SC8P171xE User Manual

Enhanced OTP 8-bit CMOS microcontrollers

Rev. 1.4.2

Please note the following CMS IP policy

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

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

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

Contents

1. Product Description

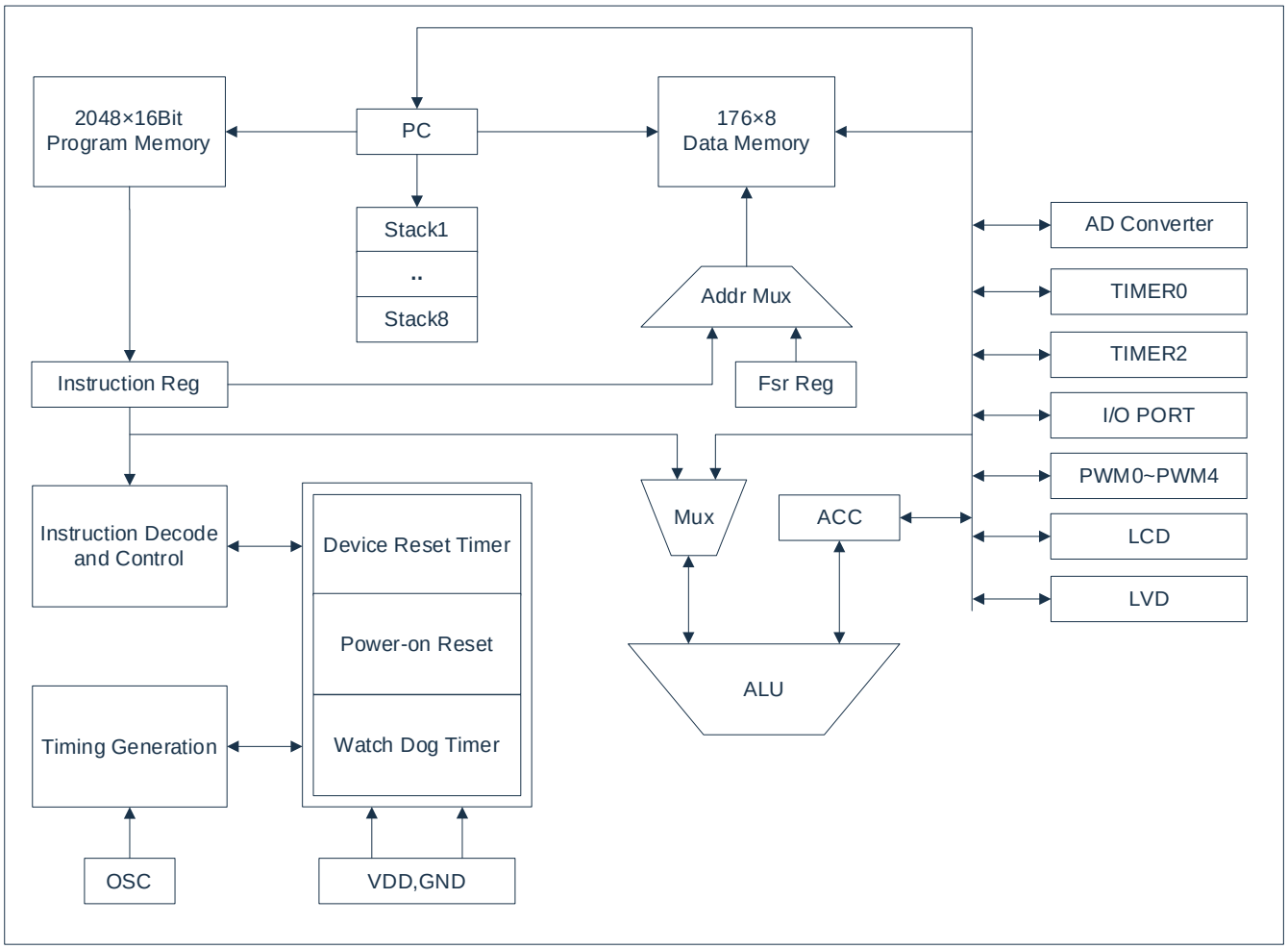
1.1 Features

- Memory
  - OTP: 2Kx16
  - Universal RAM: 176x8
- 8-level stack buffer
- Look-up table function
- Short and clear instruction system (68 instructions)
- Built-in low voltage detection circuit
- Interrupt sources
  - 2 timer interrupts
  - RA, RB ports interrupt on change
  - Other peripheral interrupts
- Timer
  - 8-bit timer TIMER0, TIMER2
- PWM module
  - 5-channel PWM output, selectable polarity
  - 4-channel PWM with shared period, independent duty cycle
  - 1-channel PWM with shared period, independent duty cycle
  - 10-bit PWM accuracy
- Operating voltage: 2.6V—5.5V@16MHz
  1.8V—5.5V@8MHz
  Operating temperature: -40℃—85℃
- Built-in high-precision RC oscillator: designed frequency of 8MHz/16MHz
- Instruction period (single or dual instruction)
- Built-in LCD1/2 Bias COM drive module
  - SEG/COM outputs are optional for all I/O ports (except RB2).
  - Drive current is selectable for all I/O ports (except RB2)
- Built-in high precision 1.2V reference voltage
  - ±1.5% @VDD=2.5V~5.5V T_A=25℃
  - ±2% @VDD=2.5V~5.5V T_A=-40℃~85℃
- High-precision 12-bit ADC
  - The reference voltage can be selected from 2V, 2.4V, and VDD.
- LVD module
  - Supports multiple voltage options: 2.2V/ 2.4V/2.7V/ 3.0V/ 3.3V/3.7V/4.0V/4.3V

Model description

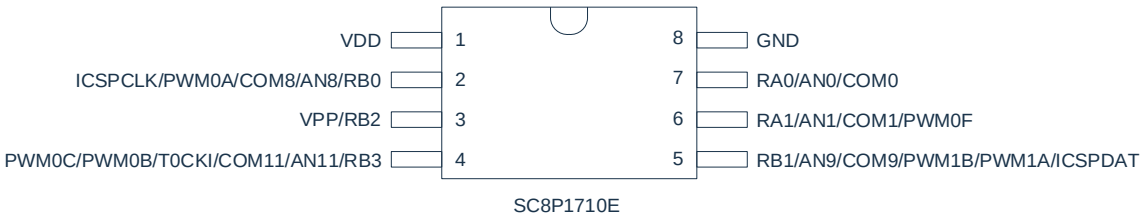
Product	ROM	RAM	I/O	LCD	ADC	PACKAGE
SC8P1710E	2K×16Bit	176×8Bit	5+1	5COM	5×12Bit	SOP8
SC8P1715E	2K×16Bit	176×8Bit	7+1	7COM	7×12Bit	MSOP10/ DFN10
SC8P1711E	2K×16Bit	176×8Bit	11+1	11COM	11×12Bit	SOP14
SC8P1712E	2K×16Bit	176×8Bit	13+1	13COM	13×12Bit	SOP16/ QFN16

