAT
- CLK

Users can use devices to build circuits before programming and only perform programming just before the product delivery. As such, it is possible to burn the latest firmware or specific firmware into the MCU.

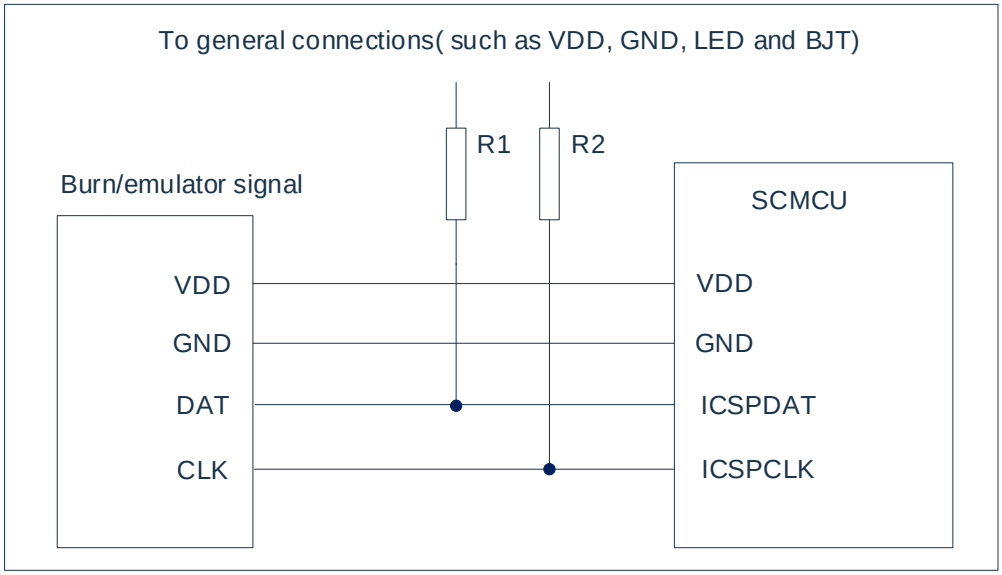


Fig 1-1: Typical In-Circuit Serial Programming example

In Fig1-1, R1, R2 are normal devices for galvanic isolation. They are usually resistors with resistance: R1≥4.7K, R2≥4.7K

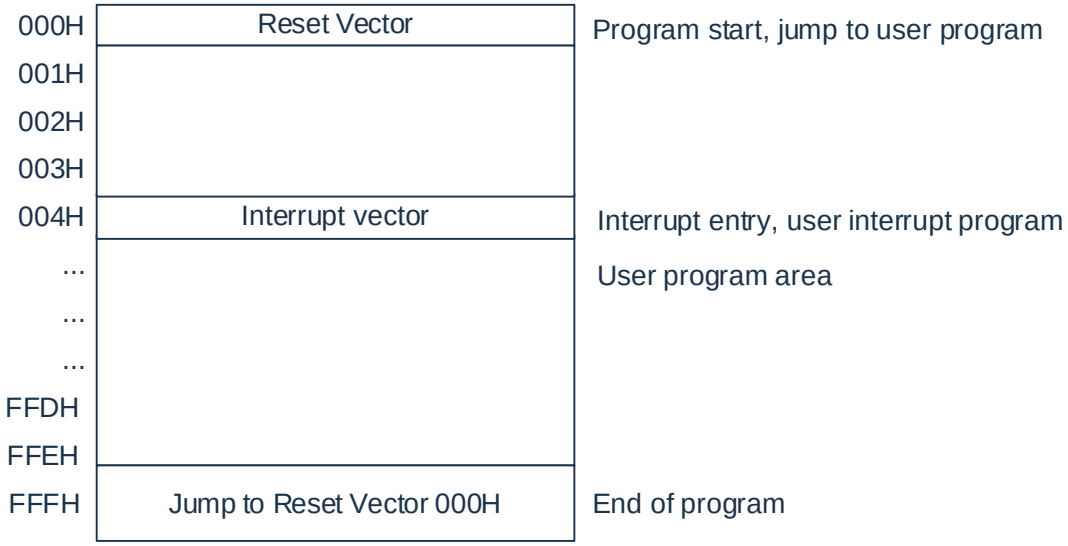
2. Central Processing Unit (CPU)

2.1 Memory

2.1.1 Program Memory

SC8F677x program memory space

ROM:4K



2.1.1.1 Reset Vector (0000H)

MCU has 1-byte long system reset vector (000H). 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(188H) Read/write register to indicate higher 4 bits in the table.
- TABLE_SPL(187H) Read/write register to indicate lower 8 bits in the table.
- TABLE_DATAH(189H) 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.

