SC8F37xx User Manual

Enhanced 8-bit CMOS Microcontroller with Flash Memory

Rev. 1.4.0

Contents

1. Product Description

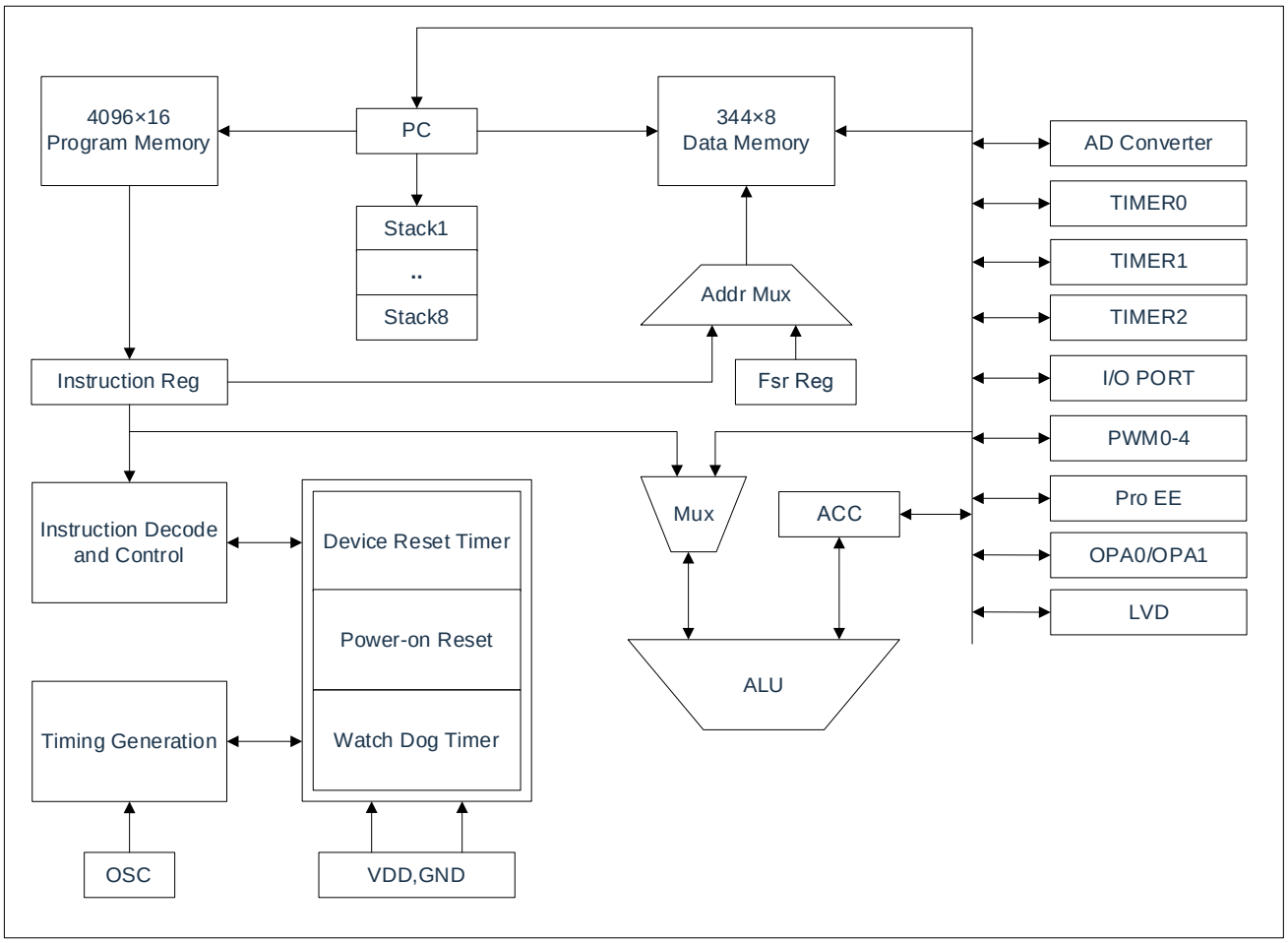
1.1 Features

- Memory
  - ROM: 4K×16Bit
  - Universal RAM: 344×8Bit
- 8-level stack buffer
- Short and clear instruction system (68 commands)
- Instructions period (single instruction or double instructions)
- Jump table function
- Built-in low voltage detection circuit
- Built-in WDT timer
- Interrupt source
  - 3 timer interrupts
  - RA port interrupt on change
  - RB port interrupt on change
  - Other peripheral interrupts
- Timer
  - 8-bit timer: TIMER0, TIMER2
  - 16-bit timer: TIMER1
- Built-in LVD module
  - Choice of voltage:
  - 2.2V/2.4V/2.7V/3.0V/3.3V /3.7V/4.0V/4.3V
- Built-in 128-byte program EEPROM
  - Rewritable times: 10,000
- Operating voltage range: 3.5V—5.5V@16MHz
  1.8V—5.5V@8MHz
  Operating temperature range: -20℃-75℃
- Two oscillation methods
  - Internal RC oscillation: design frequency 8MHz/16MHz
  - External high-speed crystal oscillation: design frequency 8MHz
- PWM mod with complementary outputs
  - 5-channel PWM, can be set to 2-channel complementary output
  - 4-channel PWM with shared period and separated duty cycle
  - 1-channel PWM with separated period and separated duty cycle
- Operational amplifier: 2-channel
  - Offset voltage <2mv
  - Can be used as a comparator
- High precision 12-bit ADC
  - High precision 12-bit ADC
    ±1.5% @VDD=2.5V~5.5V T_A=25℃
    ±2% @VDD=2.5V~5.5V T_A=-20℃~75℃
  - Selectable internal reference source: 2V/2.4V

Product specification

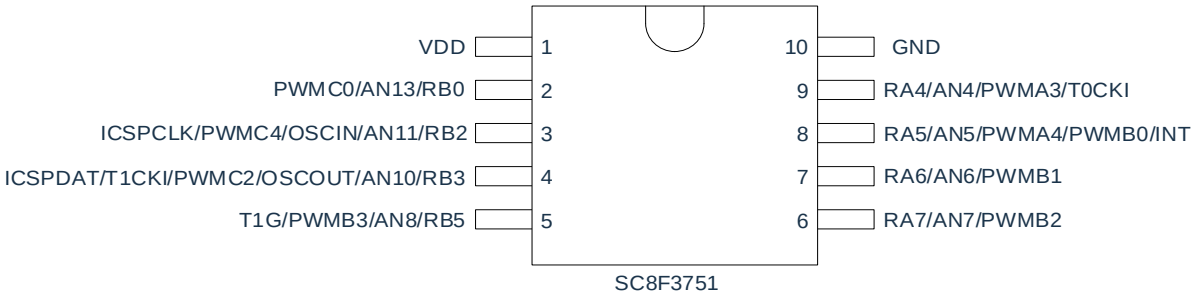
PRODUCT	ROM	RAM	Pro EE	I/O	PWM	OPA	ADC	PACKAGE
SC8F3751	4Kx16	344x8	128x8	8	5	0	12Bit x 8	MSOP10
SC8F3752	4Kx16	344x8	128x8	14	5	2	12Bit x 13	SOP16
SC8F3792	4Kx16	344x8	128x8	14	5	2	12Bit x 13	TSSOP16

1.2 System block diagram



1.3 Pinout

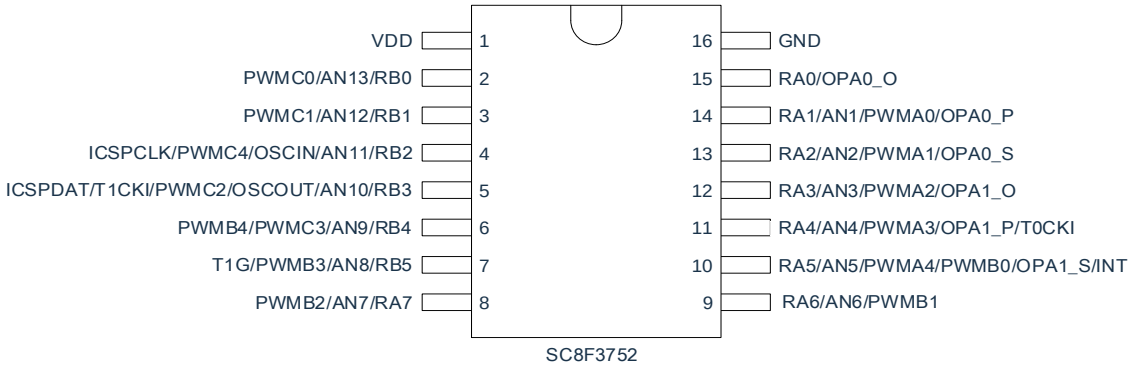
1.3.1 SC8F3751



SC8F3751 pin description:

Pin name	IO type	Description
VDD, GND	P	Supply voltage input pin, gound pin
OSCIN/OSCOUT	I/O	Crystal oscillator input/output pin
RA4-RA7	I/O	Programmable in/push-pull out pin, with pull-up resistance, pull-down resistance, interrupt on change function
RB0,RB2-RB3,RB5	I/O	Programmable in/push-pull out pin, with pull-up resistance, pull-down resistance, interrupt on change function
ICSPCLK	I	Programmable input clock pin
ICSPDAT	I/O	Programmable data input/output pin
AN4-AN7, AN8, AN10-AN11, AN13	I	12-bit ADC input pin
PWMx	O	PWM output function
INT	I	External interrupt input pin
T0CKI	I	TIMER0 external clock input pin
T1CKI	I	TIMER1 external clock input pin
T1G	I	TIMER1 external gate control input pin

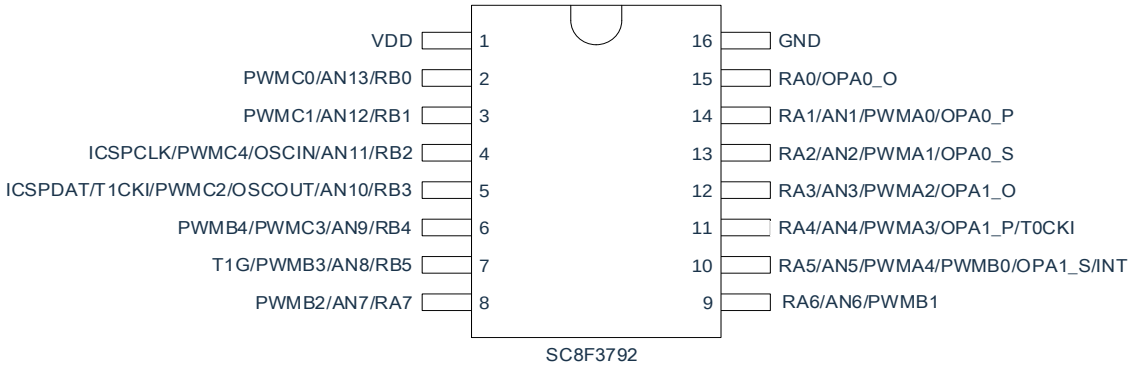
1.3.1 SC8F3752



SC8F3752 pin description:

Pin name	IO type	Description
VDD,GND	P	Supply voltage input pin, gound pin
OSCIN/OSCOUT	I/O	Crystal oscillator input/output pin
RA0-RA7	I/O	Programmable in/push-pull out pin, with pull-up resistance, pull-down resistance, interrupt on change function
RB0-RB5	I/O	Programmable in/push-pull out pin, with pull-up resistance, pull-down resistance, interrupt on change function
ICSPCLK	I	Programmable input clock pin
ICSPDAT	I/O	Programmable data input/output pin
AN1-AN13	I	12-bit ADC input pin
PWMx0-PWMx4	O	PWM0-4 output function
OPAx_P	I	Op-amp positive input pin
OPAx_S	I	Op-amp negative input pin
OPAx_O	O	Op-amp output pin
INT	I	External interrupt input pin
T0CKI	I	TIMER0 external clock input pin
T1CKI	I	TIMER1 external clock input pin
T1G	I	TIMER1 external gate control input pin

1.3.2 SC8F3792



SC8F3792 pin description:

Pin name	IO type	Description
VDD,GND	P	Supply voltage input pin, gound pin
OSCIN/OSCOUT	I/O	Crystal oscillator input/output pin
RA0-RA7	I/O	Programmable in/push-pull out pin, with pull-up resistance, pull-down resistance, interrupt on change function
RB0-RB5	I/O	Programmable in/push-pull out pin, with pull-up resistance, pull-down resistance, interrupt on change function
ICSPCLK	I	Programmable input clock pin
ICSPDAT	I/O	Programmable data input/output pin
AN1-AN13	I	12-bit ADC input pin
PWMx0-PWMx4	O	PWM0-4 output function
OPAx_P	I	Op-amp positive input pin
OPAx_S	I	Op-amp negative input pin
OPAx_O	O	Op-amp output pin
INT	I	External interrupt input pin
T0CKI	I	TIMER0 external clock input pin
T1CKI	I	TIMER1 external clock input pin
T1G	I	TIMER1 external gate control input pin

1.4 System configuration register

The system configuration register (CONFIG) is the ROM option for the initial conditions of the MCU. It can only be burned by the SC writer, and cannot be accessed and manipulated by the user. It contains the following contents:

1. OSC (choice of oscillation)
   - INTRC Internal RC oscillation
   - XT External crystal oscillation
2. INTRC_SEL (internal oscillation frequency)
   - INTRC8M F_{HSI} choose internal 8MHz RC oscillation
   - INTRC16M F_{HSI} choose internal 16MHz RC oscillation
3. WDT (watchdog choice)
   - ENABLE Enable watchdog timer
   - DISABLE Disable watchdog timer
4. PROTECT (encyption)
   - DISABLE Disable FLASH code encyption
   - ENABLE Enable FLASH code encryption, after which the read value from burning the simulator is uncertain.
5. LVR_SEL (low voltage detection selection)
   - 1.8V When this reset voltage is selected, F_{HSI} needs to select 8MHz
   - 2.0V When this reset voltage is selected, F_{HSI} needs to select 8MHz
   - 2.6V When this reset voltage is selected, F_{HSI} needs to select 8MHz
   - 3.5V
6. PWM_SEL (PWM output port selection)
   - Group A PWM0-4=RA1, RA2, RA3, RA4, RA5
   - Group B PWM0-4=RA5, RA6, RA7, RB5, RB4
   - Group C PWM0-4=RB0, RB1, RB3, RB4, RB2
7. ICSPPORT_SEL (simulation port function selection)
   - ICSP ICSPCLK and DAT ports remain as simulation ports, all functions are not available
   - NORMAL ICSPCLK and DAT ports are normal function ports

1.5 Online serial programming

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

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

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

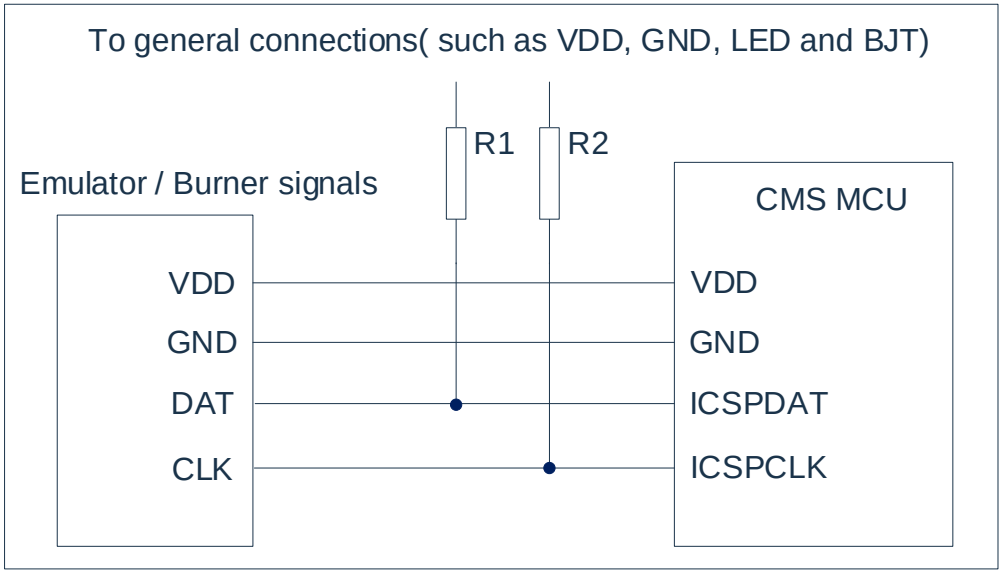


Figure 1-1: Typical connection for online serial programming

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

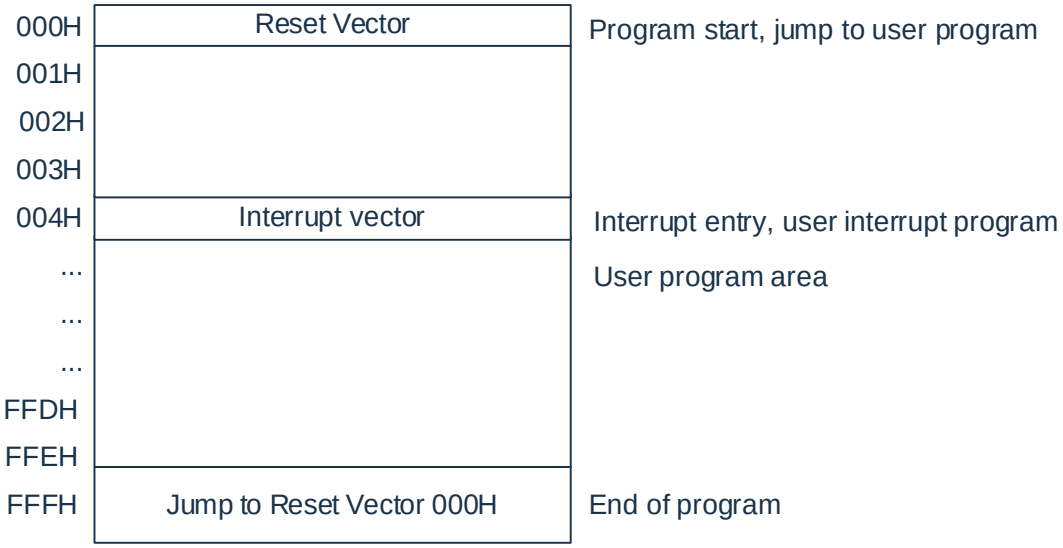
2. Central Processing Unit (CPU)

2.1 Memory

2.1.1 Program memory

SC8F37xx 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 register will be recovered to default value. PD and TO from STATUS register can determine the which reset is performed from above. The following program illustrates how to define the reset vector from FLASH.