1.2 System structure diagram



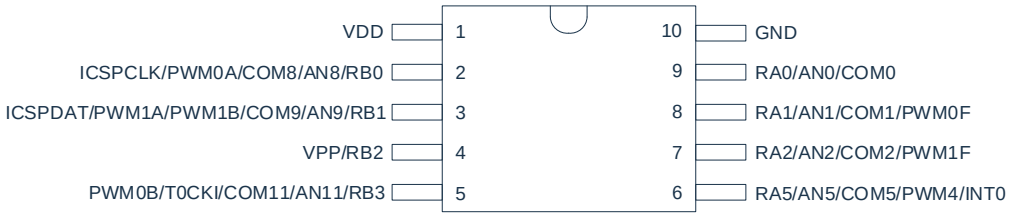
1.3 Top view

1.3.1 SC8P1710E



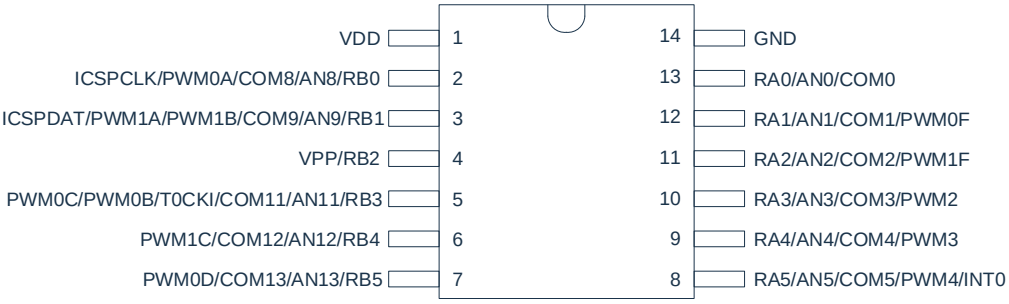
SC8P1710E

1.3.2 SC8P1715E



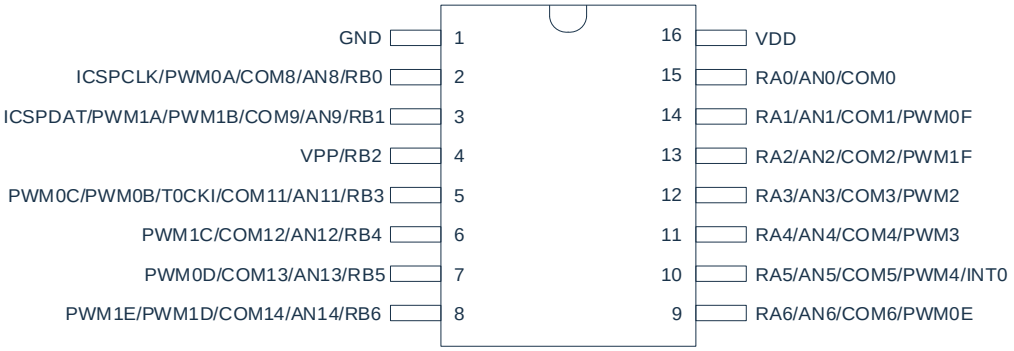
SC8P1715E

1.3.3 SC8P1711E

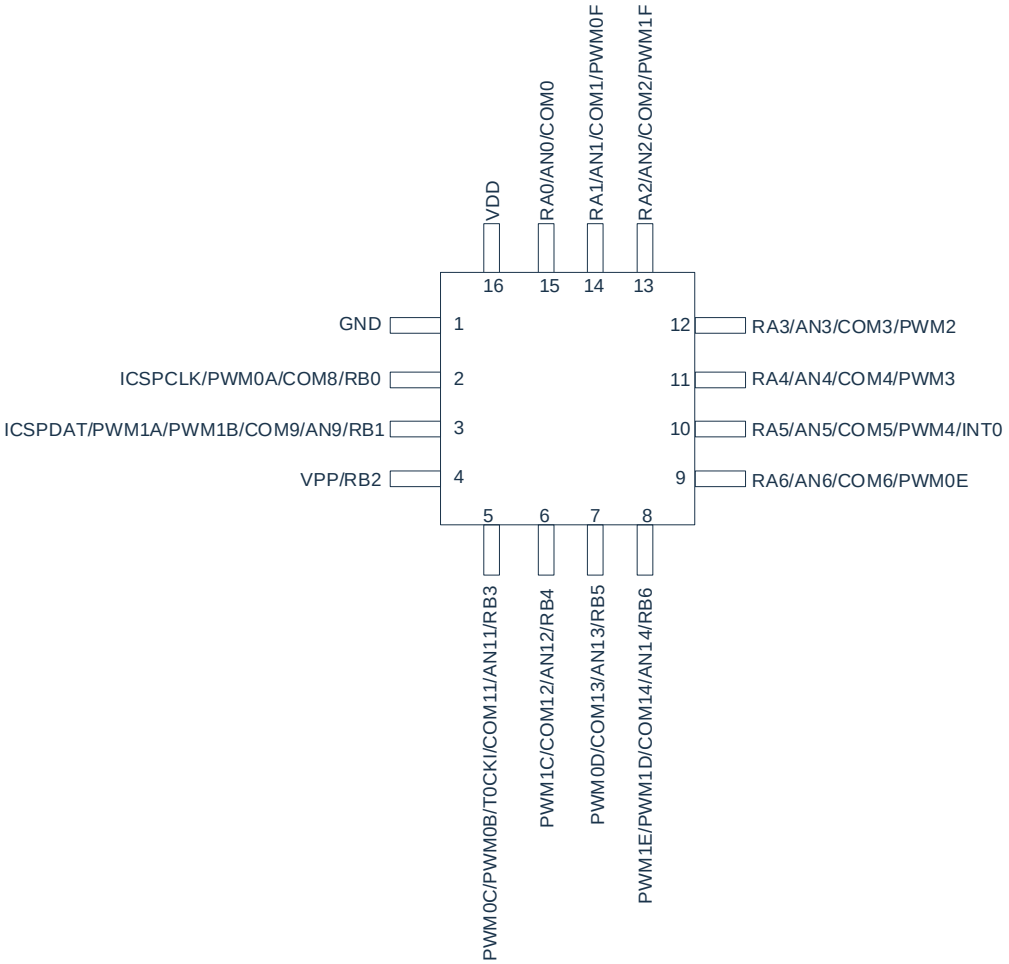


SC8P1711E

1.3.4 SC8P1712E



SC8P1712E-SOP16



SC8P1712E-QFN16

SC8P171xE pin description:

Pin name	IO type	Pin description
VDD,GND	P	Supply voltage input pin, ground pin
RA0-RA6	I/O	Programmable as input, push-pull output with pull-up resistor, pull-down resistor, interrupt on change function
RB0-RB1,RB3-RB6	I/O	Programmable as input, push-pull output with pull-up resistor, pull-down resistor, interrupt on change function
RB2	I/O	Programmable as input, open-drain low output, pull-up resistor function, interrupt on change function
ICSPCLK	I	Programming clock input pin
ICSPDAT	I/O	Programming data input/output pins
AN0-AN6, AN8-AN9, AN11-AN14	I	12-bit ADC input pin
COM0-COM6, COM8-COM9, COM11-COM14	O	LCD 1/2Bias drive common terminal
T0CKI	I	TIMER0 external clock input pin
INT0	I	External interrupt input
PWM0X	O	PWM0
PWM1X	O	PWM1
PWM2-PWM4	O	PWM2-PWM4 output
VPP	I	Programming high voltage input

Notice: PWM0X and PWM1X can only be valid for one group, e.g. PWM0A/PWM1A is valid, other PWM ports are invalid or PWM0B and PWM1B are valid, other PWM ports are invalid; and so on.

1.4 System configuration register

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

1. INTRC_SEL (watchdog selection)
   - INTRC8M F_{HSI} selects internal 8MHz RC oscillation
   - INTRC16M F_{HSI} selects internal 16MHz RC oscillation
2. WDT (watchdog selection)
   - ENABLE Turn on the watchdog timer
   - DISABLE Turn off the watchdog timer
3. PROTECT (encrypted)
   - DISABLE ROM code is not encrypted
   - ENABLE ROM code is encrypted, and the value read out by the programmed emulator will be uncertain after encryption
4. LVR_SEL (low-voltage detection selection)
   - 1.8V
   - 2.0V
   - 2.6V
   - 3.0V
5. SLEEP_LVREN (sleep LVR enable bit)
   - DISABLE LVR function off in sleep state
   - ENABLE LVR function on in sleep state
6. PWM_SEL(PWM1&PWM0)
   - RB1&RB0 PWM1=RB1, PWM0=RB0
   - RB1&RB3 PWM1=RB1, PWM0=RB3
   - RB4&RB3 PWM1=RB4, PWM0=RB3
   - RA2&RA1 PWM1=RA2, PWM0=RA1
   - RB6&RA6 PWM1=RB6, PWM0=RA6
   - RB6&RB5 PWM1=RB6, PWM0=RB5

1.5 Online serial programming

The microcontroller can be programmed serially in the final application circuit. Programming can be done simply with the following 5 wires.