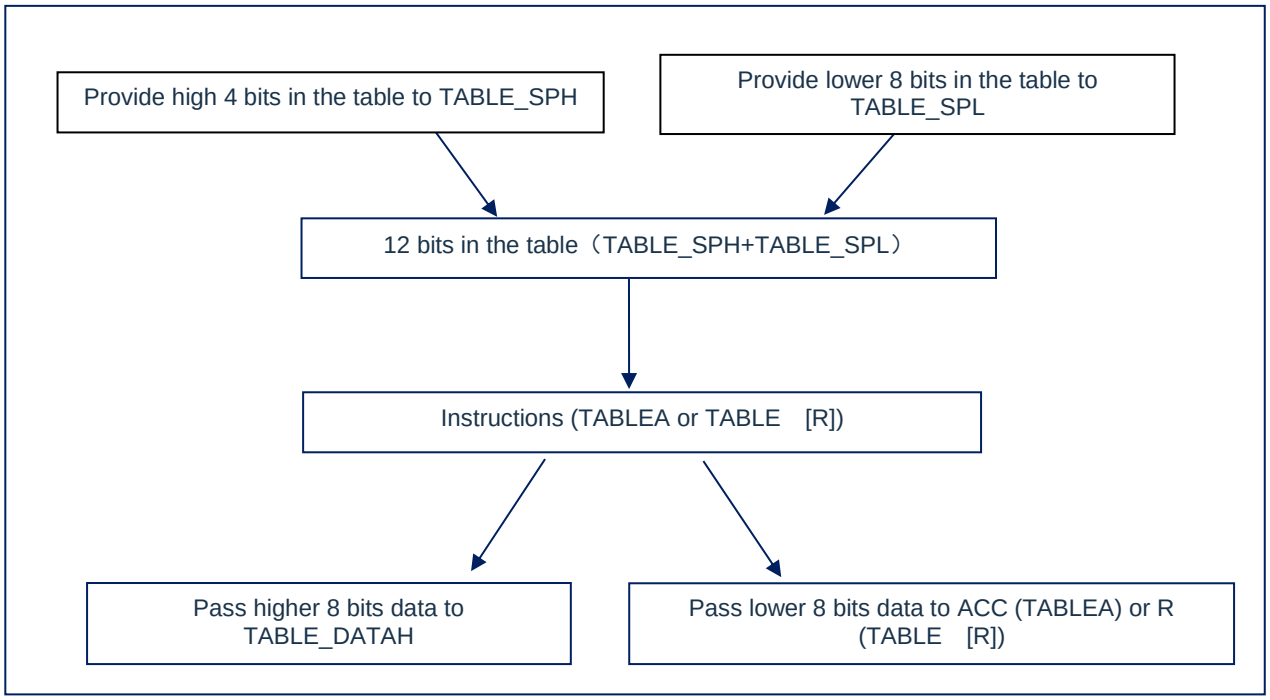


Fig 2-1: Flow chart for table usage

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 exeecution 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 地址
                CLR         PCLATH
                …
00FCH:          ADDR        PCL             ;ACC+PCL
00FDH:          JP          LOOP1           ;ACC=0, jump toLOOP1
00FEH:          JP          LOOP2           ;ACC=1, jump toLOOP2
00FFH:          JP          LOOP3           ;ACC=2, jump toLOOP3
0100H:          JP          LOOP4           ;ACC=3, jump to0000H address
0101H:          JP          LOOP5           ;ACC=4, jump to0001H address
0102H:          JP          LOOP6           ;ACC=5, jump to0002H address
```

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

2.1.2 Data Memory

List of data memory of SC8F677x

	address		address		address		address
INDF	00H	INDF	80H	INDF	100H	INDF	180H
TMR0	01H	OPTION_REG	81H	----	101H	----	181H
PCL	02H	PCL	82H	PCL	102H	PCL	182H
STATUS	03H	STATUS	83H	STATUS	103H	STATUS	183H
FSR	04H	FSR	84H	FSR	104H	FSR	184H
PORTA	05H	TRISA	85H	----	105H	----	185H
PORTB	06H	TRISB	86H	----	106H	----	186H
WPUA	07H	WPDB	87H	PIR2	107H	TABLE_SPL	187H
WPUB	08H	OSCCON	88H	PIE2	108H	TABLE_SPH	188H
IOCB	09H	----	89H	----	109H	TABLE_DATAH	189H
PCLATH	0AH	PCLATH	8AH	PCLATH	10AH	PCLATH	18AH
INTCON	0BH	INTCON	8BH	INTCON	10BH	INTCON	18BH
PIR1	0CH	EECON1	8CH	----	10CH	----	18CH
PIE1	0DH	EECON2	8DH	----	10DH	----	18DH
PWMD23H	0EH	EEDAT	8EH	----	10EH	----	18EH
PWM01DT	0FH	EEDATH	8FH	----	10FH	----	18FH
PWM23DT	10H	EEADR	90H	ANSEL0	110H	----	190H
TMR2	11H	PR2	91H	ANSEL1	111H	----	191H
T2CON	12H	----	92H	----	112H	----	192H
PWMCON0	13H	----	93H	----	113H	----	193H
PWMCON1	14H	----	94H	----	114H	----	194H
PWMTL	15H	IOCA	95H	----	115H	----	195H
PWMTH	16H	EEADRH	96H	----	116H	----	196H
PWMD0L	17H	WPDA	97H	----	117H	----	197H
PWMD1L	18H	----	98H	----	118H	----	198H
PWMD2L	19H	----	99H	----	119H	----	199H
PWMD3L	1AH	----	9AH	----	11AH	----	19AH
PWMD4L	1BH	----	9BH	----	11BH	----	19BH
PWMD01H	1CH	ADCON1	9CH	----	11CH	----	19CH
PWMCON2	1DH	ADCON0	9DH	----	11DH	----	19DH
PWM4TL	1EH	ADRESH	9EH	----	11EH	----	19EH
----	1FH	ADRESL	9FH	LVDCON	11FH	----	19FH
Universal register 96 byte	20H 6FH	Universal register 80byte	A0H BF C0 EFH	Universal register 80byte	120H 16FH	----	1A0H 1EFH
	70H -- 7FH	Fast memory spce 70H-7FH	F0H -- FFH	Fast memory spce 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, 80H-9FH, 100H-11FH, 180H-18BH.

Summary of special registers in SC8F677x Bank0

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
00H	INDF	Look-up for this unit will use FSR, not physical register.								xxxxxxxx
01H	TMR0	TIMER0 data register								xxxxxxxx
02H	PCL	Lower bit of program counter								00000000
03H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
04H	FSR	memory pointers for indirect addressing of data memory								xxxxxxxx
05H	PORTA	----	----	----	----	RA3	RA2	RA1	RA0	xxxxxxxx
06H	PORTB	----	----	----	----	RB3	RB2	RB1	RB0	xxxxxxxx
07H	WPUA	----	----	----	----	WPUA3	WPUA2	WPUA1	WPUA0	----0000
08H	WPUB	----	----	----	----	WPUB3	WPUB2	WPUB1	WPUB0	----0000
09H	IOCB	----	----	----	----	IOCB3	IOCB2	IOCB1	IOCB0	----0000
0AH	PCLATH	----	----	----	----	Write buffer of higher 4 bits of program counter				-----0000
0BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
0CH	PIR1	----	EEIF	----	----	----	PWMIF	TMR2IF	ADIF	-0---000
0DH	PIE1	----	EEIE	----	----	----	PWMIE	TMR2IE	ADIE	-0---000
0EH	PWMD23H	----	----	PWMD3[9:8]		----	----	PWMD2[9:8]		--00--00
0FH	PWM01DT	----	----	PWM01 dead zone delay time						--00000
10H	PWM23DT	----	----	PWM23 dead zone delay time						--00000
11H	TMR2	TIMER2 module register								00000000
12H	T2CON	CLK_SEL	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	00000000
13H	PWMCON0	CLKDIV[2:0]			PWM4EN	PWM3EN	PWM2EN	PWM1EN	PWM0EN	00000000
14H	PWMCON1	PWMIO_SEL[1:0]		PWM2DTEN	PWM0DTEN	----	----	DT_DIV[1:0]		0000--00
15H	PWMTL	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 SC8F677x 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	The low byte of the program counter (PC)								00000000
83H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
84H	FSR	Indirect data memory address pointer								xxxxxxxx
85H	TRISA	----	----	----	----	TRISA3	TRISA2	TRISA1	TRISA0	----1111
86H	TRISB	----	----	----	----	TRISB3	TRISB2	TRISB1	TRISB0	----1111
87H	WPDB	----	----	----	----	WPDB3	WPDB2	WPDB1	WPDB0	----0000
88H	OSCCON	----	IRCF2	IRCF1	IRCF0	----	----	SWDTEN	----	-101—0-
8AH	PCLATH	----	----	----	----	Write buffer for the upper 4 bits of the program counter				-----0000
8BH	INTCON	GIE	PEIE	T01E	INTE	RBIE	T0IF	INTF	RBIF	00000000
8CH	EECON1	EEPGD	----	----	----	WRERR	WREN	WR	RD	0--0x000
8DH	EECON2	EEPROM control register 2 (not a physical register)								--------
8EH	EEDAT	EEDAT7	EEDAT6	EEDAT5	EEDAT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0	xxxxxxxx
8FH	EEDATH	EEDATH7	EEDATH6	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0	xxxxxxxx
90H	EEADR	EEADR7	EEADR6	EEADR5	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0	00000000
91H	PR2	TIMER2 period register								00000000
95H	IOCA	----	----	----	----	IOCA3	IOCA2	IOCA1	IOCA0	----0000
96H	EEADRH	----	----	----	----	EEADRH3	EEADRH2	EEADRH1	EEADRH0	----0000
97H	WPDA	----	----	----	----	WPDA3	WPDA2	WPDA1	WPDA0	----0000
9CH	ADCON1	ADFM	CHS4	ADCS2	----	----	LDO_EN	LDO_SEL[1:0]		0----000
9DH	ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/ $\overline{\text{DONE}}$	ADON	00000000
9EH	ADRESH	High byte of ADC result register								xxxxxxxx
9FH	ADRESL	Low byte of ADC result register								xxxxxxxx

Summary of special registers in SC8F677x 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	The low byte of the program counter (PC)								00000000
103H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
104H	FSR	Indirect data memory address pointer								xxxxxxxx
107H	PIR2	----	----	----	----	----	----	RACIF	LVDIF	------00
108H	PIE2	----	----	----	----	----	----	RACIE	LVDIE	------00
10AH	PCLATH	----	----	---	---	Write buffer for the upper 4 bits of the program counter				----0000
10BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
110H	ANSEL0	----	----	----	----	ANS3	ANS2	ANS1	ANS0	----0000
111H	ANSEL1	ANS15	ANS14	ANS13	ANS12	----	----	----	----	0000----
11FH	LVDCON	LVD_RES	----	----	----	LVD_SEL2	LVD_SEL1	LVD_SEL0	LVDEN	0---0000

Summary of special registers in SC8F677x 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	The low byte of the program counter (PC)								00000000
183H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
184H	FSR	Indirect data memory address pointer								xxxxxxxx
187H	TABLE_SPH	Table low pointer								xxxxxxxx
188H	TABLE_SPH	----	----	----	----	Table high address				----xxxx
189H	TABLE_DATAH	High 8-bit data of the table								xxxxxxxx
18AH	PCLATH	----	----	----	----	Write buffer for the upper 4 bits of the program counter				----0000
18BH	INTCON	GIE	PEIE	T01E	INTE	RBIE	T01F	INTF	RBIF	00000000

2.2 Addressing Mode

2.2.1 Direct Addressing

Operate on RAM through accumulator (ACC)

Example: pass the value in ACC to 30H register

```
LD      30H,A
```

Example: pass the value in 30H register to ACC

```
LD      A,30H
```

2.2.2 Immediate Addressing

Pass the immediate value to accumulator (ACC).

Example: pass immediate value 12H to ACC

```
LDIA    12H
```

2.2.3 Indirect Addressing

Data memory can be direct or indirect addressing. Direct addressing can be achieved through INDF register, INDF is not physical register. When load/save value in INDF, address is the value in FSR register (lower 8 bits) and IRP bit in STATUS register (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
```

例：间接寻址清 RAM(20H-7FH)举例：

```
        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.

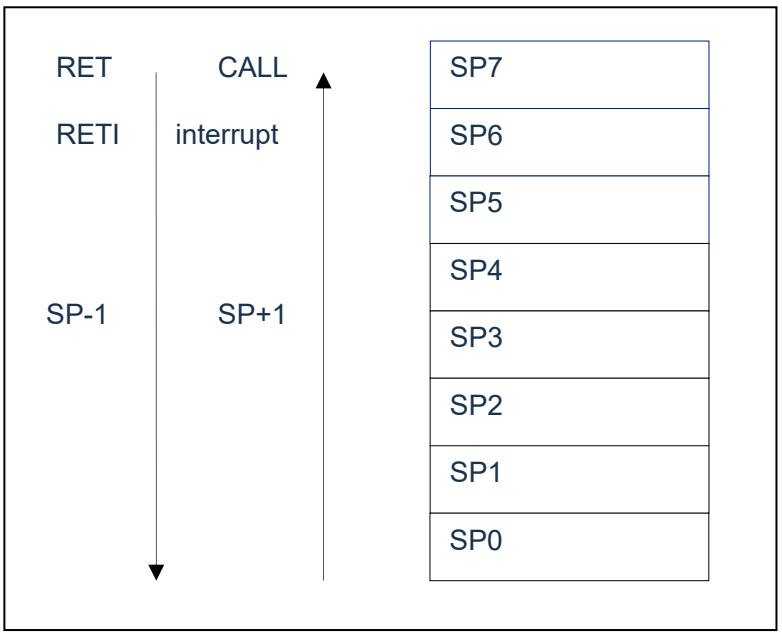


Fig 2-1: stack buffer working principle

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=	Result is 0；
	0=	Result is not 0
Bit1	DC:	Carry bit;
	1=	The 4th 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=	When carry happens at the highest bit or no borrow happens；
	0=	When no carry happens at the highest bit or borrow happens

TO and PD bit can reflect the reason for reset of chip. The following is the events which affects the TO and PD and the status of TO 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_{CPU})。

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

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 12 Bit, user can access lower 8 bits through PCL (02H). The higher 4 bits cannot be accessed. It can hold address for 4K×16Bit program. Passing a value to PCL will cause a short jump which range until the 256 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

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

Note:

1) SWDTEN is located at Bit1 of the OSCCON register.
2) if WDT configuration bit in CONFIG equals 1, then WDT is always enabled and is unrelated to the status of control bit of SWDTEN. if WDT configuration bit in CONFIG equals 0, then it is able to disable WDT using the control bit of SWDTEN.

3. System Clock

3.1 General

When clock 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.

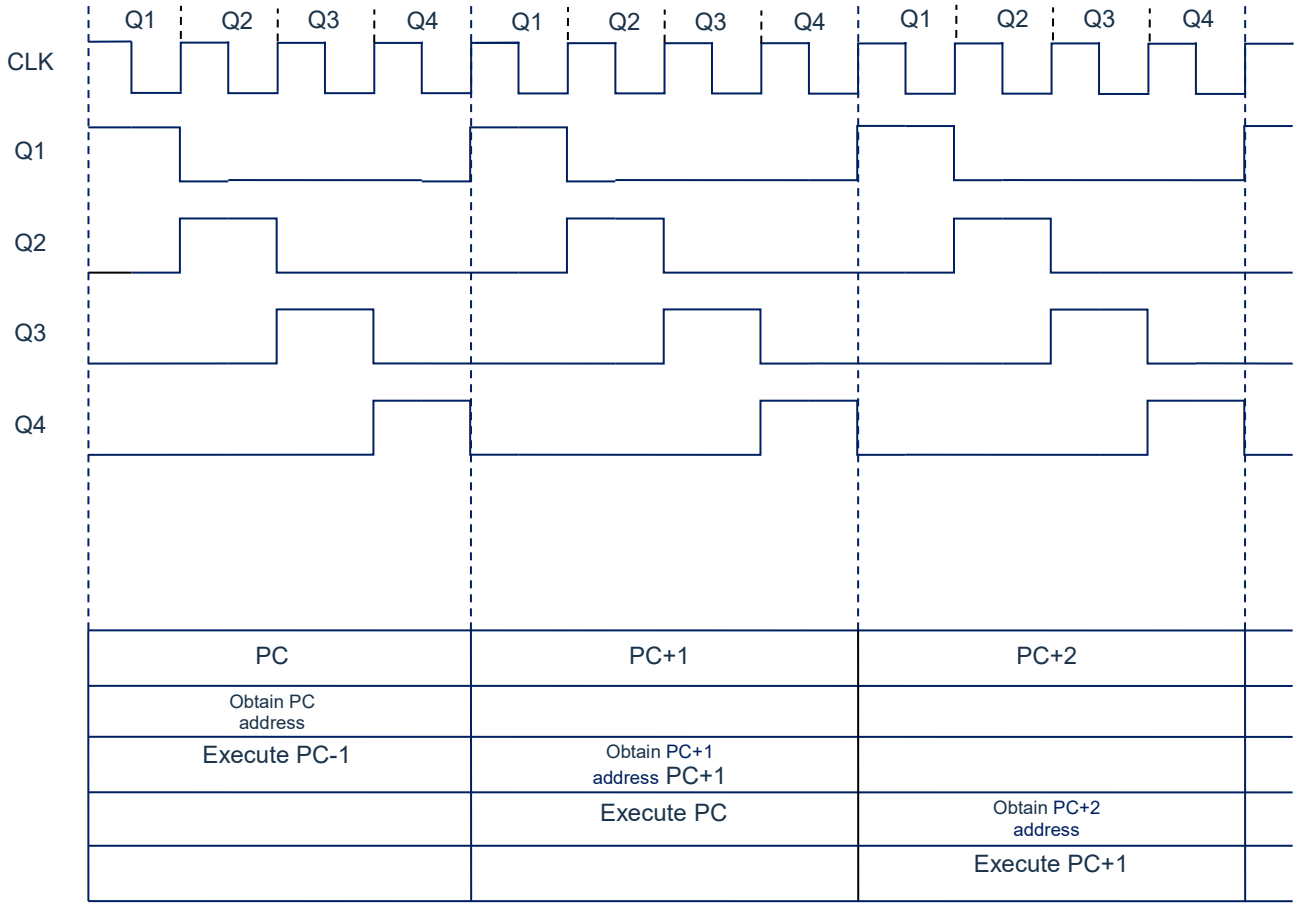


Fig 3-1: time series for clock and instruction period

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
16MHz	500ns	250ns

3.2 System Oscillator

he chip integrates 8M/16M RC oscillator.

3.2.1 Internal RC Oscillation

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

3.3 Reset Time

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

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

3.4 Oscillator Control Register

Oscillator control (OSCCON)register controls the system clock and frequency selection. Oscillator tune register OSCTUNE can tune the frequency of internal oscillationin the software.

Oscillator control register OSCCON (88H)

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

Bit7	Not used, read 0	
Bit6~Bit4	IRCF<2:0>:	Slection bit for frequency division of Internal oscillator
	111=	F_{SYS} = F_{OSC}/1
	110=	F_{SYS} = F_{OSC} /2
	101=	F_{SYS} = F_{OSC} /4 (default)
	100=	F_{SYS} = F_{OSC} /8
	011=	F_{SYS} = F_{OSC} /16
	010=	F_{SYS} = F_{OSC} /32
	001=	F_{SYS} = F_{OSC} /64
	000=	F_{SYS} = 32kHz (LFINTOSC)
Bit3~Bit2	Not used	
Bit1	SWDTEN:	Software enable or disable watchdog timer bit.
	1=	Enable WDT
	0=	Disable WDT (reset value)。
Bit0	Not used	

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

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.

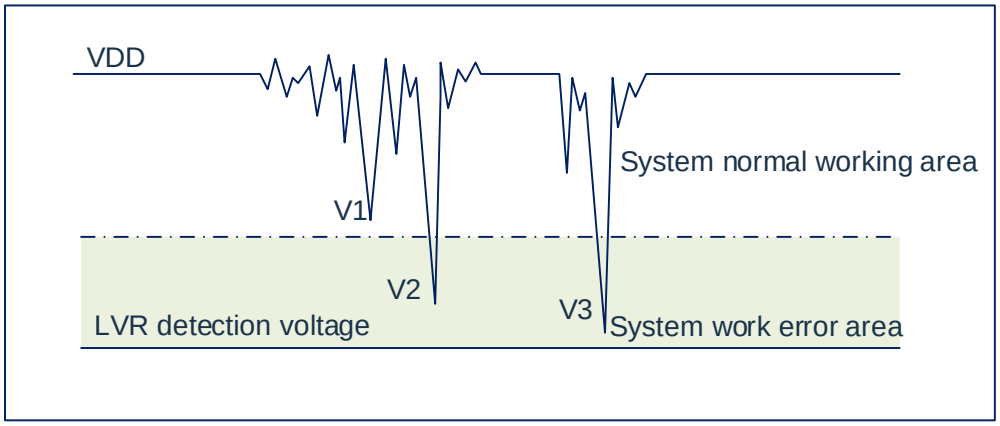


Fig 4-1: power off reset

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/PORTB 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
        ...
        LDIA    0A5H        ;turn off other functions
        LD      SP_FLAG,A
        CLRWDT              ;set sleep status memory register
        STOP                ;clear WDT
```