Example: define reset vector

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

2.1.1.2 Interrupt vector

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

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

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

Note: MCU does not provide specific unstack and push instructions, so 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 Jump table

The jump table enables the multi-address jumping. Since the values of PCL and ACC can be added together to obtain a new PCL, multiple address jumps can be achieved by adding different ACC values to the PCL. If the ACC value is n, PCL+ACC means the current address plus n. The PCL value will also add 1 to itself after the current instruction is executed, see the following example. If an overflow occurs after PCL+ACC, the PC will not carry, so care should be taken when writing the program. In this way, the user can easily implement multi-address jumps by modifying the value of ACC.

PCLATH is the PC high-bit buffer register. When operating on PCL, you must first assign a value to PCLATH.

Example: Example of a correct multi-address jump program

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

Example: Example of an incorrect multi-address jump program

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

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

2.1.2 Data memory

SC8F37xx data memory list

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

Data memory consists of 512×8 bits. It can be divided into two space: special function register and general-purpose data memory. Most of data memory are able to write/read data, only some data memory is read-only. Special register address is from 00H-1FH, 80-9FH, 100-11FH, 180-19FH.

SC8F37xx Special Function Register Summary Bank0

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
00H	INDF	Look-up for this unit will use FSR, not physical register.								xxxxxxxx
01H	TMR0	TIMER0 data register								xxxxxxxx
02H	PCL	Lower bit of program counter								00000000
03H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
04H	FSR	Indirect data memory address pointer								xxxxxxxx
05H	PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	xxxxxxxx
06H	PORTB	----	----	RB5	RB4	RB3	RB2	RB1	RB0	--xxxxxx
07H	WPDA	WPDA7	WPDA6	WPDA5	WPDA4	WPDA3	WPDA2	WPDA1	WPDA0	00000000
08H	WPDB	----	----	WPDB5	WPDB4	WPDB3	WPDB2	WPDB1	WPDB0	--000000
0AH	PCLATH	----	----	----	----	Write buffer for the high 4 bits of the program counter				-----0000
0BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
0CH	PIR1	RAIF	ADIF	----	----	EEIF	PWMIF	TMR2IF	TMR1IF	00000000
0DH	PIR2	----	----	----	----	----	----	----	LVDIF	-------0
0EH	TMR1L	Data register for the low byte of the 16-bit TIMER1 register								xxxxxxxx
0FH	TMR1H	Data register for the high byte of the 16-bit TIMER1 register								xxxxxxxx
10H	T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	----	T1SYNC	TMR1CS	TMR1ON	0000-000
11H	TMR2	TIMER2 module register								00000000
12H	T2CON	----	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-0000000
1EH	ADRESH	High byte of A/D result register								xxxxxxxx
1FH	ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/ $\overline{\text{DONE}}$	ADON	00000000

SC8F37xx Special Function Register Summary Bank1

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
80H	INDF	Addressing this unit will use FSR (not physical register)								xxxxxxxx
81H	OPTION_REG	----	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	-1111011
82H	PCL	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	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	11111111
86H	TRISB	----	----	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	--111111
87H	IOCA	IOCA7	IOCA6	IOCA5	IOCA4	IOCA3	IOCA2	IOCA1	IOCA0	00000000
8AH	PCLATH	----	----	----	----	Write buffer for the high 4 bits of the program counter				-----0000
8BH	INTCON	GIE	PEIE	T01E	INTE	RBIE	T0IF	INTF	RBIF	00000000
8CH	PIE1	RAIE	ADIE	----	----	EEIE	PWMIE	TMR2IE	TMR1IE	00000000
8DH	PIE2	----	----	----	----	----	----	----	LVDIE	-------0
8FH	OSCCON	----	IRCF2	IRCF1	IRCF0	----	----	----	SCS	-110---0
90H	OSCTUNE	----	----	----	TUN4	TUN3	TUN2	TUN1	TUN0	---00000
92H	PR2	TIMER2 cycle register								11111111
93H	PWM01DT	----	----	PWM01 deadtime delay time						--00000
94H	PWM23DT	----	----	PWM23 deadtime delay time						--00000
95H	WPUB	----	----	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0	--000000
96H	IOCB	----	----	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	IOCB0	--000000
97H	LVDCON	LVD_RES	—	—	—	LVD_SEL[2:0]			LVDEN	x--0000
9AH	OPA0CON	OPA0EN	OPA0O	OPA0_CMP	OPA0_ADC	OPA0_FW	OPA0_BG	----	----	000000--
9BH	OPA0ADJ	OPA0OUT	OPA0COFM	OPA0CRS	OPA0ADJ[4:0]					000xxxxx
9CH	OPA1CON	OPA1EN	OPA1O	OPA1_CMP	OPA1_ADC	OPA1_FW	OPA1_BG	----	----	000000--
9DH	OPA1ADJ	OPA1OUT	OPA1COFM	OPA1CRS	OPA1ADJ[4:0]					000xxxxx
9EH	ADRESL	Low byte of ADC result register								xxxxxxxx
9FH	ADCON1	ADFM	----	----	----	----	LDO_EN	----	LDO_SEL	0----0-0

SC8F37xx Special Function Register Summary Bank2

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

SC8F37xx Special Function Register Summary Bank3

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

2.2 Addressing modes

2.2.1 Direct addressing

The RAM is operated through the accumulator (ACC).

Example: pass the value in ACC to 30H register

```
LD      30H,A
```

Example: pass the value in 30H register to ACC

```
LD      A,30H
```

2.2.2 Immediate addressing

Pass the immediate value to accumulator (ACC).

Example: pass immediate value 12H to ACC

```
LDIA    12H
```

2.2.3 Indirect addressing

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

Example: application of FSR and INDF

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

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

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

2.3 Stack

The stack buffer of the chip has 8 levels. The stack buffer is neither part of the data memory nor part of the program memory, and it can neither be read nor written. 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 subroutine calls or interrupts happens, value in program counter (PC)will be transferred to stack buffer. When return from interrupt or return from subroutine, value is transferred back to PC. The diagram below illustrates how this works:

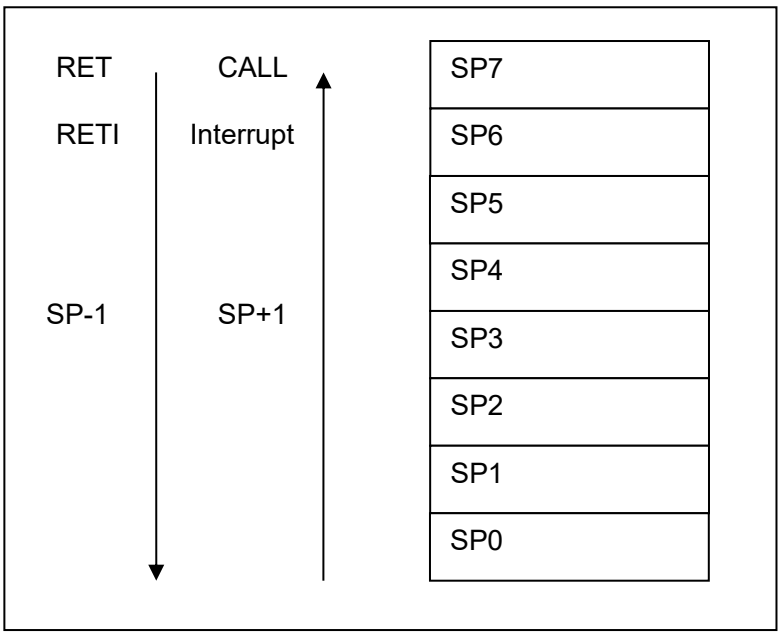


Figure 2-1: How the stack buffer works

Stack buffer will follow one principle: 'first in last out'

Note: stack buffer has only 8 levels, if the stack is full and interrupt happens which cannot be screened out, then only the flag 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 subroutine calls, then stack will overflow and the contents which enter the stack first will be lost, only the last 8 return address will be saved.

2.4 Accumulator (ACC)

2.4.1 Overview

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

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:

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

For example, CLRSTATUS will clear higher 3 bits and set the Z bit to 1. Hence the value of STATUS will be 000u u1uu (u = unchange). 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 bit;
	1=	Power on or CLRWDT instructions;
	0=	STOP instructions.
Bit2	Z:	Result is zero bit;
	1=	Result is 0;
	0=	Result is not 0
Bit1	DC:	Carry bit;
	1=	When carry happens to higher bits or no borrow happens in Lower 4 bits in the result;
	0=	When no carry happens to higher bits or borrow happens in Lower 4 bits in the result.
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 and PD after these resets.

Events	TO	PD
Power on	1	1
WDT overflow	0	X
STOP instructions	1	0
CLRWDT instructions	1	1
Sleep	1	0

Events which affect TO/PD

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

TO/PD status after reset

2.6 Pre-scaler (OPTION_REG)

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

- TIMER0/WDT pre-scaler
- TIMER0

Pre-scaler: OPTION_REG (81H)

81H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION_REG	---	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
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 Not used

Bit6 INTEDG: Edge selection bit for triggering interrupt
1= INT pin rising edge triggered interrupt
0= INT pin falling edge triggered interrupt

Bit5 T0CS: Selection bit for TIMER0 clock source.
0= Internal instructions period clock (F_{SYS}/4).
1= Transition edge on T0CKI pin

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

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

Bit2~Bit0 PS2~PS0: Configuration bit for pre-allocation parameters.

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

The pre-scaler register is actually an 8-bit counter that is used as a post-scaler when used for the monitoring register WDT and as a pre-scaler when used for the timer/counter, often collectively referred to as a pre-scaler. There is only one physical divider on the chip, which can only be used for either WDT or TIMER0, and both cannot be used at the same time. That is, if it is used for TIMER0, the WDT cannot use the pre-scaler, and vice versa.

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

When used for TIMER0, all instructions about writing to TIMER0 (e.g., CLRTMR0, SETB TMR0,1, etc.) will clear the pre-scaler.

Whether TIMER0 or WDT uses a pre-scaler is completely controlled by software. It can be changed dynamically. To avoid a chip reset that should not occur, the following instruction should be executed when switching from TIMER0 to WDT.

```
CLRB    INTCON,GIE          ;Turn off the enable bit for interrupt to avoid
                            entering interrupt during the following timings
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 a new pre-scaler
LD      OPTION_REG,A
CLRWDT                      ;clear WDT
SETB    INTCON,GIE          ;when an interrupt is needed, enable bit is
                            turned on here
```

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

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

Note: To enable TIMER0 to obtain a 1:1 prescaler ratio configuration, assign WDT to the prescaler by setting the PSA bit of the option register to 1.

2.7 Program counter (PC)

The program counter (PC) controls the order of instruction execution in the program memory FLASH, which can address the entire range of FLASH. After obtaining an instruction code, the program counter (PC) will automatically increase by 1 and point to the address of the next instruction code. However, when executing operations such as jump, condition jump, assignment to PCL, subroutine call, initialization reset, interrupt, interrupt return, subroutine return, etc., the PC will load the address associated with the instruction, rather than the address of the next instruction.

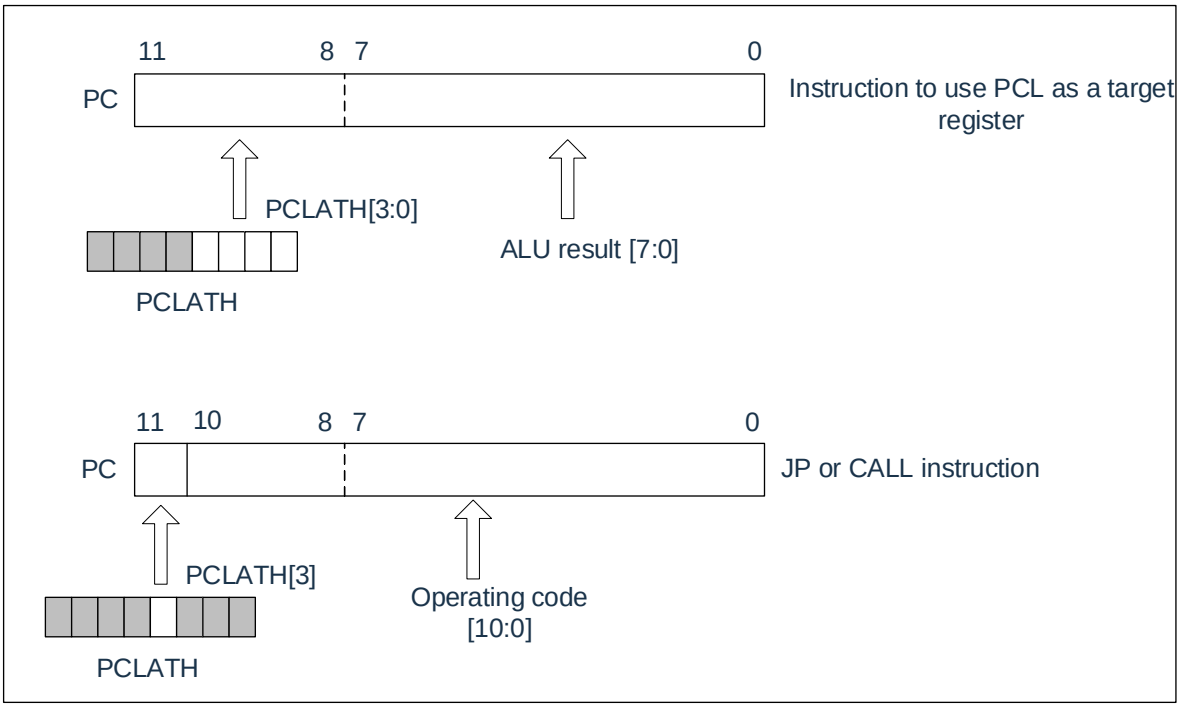
When a condition jump instruction is encountered and the jump condition is met, the next instruction read during the current instruction execution will be discarded and an empty instruction operation cycle will be inserted before the correct instruction is obtained. Or, the next instruction will be executed sequentially.