- Power wire
- Ground wire
- Data wire
- Clock wire
- High-voltage wire

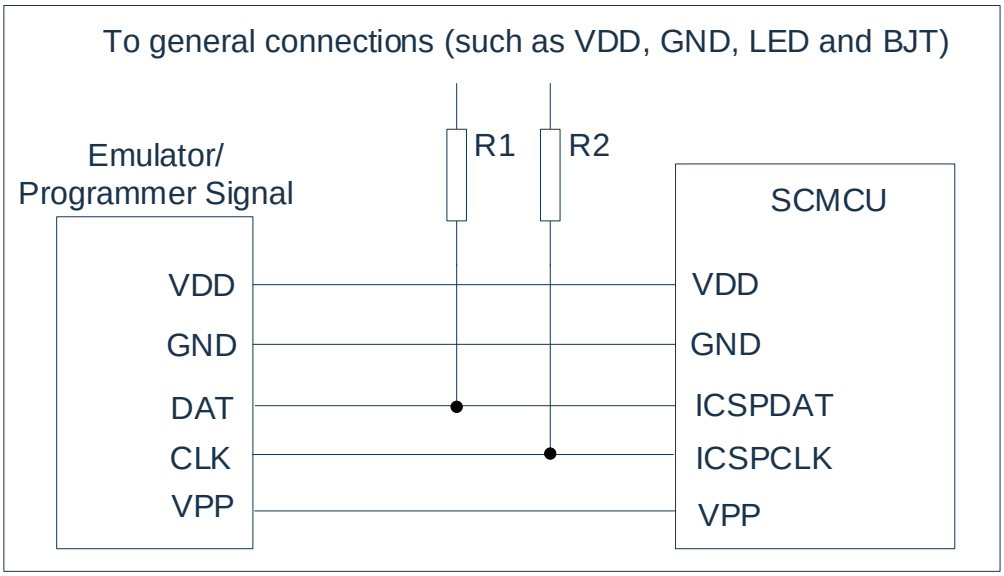


Figure 1-1: Typical connection for online serial programming

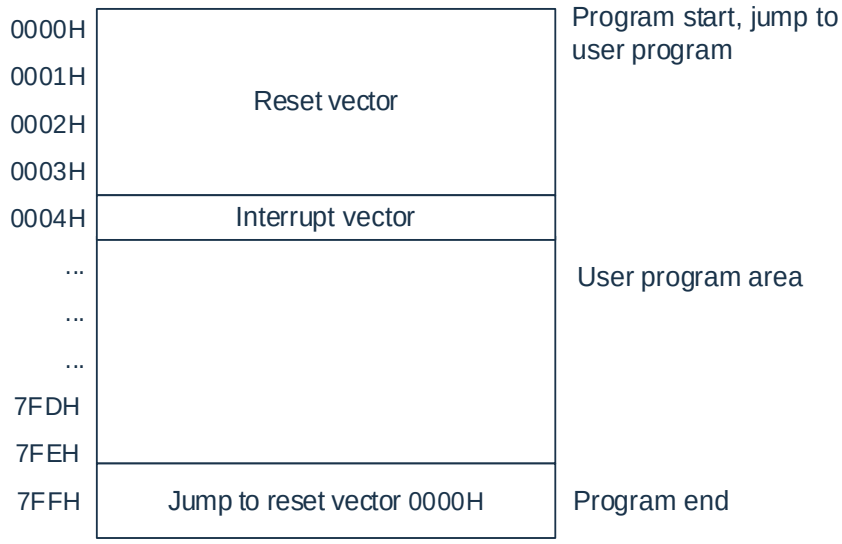
In the above figure, R1 and R2 are the electrical isolation devices, normally represented by resistors with the following resistance values: R1≥4.7K, R2≥4.7K.

2. Central Processing Unit (CPU)

2.1 Memory

2.1.1 Program memory

OTP: 2K



2.1.1.1 Reset vector (0000H)

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

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

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

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 interrupt will enter 0004H. Users will determine which interrupt to execute according to the bit of the interrupt request flag bit register. The following program illustrates how to write interrupt service program.

Example: define 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 users need to protect interrupt scene.

Example: interrupt-in protection

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

Example: interrupt-out restore

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

2.1.1.3 Look-up table

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

Related instructions:

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

Related registers:

- TABLE_SPH Read/write register to indicate higher 3 bits in the table.
- TABLE_SPL Read/write register to indicate lower 8 bits in the table.
- TABLE_DATAH Read only register to save higher bit information in the table

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

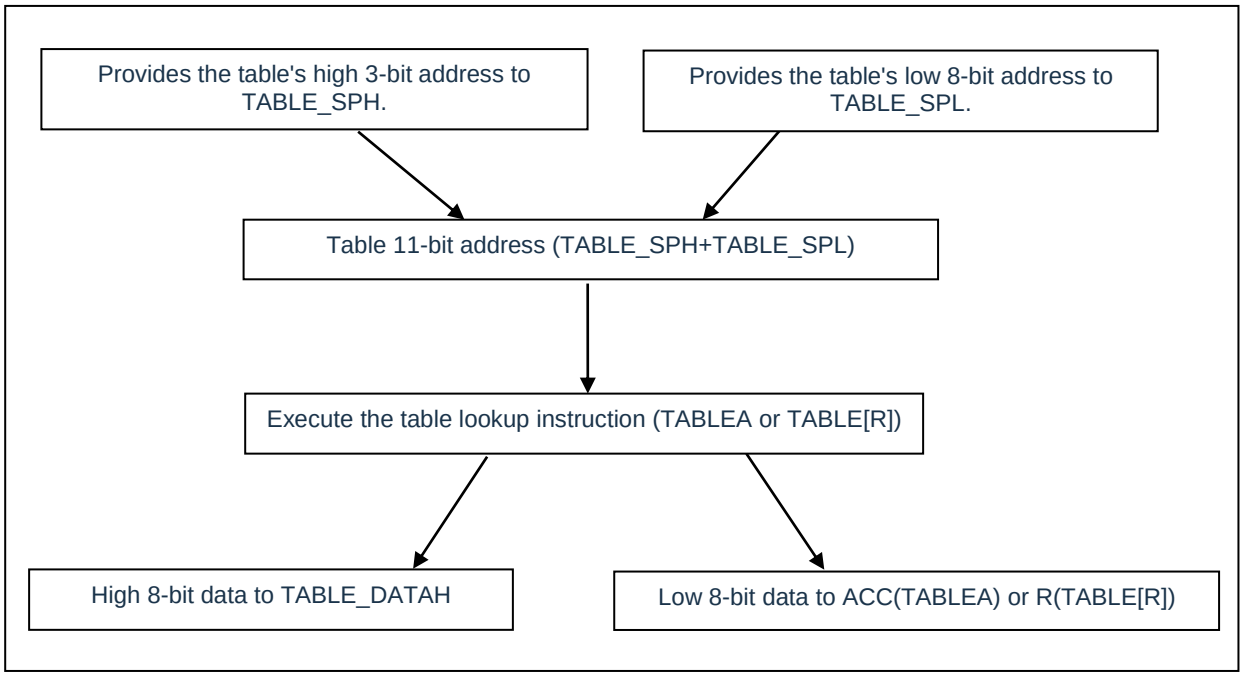


Figure 2-1: Flow chart of table call

The following example explains how to call the table from within the program.