5.5 Sleep Mode Awaken Time

When MCU is awaken fromsleep mode, oscillation reset time is needed. This time is 16 F_{OSC} clock cycles + 135 T_{SYS} clock cycles in the internal high-speed oscillation mode, and 11 T_{SYS} clock cycles in the internal low-speed oscillation mode. 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})	$F_{SYS}=F_{OSC}$	$T_{WAIT}=16/F_{OSC}+135*1/F_{OSC}$
	$F_{SYS}=F_{OSC}/2$	$T_{WAIT}=16/F_{OSC}+135*2/F_{OSC}$
	…	…
	$F_{SYS}=F_{OSC}/64$	$T_{WAIT}=16/F_{OSC}+135*64/F_{OSC}$
Internal low-speed RC oscillation ($F_{LFINTOSC}$)	----	$T_{WAIT}=11/F_{LFINTOSC}$

6. I/O Port

Chip has 2 I/O port: PORTA, PORTB（max. of 8 I/O）. read/write port data register can directly read/write these ports.

Port	Bit	Pin Description	I/O
PORTA	0	Schmitt trigger input, push-pull output, AN0, PWM output	I/O
	1	Schmitt trigger input, push-pull output, AN1, PWM output	I/O
	2	Schmitt trigger input, push-pull output, AN2, PWM output	I/O
	3	Schmitt trigger input, push-pull output, AN3, PWM output	I/O
PORTB	0	Schmitt trigger input, push-pull output, AN15, PWM output, program data input/output, Oscillation input port	I/O
	1	Schmitt trigger input, push-pull output, AN14, PWM output, program clock input, Oscillation output port	I/O
	2	Schmitt trigger input, push-pull output, AN13, PWM output	I/O
	3	Schmitt trigger input, push-pull output, AN12, PWM output	I/O

< Table 6-1: port configuration summary >

6.1 I/O Port Structure

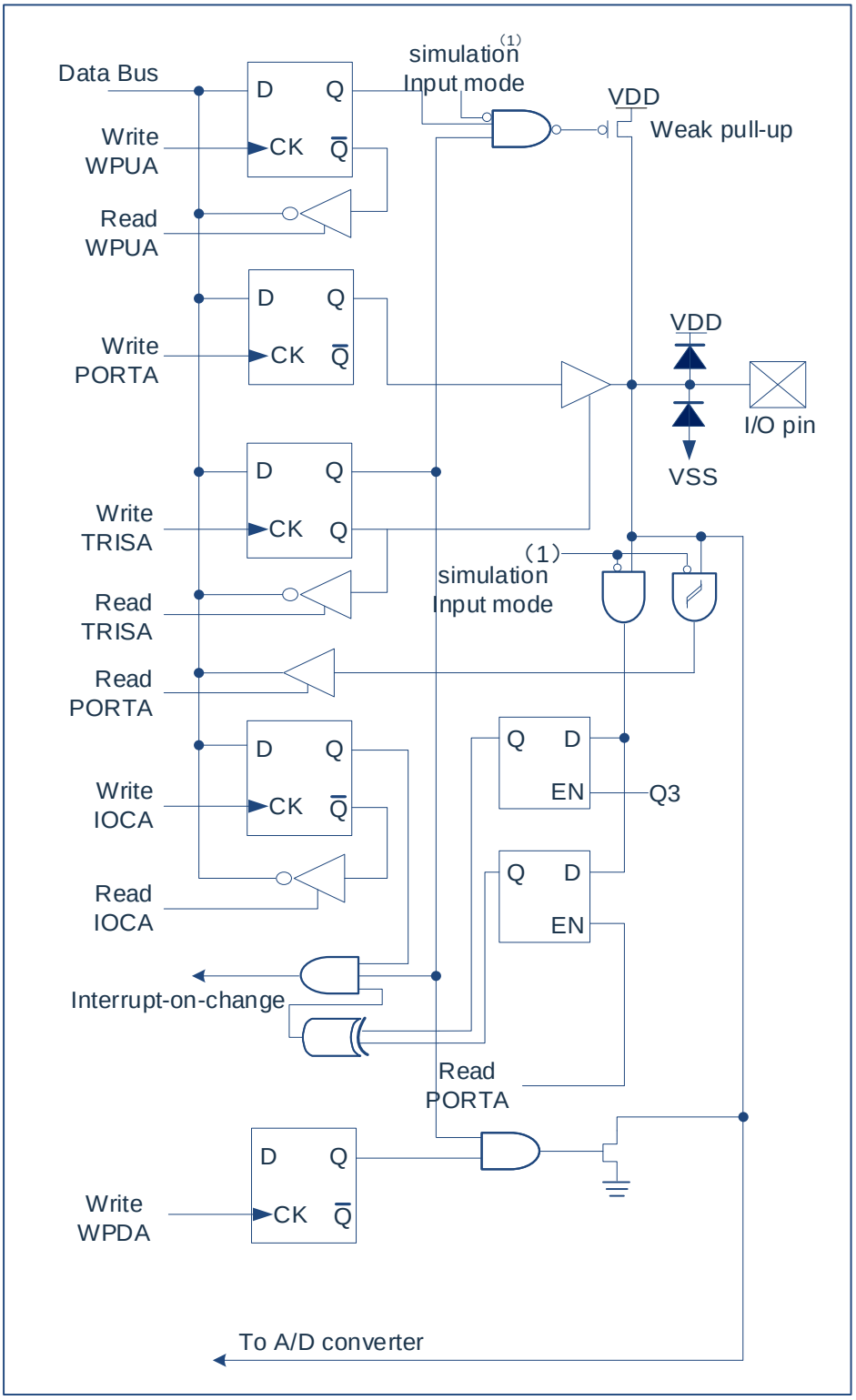


Fig 6-1: I/Oport structure

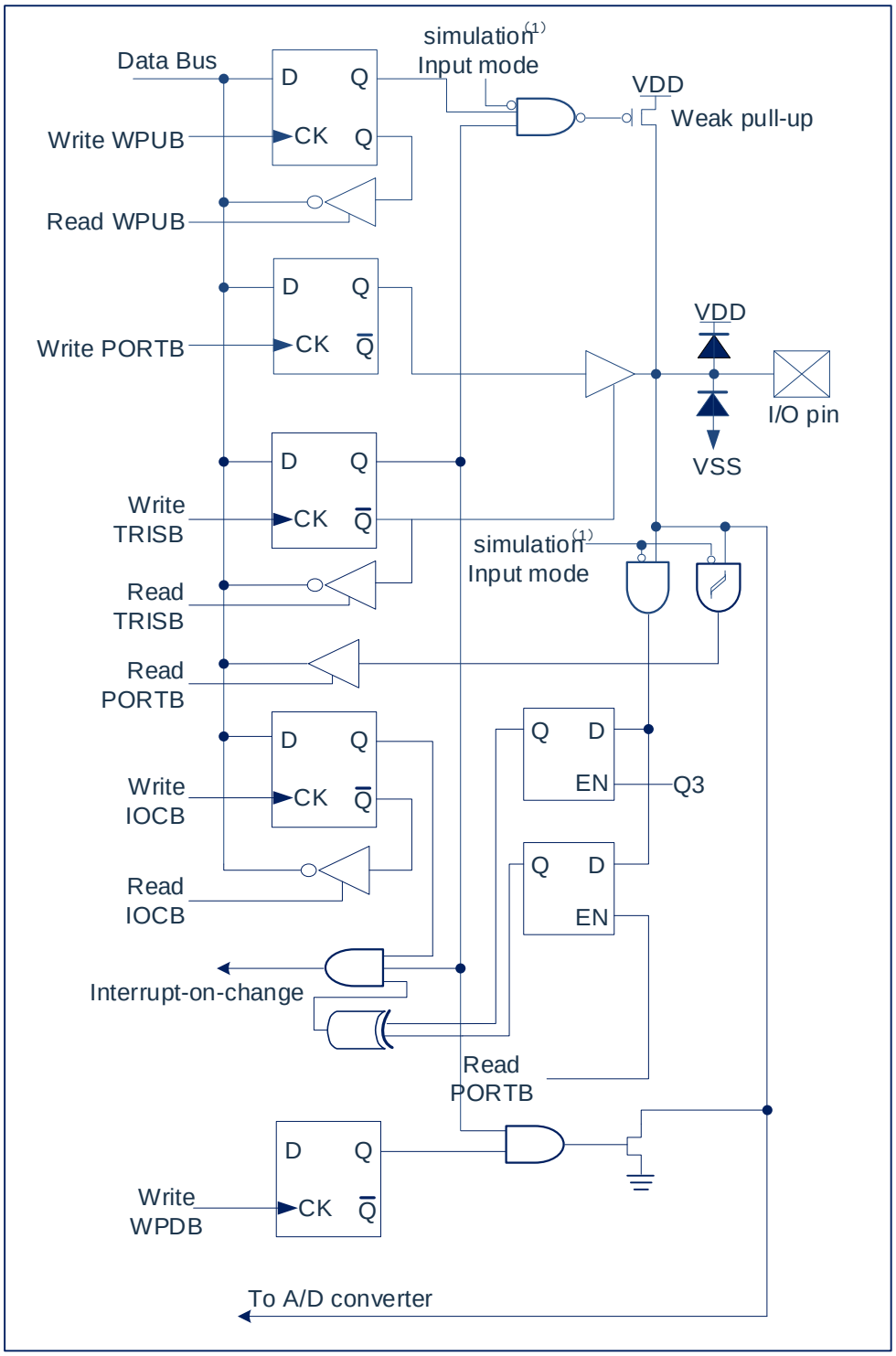


Fig 6-2: I/O port structure

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. I/O pins configured as analog inputs always read as 0.