The program counter (PC) is 12-bit, the lower 8 bits are user accessible through the PCL (02H) register, and the upper 4 bits are not user accessible. It can hold 4K x 14Bit program addresses. Assigning a value to PCL will generate a short jump action to 256 addresses of the current page.

Note: When the programmer makes JP or CALL operation with more than 2K space, or makes short jumps with PCL, the PC high buffer register (PCLATH) should be assigned first.

The following are the value of PC under special conditions.

Reset	PC=0000;
Interrupt	PC=0004 (the original PC+1 will be automatically added into the stack);
CALL	PC[11] is determined by PCLATH[3] and PC[10:0] is determined by the opcode. (The original PC+1 is automatically added to the stack);
RET, RETI, RET i	PC = the value from the stack;
Operating on PCL	PC[11:8] determined by PCLATH[3:0], PC[7:0] = user-specified value;
JP	PC[11] is determined by PCLATH[3] and PC[10:0] is determined by the opcode.
Other instructions	PC=PC+1;



The following sample program gives precautions for using JP or CALL instructions.

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

2.8 Watchdog timer (WDT)

Watch Dog Timer (WDT) is an on-chip and self-oscillating RC oscillation timer that does not require any peripheral components and keeps the WDT timing even if the chip's main clock is stopped.

2.8.1 WDT period

WDT and TIMER0 share the same 8-bit pre-scaler. After all resets, the WDT overflow period is 128ms, if you need to change the WDT period, you can set the OPTION_REG register. the WDT overflow period will be affected by the ambient temperature, supply voltage and other parameters.

The "CLRWDT" and "STOP" instructions will clear the WDT timer and the count value in the pre-scaler (when the pre-scaler is assigned to the WDT). The WDT is generally used to prevent the system from going out of control, or to prevent the MCU program from going out of control. Normally, the WDT should be cleared by the "CLRWDT" instruction before it overflows to prevent a reset. If the program goes out of control due to some disturbance, then the "CLRWDT" instruction cannot be executed before the WDT overflows, which will cause the WDT to overflow and generate a reset. The system is restarted without losing control. If the reset is generated by WDT overflow, the "TO" bit of the status register (STATUS) will be cleared to zero, and the user can determine whether the reset is caused by WDT overflow according to this bit.

Notes:

1. If you use the WDT function, you must place the "CLRWDT" instruction in some places of the program to ensure that it can be cleared before the WDT overflows. Otherwise, the chip will keep resetting and the system will not work properly.
2. The WDT cannot be cleared in the interrupt program, otherwise the main program "run away" cannot be detected.
3. The program should have a clear WDT operation in the main program and try not to clear WDT in multiple subroutines, this method can maximize the protection function of the watchdog counter.
4. The overflow time of the watchdog counter varies from chip to chip, so when setting the clear WDT time, there should be a large redundancy with the overflow time of the WDT to avoid unnecessary WDT resets.

2.8.2 Watchdog timer control register (WDTCON)

Watchdog timer control register: WDTCON (105H)

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

Bit7~Bit1 Not used, reads 0.

Bit0 SWDTEN: Software enables or disables the watchdog timer bit.

1= Enables WDT.

0= WDT (reset value) is disabled.

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

3. System Clock

3.1 Overview

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

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

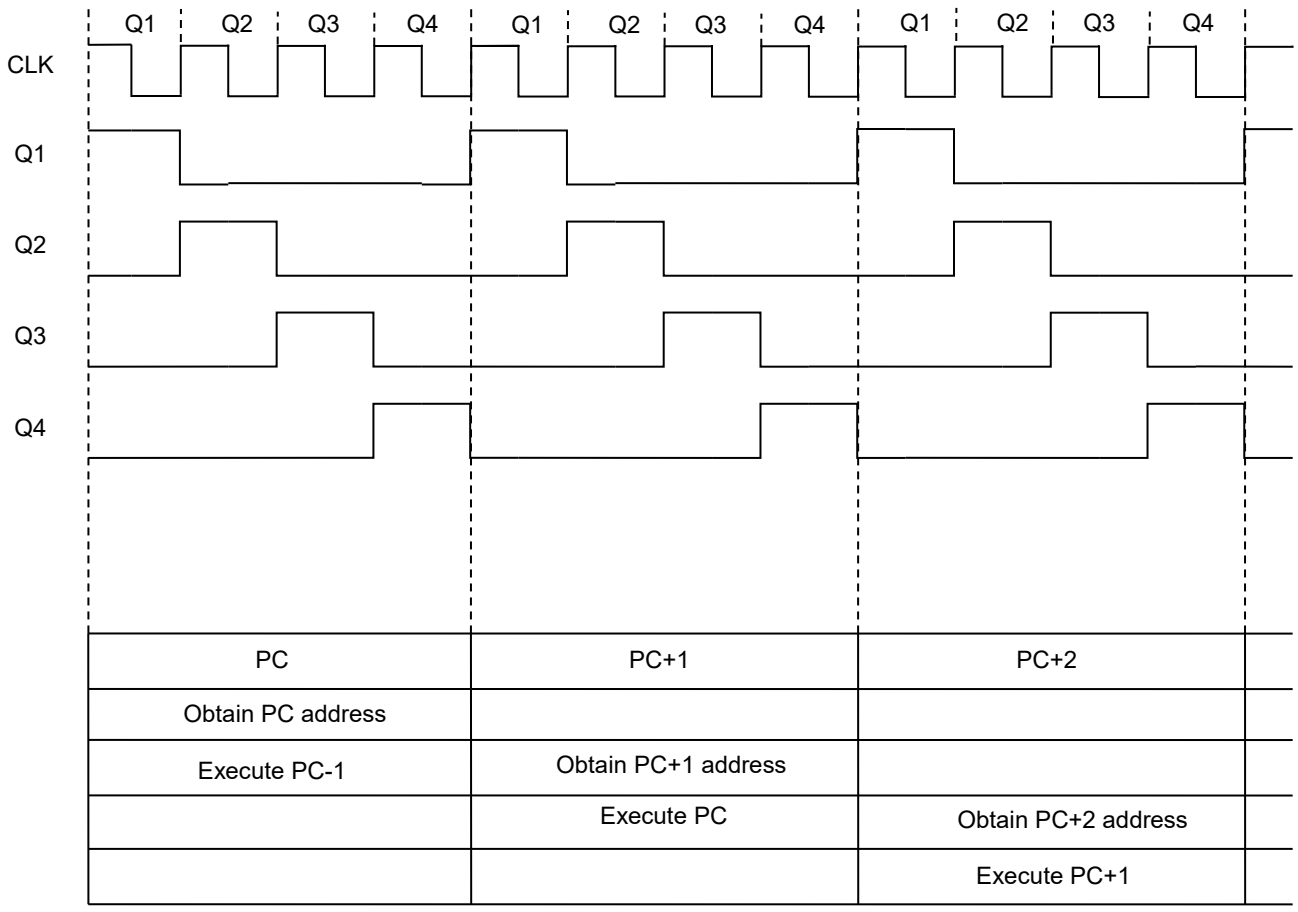


Figure 3-1: Timing diagram of clock and instruction cycles

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

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

3.2 System oscillator

The chip has 2 types of oscillation, internal RC oscillation and external XT oscillation.

3.2.1 Internal RC oscillation

The default oscillation mode of the chip is internal RC oscillation, and its oscillation frequency is 8MHz.

3.2.2 External XT oscillation

The chip operates in the external XT oscillation mode when the OSC in the CONFIG option is selected as XT during the burn-in, while the internal RC oscillation stops working and OSCIN and OSCOUT are used as oscillation ports.

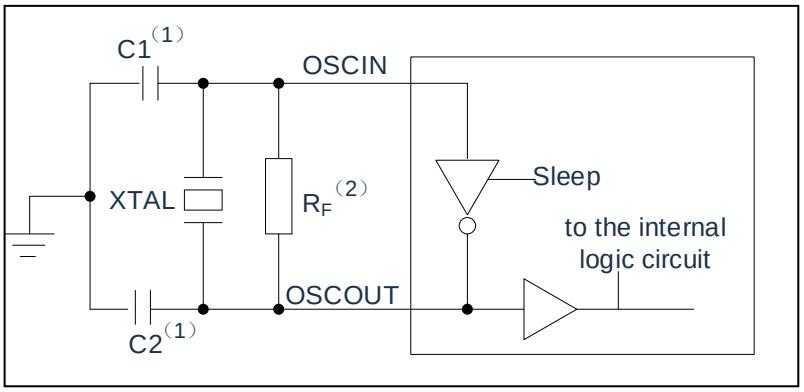


Figure 3-2: Typical XT oscillation method

Suggested parameters:

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

3.3 Reset time

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

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

3.4 Oscillator control register

The oscillator control (OSCCON) register controls the system clock and frequency selection, and the oscillator regulation register (OSCTUNE) enables software adjustment of the internal oscillation frequency.

Oscillator control register: OSCCON(8FH)

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

Bit7 Not used, reads 0.

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

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

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

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

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

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

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

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

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

Bit3~Bit1 Not used.

Bit0 SCS: System clock select bit.

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

0= The clock source is defined by CONFIG.

Note: F_{HSI} is the internal high-speed oscillator frequency, 8MHz or 16MHz can be selected; F_{SYS} is the system operating frequency.

3.5 Clock block diagram

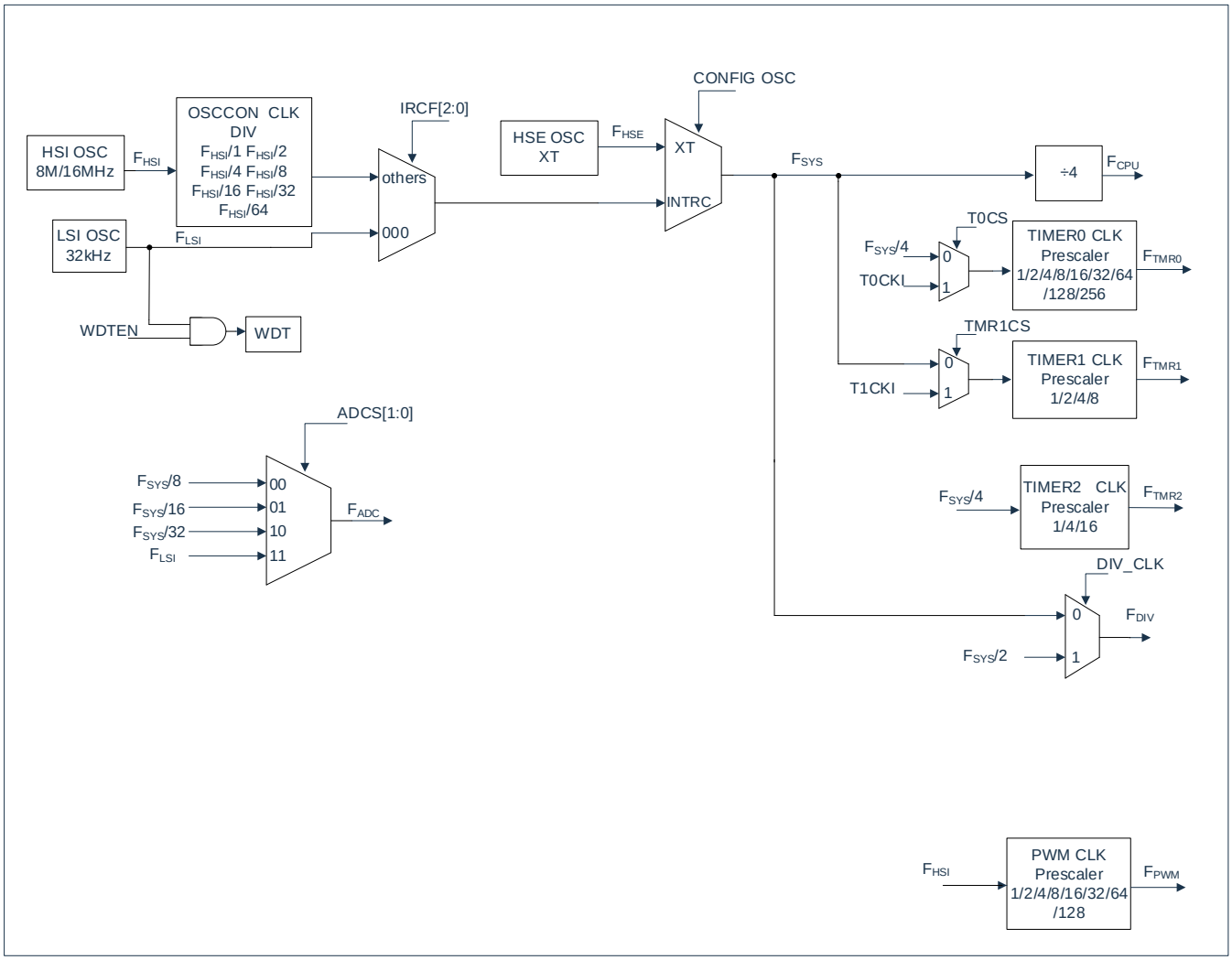


Figure 3-2: Clock block diagram

4. Reset

The chip can be reset in 3 ways as follows:

- Power-on reset.
- Low voltage reset.
- Watchdog overflow reset under normal operation.

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

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

4.1 Power-on reset

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

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

4.2 Power-off reset

4.2.1 Overview

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

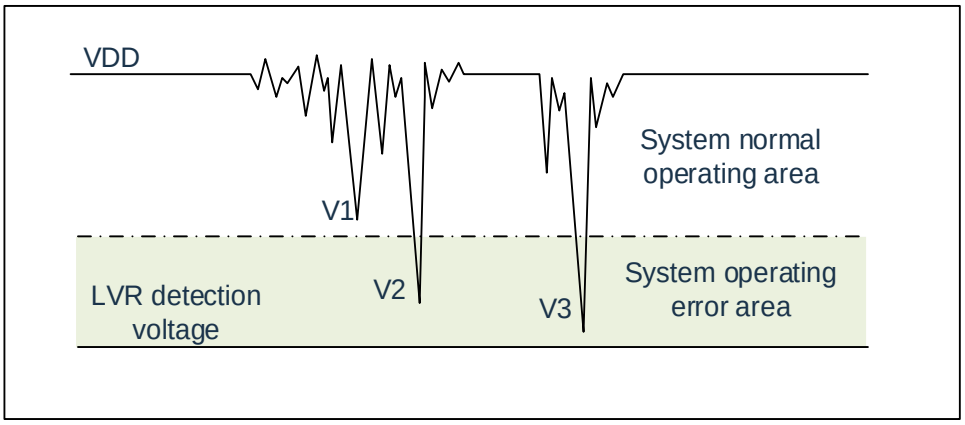


Figure 4-1: Power off reset

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

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

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

As shown above, the system normal operating voltage is generally higher than the system reset voltage, while the reset voltage is determined by the low voltage detection (LVR) level. When the system execution speed increases, the system minimum operating voltage also increases accordingly. However, as the system reset voltage is fixed, there will be a voltage region between the system minimum operating voltage and the system reset voltage where the system cannot work normally and will not reset. This area is known as the dead zone.

4.2.2 Improvement of power-off reset

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

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

Watchdog Timer

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

Reduce the operating speed of the system

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

Increase the voltage drop slope

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

4.3 Watchdog reset

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

The timing of the watchdog reset is as follows:

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

For the application of the watchdog timer, please refer to the Chapter 2.8 WDT Application.

5. Sleep Mode

5.1 Enter sleep mode

Execute STOP instruction to enter sleep mode. If WDT is enabled, then:

- The WDT is cleared and continues to run.
- The PD bit in the STATUS register is cleared to zero.
- The TO bit is set to 1.
- Turn off the oscillator driver.
- The I/O port remains the state before the STOP instruction was executed (driven high-level, low-level, or high impedance).

In sleep mode, to minimize current consumption, all I/O pins should be kept as VDD or GND, with no external circuitry consuming current from the I/O pins. To avoid input pin floating and invoke current, high impedance I/O should be pulled to high or low level externally. Internal pull up resistance should also be considered.