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

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

2.1.1.4 Jump table

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

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

Example: correct illustration of multi-address jump

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

Example: wrong illustration of multi-address jump

```
OTP 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 OTP space when using PCL to achieve multi-address jump.

2.1.2 Data memory

SC8P171xE data memory list

BANK0		BANK1		BANK2		BANK3	
INDF	00H	INDF	80H		100H		180H
TMR0	01H	OPTION_REG	81H		101H		181H
PCL	02H	PCL	82H		102H		182H
STATUS	03H	STATUS	83H		103H		183H
FSR	04H	FSR	84H		104H		184H
PORTA	05H	TRISA	85H		105H		185H
PORTB	06H	TRISB	86H		106H		186H
WPUA	07H	WPDB	87H		107H		187H
WPDA	08H	WDTCON	88H		108H		188H
IOCA	09H	----	89H		109H		189H
PCLATH	0AH	PCLATH	8AH		10AH		18AH
INTCON	0BH	INTCON	8BH		10BH		18BH
PIR1	0CH	PIE1	8CH		10CH		18CH
PIR2	0DH	PIE2	8DH		10DH		18DH
PWMD2L	0EH	PWMCON1	8EH		10EH		18EH
PWMD3L	0FH	OSCCON	8FH		10FH		18FH
PWMD4L	10H	----	90H		110H		190H
TMR2	11H	LVDCON	91H		111H		191H
T2CON	12H	PR2	92H		112H		192H
LCDCON0	13H	ANSEL	93H		113H		193H
LCDCON1	14H	ANSELH	94H		114H		194H
LCDCON	15H	WPUB	95H		115H		195H
PWMD0L	16H	IOCB	96H		116H		196H
PWMD1L	17H	TABLE_SPH	97H		117H		197H
PWMD01H	18H	TABLE_SPL	98H		118H		198H
PWMTL	19H	TABLE_DATAH	99H		119H		199H
PWMTH	1AH	----	9AH		11AH		19AH
PWMCON	1BH	----	9BH		11BH		19BH
PWMD23H	1CH	----	9CH		11CH		19CH
PWM4TL	1DH	----	9DH		11DH		19DH
ADRESH	1EH	ADRSEL	9EH		11EH		19EH
ADCON0	1FH	ADCON1	9FH		11FH		19FH
Universal register 96-byte	20H 6FH	Universal register 80-byte	A0H EFH		120H 16FH		1A0H 1EFH
	70H -- 7FH	Fast memory space 70H-7FH	F0H -- FFH		170H -- 17FH		1F0H -- 1FFH

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

SC8P171xE special function register summary Bank0

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
00H	INDF	Addressing this unit will address data memory (not a physical register) using the contents of the FSR.								xxxxxxxx
01H	TMR0	TIMER0 data register								xxxxxxxx
02H	PCL	Program counter low byte								00000000
03H	STATUS	----	----	RP0	TO	PD	Z	DC	C	--011xxx
04H	FSR	Indirect data memory address pointer								xxxxxxxx
05H	PORTA	----	RA6	RA5	RA4	RA3	RA2	RA1	RA0	-xxxxxxx
06H	PORTB	----	RB6	RB5	RB4	RB3	RB2	RB1	RB0	-xxxxxxx
07H	WPUA	----	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0	-0000000
08H	WPDA	----	WPDA6	WPDA5	WPDA4	WPDA3	WPDA2	WPDA1	WPDA0	-0000000
09H	IOCA	----	IOCA6	IOCA5	IOCA4	IOCA3	IOCA2	IOCA1	IOCA0	-0000000
0AH	PCLATH	----	---	----	----	----	Write buffer for the high 3 bits of the program counter			-----000
0BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
0CH	PIR1	----	----	----	----	RAIF	PWMIF	TMR2IF	ADIF	----0000
0DH	PIR2	----	----	----	----	----	----	----	LVDIF	-------0
0EH	PWMD2L	PWM2 duty cycle low register								00000000
0FH	PWMD3L	PWM3 duty cycle low register								00000000
10H	PWMD4L	PWM4 duty cycle low register								00000000
11H	TMR2	TIMER2 module register								00000000
12H	T2CON	----	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-0000000
13H	LCDCON0	----	COM6EN	COM5EN	COM4EN	COM3EN	COM2EN	COM1EN	COM0EN	-0000000
14H	LCDCON1	----	COM14EN	COM13EN	COM12EN	COM11EN	----	COM9EN	COM8EN	-0000000
15H	LCDCON	LCDEN	----	----	----	----	----	LCD_ISLE[1:0]		0-----00
16H	PWMD0L	PWM0 duty cycle low register								00000000
17H	PWMD1L	PWM1 duty cycle low register								00000000
18H	PWMD01H	----	----	----	----	PWMD1[9:8]		PWMD0[9:8]		----0000
19H	PWMTL	PWM period low register								00000000
1AH	PWMTH	----	----	----	----	----	----	PWM period high register		------00
1BH	PWMCON	CLKDIV[2:0]			----	PWM1DIR	PWM0DIR	PWM1EN	PWM0EN	00-00000
1CH	PWMD23H	----	----	----	----	PWMD3[9:8]		PWMD2[9:8]		----0000
1DH	PWM4TL	PWM4 period low register								00000000
1EH	ADRESH	A/D result register high byte								xxxxxxxx
1FH	ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/ $\overline{\text{DONE}}$	ADON	00000000

SC8P171xE special function register summary Bank1

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
80H	INDF	Addressing this unit will address data memory (not a physical register) using the contents of the FSR.								xxxxxxxx
81H	OPTION_REG	----	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	-1111011
82H	PCL	Program counter (PC) low byte								00000000
83H	STATUS	----	----	RP0	TO	PD	Z	DC	C	--011xxx
84H	FSR	Indirect data memory address pointer								xxxxxxxx
85H	TRISA	----	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	-1111111
86H	TRISB	----	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	-1111111
87H	WPDB	----	WPDB6	WPDB5	WPDB4	WPDB3	----	WPDB1	WPDB0	-0000-00
88H	WDTCON	----	----	----	----	----	----	----	SWDTEN	-------0
8AH	PCLATH	----	----	----	----	----	Write buffer for the high 3 bits of the program counter			-----000
8BH	INTCON	GIE	PEIE	T01E	INTE	RBIE	T0IF	INTF	RBIF	00000000
8CH	PIE1	----	----	----	----	RAIE	PWMIE	TMR2IE	ADIE	----0000
8DH	PIE2	----	----	----	----	----	----	----	LVDIE	-------0
8EH	PWMCON1	----	----	PMW4DIR	PWM3DIR	PWM2DIR	PWM4EN	PWM3EN	PWM2EN	--000000
8FH	OSCCON	----	IRCF2	IRCF1	IRCF0	----	----	----	----	-101----
91H	LVDCON	LVD_RES	---	---	---	LVD_SEL[2:0]			LVDEN	x---0000
92H	PR2	TIMER2 period register								11111111
93H	ANSEL	----	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0	-0000000
94H	ANSELH	----	ANS14	ANS13	ANS12	ANS11	----	ANS9	ANS8	-0000-00
95H	WPUB	----	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0	-0000000
96H	IOCB	----	IOCB6	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	IOCB0	-0000000
97H	TABLE_SPH	----	----	----	----	----	Table High 3 bits pointer			-----xxx
98H	TABLE_SPL	Table low pointer								xxxxxxxx
99H	TABLE_DATAH	Table high data								xxxxxxxx
9EH	ADRESL	A/D result register low byte								xxxxxxxx
9FH	ADCON1	ADFM	----	----	----	TADSEL	LDOEN	----	LDOSEL	0---00-0

2.2 Addressing mode

2.2.1 Direct addressing

It operates the RAM through the operation register (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 the immediate value 12H to ACC

```
LDIA    12H
```

2.2.3 Indirect addressing

Data memory can be addressed directly or indirectly. Direct addressing can be achieved through INDF register, and INDF is not a physical register. When INDF is accessed, it is addressed according to the value in the FSR register (lower 8 bits) and the IRP bit (bit 9) of the STATUS register, and points to the register at that address. Therefore, after setting the FSR register and the IRP bit of STATUS register, INDF register can be regarded as a target register. Read INDF (FSR=0) indirectly will produce 00H. Write to the INDF register indirectly will cause an empty operation. The following example shows how indirect addressing works.

Example: application of FSR and INDF

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

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

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

2.3 Stack

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

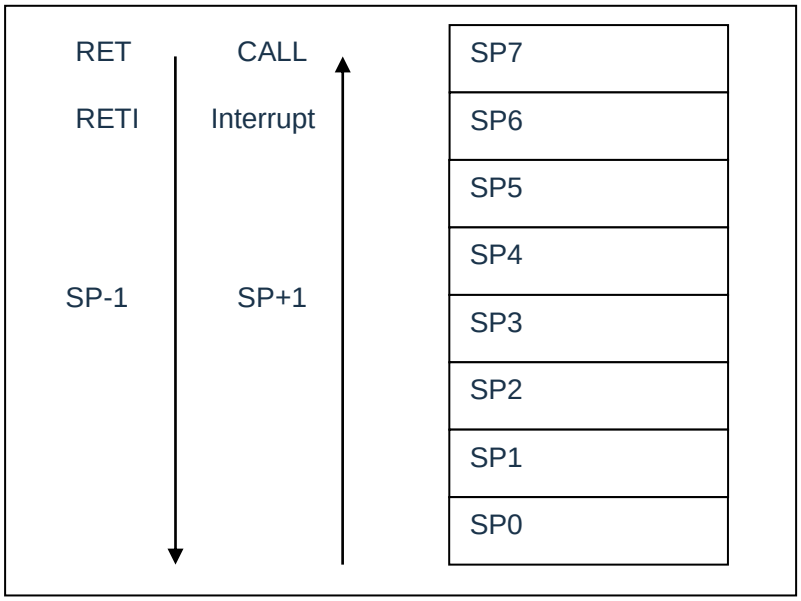


Figure 2-2: Working principle of stack buffer

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

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

2.4 Accumulator (ACC)

2.4.1 Overivew

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

ACC register is an 8-bit register to store the ALU calculation result. 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 provided instructions.

2.4.2 ACC application

Example: use ACC for data transferring