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

PORTA data register PORTA (05H)

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

Bit7~Bit4 Unused

Bit3~Bit0 PORTA<3:0>: PORTA I/O pin bit;

1= Port pin level >V_{IH};

0= Port pin level <V_{IL}。

PORTA direction register TRISA(85H)

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

Bit7~Bit4 Unused

Bit3~Bit0 TRISA<3:0>: PORTA tri-state control bit;

1= PORTA pin set to be input; (tri-state);

0= PORTA pin set to be output

Example: procedure for PORTA

```
LDIA    B'00001100'     ; set PORTA<1:0> as output port, PORTA<3:2> as input port
LD      TRISA,A
LDIA    01H             ;PORTA<0> output high level, PORTA<1> output low level
LD      PORTA,A         ; since PORTA<3:2> are input ports, 0 or 1 does not matter
```

6.2.2 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	----	----	----	----	ANS3	ANS2	ANS1	ANS0
R/W	----	----	----	----	R/W	R/W	R/W	R/W
Reset value	----	----	----	----	0	0	0	0

Bit7~Bit4 Not used.

Bit3~Bit0 ANS<3:0>: Analog selection bit, select the digital or analog function of pin AN <3:0>

1= Analog input. The pins are assigned as analog inputs.

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

6.2.3 PORTA Pull-up resistor

Each PORTA pin has an internal weak pull-up that can be individually configured. The control bits WPUA<3:0> enable or disable each weak pull-up. When a port pin is configured as an output, its weak pull-up will be automatically cut off.

PORTA Pull-up resistor register WPUA(07H)

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

Bit7~Bit4 Not used.

Bit3~Bit0 WPUA<3:0>: Weak pull-up register bit.

1= Enable pull up.

0= Disable pull up.

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

6.2.4 PORTA Pull Down Resistance

Each PORTA pin has an internal weak pull-down that can be individually configured. The control bits WPDA<3:0> enable or disable each weak pull-down. When the port pin is configured as output, its weak pull-down will automatically cut off.

PORTApull down resistance register WPDA（97H）

97H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPDA	----	----	----	----	WPDA3	WPDA2	WPDA1	WPDA0
R/W	----	----	----	----	R/W	R/W	R/W	R/W
Reset value	----	----	----	----	0	0	0	0

Bit7~Bit4 Not used

Bit3~Bit0 WPDA<3:0>: 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.

6.2.5 PORTA Level Chage Interrupt Interrupt

All PORTA pins can be individually configured as level change interrupt pins. The control bit IOCA<3:0> 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	----	----	----	----	IOCA3	IOCA2	IOCA1	IOCA0
R/W	----	----	----	----	R/W	R/W	R/W	R/W
Reset value	----	----	----	----	0	0	0	0

Bit7~Bit4 Not used.

Bit3~Bit0 IOCA<3:0> 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 4Bit 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, IOCB, WPDB, ANSEL1, etc.

PORTB data register PORTB(06H)

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

Bit7~Bit4 Not used.

Bit3~Bit0 PORTB<3:0>: 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	----	----	----	----	TRISB3	TRISB2	TRISB1	TRISB0
R/W	----	----	----	----	R/W	R/W	R/W	R/W
Reset value	----	----	----	----	1	1	1	1

Bit7~Bit4 Not used.