5.2 Awaken from sleep mode

The device can be awakened from sleep by any of the following events:

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

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

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

5.3 Interrupt awakening

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

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

5.4 Sleep mode application examples

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

Example: Procedures for entering sleep mode

```
SLEEP_MODE:
        CLR     INTCON          ;Shutdown interrupt enable
        LDIA    B'00000000'
        LD      TRISA,A
        LD      TRISB,A         ;All I/O are set to output ports
        LD      TRISC,A
        LD      TRISE,A
        ...                     ;Turn off other functions
        LDIA    0A5H
        LD      SP_FLAG,A       ;Set  sleep status memory register(user-defined)
        CLRWDT                  ;Clear WDT
        STOP                    ;Execute STOP command
```

5.5 Sleep mode wake-up time

When the MCU is awakened from sleep, it needs to wait for an oscillation stabilization time (Reset Time), which is 1024 T_{SYS} clock cycles in the internal high-speed oscillation mode and 8 T_{SYS} clock cycles in the internal low-speed oscillation mode. The relationships are shown in the table below.

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

6. I/O Ports

The chip has two I/O ports: PORTA, PORTB (Max.14 I/Os). The read/write port data registers can directly read/write these ports.

Port	Bit	Pin description	I/O
PORTA	0	Schmitt trigger input, push-pull output, op-amp 0 output	I/O
	1	Schmitt trigger input, push-pull output, AN1, PWM output, op-amp 0 positive input	I/O
	2	Schmitt trigger input, push-pull output, AN2, PWM output, op-amp 0 negative input	I/O
	3	Schmitt trigger input, push-pull output, AN3, PWM output, op-amp 1 output	I/O
	4	Schmitt trigger input, push-pull output, AN4, PWM output, op-amp 1 positive input	I/O
	5	Schmitt trigger input, push-pull output, AN5, PWM output, op-amp 1 negative input, external interrupt input	I/O
	6	Schmitt trigger input, push-pull output, AN6, PWM output	I/O
	7	Schmitt trigger input, push-pull output, AN7, PWM output	I/O
PORTB	0	Schmitt trigger input, push-pull output, AN13, PWM output	I/O
	1	Schmitt trigger input, push-pull output, AN12, PWM output	I/O
	2	Schmitt trigger input, push-pull output, AN11, PWM output, programming clock input, oscillation output port	I/O
	3	Schmitt trigger input, push-pull output, AN10, PWM output, Programming data input/output, oscillation input port	I/O
	4	Schmitt trigger input, push-pull output, AN9, PWM output	I/O
	5	Schmitt trigger input, push-pull output, AN8, PWM output	I/O

<Table 6-1: General overview of port configuration>

6.1 I/O port block diagram

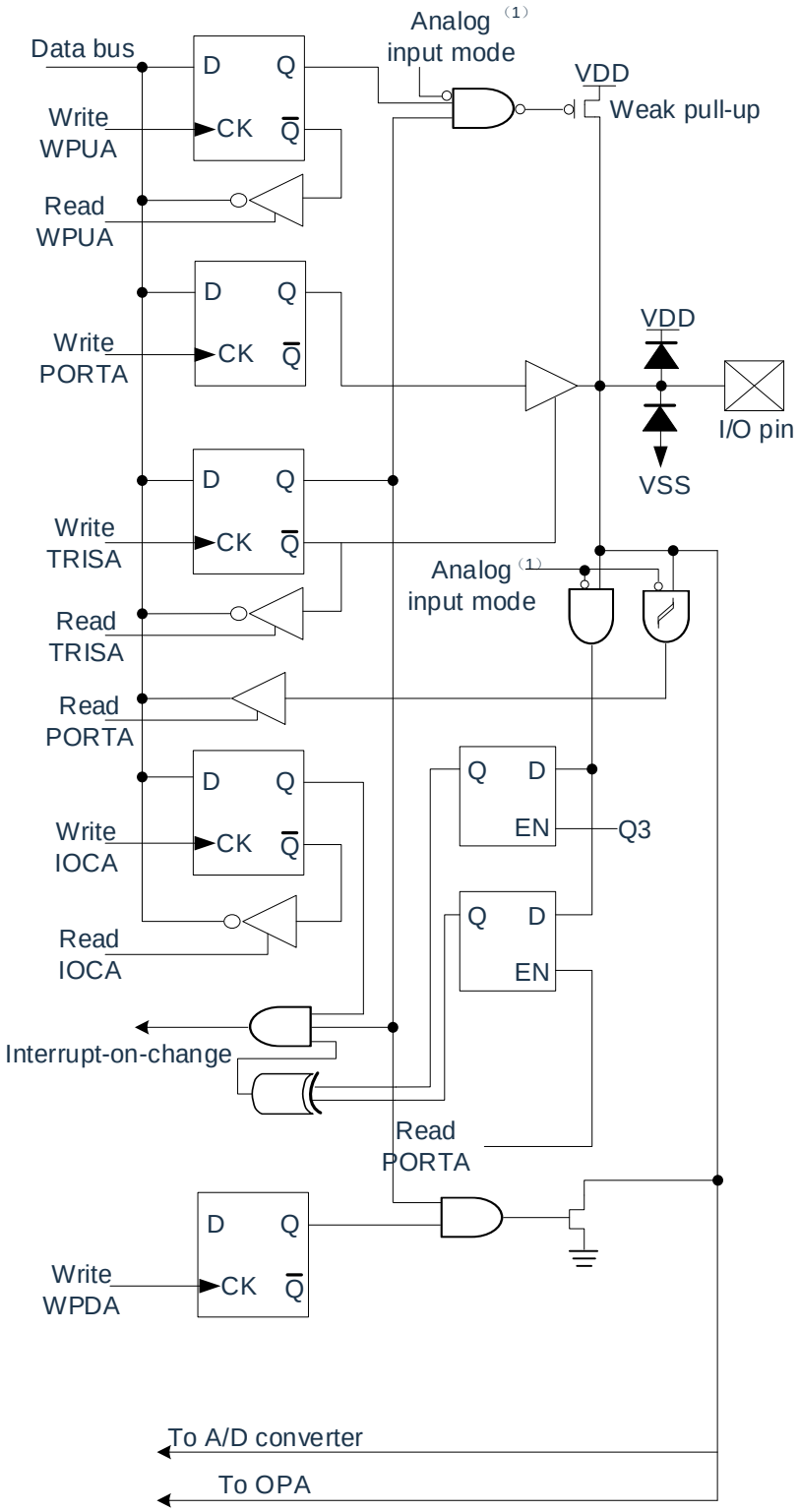


Figure 6-1: I/O port structure

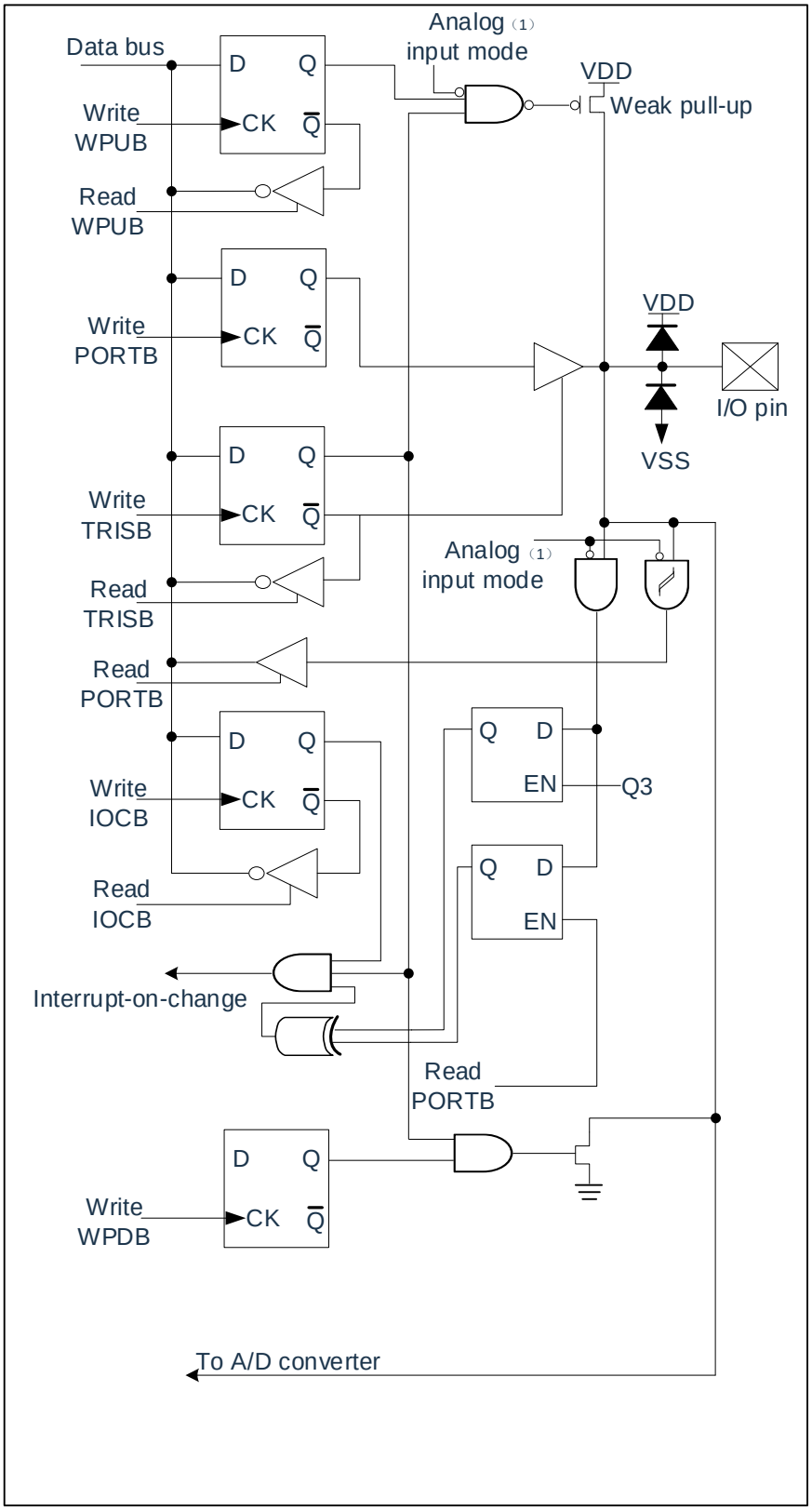


Figure 6-2: I/O port structure

6.2 PORTA

6.2.1 PORTA data and direction control

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

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

The registers associated with the PORTA port are PORTA, TRISA, WPUA, WPDA, IOCA, ANSEL.

PORTA data register: PORTA (05H)

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

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

PORTA direction register: TRISA (85H)

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

Bit7~Bit0 TRISA<7:0>: PORTA tri-state control bit;
1= PORTA pin is configured as an input (tri-state);
0= PORTA pin is configured as an output.

Example: Procedure for PORTA Port

```
LDIA    B'11110000'    ;Set PORTA<3:0> as an output port, PORTA<7:4> as an input port
LD      TRISA,A
LDIA    03H            ;PORTA<1:0> output high, PORTA<3:2> output low
LD      PORTA,A        ;Since PORTA<7:4> is an input port, assigning 0 or 1 has no effect.
```

6.2.2 PORTA analog selection control

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

PORT analog selection register: ANSEL (188H)

188H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	---
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	---
Reset value	0	0	0	0	0	0	0	---

Bit0 Not used

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

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

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

6.2.3 PORTA pull-up resistance

Each PORTA pin has an individually configurable internal weak pull-up. The control bit WPUA<7:0> enables or disables each weak pull-up. When a port pin is configured as an output, its weak pull-up is automatically cut-off.

PORTA pull-up resistance register: WPUA (18EH)

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

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

1= Enable pull-up.

0= Disable pull-up.

Note: If a pin is configured as output, the weak pull-up will be 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 bit WPDA<7:0> enables or disables each weak pull down.

PORTA pull down resistance: WPDA (07H)

07H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPDA	WPDA7	WPDA6	WPDA5	WPDA4	WPDA3	WPDA2	WPDA1	WPDA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

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

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

6.2.5 PORTA interrupt on change

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

For the pin that has enabled interrupt-on-change, compare the value on the pin with the old value latched when PORTA was read last time. A logical or operation is performed with the "mismatch" output of the last read operation to set the PORTA interrupt-on-change flag bit (RAIF) in the PIR1 register to 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 enable the RACIF flag to be cleared. The latch will keep the last read value from the under voltage reset. After reset, if the mismatch still exists, the 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 interrupt-on-change register: IOCA (87H)

87H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOCA	IOCA7	IOCA6	IOCA5	IOCA4	IOCA3	IOCA2	IOCA1	IOCA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 IOCA<7:0> Control bit of interrupt on change of PORTA

1= Enable interrupt on change

0= Disable interrupt on change

6.3 PORTB

6.3.1 PORTB data and direction

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

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

Related registers with PORTB port include PORTB, TRISB, WPUB, WPDB, IOCB, ANSELH and etc.

PORTB data register: PORTB (06H)

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

Bit7~Bit6	Not used	
Bit5~Bit0	PORTB<5:0>:	PORTB I/O pin bit.
	1=	Port pin level > V_{IH}.
	0=	Port pin level < V_{IL}.

PORTB direction register TRISB (86H)

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

Bit7~Bit6	Not used	
Bit5~Bit0	TRISB<5:0>:	PORTB tristate control bit.
	1=	PORTB pin configured as input(tri-state).
	0=	PORTB pin configured as output.

Example: PORTB port procedure

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

6.3.2 PORTB analog selection control

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

PORTB analog selection register ANSELH (189H)

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

Bit7~Bit6 Not used

Bit5~Bit0 ANS<13:8>: Analog selection bit, selecting the analog or digital function of pin AN<13:8>.

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

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

6.3.3 PORTB pull-down resistance

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

PORTB pull-down resistance register: WPDB (08H)

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

Bit7~Bit6 Not used

Bit5~Bit0 WPDB<5:0>: Weak pull-down register bit.

1= Enable pull-down.

0= Disable pull-down.

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

6.3.4 PORTB pull-up resistance

Each PORTB pin has an individually configurable internal weak pull-up. Control bit WPUB<5:0> enables or disables each weak pull-up. When a port pin is configured as output, its weak pull-up is automatically cut-off.

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

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

Bit7~Bit6 Not used

Bit5~Bit0 WPUB<5:0>: Weak pull-up register bit.

1= Enable pull-up.

0= Disable pull-up.

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

6.3.5 PORTB interrupt on change

All PORTB pins can be individually configured as interrupt-on-change pins. The control bit IOCB <5:0> enables or disables this interrupt function for each pin. The interrupt-on-change pin is disabled on power-on reset.

For pins that have enabled interrupt on change, the value on the pin is compared to the old value latched during the last PORTB read. A logical or operation is performed with the "mismatch" output of the last read operation to set the PORTB interrupt-on-change flag bit (RBIF) in the INTCON register to 1.

This interrupt wakes up the device from its sleep state and can be cleared by the user in the interrupt service program by the following methods:

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

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

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

PORTB interrupt-on-change register: IOCB (96H)

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

Bit7~Bit6	Not used	
Bit5~Bit0	IOCB<5:0>	Control bit of PORTB interrupt on change.
	1=	Enable level change interrupt.
	0=	Disable level change interrupt.

6.4 I/O usage

6.4.1 Write I/O port

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

Example: write I/O port program

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

6.4.2 Read I/O port

Example: read I/O port program

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

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

6.5 Cautions 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 to prevent the I/O port from scanning the level by mistake.
3. When the I/O port is an input port, its input level should be between "VDD+0.7V" and "GND-0.7V". If the input port voltage is not within this range, the method shown in the figure below can be used.