```
        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 a double operand instruction

```
        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 a double operand instruction

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

2.5 Program status register (STATUS)

STATUS register includes:

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

Just like other registers, STATUS register can be the target register of any 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 will not change). It is recommended to only use CLRB, SETB, SWAPA and SWAPR instructions to change STATUS register because these will not affect any status bits.

Program status register STATUS (03H)

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

Bit7~Bit6 Unused

Bit5 RP[1:0]: Memory select bit
00: Select Bank0
01: Select Bank1

Bit4 TO: Time out bit
1= Power on or CLRWDT instruction or STOP instruction
0= WDT time out

Bit3 PD: Power down
1= Power on or CLRWDT instruction
0= Execute STOP instruction

Bit2 Z: Result bit
1= The result of an arithmetic or logical operation is zero
0= The result of an arithmetic or logical operation is not zero

Bit1 DC: Half carry bit/borrow bit
1= Carry happens to higher bits from the lower 4 bits of the result
0= No carry happens to higher bits from the lower 4 bits of the result

Bit0 C: Carry/borrow bit
1= Carry happens at the highest bit
0= No carry happens at the highest bit

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

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

Table of events affecting PD, TO

TO	PD	Reset reason
0	0	Sleep state WDT overflow
0	1	Non-sleep state WDT overflow
1	1	Power on
---	---	----

Status of TO/PD after reset

2.6 Pre-scaler (OPTION_REG)

OPTION_REG register is a read/write register that contains various control bits for configuration.

- TIMER0/WDT pre-scaler.
- TIMER0.

Prescaler control register OPTION_REG (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 Unused

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

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

Bit4 T0SE: Transition edge on the T0CKI pin
1= Increase when T0CKI pin signal transited from low to high
0= Increase when T0CKI pin signal transited from high to low

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

Bit2~Bit0 PS2~PS0: Pre-allocation parameter configure 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

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

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

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

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

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

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

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

Note: To enable TIMER0 to acquire a 1:1 prescaler ratio configuration, the prescaler can be assigned to the WDT by setting the PSA bit of the option register to 1.

2.7 Program counter (PC)

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

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

Program counter (PC) is 11-bit width, users can access lower 8 bits by PCL (02H). The higher 3 bits cannot be accessed. It can hold address for 1.5K×16Bit program. Assigning a value to PCL results in a short jump to the 256 addresses of the current page.

Note: When the programmer uses PCL to make short jumps, the PC high buffer register PCLATH should be assigned first.

The PC values for several special cases are given below

reset	PC=0000;
interrupt	PC=0004 (original PC+1 will be add to stack automatically);
CALL	PC= program specified address (original PC+1 will be add to stack automatically);
RET, RETI, RETI	PC=value from stack;
PCL operation	PC[10:8] unchanged, PC[7:0]=user defined value;
JP	PC= program specified value;
Other instructions	PC=PC+1;

2.8 Watchdog timer (WDT)

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

2.8.1 WDT period

WDT and TIMER0 share the same 8-bit prescaler. 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 overflow period of the WDT will be affected by the ambient temperature, the supply voltage and other parameters.

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

Note:

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

2.8.2 Watchdog timer control register WDTCON

Watchdog timer control register WDTCON(88H)

88_H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WDTCON	---	---	---	---	---	---	---	SWDTEN
R/W	---	---	---	---	---	---	---	R/W
Reset value	---	---	---	---	---	---	---	0

Bit7~Bit1 Unused, read 0

Bit0 SWDTEN: Software enable or disable watchdog timer bit

1= Enable WDT

0= Disable WDT (reset value)

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

3. System Clock

3.1 Overview

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

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

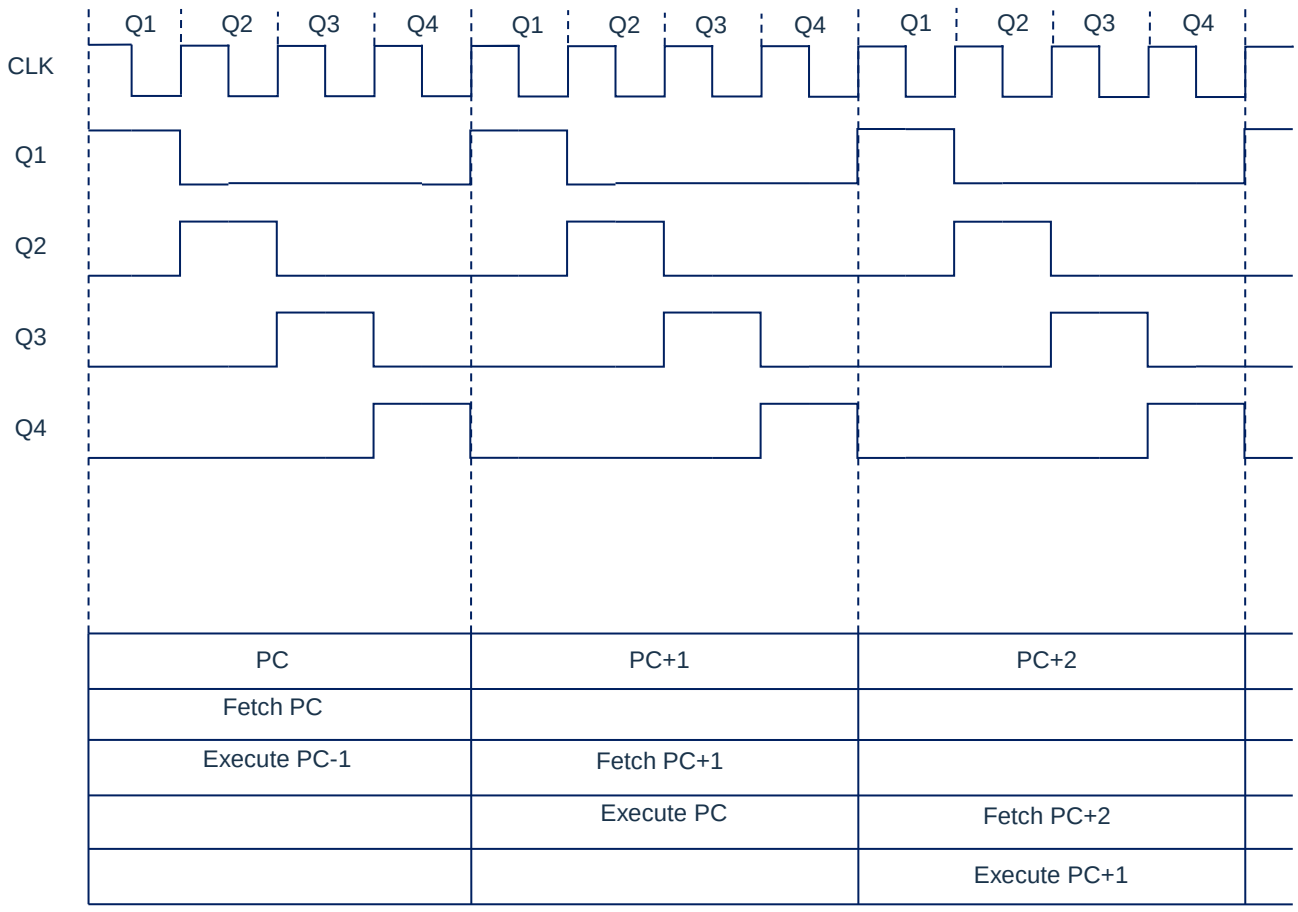


Figure 3-1: Clock and instruction cycle timing chart

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

System frequency	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 one type of oscillation: internal RC oscillation.

3.2.1 Internal RC oscillation

The default oscillation mode of the chip is internal RC oscillation, and the oscillation frequency can be configured to 8MHz or 16MHz through CONFIG. On this basis, the operating frequency of the chip can be set through the OSCCON register.

3.3 Reset time