Bit3~Bit0 TRISB<3:0>: 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'00000011'    ; set PORTB<1:0> as input port, others as output port
LD      TRISB,A
```

6.3.2 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	----	----	----	----
R/W	R/W	R/W	R/W	R/W	----	----	----	----
Reset value	0	0	0	0	----	----	----	----

Bit7~Bit4 ANS<15:12>: Analog selection bits, select the analog or digital functions of pin AN <15:12>.
1= Analog input. The pins are assigned as analog inputs.
0= Digital I/O. Pins are assigned to ports or special functions.

Bit3~Bit0 No used.

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<3:0> enable or disable each weak pull-down. When the port pin is configured as output, its weak pull-down will automatically cut off.

PORTB pull down resistance register WPDB (87H)

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

Bit7~Bit4 No used.
Bit3~Bit0 WPDB<3:0>: 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 Pull up Resistance

Each PORTB pin has an internal weak pull up that can be individually configured. The control bits WPUB<3:0> 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	----	----	----	----	WPUB3	WPUB2	WPUB1	WPUB0
R/W	----	----	----	----	R/W	R/W	R/W	R/W
Reset value	----	----	----	----	0	0	0	0

Bit7~Bit4 No used.

Bit3~Bit0 WPUB<3:0>: 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.

6.3.5 PORTB Level Chage Interrupt Interrupt

All PORTB pins can be individually configured as level change interrupt pins. The control bit IOCB<3:0> allows or disables the interrupt function of each pin. Disable pin level change interrupt function when power on reset.

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

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

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

-Clear the flag bit RBIF.

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

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

PORTB level change interrupt register IOCB(09H)

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

Bit7~Bit4	Not used.	
Bit3~Bit0	IOCB<3:0>	Control bit of level change interrupt of PORTB.
	1=	Enable level change interrupt
	0=	Disable level change interrupt

6.4 I/O Usage

6.4.1 Write I/O Port

The I/O port registers of the chip, like general general-purpose registers, can be written by data transfer instructions, bit manipulation instructions, etc.

Example: write I/O port program

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

6.4.2 Read I/O Port

Example: write I/O port program

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

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

6.5 Precautions for I/O Port Usage

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

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

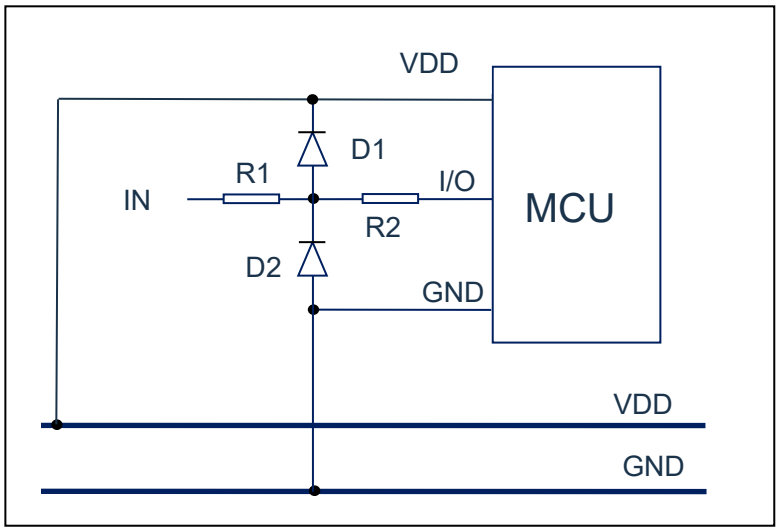


Fig 6-3: The input voltage is not within the specified range

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

7. Interrupt

7.1 Interrupt General

The chip has the following interrupt source:

- TIMER0 overflow interrupt
- TIMER2 match interrupt
- PORTA level change interrupt
- PORTB level change interrupt
- AD interrupt
- PWM interrupt
- INT 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.

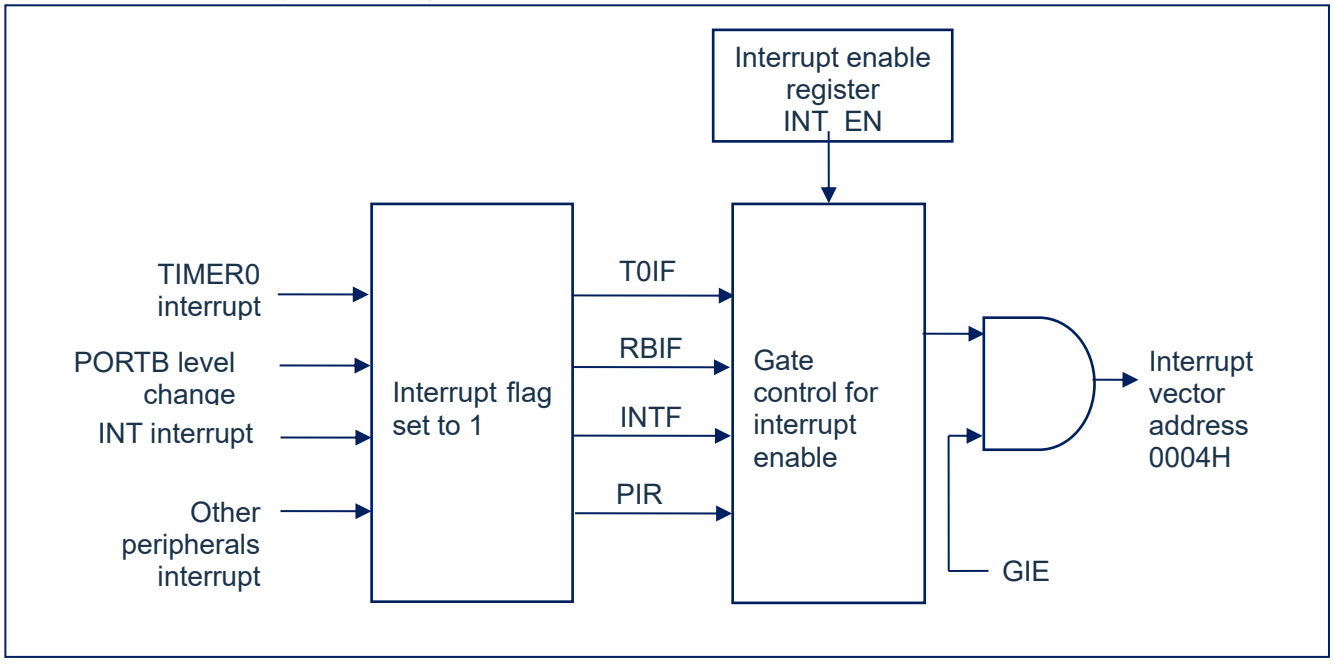


Fig 7-1: interrupt theory

7.2 Interrupt control Register

7.2.1 Interrupt Control Register

The interrupt control registerINTCON is a readable and writable register, including the allowable and flag bits for TMR0 register overflow and PORTB port level change interrupt.

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

Interrupt control register INTCON (0BH)

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

Bit7 GIE: Global interrupt enable bit;
1= Enable all unshielded interrupt;
0= Disable all interrupt

Bit6 PEIE: Peripherals interrupt enable bit;
1= Enable all unshielded peripherals interrupt;
0= Disable all peripherals interrupt.

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

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

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

Bit2 T0IF: TIMER0 overflow interrupt enable bit (2);
1= TMR0 register overflow already (must clear through software);
0= TMR0 register not overflow

Bit1 INTF: INT external interrupt flag bit;
1= INT external interrupt happens (must clear through software);
0= INT external interrupt not happen

Bit0 RBIF: PORTB level change interrupt flag bit;
1= The level of at least one pin in the PORTB port has changed (must clear through software);
0= None of the PORTB universal I/O pin status has changed.

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

7.2.2 Peripherals Interrupt Enable Register

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

Peripherals interrupt enable register PIE1 (0DH)

0DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE1	----	EEIE	----	----	----	PWMIE	TMR2IE	ADIE
R/W	----	R/W	----	----	----	R/W	R/W	R/W
Reset value	----	0	----	----	----	0	0	0

Bit7 Not used.

Bit6 EEIE: EEDATA interrupt enable bit;
1= Enable EEDATA write interrupt;
0= Disable EEDATA write interrupt.

Bit5~3 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 ADIE: A/D converter (ADC)interrupt enable bit;
1= enable ADC interrupt;
0= disable ADC interrupt

Peripherals interrupt enable register PIE2(108H)

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

Bit7~2 Not used.

Bit1 RACIE: PORTA interrupt-on-change enable bit
1= Enable PORTA interrupt on change;
0= Disable PORTA interrupt on change

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

7.2.3 Peripherals Interrupt Request Register

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

Peripherals interrupt request register PIR1(0CH)

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

Bit7 Not used.

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

Bit5~3 Not used.

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

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

Bit0 ADIF: A/D converter interrupt flag bit;
1= A/D conversion complete (must clear through software);
0= A/D conversion not complete or not start.

Peripherals interrupt request register PIR2(107H)

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

Bit7~2 Not used.

Bit1 RACIF: PORTA level change interrupt flag bit;
1= PORTA level change interrupt flag bit happens (must clear through software);
0= PORTA level change interrupt flag bit not happen

Bit0 LVDIF: LVD interrupt flag bit
1= The power supply voltage is lower than the voltage point set by LVD (must clear through 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.

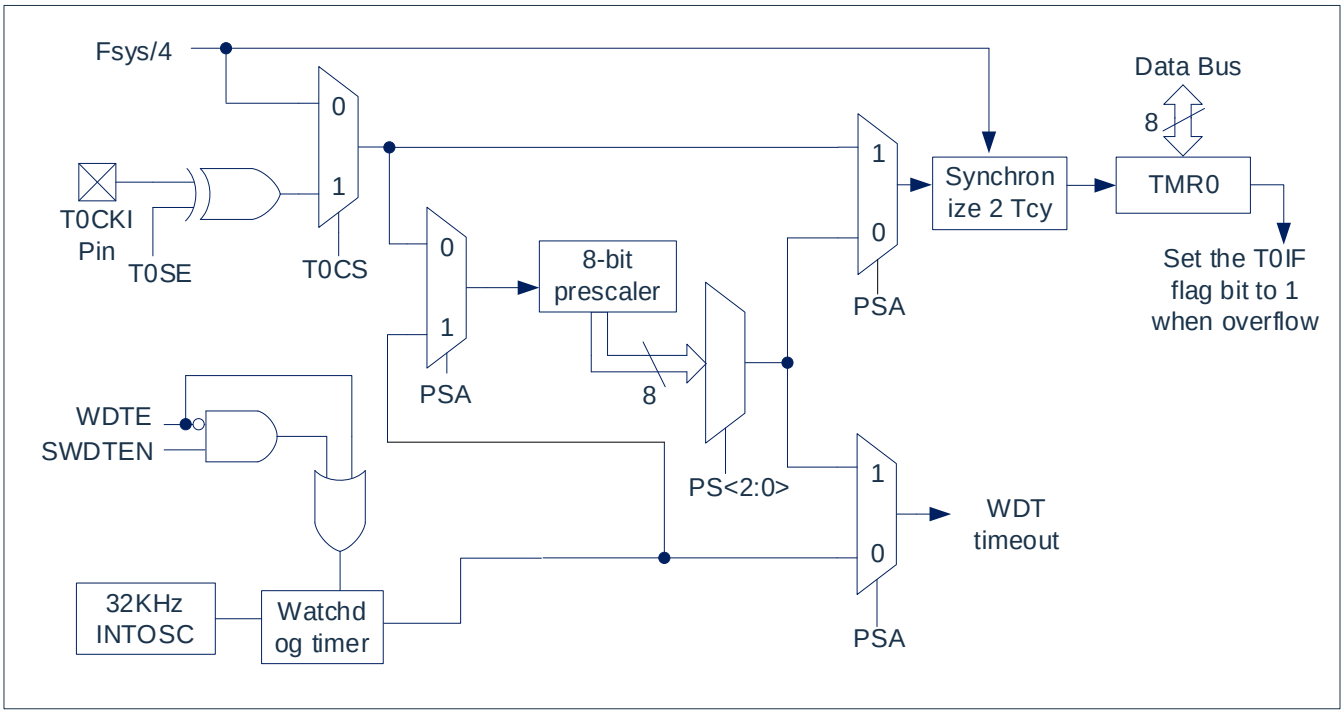


Fig 8-1: TIMER0/WDT mod structure

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 2.6 prescaler register (OPTION_REG).

8-bit timer/counter TMR0(01H)

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

OPTION_REG register (81H)

81H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION_REG	----	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
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 (WDT_DIV=DISABLE)	WDT Frequency division ratio (WDT_DIV=ENABLE)
0	0	0	1:2	1:1	1:3
0	0	1	1:4	1:2	1:6
0	1	0	1:8	1:4	1:12
0	1	1	1:16	1:8	1:24
1	0	0	1:32	1:16	1:48
1	0	1	1:64	1:32	1:96
1	1	0	1:128	1:64	1:192
1	1	1	1:256	1:128	1:384

9. TIMER2

9.1 TIMER2 General

TIMER1 mod is a 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 post-division ratio (1:1 to 1:16).

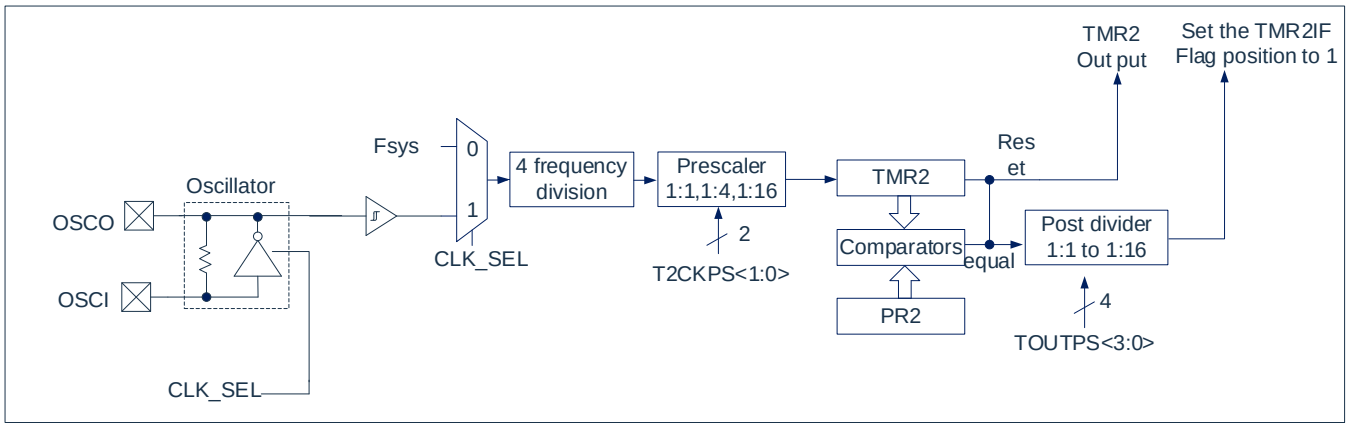


Fig 10-1: TIMER2 structure

9.2 Working Principle of TIMER2

The input clock of the TIMER2 module is the system clock (Fsys) or external 32.768kHz oscillation. After the clock is divided by 4, it is input to the TIMER2 prescaler, 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:

- Write to the TMR2 register
- Write to the T2CON register
- Any device reset occurs (power-on reset, watchdog timer reset, or undervoltage reset).

Note:
1. When TMR2ON is 0, the TMR2 register is forced to clear
2. The reset value of the PR2 register is 00H. When using TIMER2, please set the PR2 register first to avoid false triggering of the match

9.3 TIMER2 related register

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

TIMER2 data register TMR2(11H)

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

TIMER2 control register T2CON(12H)

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

Bit7 CLK_SEL: Clock source selection;
1= Select external 32.768kHz oscillation/4 as TMR2 clock source (can continue to count in sleep state);
0= Select the internal F_{CPU} as the TMR2 clock source.

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

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

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

Note: To enable LP oscillation, the theoretical value of the required start-up time is 32ms, which may be larger in practice.

10. Analog to Digital Conversion (ADC)

10.1 ADC general

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

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

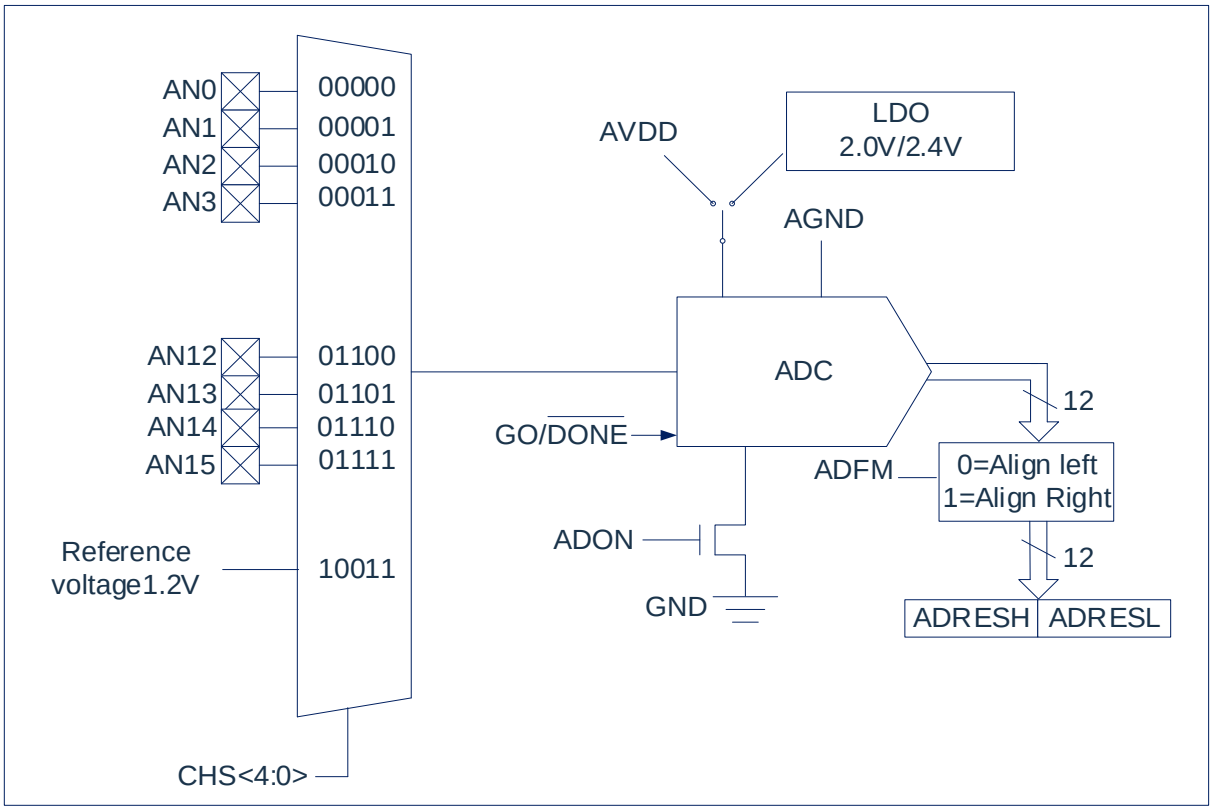


Fig 10-1: ADC structure

10.2 ADC configuration

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

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

10.2.1 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.

10.2.2 Channel selection

The CHS bit of the ADCON0 and ADCON1 register 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.

10.2.3 ADC internal reference voltage

The chip has a built-in reference voltage. To detect the reference voltage, set the CHS[4:0] bits to 10011.

10.2.4 ADC reference voltage

The ADC reference voltage is always provided by the chip's VDD and GND. The internal reference voltage can be 2.0V/2.4V. When selecting the internal reference voltage, you need to select a slower conversion clock, refer to the chapter on conversion clock.

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

10.2.5 Converter clock

The ADCS bit of the ADCON0 register can be set by software to select the clock source for conversion. There are 6 possible clock frequencies to choose from:

- Fosc/4
- Fosc/8
- Fosc/16
- Fosc/32
- Fosc/64
- Fosc/128

The time to complete one-bit conversion is defined as T_{ADCCLK}. A complete 12-bit conversion requires 16 T_{ADCCLK} 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=5V)

ADC clock selection		AD conversion time	
ADC clock source	ADCS<2:0>	Fosc = 16MHz	Fosc = 8MHz
Fosc/4	010		8μs
Fosc/8	011	8μs	16μs
Fosc/16	100	16μs	32μs
Fosc/32	101	32μs	64μs
Fosc/64	110	64μs	128μs
Fosc/128	111	128μs	256μs

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

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

Reference voltage	Working voltage (V)	Fastest division setting		Conversion rate (ksps)
		Fosc = 16MHz	Fosc = 8MHz	
VDD	3.0~5.5	Fosc/16	Fosc/8	62.5
VDD	2.5~5.5	Fosc/32	Fosc/16	31.3
2.4	2.6~5.5	Fosc/64	Fosc/32	15.6
2.0	2.5~5.5	Fosc/64	Fosc/32	15.6

10.2.6 ADC Interrupt

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

10.2.7 Output Formatting

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

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

10.3 ADC working principle

10.3.1 Start conversion

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

10.3.2 Complete conversion

When the conversion is complete, the ADC mod will:

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

10.3.3 Stop conversion

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

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

10.3.4 Working principle of ADC in sleep mode

Note: ADC module cannot wake up the chip in sleep mode.

10.3.5 AD conversion procedure

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

1. port configuration:
   - Disable pin output driver (see TRIS register);
   - Configure the pin as an analog input pin.
2. configuration ADC mod:
   - Select ADC reference voltage (when the reference voltage is switched from VDD to the internal LDO, a delay of more than 100us is required before AD conversion can be performed);
   - Select ADC conversion clock;
   - Select ADC input channel;
   - Choose the format of the result;
   - Start the ADC mod.
3. configuration ADC interrupt (optional):
   - Clear ADC interrupt flag bit;
   - Allow ADC interrupt;
   - Allow peripherals interrupt;
   - Allow global interrupt.
4. Wait for the required acquisition time.
5. Set GO/$\overline{\text{DONE}}$ to 1 to start conversion.
6. Wait for the ADC conversion to end by one of the following methods:
   - Query GO/ ("$\overline{\text{DONE}}$") bit
   - Wait for ADC interrupt (allow interrupt).
7. Read ADC results.
8. Clear the ADC interrupt flag bit (if interrupt is allowed, this operation is required).
9. When the GO/$\overline{\text{DONE}}$ bit changes from 1 to 0 or ADIF changes from 0 to 1, it is necessary to wait at least two TAD time before AD conversion can be started again.

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
```