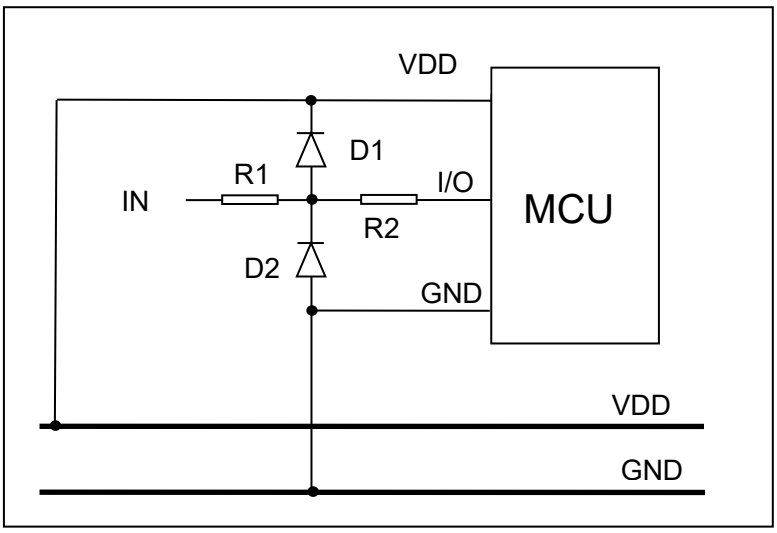


Figure 6-3: The input voltage is not within the specified range

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 Overview

The chip has multiple interrupt sources as follows.

- TIMER0 overflow interrupt
- TIMER2 match interrupt
- PORTA interrupt on change
- PORTB interrupt on change
- Program EEPROM write interrupt;
- TIMER1 overflow interrupt
- AD interrupt
- PWM interrupt
- INT interrupt

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

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

Executing the "Return from Interrupt" instruction (RETI) will exit the interrupt service program and set the GIE bit to 1, thereby re-allowing unmasked interrupts.

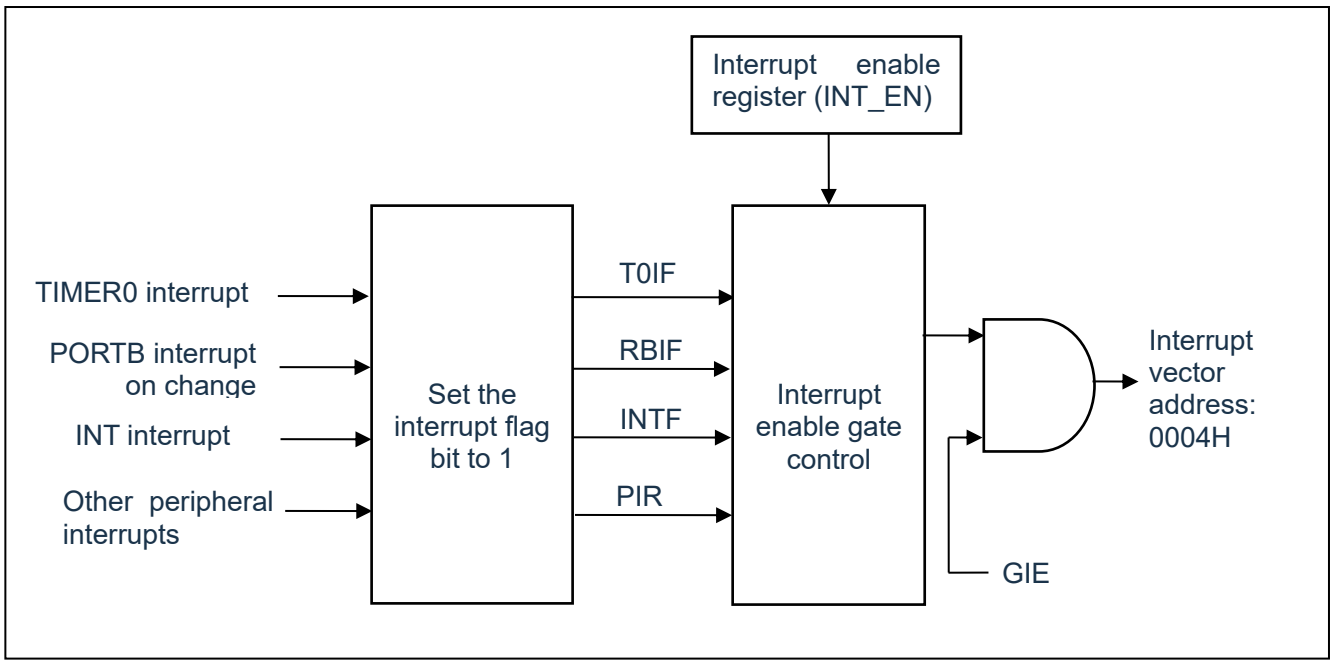


Figure 7-1: Interrupt principle diagram

7.2 Interrupt control register

7.2.1 Interrupt control register

The interrupt control register (INTCON) is a readable and writable register, including the enable 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 enabling an interrupt.

Interrupt control register: INTCON (0BH)

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

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

Bit6 PEIE: Peripheral interrupt enable bit;
1= Enable all unshielded peripheral interrupt;
0= Disable all peripheral 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 interrupt-on-change enable bit (1);
1= Enable PORTB interrupt-on-change;
0= Disable PORTB interrupt-on-change.

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

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

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

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 1 when TMR0 rolls over to 0. Reset will not change TMR0 and should be initialized before clearing the T0IF bit.

7.2.2 Peripheral interrupt enable register

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

Peripheral interrupt enable register: PIE1 (8CH)

8CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE1	RAIE	ADIE	----	----	EEIE	PWMIE	TMR2IE	TMR1IE
R/W	R/W	R/W	----	----	R/W	R/W	R/W	R/W
Reset value	0	0	----	----	0	0	0	0

Bit7 RAIE: PORTA interrupt-on-change enble bit;
1= Enable PORTA interrupt on change;
0= Disable PORTA interrupt on change.

Bit6 ADIE: AD converter (ADC) interrupt enable bit;
1= Enable ADC interrupts;
0= Disable ADC interrupts.

Bit5~Bit4 Reserved

Bit3 EEIE: EEDATA interrupt enable bit
1= Enable EEDATA write operation interrupt
0= Disable EEDATA write operation interrupt

Bit2 PWMIE: PWM interrupt enable bit;
1= enable PWM interrupt;
0= disable PWM interrupt.

Bit1 TMR2IE: TIMER2 and PR2 match interrupt enable bit;
1= enable TMR2 and PR2 match interrupt;
0= disable TMR2 and PR2 match interrupt.

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

Peripheral interrupt enable register: PIE2(8DH)

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

Bit7~Bit1 Not used.

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

7.2.3 Peripheral interrupt request register

The peripheral interrupt request 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 corresponding interrupt flag bit is cleared before enabling an interrupt.

Peripheral interrupt request register: PIR1 (0CH)

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

Bit7 RAIF: PORTA interrupt-on-change flag bit;
1= A PORTA interrupt-on-change flag bit is generated;
0= A PORTA interrupt-on-change flag bit is not generated.

Bit6 ADIF: AD converter interrupt flag bit;
1= AD conversion completed (must clear by software);
0= AD conversion is not completed or has not been started.

Bit5~Bit4 Reserved

Bit3 EEIF: Program EEPROM write operation interrupt bit;
1= Program EEPROM write operation completed (must clear by software);
0= Program EEPROM write operation is not complete or has not been started.

Bit2 PWMIF: PWM interrupt flag bit;
1= A PWM interrupt occurred (must be cleared by software)
0= No PWM interrupt occurred

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

Bit0 Not used, reads 0.

Peripheral interrupt request register: PIR2(0DH)

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

Bit7~Bit1 Not used.

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

7.3 Protection methods for interrupt

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

Example: Stack protection for ACC and STATUS

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

7.4 Interrupt priority and multi-interrupt nesting

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

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

8. TIMER0

8.1 Overview

TIMER0 is composed of the following functions:

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

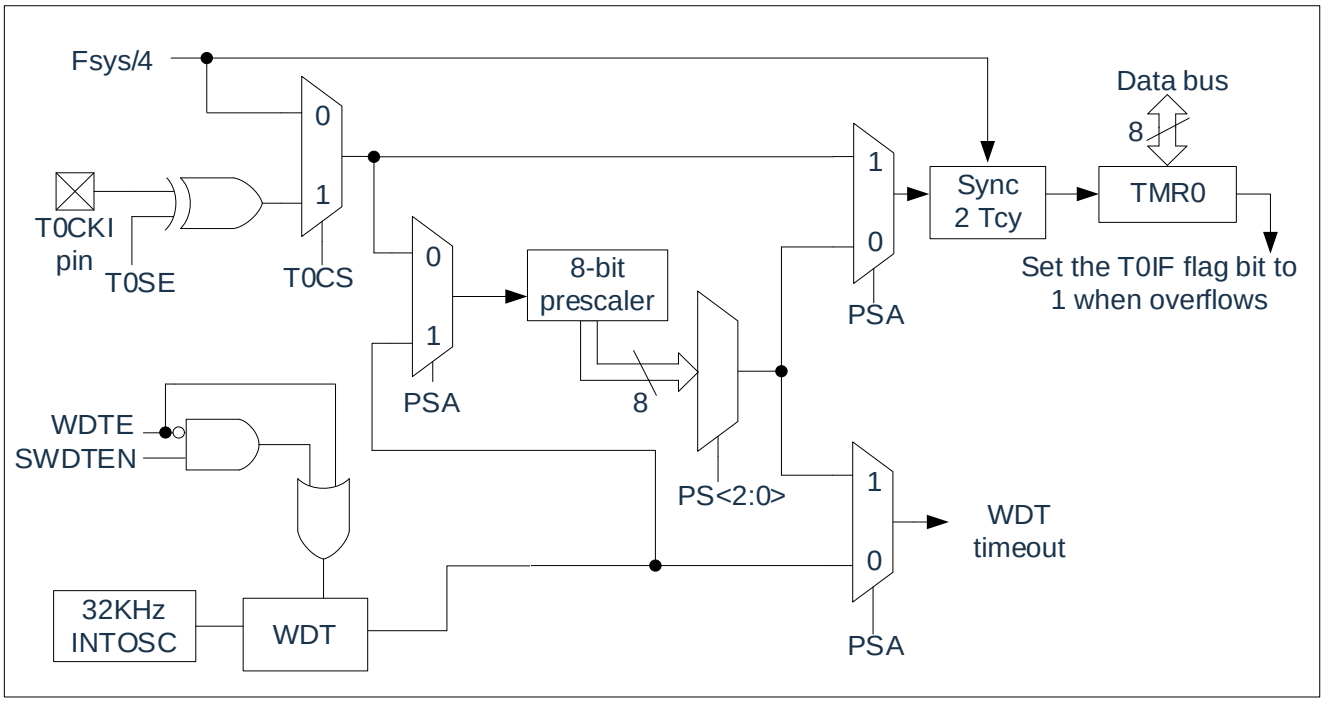


Figure 8-1: TIMER0/WDT module structure

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

8.2 Working principle for TIMER0

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

8.2.1 8-bit timer mode

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

8.2.2 8-bit counter mode

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

8.2.3 Software programmable pre-scaler

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

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

The pre-scaler is not readable and writable. When the pre-scaler is assigned to the TIMER0 mod, all instructions written to the TMR0 register will clear the pre-scaler. When the pre-scaler is assigned to the WDT, the CLRWDT instruction will clear both the pre-scaler and the 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 switch 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 general interrupt enable bit to
                                avoid entering the interrupt program when
                                executing the following specific timings
        LDIA    B'00000111'
        ORR     OPTION_REG,A    ;set pre-scaler to max. value
        CLR     TMR0            ;clear TMR0
        SETB    OPTION_REG,PSA  ;set pre-scaler to WDT
        CLRWDT                  ;clear WDT
        LDIA    B'xxxx1xxx'     ;set a 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 switch 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 a 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 by software. TIMER0 interrupt enable bit is the T0IE bit of the INTCON register.

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

8.3 TIMER0 related registers

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

TMR0 is an 8-bit readable and writable timer/counter, OPTION_REG is an 8-bit write-only register, the user can change the value of OPTION_REG to modify the working mode of TIMER0, etc. See 2.6 for the application of the 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= Internal instruction period clock (F_{sys}/4).

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

Bit3 PSA: Pre-scaler allocation bit.
1= Pre-scaler allocated to WDT.
0= Pre-scaler allocated toTIMER0 mod.

Bit2~Bit0 PS2~PS0: Pre-scaler parameter configuration bit.

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

9. TIMER1

9.1 Overview

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

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

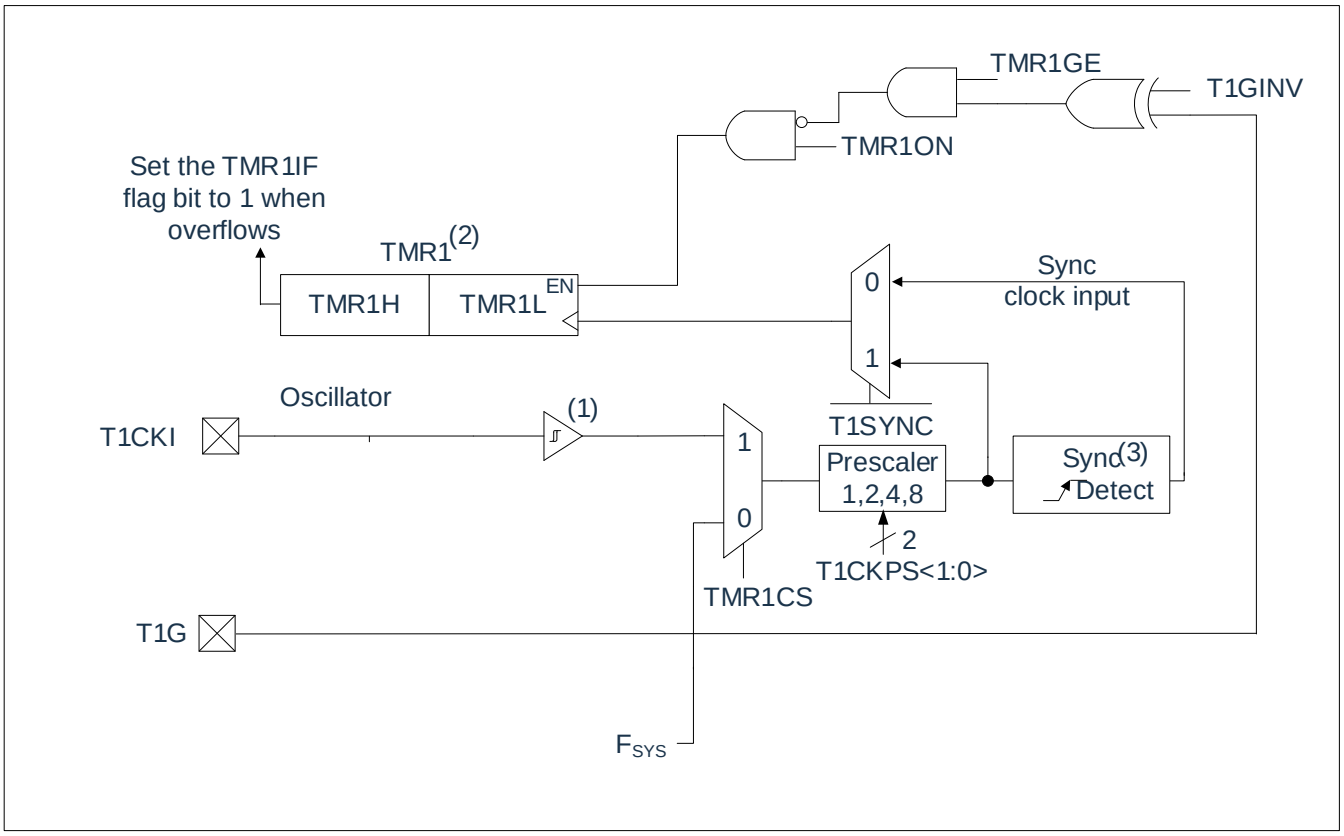


Figure 9-1: Block diagram of TIMER1

Note:

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

9.2 Working principle for TIMER1

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

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

9.3 Clock source selection

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

Clock source	TMR1CS
F_{SYS}	0
T1CKI pin	1

9.3.1 Internal clock source

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

9.3.2 External clock source

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

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

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

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

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

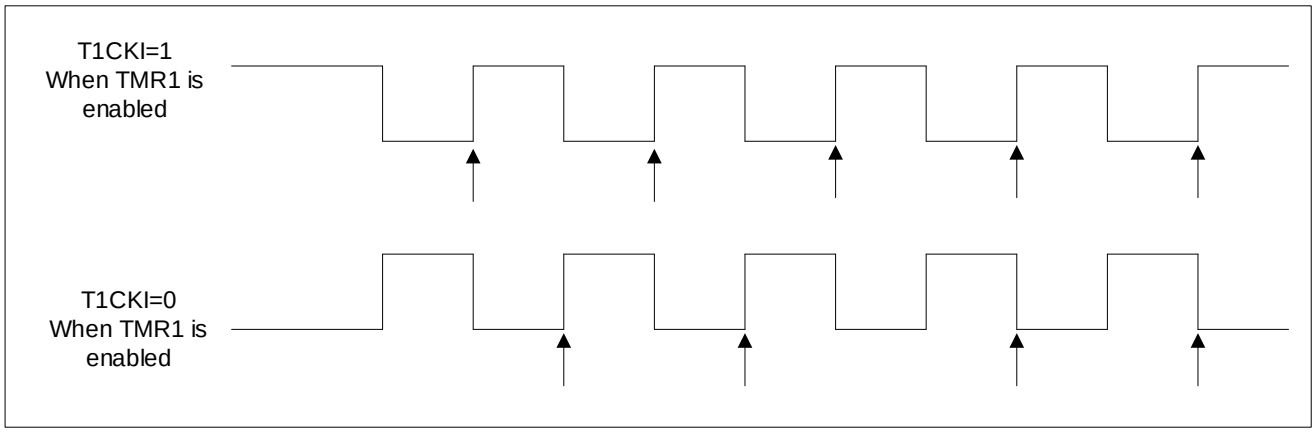


Figure 9-2: Incremental edge of TIMER1

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

9.4 TIMER1 pre-scaler

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

9.5 TIMER1 working principle in asynchronous counter mode

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

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

9.5.1 Read and write operations to TIMER1 in asynchronous counter mode

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

For write operations, it is recommended that the user stop the timer before writing the required value. When the register is counting, writing data to the timer register may cause a write conflict. This will generate unpredictable values in the TMR1H:TMR1L pair of registers.