Reset Time is the time from the chip reset to the chip oscillation stabilization, its design value is about 18ms.

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

3.4 Oscillator control register

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

Oscillator control register OSCCON(8FH)

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

Bit7 Unused, read 0

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

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

110= $F_{SYS} = F_{HSI}$ /2

101= $F_{SYS} = F_{HSI}$ /4 (default)

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

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

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

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

000= F_{SYS} = 32kHz (LSI)

Bit3~Bit0 Unused

3.5 Clock Block Diagram

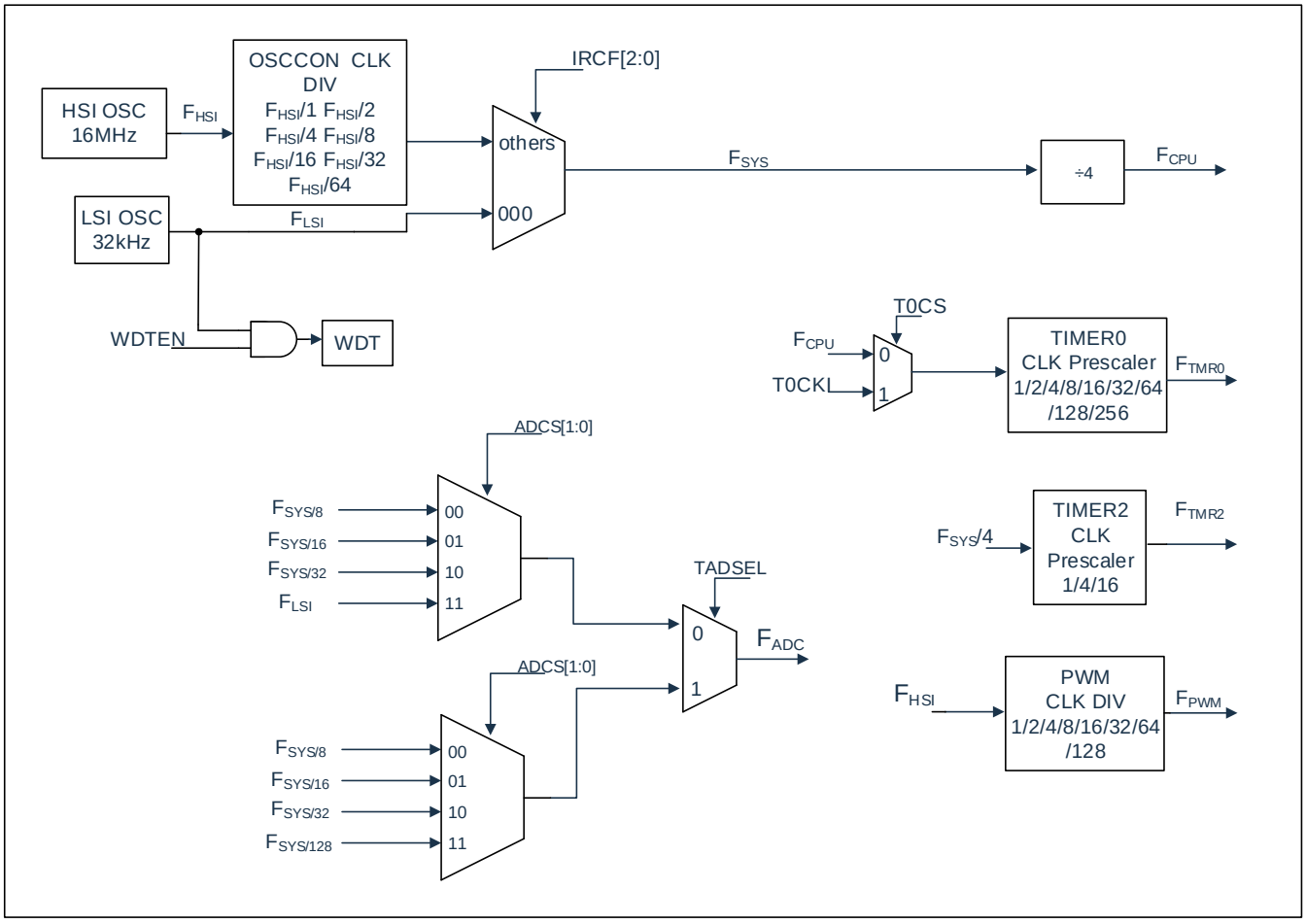


Figure 3-3: Clock Block Diagram

4. Reset

The chip can be reset in the following three ways:

- Power on reset
- Low voltage reset
- Watchdog overflow reset during normal operation

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

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

4.1 Power on reset

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

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

4.2 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.

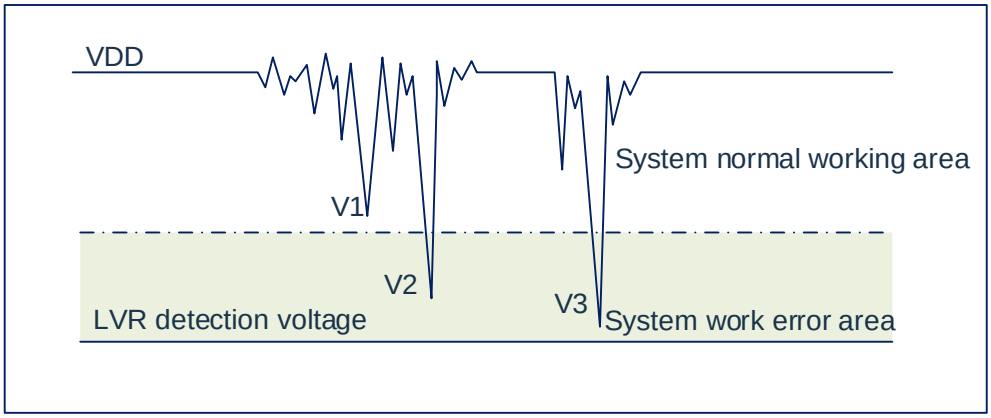


Figure 4-1: Power off reset

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 Improvements for 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, error may happen. It is then suggested to insert a resistor between power source and ground to ensure the MCU pass the dead zone and enter the reset zone faster.

4.3 Watchdog reset

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

The timing of the watchdog reset is as follows:

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

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

5. Sleep Mode

5.1 Enter sleep mode

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

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

In sleep mode, to avoid current consumption, all I/O pins should keep at VDD or GND to make sure no external circuit is consuming the current from I/O pins. To avoid input pin, float 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 woken from sleep by any of the following events.

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

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

5.4 Sleep mode application

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

Example: procedures for entering sleep mode

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

5.5 Sleep mode awaken time

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

System main clock source	System clock frequency (IRCF<2:0>)	Sleep wakeup wait time T_{WAIT}
Internal high-speed RC oscillation (F_{HSI})	$F_{SYS}= F_{HSI}$	$T_{WAIT}=128*1/ F_{HSI}$
	$F_{SYS}= F_{HSI} /2$	$T_{WAIT}=128*2/ F_{HSI}$
	…	…
	$F_{SYS}= F_{HSI} /64$	$T_{WAIT}=128*64/ F_{HSI}$
Internal low-speed RC oscillation (F_{LSI})	---	$T_{WAIT}=4/F_{LSI}$

6. I/O Ports

The chip has four I/O ports: PORTA, PORTB (up to 14 I/Os). These ports can be accessed directly from the read/write port data registers.

Port	Bit	Pin description	I/O
PORTA	0	Schmitt trigger input, push-pull output, AN0, LCD drive port	I/O
	1	Schmitt trigger input, push-pull output, AN1, LCD drive port	I/O
	2	Schmitt trigger input, push-pull output, AN2, LCD drive port	I/O
	3	Schmitt trigger input, push-pull output, AN3, LCD drive port	I/O
	4	Schmitt trigger input, push-pull output, AN4, LCD drive port	I/O
	5	Schmitt trigger input, push-pull output, AN5, LCD drive port	I/O
	6	Schmitt trigger input, push-pull output, AN6, LCD drive port	I/O
PORTB	0	Schmitt trigger input, push-pull output, AN8, LCD drive port, programming clock input	I/O
	1	Schmitt trigger input, push-pull output, AN9, LCD drive port, programming data input/output	I/O
	2	Schmitt trigger input, open drain low output, high voltage input port	I/O
	3	Schmitt trigger input, push-pull output, TMR0 clock input, AN11, LCD drive port	I/O
	4	Schmitt trigger input, push-pull output, AN12, LCD drive port	I/O
	5	Schmitt trigger input, push-pull output, AN13, LCD drive port	I/O
	6	Schmitt trigger input, push-pull output, AN14, LCD drive port	I/O

＜Table 6-1: Port configuration overview＞

6.1 I/O port structure

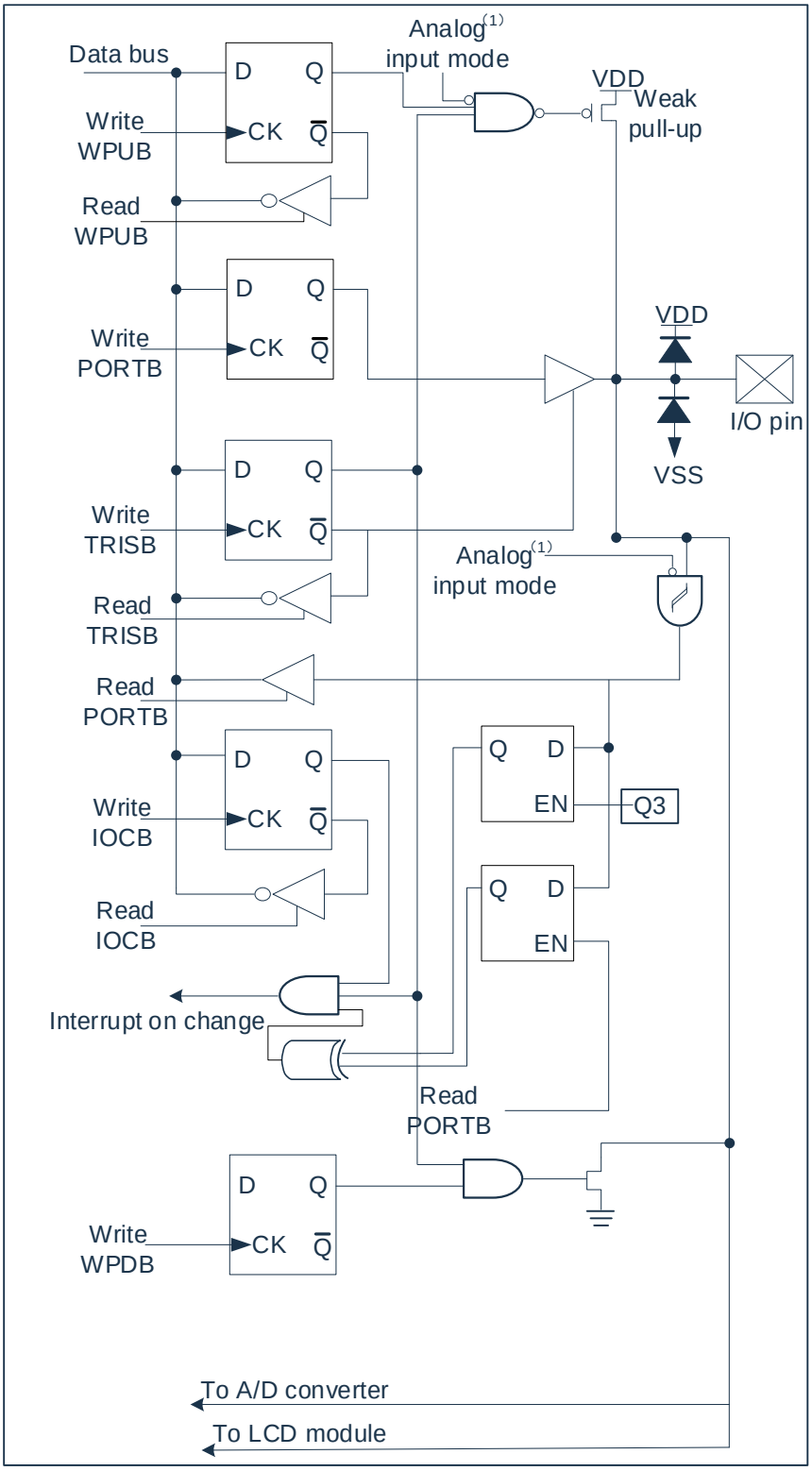


Figure 6-1: I/O port stucture

Note: ANSEL and ANSELH determine the analog input mode.

6.2 PORTA

6.2.1 PORTA data and direction control

PORTA is a 7-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 related to the PORTA port are PORTA, TRISA, WPUA, WPDA, ANSEL, IOCA and so on.

Note: The ANSEL register must be initialized to configure the analog channel as a digital input. Pins configured as analog inputs will read 0.

PORTA data register PORTA(05H)

05H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTA	---	RA6	RA5	RA4	RA3	RA2	RA1	RA0
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

Bit7	Unused	
Bit6~Bit0	PORTA<6:0>:	PORTAI/O pin bit
	1=	Port pin level >V_{IH}
	0=	Port pin level <V_{IL}

PORTA direction register TRISA(85H)

85H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISA	---	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
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

Bit7	Unused	
Bit6~Bit0	TRISA<6:0>:	PORTA tristate control bit
	1=	PORTA pin set to be input (tristate)
	0=	PORTA pin set to be output

Example: procedure for PORTA