10.4 ADC Related Register

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

AD control register ADCON0 (9DH)

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

Bit7~Bit6 ADCS<1:0>: A/D conversion clock selection bit.
ADCS<2:0> ADCS2 in ADCON1 register
000= Reserve
001= Reserve
010= Fosc/4
011= Fosc/8
100= Fosc/16
101= Fosc/32
110= Fosc/64
111= Fosc/128

Bit5~Bit2 CHS<3:0>: Analog channel selection bit.
CHS<4:0> CHS4 in ADCON1 register
00000= AN0
00001= AN1
00010= AN2
00011= AN3
00100~01011 Reserve
01100= AN12
01101= AN13
01110= AN14
01111= AN15
10000~10010 Reserve
10011= 1.2V (fixed reference voltage)
other= Reserve

Bit1 GO/$\overline{DONE}$: 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.
When the GO/$\overline{DONE}$ bit changes from 1 to 0 or ADIF changes from 0 to 1, it is necessary to wait at least two TAD time before AD conversion can be started again.
0= A/D conversion complete or not in progress.

Bit0 ADON: ADC enable bit.
1= Enable ADC;
0= Disable ADC, not consuming current.

AD data register high bit ADCON1(9CH)

9CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON1	ADFM	CHS4	ADCS2	---	---	LDO_EN	LDO_SEL1	LDO_SEL0
R/W	R/W	R/W	R/W	---	---	R/W	R/W	R/W
Reset value	0	0	0	---	---	0	0	0

Bit7 ADFM: AD conversion result format selection bit;
1= Right alignment
0= left alignment

Bit6 CHS4 Enable in combination with CHS3~0 of ADCON0

Bit5 ADCS2 Enable in combination with ADCS1~0 of ADCON0

Bit4~Bit3 Not used, read 0.

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

Bit1~Bit0 LDO_SEL<1:0>: Reference voltage selection bit
00= Disabled (Note: When the internal LDO is selected as the reference voltage, it is forbidden to be 00)
01= 2.0V
10= 2.4V
11= 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
R/W	R	R	R	R	R	R	R	R
Reset value	X	X	X	X	X	X	X	X

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

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

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

Bit7~Bit4 ADRES<3:0>: ADC result register bit.
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
R/W	----	----	----	----	----	----	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
R/W	R	R	R	R	R	R	R	R
Reset value	X	X	X	X	X	X	X	X

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

The 9-2 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 9nd to 2th digits.

11. PWM Mode

11.1 PWM General

The chip has a built-in programmable 10-bit PWM module, which can be configured as 4 channels of common cycle, independent duty cycle output PWM0~3 and 1 channel of independent cycle, independent duty cycle output PWM4; among them, PWM0/PWM1, PWM2/PWM3 It can be configured with complementary forward and reverse output.

11.2 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= F_{OSC}/128
110= F_{OSC}/64
101= F_{OSC}/32
100= F_{OSC}/16
011= F_{OSC}/8
010= F_{OSC}/4
001= F_{OSC}/2
000= F_{OSC}/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.
11= PWM is allocated in group A, PWM0-RA0, PWM1-RA1, PWM2-RA2, PWM3-RA3, Reserve
10= PWM is allocated in group B, PWM0-RA0, PWM1-RA1, PWM2-RA2, PWM3-RB2, PWM4-RB1
01= Reserve
00= PWM is allocated in group D, PWM0-RB0,PWM1-RB1, PWM2-RB3, Reserve, PWM4-RB2

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 registerPWMTL (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.

PWM4 Period 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.

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 PWMDx[9:8] does not take effect immediately, it takes effect after writing to PWMDxL.

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 RegisterPWM23DT (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.

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.

11.3 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;

11.4 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.

11.5 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.

11.6 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.

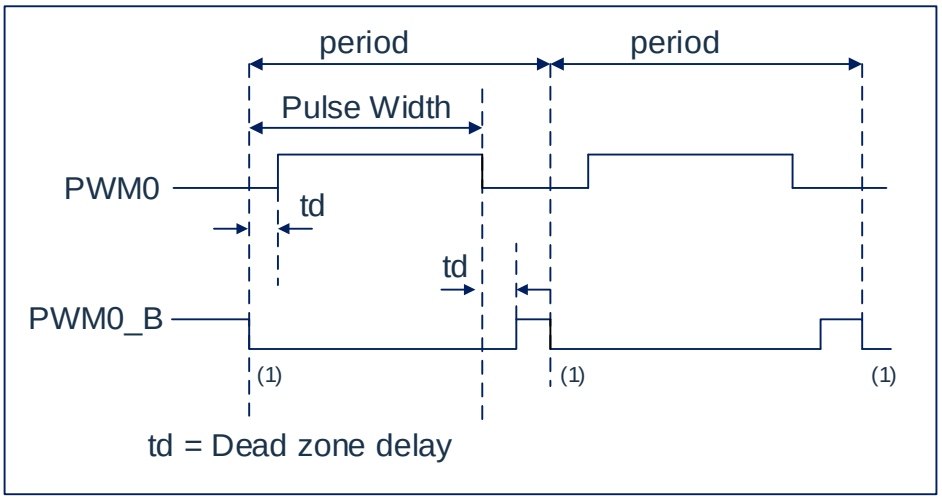


Fig 11-1: Example of PWM dead zone delay output

The dead time calculation formula is:

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

11.7 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[5:4] 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.

12. Program EEPROM and Program Memory Control

12.1 General

The devices in this series have 4K 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 12-bit EEPROM cell address to be read.

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

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

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

12.2 Related Register

12.2.1 EEADR and EEADRH Register

The EEADR and EEADRH registers can address up to 128 bytes of program EEPROM or up to 4K bytes of program memory.

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

12.2.2 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	---	---	---	---	EEADRH3	EEADRH2	EEADRH1	EEADRH0
R/W	---	---	---	---	R/W	R/W	R/W	R/W
Reset value	---	---	---	---	0	0	0	0

Bit7~Bit4 Not used，read 0.

Bit3~Bit0 EEADRH<3:0>: Specify the upper 4 address of the program memory read operation.

EEPROM control register EECON1(8CH)

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

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

Bit6~Bit4 Not used.

Bit3 WRERR: EEPROM error flag bit;
1= Write operation terminated prematurely（any WDT reset or undervoltage reset during normal operation）;
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.