9.6 TIMER1 gate control

The TIMER1 gating signal source can be configured in software to the T1G pin. This allows the device to use T1G directly for external event timing. For information on how to select the TIMER1 gating source, see "2.1.2 Data memory". This functional component can be just the software for the △-Σ A/D converter, or it can be used for many other applications.

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

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

9.7 TIMER1 interrupt

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

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

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

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

9.8 TIMER1 working principle during sleep

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

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

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

9.9 TIMER1 control register

TIMER1 control register: T1CON (10H)

10H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	---	T1SYNC	TMR1CS	TMR1ON
R/W	R/W	R/W	R/W	R/W	---	R/W	R/W	R/W
Reset value	0	0	0	0	---	0	0	0

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

Bit6 TMR1GE: TIMER1 gate control enable bit.
If TMR1ON=0, ignore this bit.
If TMR1ON=1: 1=TIMER1 count controlled by the TIMER1 gating function;
0=TIMER1 always counts.

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

Bit3 Reserved bit, disabled

Bit2 T1SYNC: TIMER1 external clock input synchronous control bit.
TMR1CS=1: 1= Not synchronized with external clock inputs;
0= Synchronized with external clock input.
TMR1CS=0: Ignoring this bit, TIMER1 uses the internal clock.

Bit1 TMR1CS: TIMER1 clock source selection bit;
1= Clock source from the T1CKI pin (rising edge triggered);
0= Internal clock source F_{SYS}.

Bit0 TMR1ON: TIMER1enable bit;
1= Enable TIMER1;
0= Disable TIMER1.

10. TIMER2

10.1 Overview

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

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

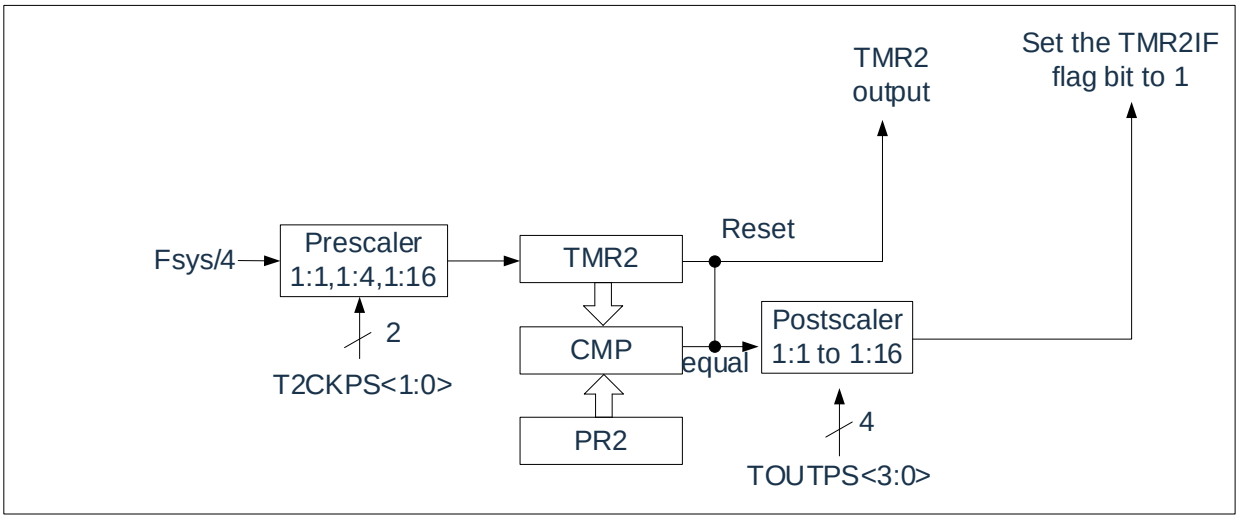


Figure 10-1: Block diagram of TIMER2

10.2 Working principle for TIMER2

The input clock of the TIMER2 mod is the system instruction clock (F_{SYS}/4). The clock is input to the TIMER2 pre-scaler. There are several division ratios to choose from: 1:1, 1:4 or 1:16. The output of the pre-scaler will be used to increment the TMR2 register.

The values of TMR2 and PR2 are continuously compared to determine when they match. TMR2 will be incremented 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 post-scaler counters are cleared under the following conditions:

- TMR2 write operation
- T2CON write operation
- Any device reset occurs (power-on reset, watchdog timer reset, or under voltage reset).

Note: Writing T2CON will not clear TMR2.

10.3 TIMER2 related registers

There are 2 registers related to TIMER2, namely data memory TMR2 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	---	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
Read/write	---	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	0	0	0	0	0	0	0

Bit7 Not used

Bit6~Bit3 TOUTPS<3:0>: TIMER2 output post-scaler 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 pre-scaler selection bit;

00= 1;
01= 4;
1x= 16.

11. Analog to Digital Conversion (ADC)

11.1 Overview

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

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

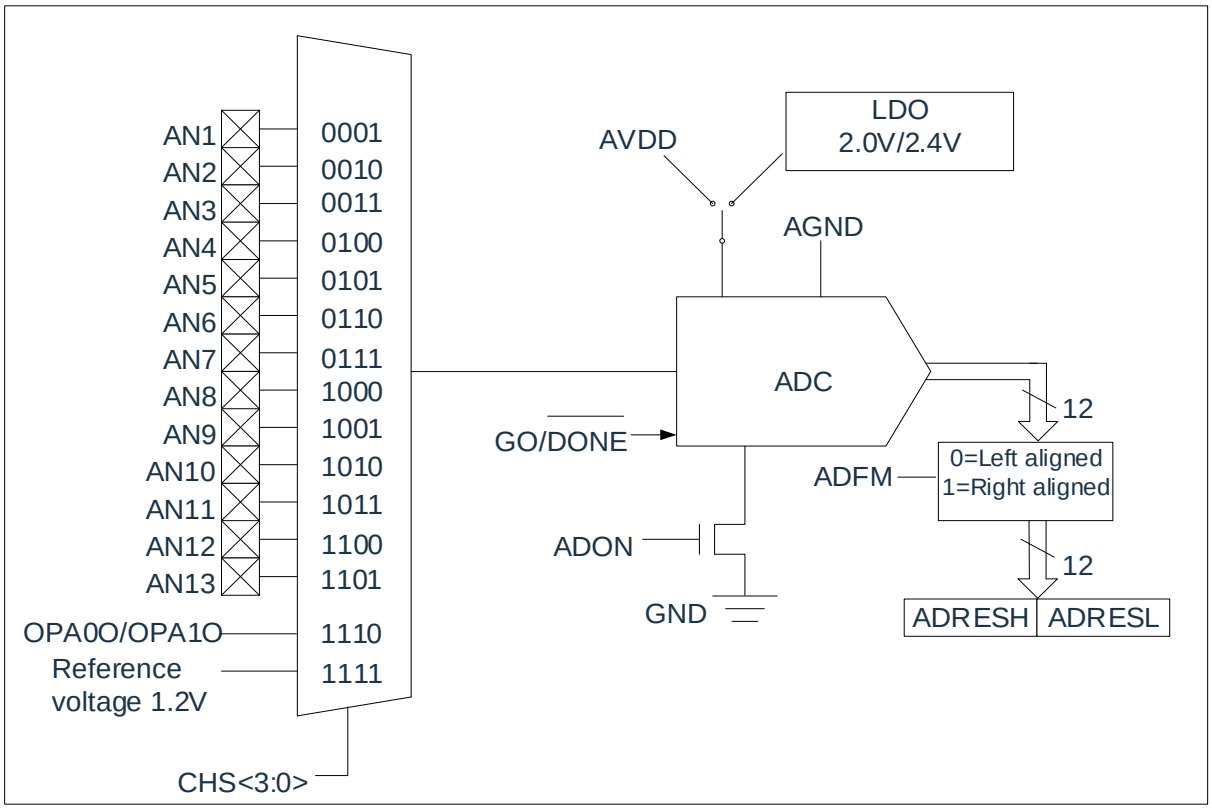


Figure 11-1: Block diagram of ADC

11.2 ADC configuration

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

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

11.2.1 Port configuration

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

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

11.2.2 Channel selection

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

If the channel is changed, a certain delay will be required before the next conversion starts. For more information, please refer to the chapter "11.3ADC operation principle".

11.2.3 ADC internal reference voltage

The chip has a built-in 1.2V reference voltage. To detect this reference voltage, set the ADCON0[5:2] bits to 1111.

11.2.4 ADC reference voltage

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

Note: When the internal LDO is selected as the reference voltage, the maximum effective accuracy of the ADC will be reduced. The lower the detection voltage is, the higher the ADC accuracy is obtained, and the recommended input voltage setting value is <1V.

11.2.5 Converter clock

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

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

The time to complete a one-bit conversion is defined as TAD. 49 TAD cycles are required for a complete 12-bit conversion.

The corresponding TAD specification must be met to obtain the correct conversion results. The following table shows an example of the correct ADC clock selection.

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

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

ADC clock period		Device frequency		
ADC clock source	ADCS<1:0>	8MHz	4MHz	1MHz
$F_{SYS}/8$	00	49.0μs	98.0μs	392.0μs
$F_{SYS}/16$	01	98.0μs	196.0μs	784.0μs
$F_{SYS}/32$	10	196.0μs	392.0μs	1.5ms
F_{RC}	11	1-3ms	1-3ms	1-3ms

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

11.2.6 ADC Interrupt

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

11.2.7 Result formatting

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

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

11.3 ADC operation principle

11.3.1 Start conversion

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

Note: You cannot use the same command that turns on the AD module to set GO/$\overline{\text{DONE}}$ to 1.

11.3.2 Complete conversion

When the conversion is complete, the ADC module will:

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

11.3.3 Stop conversion

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

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

11.3.4 Working principle for ADC in sleep mode

The ADC module can operate in sleep mode. This operation requires the ADC clock source to be set to the F_{RC} option. If the F_{RC} clock source is selected, the ADC waits one more instruction cycle before starting the conversion. This allows the STOP instruction to be executed to reduce system noise during conversion. If ADC interrupts are allowed, the device will wake up from sleep mode when the conversion is completed. If ADC interrupts are disabled, the ADC module will be turned off at the end of the conversion even if the ADON bit is held at 1. If the ADC clock source is not F_{RC}, executing the STOP instruction will abort the current conversion and shut down the AD module even if the ADON bit remains set to 1.

11.3.5 AD conversion procedure

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

1. Port configuration:
   - Disable pin output driver (see TRIS register);
   - Configure pin as analog input pin.
2. ADC mod configuration:
   - Select ADC reference voltage (AD conversion must be wait for up to 100us if the reference voltage switches from VDD to internal LDO);
   - Select ADC conversion clock;
   - Select ADC input channel;
   - Choose the format of the result;
   - Start the ADC mod.
3. ADC interrupt (optional) configuration:
   - Clear ADC interrupt flag bit;
   - Enable ADC interrupt;
   - Enable peripheral interrupt;
   - Enable global interrupt.
4. Wait for the required acquisition time.
5. Set GO/$\overline{\text{DONE}}$to 1 to start the conversion.
6. Wait for the ADC conversion to finish by one of the following methods:
   - Query the GO/$\overline{\text{DONE}}$ bit.
   - Wait for ADC interrupt (enable interrupt).
7. Read ADC results.
8. Clear the ADC interrupt flag bit to zero (this operation is required if interrupts are enabled).

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

Example: AD conversion

```
        LDIA    B'10000000'
        LD      ADCON1,A
        SETB    TRISA,1         ;Set PORTA.1 as input port
        SETB    ANSEL,1         ;Set PORTA,1 as analog port
        LDIA    B'11000101'
        LD      ADCON0,A
        CALL    DELAY           ;Delayed for a period of time
        SETB    ADCON0,GO
        SZB     ADCON0,GO       Wait for AD conversion to finish
        JP      $-1
        LD      A,ADRESH        ;Save the high bit of AD conversion result
        LD      RESULTH,A
        LD      A,ADRESL        ;Save the low bit of AD conversion result
        LD      RESULTL,A
```

11.4 ADC related registers

There are four main registers related to AD conversion, namely control register ADCON0, ADCON1, and data registers ADRESH and ADRESL.

AD control register: ADCON0 (1FH)

1FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit6 ADCS<1:0>: AD conversion clock selection bit
00= F_{SYS}/8
01= F_{SYS}/16
10= F_{SYS}/32
11= F_{RC}
(Clock generated by a dedicated internal oscillator with a frequency of up to 32 KHz = F_{HSI})

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

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

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

AD data register higher bit: ADCON1 (9FH)

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

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

Bit6~Bit3 Not used, reads 0.

Bit2 LDO_EN: Internal reference voltage enable bit.
1= Enabling the 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 the ADC reference voltage.

Bit1 Not used

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

AD data register higher bit: ADRESH(1EH), ADFM=0

1EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESH	ADRES11	ADRES10	ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4
R/W	R	R	R	R	R	R	R	R
Reset value	X	X	X	X	X	X	X	X

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

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

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

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

Bit3~Bit0 Not used.

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

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

Bit7~Bit2 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 bit: ADRESL(9EH), ADFM=1

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

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

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

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

12. PWM Module

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

The PWM outputs can be selected via CONFIG as RA1-RA5 or RA5-RA7, RB5, RB4 or RB0-RB4, where PWM0/PWM1, PWM2/PWM3 can be configured with complementary forward and reverse outputs

12.1 Pin configuration

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

12.2 Related register description

PWM control register 0: PWMCON0 (107H)

107H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMCON0	CLKDIV[2:0]			PWM4EN	PWM3EN	PWM2EN	PWM1EN	PWM0EN
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

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

Bit4~Bit0 PWMxEN: PWMx enable bit.
1= Enable PWMx.
0= Disable PWMx.

PWM control register 1: PWMCON1(108H)

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

Bit7~6 Not used

Bit5 PWM2DTEN: PWM2 dead-time enable bit.
1= Enable PWM2 dead-time function, PWM2 and PWM3 form a complementary output pair.
0= Disable PWM2 dead-time function.

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

Bit3~Bit2 Not used.

Bit1~Bit0 DT_DIV[1:0] Dead time clock source dividing frequency.
11= F_{HSI} /8
10= F_{HSI} /4
01= F_{HSI} /2
00= F_{HSI} /1

PWM control register: 2PWMCON2(109H)

109H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMCON2	---	---	---	PWM4DIR	PWM3DIR	PWM2DIR	PWM1DIR	PWM0DIR
R/W	---	---	---	R/W	R/W	R/W	R/W	R/W
Reset value	---	---	---	0	0	0	0	0

Bit7~Bit5 Not used.