```
LDIA    B'01110000'     ;set PORTA<3:0> as output port, PORTA<6:4>as input port
LD      TRISA,A
LDIA    03H             ;PORTA<1:0>output high level, PORTA<3:2>output low level
LD      PORTA,A         ;since PORTA<6:4> 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 an 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 ANS bit has no effect on the digital output function. The pin with TRIS cleared and ANS set to 1 will still be used as a digital output, but the input mode will become an analog mode. This can cause unpredictable results when performing read-modify-write operations on the affected port.

PORTA analog selection register ANSEL(93H)

93H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL	---	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0
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 Unused

Bit6~Bit0 ANS<6:0>: Analog selection bit, select the digital or analog function of pin AN<6:0>

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 resistor

Each PORTA pin has an individually configurable internal weak pull-up. Control bits WPUA<6:0> enable or disable each weak pull-up. When a port pin is configured as an output, its weak pull-up is automatically cut off.

PORTA pull-up resistor register WPUA(07H)

07H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUA	---	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
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 Unused

Bit6~Bit0 WPUA<6:0>: Weak pull up register bit

1= Enable pull up

0= Disable pull up

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

6.2.4 PORTA pull-down resistor

Each PORTA pin has an internal weak pull-down that can be individually configured. The control bits WPDA<6:0> enable or disable each weak pull down. When a port pin is configured as an output, its weak pull-down is automatically cut off.

PORTA pull-down resistor register WPDA(08H)

08H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPDA	---	WPDA6	WPDA5	WPDA4	WPDA3	WPDA2	WPDA1	WPDA0
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 Unused

Bit6~Bit0 WPDA<6:0>: Weak pull-down register bit

1= Enable pull down

0= Disable pull down

Note: If pin is configured as output, 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<6:0> enables or disables the interrupt function of each pin. Disable pin level change interrupt function when power on reset.

For the pin that has allowed level change interrupt, compare the value on the pin with the old value latched when PORTA was read last time. Perform a logical OR operation with the output "mismatch" of the last read operation to set the PORTA level change interrupt flag (RAIF) of the PIR1 register as 1.

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

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

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

PORTA interrupt-on-change register IOCA(09H)

09H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOCA	---	IOCA6	IOCA5	IOCA4	IOCA3	IOCA2	IOCA1	IOCA0
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	Unused	
Bit6~Bit0	IOCA<6:0>	PORTA interrupt-on-change control bit
	1=	Enable interrupt-on-change
	0=	Disable interrupt-on-change

6.3 PORTB

6.3.1 PORTB data and direction

PORTB is a 6-bit wide bi-directional port. The corresponding data direction register is TRISB. Set a bit in TRISB to 1 (=1) to make the corresponding PORTB pin as the input pin. Clearing a bit in TRISB (=0) will make the corresponding PORTB pin as the output pin.

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

The registers related to the PORTB port are PORTB, TRISB, WPUB, WPDB, IOCB, ANSELH, and so on.

PORTB data register PORTB(06H)

06H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTB	---	RB6	RB5	RB4	RB3	RB2	RB1	RB0
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

Bit7	Unused	
Bit6~Bit0	PORTB<6:0>:	PORTBI/O pin bit
	1=	Port pin level >V_{IH}
	0=	Port pin level <V_{IL}

PORTB direction register TRISB (86H)

86H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISB	---	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
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

Bit7	Unused	
Bit6~Bit0	TRISB<6:0>:	PORTB tristate control bit
	1=	PORTB pin configured as input(tristate)
	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 an 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(94H)

94H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSELH	---	ANS14	ANS13	ANS12	ANS11	---	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	Unused	
Bit6~Bit3	ANS<14:11>:	Analog selection bits, select the analog or digital functions of pin AN<14:11>
	1=	Analog input. The pin is selected as analog input
	0=	Digital I/O. The pin is selected as port or special function
Bit2	Unused	
Bit1~Bit0	ANS<9:8>:	Analog selection bits, select the analog or digital functions of pin AN<9:8>
	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-up resistor

Each PORTB pin has an internal weak pull up that can be individually configured. The control bits WPUB<6:0> enable or disable each weak pull up. When a port pin is configured as an output, its weak pull-up is automatically cut off.

PORTB pull-up resistor register WPUB(95H)

95H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUB	---	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0
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	Unused	
Bit6~Bit0	WPUB<6:0>:	Weak pull-up register bit
	1=	Enable pull-up
	0=	Disable pull-up

Note:
1. If pin is configured as output, weak pull-up will be automatically disabled.
2. The internal pull-up resistor at pin RB2 is forced on during power-on reset.