12.3 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

```
        LD      A,EE_ADD        ; Put the address to be read into the EEADR register
        LD      EEADR,A
        CLRB    EECON1,EEPGD    ; Select program EEPROM
        SETB    EECON1,RD       ;Enable read signal
        NOP                     ; Here to read the data, the NOP instruction must be added
        NOP
        LD      A,EEDAT         ;read and load data to ACC
```

12.4 Write Program EEPROM

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

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

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

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

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

Example: write program EEPROM

```
        LD      A,EE_ADDL       ; Put the address to be written into the EEADR register
        LD      EEADR,A
        LD      A,EE_DATAL      ; Put the lower 8 bits of data to be written into the EEDAT register
        LD      EEDAT,A
        LD      A,EE_DATAH      ; Put the high 8 bits of data to be written into the EEDATH register
        LD      EEDATH,A
        CLRWDT
        CLRB    EECON1,EEPGD
        SETB    EECON1,WREN     ; Allow write operations
        CLRB    INTCON,GIE      ;Disable interrupt
        SZB     INTCON,GIE      ;Ensure interrupt is disabled
        JP      $-2
        LDIA    055H            ; Write 55H to EECON2
        LD      EECON2,A
        LDIA    0AAH            ; Write 0AAH to EECON2
        LD      EECON2,A
        SETB    EECON1,WR       ;Enable write signal
        NOP
        NOP
        CLRWDT
        CLRB    EECON1,WREN     ; End of writing, turn off the write enable bit
        SETB    INTCON,GIE
```

12.5 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,EE_ADDL       ; Put the address to be read into the EEADR register
        LD      EEADR,A
        LD      A,EE_ADDH       ; Put the high bit of the address to be read into 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      EE_DATL,A
        LD      A,EEDATH
        LD      EE_DATH,A
```

12.6 Write Program Memory

Program memoryis read only, cannot be written.

12.7 Precautions on Program EEPROM

12.7.1 Programming Time for Program EEPROM

The program EEPROM programming time is not fixed. the time required to program different data is about 4.6ms, and the CPU stops working during the program, and the program needs to do the relevant processing.

12.7.2 Write Verification

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

12.7.3 Protection to Avoid Writing Wrongly

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

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

- Undervoltage
- Power glitch
- Software failure

13. Low Voltage detection (LVD)

13.1 LVD Mod General

The 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.

13.2 LVD Related Register

There is 1 register related to LVD mod.

LVD control register LVDCON(11FH)

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

Bit7 LVD_RES: LVD output result
0= VDD> Set LVD voltage;
1= VDD< Set LVD voltage;

Bit6~Bit4 Not used.

Bit3~Bit1 LVD_SEL[2:0]: LVD voltage selection
000= 2.2V;
001= 2.4V;
010= 2.7V;
011= 3.0V;
100= 3.3V;
101= 3.7V;
110= 4.0V;
111= 4.3V;

Bit0 LVDEN: LVD enable bit
0= Disable;
1= Enable;

13.3 LVD Operation

By setting the LVD voltage value V_{SET} in the LVDCON register, after enabling LVDEN, the frequent fluctuation of LVD output results caused by the fluctuation of power supply voltage near V_{SET} is reduced due to the filtering process inside the chip. LVD_RES and LVDIF can be set high only when the power supply voltage is lower than V_{SET} for 1.5ms. An LVD interrupt is generated if the corresponding interrupt enable bit is enabled.

LVD cannot be used for interrupt wake up mode

14. Electrical parameters

14.1 Limit parameters

Power supply voltage…… GND-0.3V~GND+6V
Storage temperature range…… 50℃~125℃
Operating temperature…… - 20℃~85℃
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.

14.2 DC Electrical Characteristic

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		VDD	condition				
VDD	Operating voltage	-	F_{SYS}=16MHz	V_{LVR2}		5.5	V
		-	F_{SYS}=8MHz	V_{LVR1}		5.5	V
I_{DD}	Operating voltage	5V	F_{SYS}=16MHz		3		mA
		3V	F_{SYS}=16MHz		2		mA
		5V	F_{SYS}=8MHz		2		mA
		3V	F_{SYS}=8MHz		1		mA
		5V	Program EEPROM		4		mA
		3V	Program EEPROM		3		mA
I_{STB}	Quiescent current	5V	LVR=DIS WDT=DIS	-	0.6	2	μA
		3V	LVR=DIS WDT=DIS	-	0.4	2	μA
		5V	LVR=DIS WDT=EN		3.5		μA
		3V	LVR=DIS WDT=EN		3.0		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	-	----	2.5	-	5.5	V
R_{PH}	Pull-up resistor value	5V	V_O=0.5VDD	-	32	-	KΩ
		3V	V_O=0.5VDD	-	52	-	KΩ
R_{PD}	Pull-down resistor value	5V	V_O=0.5VDD	-	32	-	KΩ
		3V	V_O=0.5VDD	-	48	-	KΩ
I_{OL}	Output sink current	5V	V_{OL}=0.3VDD	-	50	-	mA
		3V	V_{OL}=0.3VDD	-	24	-	mA
I_{OH1}	Output high current	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 TA=25℃		-1.5%	1.2	1.5%	V
		VDD=2.5~5.5V T_A=－40~85℃		-2.0%	1.2	2.0%	V

14.3 ADC Electrical Characteristic

(T_A= 25˚C, unless otherwise specified)

Symbol	Parameter	Min	Typ	Max	Unit	Symbol
V_{ADC}	Operating voltage range	V_{ADREF}= VDD, F_{ADCCLK}=1MHz	3.0		5.5	V
		V_{ADREF}= VDD, F_{ADCCLK}=500kHz	2.5		5.5	V
		V_{ADREF}=2.0V, F_{ADCCLK}=250kHz	2.5		5.5	V
		V_{ADREF}=2.4V, F_{ADCCLK}=250kHz	2.6		5.5	V
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_{AIN}	Input voltage range	V_{ADC}=5V, V_{ADREF} = VDD, F_{ADCCLK}=500kHz	0		V_{ADC}	V
DNL1	Differential non-linearity error 1	V_{ADC}=5V, V_{ADREF} = VDD, F_{ADCCLK}=1MHz			±2	LSB
INL1	Integral non-linearity error 1	V_{ADC}=5V, V_{ADREF} = VDD, F_{ADCCLK}=1MHz			±2	LSB
DNL2	Differential non-linearity error 2	V_{ADC}=5V, V_{ADREF} = 2.4V, F_{ADCCLK}=250kHz, V_{AIN}<0.8V			±3	LSB
INL2	Integral non-linearity error 2	V_{ADC}=5V, V_{ADREF} = 2.4V, F_{ADCCLK}=250kHz, V_{AIN}<0.8V			±3	LSB
DNL3	Differential non-linearity error 3	V_{ADC}=5V, V_{ADREF} = 2.0V, F_{ADCCLK}=250kHz, V_{AIN}<0.7V			±3	LSB
INL3	Integral non-linearity error 3	V_{ADC}=5V, V_{ADREF} = 2.0V, F_{ADCCLK}=250kHz, V_{AIN}<0.7V			±3	LSB
T_{ADC}	Conversion time	-		16		T_{ADCCLK}

14.4 ADC Referen LDO Characteristic

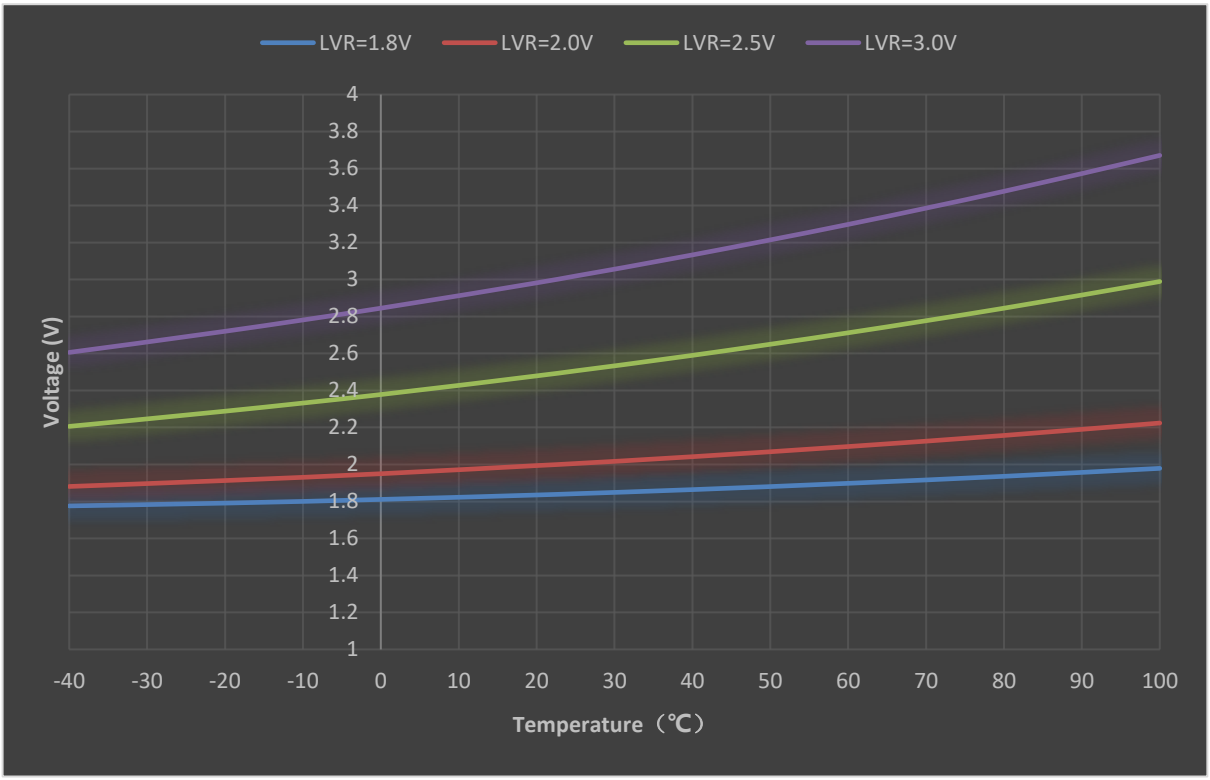
(T_A= 25˚C, unless otherwise specified)

Symbol	Parameter	Min	Typ	Max	Unit	Symbol
V_{ADREF1}	LDO_OUT=2.0V	VDD=2.5~5.5V	-0.6%	2.0	+0.6%	V
V_{ADREF2}	LDO_OUT=2.4V	VDD=2.6~5.5V	-0.6%	2.4	+0.6%	V

14.5 LVR Electrical Characteristic

(T_A= 25˚C, unless otherwise specified)

Symbol	Parameter	Condittion	Min	Typ	Max	Unit
V_{LVR1}	LVR=1.8V	VDD=1.6~5.5V	1.7	1.8	1.9	V
V_{LVR2}	LVR=2.0V	VDD=1.8~5.5V	1.9	2.0	2.1	V
V_{LVR3}	LVR=2.5V	VDD=2.4~5.5V	2.4	2.5	2.6	V
V_{LVR4}	LVR=3.0V	VDD=2.8~5.5V	2.9	3.0	3.1	V



14.6 LVD Electrical Characteristic

(T_A= 25˚C, unless otherwise specified)

Symbol	Parameter	Condittion	Min	Typ	Max	Unit
V_{LVD}	Voltage	-	2.0		5.5	V
	Precision	VDD=2.0~5.5V T_A=－40~85˚C	-5%	V_{SET}	+5%	V

14.7 AC electrical characteristics

(T_A= 25˚C, unless otherwise specified)

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		VDD	Condition				
T_{WDT}	WDT reset time	5V	-		16		ms
		3V	-		16		ms
T_{EEPROM}	EEPROM programming time	5V	F_{OSC}=8MHz		4.6		ms
		3V	F_{OSC}=8MHz		4.6		ms
		5V	F_{OSC}=16MHz		4.6		ms
		3V	F_{OSC}=16MHz		4.6		ms
F_{RC}	Internal frequency stability	VDD=2.5~5.5V	T_A=25℃	-1.5%	8	+1.5%	MHz
		VDD=2.5~5.5V	T_A=-40~85℃	-2.5%	8	+2.5%	MHz
		VDD=1.8~5.5V	T_A=25℃	-2%	8	+2%	MHz
		VDD=1.8~5.5V	T_A=-40~85℃	-3.5%	8	+3.5%	MHz
		VDD=2.5~5.5V	T_A=25℃	-1.5%	16	+1.5%	MHz
		VDD=2.5~5.5V	T_A=-40~85℃	-2.5%	16	+2.5%	MHz
		VDD=1.8~5.5V	T_A=25℃	-3%	16	+3%	MHz
		VDD=1.8~5.5V	T_A=-40~85℃	-4.5%	16	+4.5%	MHz

14.8 EMC Characteristics

14.8.1 EFT Characteristics

Symbol	Parameter	Test Conditions	Max	Unit	Level
V_{EFTB}	Fast transient voltage burst limits to beapplied through 0.1uF(capacitance) on VDDand VSSpins to induce a functional disturbance	T_A = + 25 ˚C, F_{SYS}=8MHz, conforms to IEC 61000-4-4	4800	V	4B

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

14.8.2 ESD Characteristics

Symbol	Parameter	Test Conditions	Max	Unit	Level
V_{ESD}	Electrostatic discharge (Human body discharge mode - HBM)	T_A = + 25˚C, JEDEC EIA/JESD22- A114	2000	V	2
	Electrostatic discharge (Machine discharge mode - MM)	T_A = + 25˚C, JEDEC EIA/JESD22- A115	400	V	C

14.8.3 Latch-Up Characteristics

Symbol	Parameter	Test Conditions	Max	Unit	Level
LU	Static latch-up class	JEDEC STANDARD NO.78D NOVEMBER 2011	Class I (T_A = +25˚C)	±200	mA

15. Instructions

15.1 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

15.2 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
HSUBC   R01     ;execute：R01= (80H-77H-C)=09H (C=0)
R                         R01= (80H-77H-C)=08H (C=1)
```