Bit4~Bit0 PWMxDIR PWM output inversion control bit.
1= PWMx inversed output.
0= PWMx normal output.

PWM0~PWM3 period lower bit register: PWMTL (18FH)

18FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMTL	PWMT[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMT[7:0]: Lower 8 bits of PWM0~PWM3 period register.

PWM4 period lower bit register: PWMT4L(191H)

191H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMT4L	PWM4T[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWM4T[7:0]: Lower 8 bits of PWM4 period register.

Period higher bit register: PWMTH (190H)

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

Bit7~Bit6 Not used.
Bit5~Bit4 PWM4D[9:8]: Higher 2 bits of PWM4 duty register.
Bit3~Bit2 PWM4T[9:8]: Higher 2 bits of PWM4 period register.
Bit1~Bit0 PWMT[9:8]: Higher 2 bits of PWM0~PWM3 period register.

PWM0 duty cycle lower bit register: PWMD0L (193H)

193H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD0L	PWMD0[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

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

PWM1 duty cycle lower bit register: PWMD1L (194H)

194H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD1L	PWMD1[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

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

PWM2 duty cycle lower bit register: PWMD2L(195H).

195H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD2L	PWMD2[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

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

PWM3 duty cycle lower bit register: PWMD3L(196H)

196H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD3L	PWMD3[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

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

PWM4 duty cycle lower bit register: PWMD4L(197H)

197H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD4L	PWMD4[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD4[7:0]: Lower 8 bits of PWM4 duty cycle.

PWM0 and PWM1 duty cycle higher bit register: PWMD01H (11CH)

11CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD01H	---	---	PWMD1[9:8]		---	---	PWMD0[9:8]	
R/W	---	---	R/W	---	---	----	R/W	R/W
Reset value	---	---	0	---	---	----	0	0

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

PWM2 and PWM3 duty cycle higher bit register: PWMD23H (11DH)

11DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD23H	---	---	PWMD3[9:8]		---	---	PWMD2[9:8]	
R/W	---	---	R/W	---	---	----	R/W	R/W
Reset value	---	---	0	---	---	----	0	0

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

PWM0 and PWM1 dead time register: PWM01DT (93H)

93H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM01DT	---	---	PWM01DT[5:0]					
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	---	0	0	0	0	0	0

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

PWM2 and PWM3 dead time register: PWM23DT (94H)

94H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM23DT	---	---	PWM23DT[5:0]					
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	---	0	0	0	0	0	0

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

12.3 PWM period

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

Formula 1: PWM period:

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

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

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

- PWM period counter is cleared;
- PWMx pin is set to 1;
- New period of PWM is latched;
- New duty of PWMx is latched;
- Generating a PWM interrupt flag bit.

12.4 PWM duty cycle

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

You can write the PWMDxL and PWMDxxH register at any time, but until the values in PWM period counter and PWMT match (that is, the period ends), the value of the duty cycle is latched into internal latch.

Formula 2: Pulse width calculation formula:

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

Formula 3: PWM duty cycle calculation formula:

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

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

12.5 System clock frequency changes

The PWM frequency is generated by the system clock frequency. Any change in the system clock frequency will not change the PWM frequency.

12.6 Programmable dead-time delay mode

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

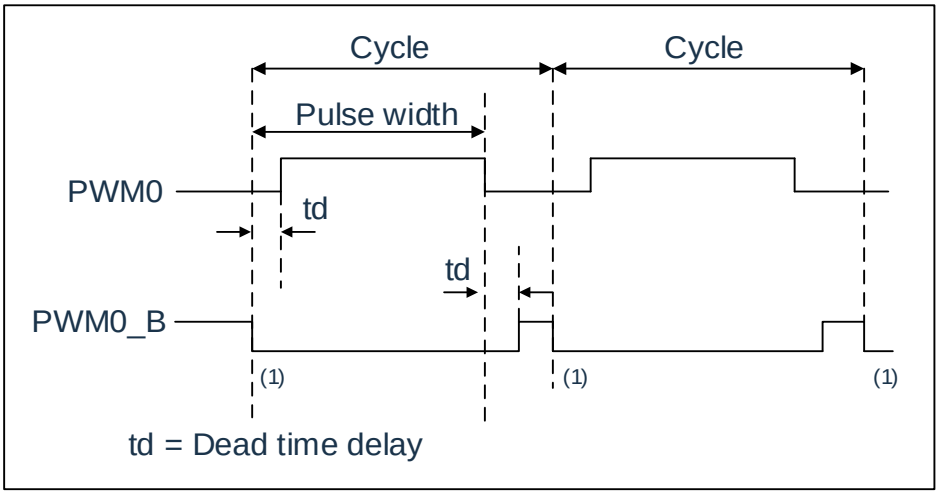


Figure 12-1: Example of PWM dead time delay output

Dead-time calculation formula:

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

12.7 PWM configuration

The following steps should be performed when configuring PWM mod:

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

13. Program EEPROM and Program Memory Control

13.1 Overview

The devices in this series have 4K words of program memory, the address range is from 0000h to 0FFFh, which is read-only in all address ranges; the device has a 128-byte program EEPROM, and the address range is 0000h to 007Fh, which is readable/writable in all address ranges.

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

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

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

When accessing the program memory of the device, the EEDAT and EEDATH register form a double byte word to save the 16-bit data to be read, and the EEADR and EEADRH register form a double byte word to save the 12-bit EEPROM unit address to be read.

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

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

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

Note:

1. Program memory means ROM space, i.e. the space where instruction codes are stored, read only; Program EEPROM is the space where user data can be stored, read and written.
2. The normal write voltage range of program EEPROM is 3.5V~5.5V, and the write current is 20mA@VDD=5V.

13.2 Related registers

13.2.1 EEADR and EEADRH registers

The EEADR and EEADRH registers can address up to 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.

13.2.2 EECON1 and EECON2 Registers

EECON1 is the control register to access the program EEPROM.

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

The control bits RD and WR start reading and writing respectively. Software can only set these bits to 1 but cannot clear them. 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, write operations to the program EEPROM are allowed. At power-on, the WREN bit is cleared to zero. The WRERR bit is set to 1 when a normal write operation is interrupted by an LVR reset or a WDT timeout reset. in these cases, the user can check the WRERR bit and rewrite the corresponding units after the reset.
- The interrupt flag bit EEIF in the PIR1 register is set to 1 when the write operation is completed. This flag bit must be cleared by software.

EECON2 is not a physical register. Reading results of EECON2 are all 0.

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

EEPROM data register: EEDAT (10CH)

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

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

EEPROM address register: EEADR (10DH)

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

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

EEPROM data register: EEDATH (10EH)

10EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEDATH	EEDATH7	EEDATH6	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	X	X	X	X	X	X	X	X

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

EEPROM address register: EEADRH (10FH)

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

Bit7~Bit4 Not used, reads 0.

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

EEPROM control register: EECON1(18CH)

18CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EECON1	EEPGD	---	---	---	WRERR	WREN	WR	RD
R/W	R/W	---	---	---	R/W	R/W	R/W	R/W
Reset value	0	---	---	---	X	0	0	0

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

Bit6~Bit4 Not used.

Bit3 WRERR: EEPROM error flag bit;
- 1= Premature termination of the write operation (any WDT reset or undervoltage reset during normal operation);
- 0= Write operation completed.

Bit2 WREN: EEPROM write enable bit;
- 1= Enable write cycles;
- 0= Write to memory is prohibited.

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 completed.

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.

13.3 Read program EEPROM

To read the program EEPROM unit, the user must write the address to the EEADR register, clear the EEPGD control bit of the EECON1 register, and then set the control bit RD to 1. Once the read control bit is set, the program EEPROM controller will use the second instruction period to read data. This will cause the second instruction following the “SETB EECON1, RD” instruction to be ignored (1). In the next clock period, the corresponding address value of the program EEPROM will be latched into the EEDAT register 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 a program memory read operation must be NOP, which prevents the user from executing a double-cycle instruction on the next instruction after the RD bit is set to 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                     ; To read data here, NOP instruction must be added.
NOP
LD      A,EEDAT         ; Read data to ACC
```

13.4 Write program EEPROM

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

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

In addition, the WREN bit in EECON1 must be set to 1 to enable write operations. This mechanism can prevent EEPROM from being written by mistake due to code execution errors (abnormal) (program 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 complete interrupt flag bit (EEIF) is set to 1. Users 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 and reset the chip during this period.

Example: write program EEPROM

```
        LD      A,EE_ADD        ;Put the address to be written into the EEADR register
        LD      EEADR,A
        LD      A,EE_DATA       ;put the data to be written to the EEDAT register
        LD      EEDAT,A
        CLRWDT
        CLRB    EECON1,EEPGD
        SETB    EECON1,WREN     ;enable write operation
        CLRB    INTCON,GIE      ;disable all interruptions
        SZB     INTCON,GIE
        JP      $-2
        LDIA    055H            ;Write 55H to EECON2
        LD      EECON2,A
        LDIA    0AAH            ;Write 0AAH to EECON2
        LD      EECON2,A
        SETB    EECON1,WR       ;Enable write signal
        SETB    INTCON,GIE
        SZB     EECON1,WR       ; Determines if the write operation is complete.
        JP      $-1
        CLRB    EECON1,WREN     ;write complete, turn off write enable bit
```

13.5 Read program memory

To read the program memory unit, the user must write the high and low bits of the address to the EEADR and EEADRH registers respectively, set the EEPGD bit of EECON1 register to 1, and then set the control bit RD to 1. Once the read control bit is set, the program memory controller will use the second instructions period 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 and EEDATH registers, the user can read these two registers in the subsequent instructions. The EEDAT and EEDATH register will save the values until the next time the user reads or writes data to the unit.

Note:

1. The two instructions following a program memory read operation must be NOP. This prevents the user from executing a double-cycle instruction on the next instruction after the RD bit is set to 1.
2. When EEPGD=1, if WR is 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
```

13.6 Write program memory

Program memory is read-only, not writable.

13.7 Cautions on program EEPROM

13.7.1 Program EEPROM burn-in time

The program EEPROM write time is not fixed, the time required to write different data varies from 100us to 5ms, and the CPU stops working during the write time, the program needs to do the relevant processing.

13.7.2 Write verification

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

13.7.3 Protection against miswrites

In some cases, users may not want to write data to the program EEPROM. In order to prevent accidental writing to 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:

- Under voltage
- Power glitches
- Software failure

14. Operational Amplifier (OPA0, OPA1)

There are two op-amps OPA0 and OPA1 in the chip, both of them have the same function and performance, and OPA0 is used for the following description.

14.1 OPA0

OPA0 has the following functions:

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

14.1.1 OPA0 enable

Setting the 7th bit of register OPA0CON to 1 enables the operational amplifier. Setting OPA0EN to 0 disables the operational amplifier.

After enabling the op-amp, the positive and negative terminals are automatically connected to the I/O port.

14.1.2 OPA0 port selection

14.1.2.1 OPA0 positive input

After enabling the op-amp, the positive terminal is automatically connected to the I/O port.

14.1.2.2 OPA0 negative input

After enabling the op-amp, the negative terminal is automatically connected to the I/O port.

14.1.2.3 OPA0 output

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

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

14.1.2.4 Port direction setting when OPA0 is used

OPA0 uses the relevant I/O ports that must be set to the input state, including op-amp inputs and op-amp outputs (when needed to be connected to IO).

14.1.3 OPA0 operation mode

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

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

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

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

Regulation mode workflow:

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

14.1.4 OPA0 related registers

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

OPA0 control register: OPA0CON (9AH)

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

Bit7	OPA0EN:	OPA0 enable bit;
	1=	Enable OPA0;
	0=	Disable OPA0.
Bit6	OPA0O	Op-amp output selection;
	1=	OPA0 output connected to the I/O port (OPA0O pin);
	0=	The OPA0 output is not connected to the I/O port.
Bit5	OPA0_CMP:	Comparator mode;
	1:	Comparator mode, comparator output can be read via OPA0ADJ[7];
	0:	Op-amp mode.
Bit4	OPA0_ADC:	Op-amp output to the ADC control bit;
	1=	Op-amp output to channel ADC14;
	0=	Op-amp output is not connected to the ADC.
Bit3	OPA0FW	OPA0 follow mode control bit; (valid only in op-amp mode)
	1=	PA0 negative terminal is shorted to the output, OPA0 negative terminal is automatically disconnected from the I/O port;
	0=	OPA0 negative terminal is not shorted to the output.
Bit2	OPA0BG	Internal reference 1.2V connected to the OPA0 positive control bit;
	1=	Internal reference connected to the OPA0 positive input, OPA0 positive is automatically disconnected from the I/O port;
	0=	Not connected.
Bit1~Bit0	Not used	

Note: Only one output of OPA0 and OPA1 can be selected at a time to be connected to ADC14 channel.

OPA0 offset-voltage regulation register: OPA0ADJ (9BH)

9BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPA0ADJ	OPA0OUT	OPA0COFM	OPA0CRS	OPA0ADJ[4:0]				
Read/write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	1	0	0	0	0

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

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

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

Bit4~Bit0 OPA0ADJ[4:0]: OPA0 offset voltage regulation bit.

15. Low Voltage Detection (LVD)

15.1 Overview

The SC8F37xx series microcontroller has a low voltage detection function, which can be used to monitor the supply voltage and generate an interrupt signal if the supply voltage is lower than the set value; the program can read the LVD output flag bit in real time.

15.2 LVD related registers

There is one register related to the LVD module.

LVD control register: LVDCON (97H)

97H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LVDCON	LVD_RES	—	—	—	LVD_SEL[2:0]			LVDEN
R/W	R	—	—	—	R/W	R/W	R/W	R/W
Reset value	X	—	—	—	0	0	0	0

Bit7	LVD_RES:	LVD output result
	0=	VDD> LVD set voltage;
	1=	VDD< LVD set voltage;
Bit6~Bit4	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;

15.3 Operation of LVD

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

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

16. Electrical Parameters

16.1 Limit parameters

Power supply voltage……GND-0.3V~GND+6.0V
Storage temperature……-50℃~125℃
Operating temperature……-20℃~75℃
Port input voltage……GND-0.3V~VDD+0.3V
Maximum sink current for all ports……200mA
Maximum source current for all ports……-150mA

Note: If the device operating conditions exceed the above "limit parameters", it may cause permanent damage to the device. The above values are only the maximum values of the operating conditions. We do not recommend that the device operate outside the range specified in this specification. The stability of the device is affected when it is operated for a long time at the limit value.

16.2 DC electrical characteristics

Symbol	Item	Test condition		Min.	Typ.	Max.	Unit
		VDD	Condition				
VDD	Operating voltage	-	16MHz	3.5	-	5.5	V
		-	8MHz	1.8	-	5.5	V
I_{DD}	Operating current	5V	Turn off all analog mode, Fsys=8MHz	-	3	-	mA
		3V	Turn off all analog mode, Fsys=8MHz	-	2	-	mA
		5V	Turn off all analog mode, Fsys=16MHz	-	3.8	-	mA
		3V	Turn off all analog mode, Fsys=16MHz	-	2.7	-	mA
		5V	Burning program EEPROM	-	20	-	mA
		3V	Burning program EEPROM	-	10	-	mA
I_{STB}	Static current	5V	----	-	0.1	2	μA
		3V	----	-	0.1	1	μA
V_{IL}	Low level input voltage	-	----	-	-	0.3VDD	V
V_{IH}	High level input voltage	-	----	0.7VDD	-	-	V
V_{OH}	High level output voltage	-	Without load	0.9VDD	-	-	V
V_{OL}	Low level output voltage	-	Without load	-	-	0.1VDD	V
V_{ADI}	AD port input voltage	-	----	0	-	V_{ADREF}	V
V_{AD}	AD module operating voltage	-	V_{ADREF}=VDD	2.7	-	5.5	V
		-	V_{ADREF}=2.0V CLK_{ADC}= F_{SYS} /32	2.2		5.5	V
		-	V_{ADREF}=2.4V CLK_{ADC}= F_{SYS} /32	2.6		5.5	V
V_{EEPROM}	EEPROM module operating voltage	-	----	3.5	-	5.5	V
E_{AD}	AD conversion error	-	----	-	±2	-	-
R_{PH}	Pull-up resistance value	5V	V_O=0.5VDD	-	40	-	KΩ

		3V	V_O=0.5VDD	-	73	-	KΩ
R_{PD}	Pull-down resistance value	5V	V_O=0.5VDD	-	40	-	KΩ
		3V	V_O=0.5VDD	-	70	-	KΩ
I_{OL}	Output port sink current	5V	V_{OL}=0.3VDD	-	60	-	mA
		3V	V_{OL}=0.3VDD	-	25	-	mA
I_{OH1}	Output port source current RA1~RA7, RB5	5V	V_{OH}=0.7VDD	-	21	-	mA
		3V	V_{OH}=0.7VDD	-	8	-	mA
I_{OH2}	Output port source current RB0~RB4	5V	V_{OH}=0.7VDD	-	34	-	mA
		3V	V_{OH}=0.7VDD	-	12	-	mA
V_{BG}	Internal reference voltage 1.2V	VDD=2.5~5.5V T_A=25℃		-1.5%	1.2	1.5%	V
		VDD=2.5~5.5V T_A=－20~75℃		-2.0%	1.2	2.0%	V

(V_{DD}=5V, T_A= 25°C, unless otherwise noted)

16.3 ADC internal LDO reference voltage characteristics

(T_A= 25°C, unless otherwise noted)

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

16.4 OPA electrical characteristics

(T_A= 25°C, unless otherwise noted)

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

16.5 LVR electrical characteristics

(T_A= 25°C, unless otherwise noted)

Symbol	Item	Test condition	Min.	Typ.	Max.	Unit
V_{LVR1}	LVR set voltage =1.8V	VDD=1.6~5.5V	1.7	1.8	1.9	V
V_{LVR2}	LVR set voltage =2.0V	VDD=1.8~5.5V	1.9	2.0	2.1	V
V_{LVR3}	LVR set voltage =2.6V	VDD=2.4~5.5V	2.5	2.6	2.7	V
V_{LVR4}	LVR set voltage =3.5V	VDD=3.3~5.5V	3.4	3.5	3.6	V

16.6 AC electrical characteristics

(T_A= 25°C, unless otherwise noted)

Symbol	Item	Test condition		Min.	Typ.	Max.	Unit
		VDD	Condition				
T_{WDT}	WDT reset time	5V	-		16		ms
		3V	-		32		ms
T_{EEPROM}	EEPROM programming time	5V	F_{HSI}=8MHz			5	ms
		3V	F_{HSI}=8MHz			5	ms
		5V	F_{HSI}=16MHz			5	ms
		3V	F_{HSI}=16MHz			5	ms
F_{RC}	Internal oscillation frequency stability	VDD=4.0~5.5V	T_A=25℃	-1.5%	8	+1.5%	MHz
		VDD=2.5~5.5V	T_A=25℃	-2.0%	8	+2.0%	MHz
		VDD=4.0~5.5V	T_A=－20~75℃	-2.5%	8	+2.5%	MHz
		VDD=2.5~5.5V	T_A=－20~75℃	-3.5%	8	+3.5%	MHz
		VDD=1.8~5.5V	T_A=－20~75℃	-5.0%	8	+5.0%	MHz
		VDD=4.0~5.5V	T_A=25℃	-1.5%	16	+1.5%	MHz
		VDD=3.5~5.5V	T_A=25℃	-2.0%	16	+2.0%	MHz
		VDD=4.0~5.5V	T_A=－20~75℃	-2.5%	16	+2.5%	MHz
		VDD=3.5~5.5V	T_A=－20~75℃	-3.5%	16	+3.5%	MHz

17. Instructions

17.1 Instruction set

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

mnemonic sign		operation	instruction period	symbol
DECR	[R]	ecrement 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
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

17.2 Description of instructions

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
HSUB    R01     ;execute: R01=(80H-77H-C)=09H(C=0)
CR                        R01=(80H-77H-C)=08H(C=1)
```

INCA **[R]**

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

period: 1

Affected flag bit: Z

example:

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

INCR **[R]**

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

period: 1

Affected flag bit: Z

example:

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

JP **add**

operation: Jump to add address

period: 2

Affected flag bit: None

example:

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

LD **A,[R]**

operation: Load the value of R to ACC

period: 1

Affected flag bit: Z

example:

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

LD **[R],A**

operation: Load the value of ACC to R

period: 1

Affected flag bit: none

example:

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

LDIA **i**

operation: Load i to ACC

period: 1

Affected flag bit: none

example:

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

NOP

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

```
NOP
NOP
```

ORIA **i**

operation: Perform 'OR' on I and ACC, save the result to ACC
period: 1
Affected flag bit: Z
example:

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

ORA **[R]**

operation: Perform 'OR' on R and ACC, save the result to ACC
period: 1
Affected flag bit: Z
example:

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

ORR **[R]**

operation: Perform 'OR' on R and ACC, save the result to R
period: 1
Affected flag bit: Z
example:

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

RET

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

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

RET i

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

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

RETI

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

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

RLCA [R]

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

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

RLCR **[R]**

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

period: 1

Affected flag bit: C

example:

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

RLA **[R]**

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

period: 1

Affected flag bit: none

example:

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

RLR **[R]**

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

period: 1

Affected flag bit: none

example:

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

RRCA **[R]**

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

period: 1

Affected flag bit: C

example:

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

RRCR **[R]**

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

period: 1

Affected flag bit: C

example:

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

RRA **[R]**

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

period: 1

Affected flag bit: none

example:

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

RRR **[R]**

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

period: 1

Affected flag bit: none

example:

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

SET **[R]**

operation: Set all bits in register R as 1

period: 1

Affected flag bit: none

example:

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

SETB **[R],b**

operation: Set b bit in register R to 1

period: 1

Affected flag bit: none

example:

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

STOP

operation: Enter sleep

period: 1

Affected flag bit: TO, PD

example:

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

SUBIA **i**

operation: I minus ACC, save the result to ACC

period: 1

Affected flag bit: C,DC,Z,OV

example:

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

SUBA **[R]**

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

period: 1

Affected flag bit: C,DC,Z,OV

example:

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

SUBR **[R]**

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

period: 1

Affected flag bit: C,DC,Z,OV

example:

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

SUBCA **[R]**

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

period: 1

Affected flag bit: C,DC,Z,OV

example:

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

SUBCR **[R]**

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

period: 1

Affected flag bit: C,DC,Z,OV

example:

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

SWAPA **[R]**

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

period: 1

Affected flag bit: none

example:

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

SWAPR **[R]**

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

period: 1

Affected flag bit: none

example:

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

SZB **[R],b**

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

period: 1 or 2

Affected flag bit: none

example:

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

SNZB **[R],b**

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

period: 1 or 2

Affected flag bit: none

example:

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

SZA **[R]**

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

period: 1 or 2

Affected flag bit: none

example:

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

SZR **[R]**

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

period: 1 or 2

Affected flag bit: none

example:

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

SZINCA **[R]**

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

period: 1 or 2

Affected flag bit: none

example:

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

SZINCR **[R]**

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

period: 1 or 2

Affected flag bit: none

example:

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

SZDECA **[R]**

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

period: 1 or 2

Affected flag bit: none

example:

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

SZDECR **[R]**

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

period: 1 or 2

Affected flag bit: none

example:

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

TABLE **[R]**

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

period: 2

Affected flag bit: none

example:

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

TABLEA

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

period: 2

Affected flag bit: none

example:

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

TESTZ **[R]**

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

period: 1

Affected flag bit: Z

example:

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

XORIA **i**

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

period: 1

Affected flag bit: Z

example:

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

XORA **[R]**

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

period: 1

Affected flag bit: Z

example:

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

XORR **[R]**

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

period: 1

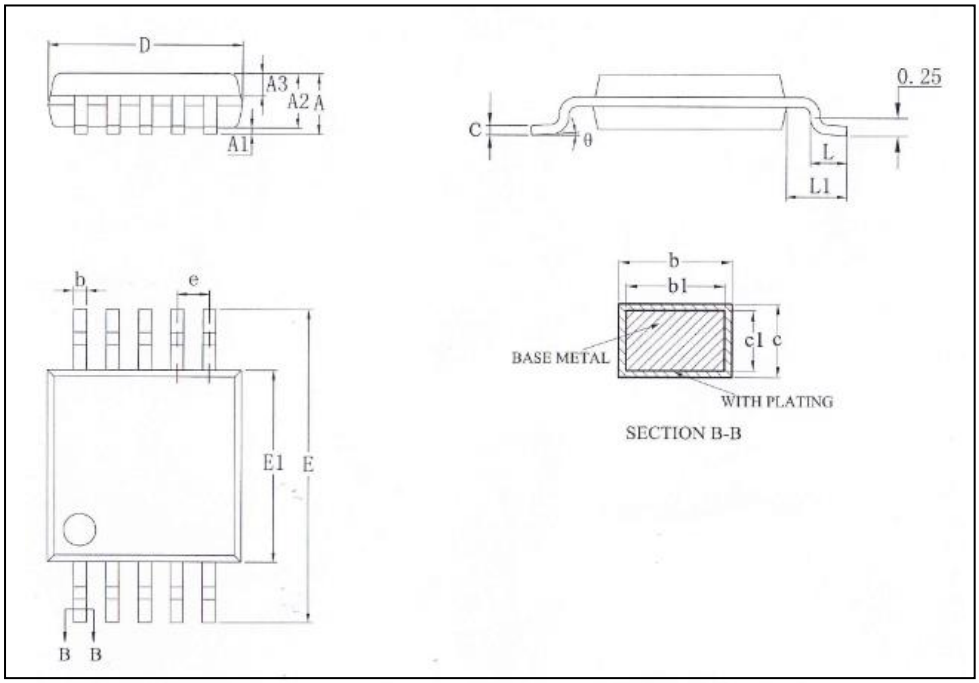
Affected flag bit: Z

example:

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

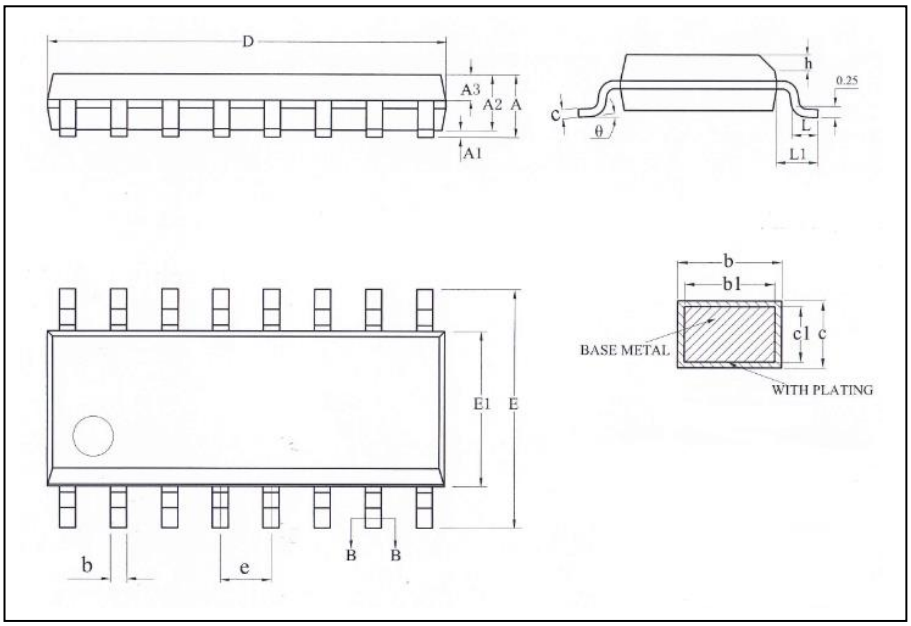
18. Packages

18.1 MSOP10



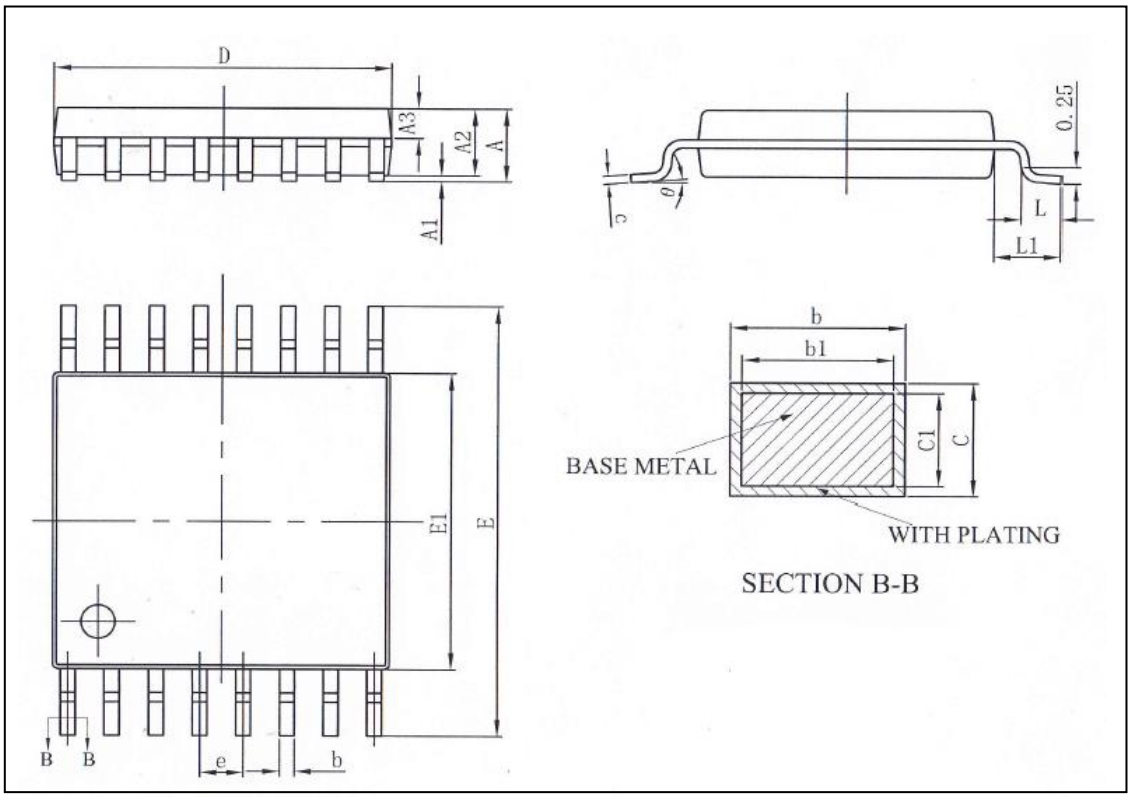
Symbol	Millimeter		
	Min	Nom	Max
A	-	-	1.10
A1	0.05	-	0.15
A2	0.75	0.85	0.95
A3	0.30	0.35	0.40
b	0.18	-	0.26
b1	0.17	0.20	0.23
c	0.15	-	0.19
c1	0.14	0.15	0.16
D	2.90	3.00	3.10
E	4.70	4.90	5.10
E1	2.90	3.00	3.10
e	0.50BSC		
L	0.40	-	0.70
L1	0.95REF		
θ	0	-	8°

18.2 SOP16



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

18.3 TSSOP16



Symbol	Millimeter		
	Min	Nom	Max
A	-	-	1.20
A1	0.05	-	0.15
A2	0.90	1.00	1.05
A3	0.39	0.44	0.49
b	0.20	-	0.28
b1	0.19	0.22	0.25
c	0.13	-	0.17
c1	0.12	0.13	0.14
D	4.90	5.00	5.10
E	6.20	6.40	6.60
E1	4.30	4.40	4.50
e	0.65BSC		
L	0.45	0.60	0.75
L1	1.05REF		
θ	0	-	8°

19. Revision History

Version	Date	Revised contents
V1.0	September 2019	Initial version
V1.1	October 2019	Corrected the description of some functions
V1.2	December 2019	Corrected the description of electrical parameters such as "Static current" and "LVR electrical characteristics"
V1.3	April 2020	Corrected some errors in the package diagram
V1.4.0	March 2022	1) Added some clock block diagrams. 2) Revised the internal high-speed oscillation frequency to FHSI and revised the clock sources of other modules according to the clock block diagram. 3) Revised the sleep wakeup wait time and ADC internal LDO reference voltage characteristics. 4) Deleted the description of sleep wake-up related to ADC interrupts.