INCA **[R]**

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

period: 1

affected flag bit: Z

example:

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

INCR **[R]**

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

period: 1

affected flag bit: Z

example:

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

JP **add**

operation: Jump to add address

period: 2

affected flag bit: none

example:

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

LD **A,[R]**

operation: Load the value of R to ACC

period: 1

affected flag bit: Z

example:

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

LD **[R],A**

operation: Load the value of ACC to R

period: 1

affected flag bit: none

example:

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

LDIA **i**

operation: Load 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

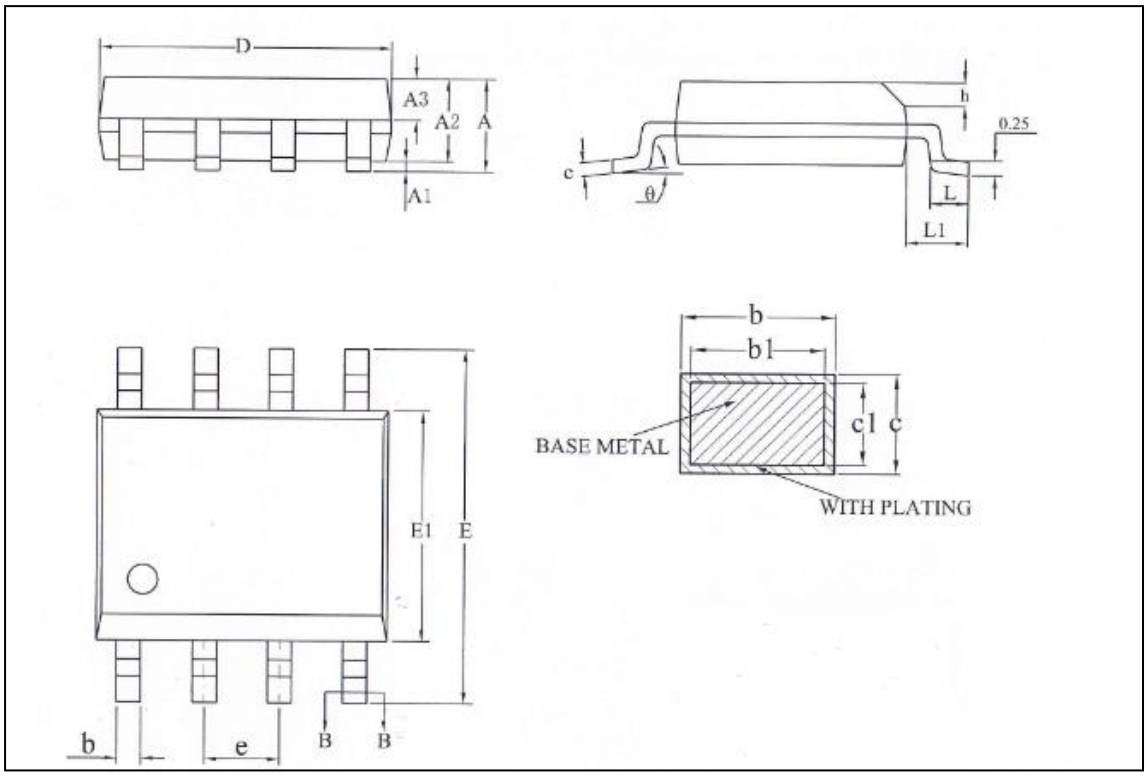
affected flag bit: Z

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
```

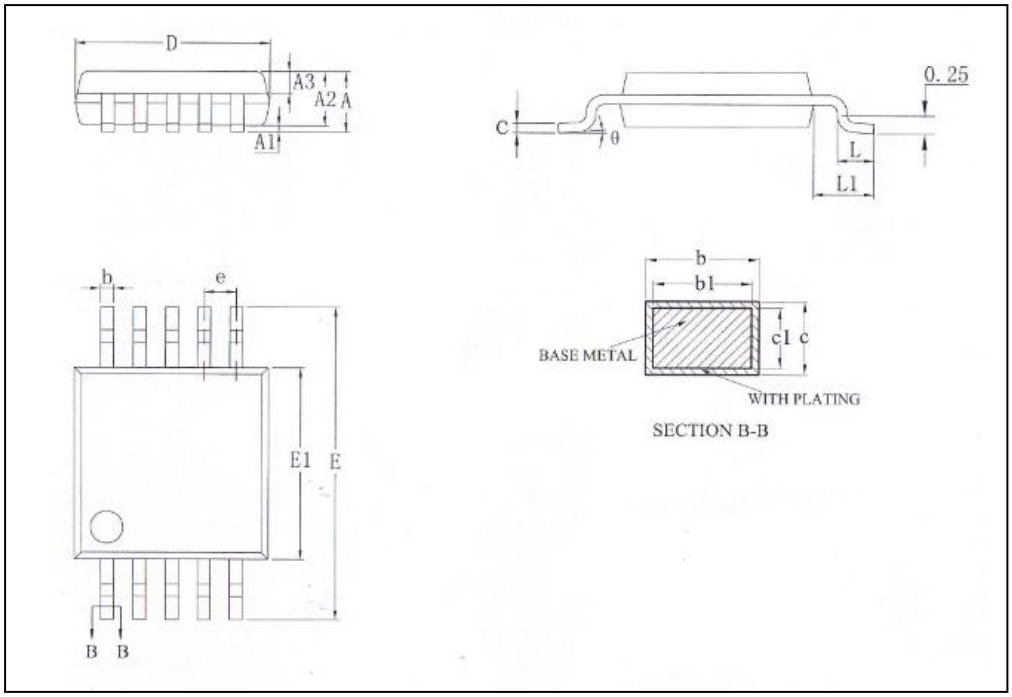
16. Packaging

16.1 SOP8



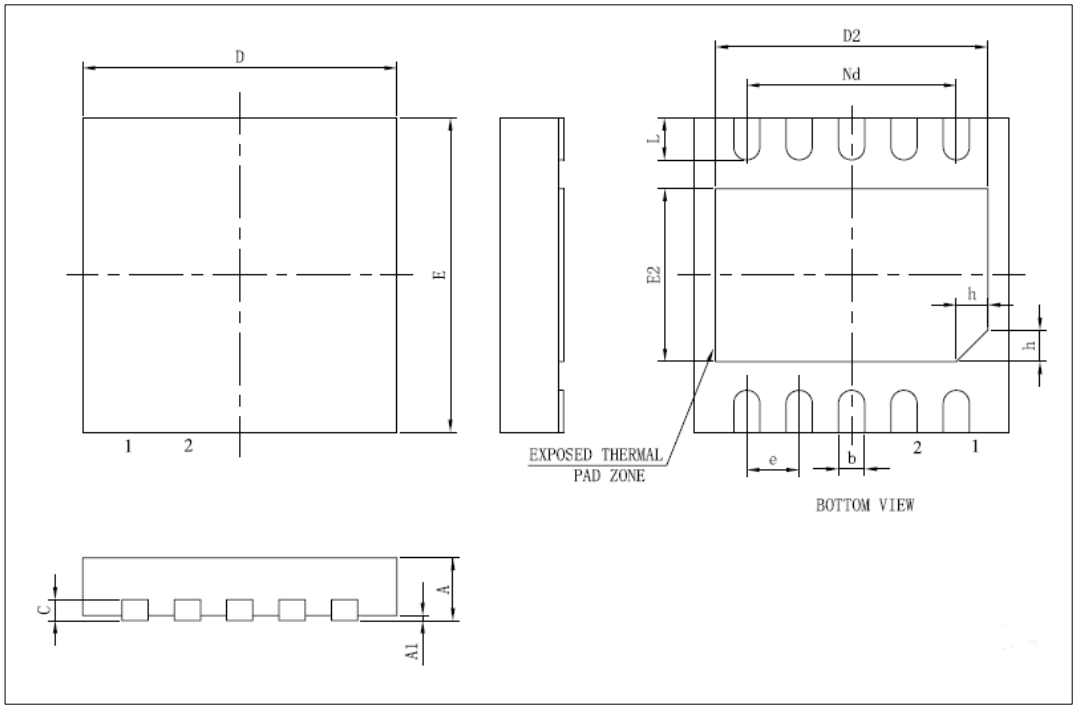
Symbol	Millimeter		
	Min	Nom	Max
A	-	-	1.75
A1	0.10	-	0.225
A2	1.30	1.40	1.50
A3	0.60	0.65	0.70
b	0.39	-	0.47
b1	0.38	0.41	0.44
c	0.20	-	0.24
c1	0.19	0.20	0.21
D	4.80	4.90	5.00
E	5.80	6.00	6.20
E1	3.80	3.90	4.00
e	1.27BSC		
h	0.25	-	0.50
L	0.5	-	0.80
L1	1.05REF		
θ	0	-	8°

16.2 MSOP10



Symbol	Millimeter		
	Min	Nom	Max
A	-	-	1.10
A1	0.05	-	0.15
A2	0.75	0.85	0.95
A3	0.30	0.35	0.40
b	0.18	-	0.26
b1	0.17	0.20	0.23
c	0.15	-	0.19
c1	0.14	0.15	0.16
D	2.90	3.00	3.10
E	4.70	4.90	5.10
E1	2.90	3.00	3.10
e	0.50BSC		
L	0.40	-	0.70
L1	0.95REF		
θ	0	-	8°

16.3 DFN10



Symbol	Millimeter		
	Min	Nom	Max
A	0.70	0.75	0.80
A1	-	0.02	0.05
b	0.18	0.25	0.30
c	0.18	0.20	0.25
D	2.90	3.00	3.10
D2	2.40	2.50	2.60
e	0.5BSC		
Nd	2.00BSC		
E	2.90	3.00	3.10
E2	1.45	1.55	1.65
L	0.30	0.40	0.50
h	0.20	0.25	0.30

17. Revision History

Revision	Date	Modify content
V1.0	Mar 2021	Initial verison
V1.1	Aug 2021	Corrected the description of the reference voltage of the ADC internal LDO
V1.2	Sep 2021	Add electrical parameters EMC characteristics
V1.3	Jan 2022	Corrected descrition in LVD operation
V1.4	Mar 2022	Corrected some electrical parameters