6.3.4 PORTB pull-down resistor

Each PORTB pin has an internal weak pull down that can be individually configured. The control bits WPDB<6:0> enable or disable each weak pull down. When the port pin is configured as an output, its weak pull-down is automatically cut off.

PORTB pull-down resistor register WPDB(87H)

87H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPDB	---	WPDB6	WPDB5	WPDB4	WPDB3	---	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	Unused	
Bit6~Bit3	WPDB<6:3>:	Weak pull-down register bit
	1=	Enable pull down
	0=	Disable pull down
Bit2	Unused	
Bit1~Bit0	WPDB<1:0>:	Weak pull-down register bit
	1=	Enable pull down
	0=	Disable pull down

6.3.5 PORTB interrupt on change

All PORTB pins can be individually configured as interrupt on change pins. The control bit IOCB<6:0> allows or disables the interrupt function of each pin. Disable pin level change interrupt function when power on reset.

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

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

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

-Clear the flag bit RBIF.

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

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

PORTB interrupt on change register IOCB(96H)

96H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOCB	---	IOCB6	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	IOCB0
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	Unused	
Bit6~Bit0	IOCB<6:0>	PORTB interrupt on change control bit
	1=	Enabele interrupt on change
	0=	Disable interrupt on change

6.4 I/O usage

6.4.1 Write to 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 to I/O port program

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

6.4.2 Read from I/O port

Example: read from I/O port program

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

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

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

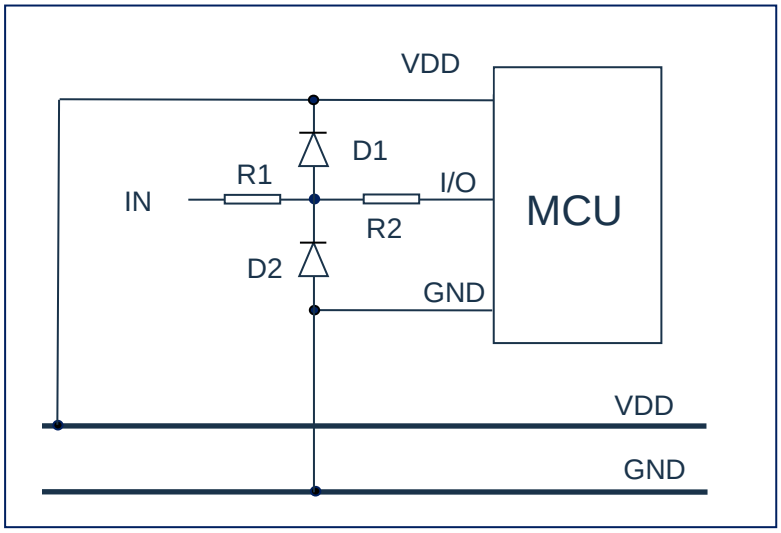


Figure 6-3: I/O port caution connection diagram

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

7. Interrupt

7.1 Overview

The chip has the following interrupt sources:

- TIMER0 overflow interrupt;
- TIMER2 match interrupt;
- PORTA interrupt on change;
- PORTB interrupt on change;
- PWM interrupt;
- INT interrupt;
- A/D interrupt;
- LVD interrupt;

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

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

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

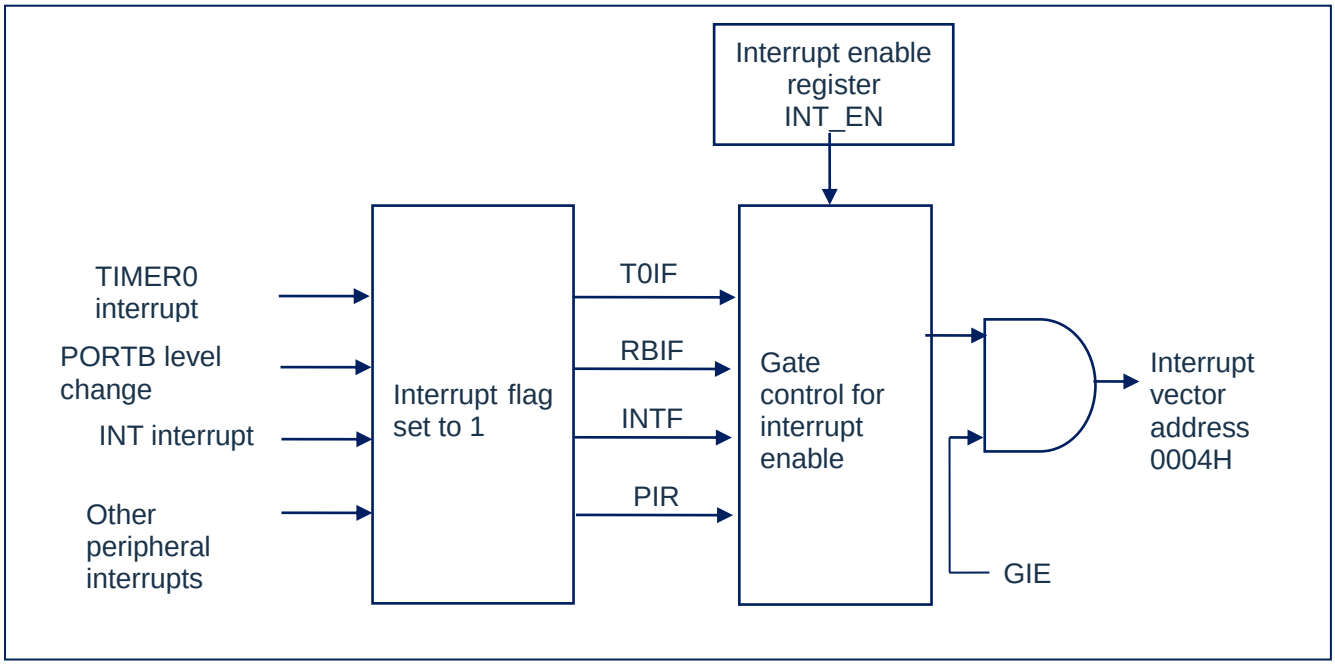


Figure 7-1: Schematic diagram of interrupt principle

7.2 Interrupt control register

7.2.1 Interrupt control register

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

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

Interrupt control register INTCON (0BH)

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

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

Bit6 PEIE: Peripheral interrupt enable bit
- 1= Enable all unshielded peripherals interrupt
- 0= Disable all peripherals interrupt

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

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

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

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

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

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

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

7.2.2 Peripheral interrupt enable register

The peripheral interrupt enable registers have PIE1 and PIE2, and before enabling any peripheral interrupts, you must first set the PEIE bit of the INTCON register to 1.

Peripheral interrupt enable register PIE1(8CH)

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

Bit7~Bit4 Unused

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

Bit2 PWMIE: PWM interrupt enable bit
1= Enable PWM interrupt
0= Disable PWM interrupt

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

Bit0 ADIE: A/D converter (ADC) interrupt enable bit
1= Enable ADC interrupt
0= Disable ADC interrupt

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 Unused

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 registers are PIR1 and PIR2. When an interrupt condition is generated, the interrupt flag bit will be set to 1 regardless of the status of the corresponding interrupt enable bit or the global enable bit GIE. The user software should ensure that the corresponding interrupt flag bit is cleared to 0 before enabling an interrupt.

Peripheral interrupt request register PIR1(0CH)

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

Bit7~Bit4 Unused

Bit3 RAIF: PORTA interrupt on change flag bit
- 1= The level state of at least one pin in the PORTA port has changed (must be cleared by software)
- 0= None of the PORTA general-purpose I/O pins have changed state

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

Bit1 TMR2IF: TIMER2 and PR2 match interrupt flag bit
- 1= TIMER2 matches PR2 (must be cleared by software)
- 0= TIMER2 does not match PR2

Bit0 ADIF: A/D converter interrupt flag bit
- 1= A/D conversion is complete (cleared by software)
- 0= A/D conversion is not complete or not start

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 Unused

Bit0 LVDIF: LVD interrupt flag bit
- 1= The supply voltage is below the voltage point set by the LVD (cleared by software)
- 0= The supply voltage is higher than the voltage point set by the LVD

7.3 Protection methods for interrupt

After an interrupt request occurs and is responded, the program goes to 0004H to execute the interrupt 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" instructions are executed, the next interrupt can be responded to.

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

8. TIMER0

8.1 TIMER0 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.

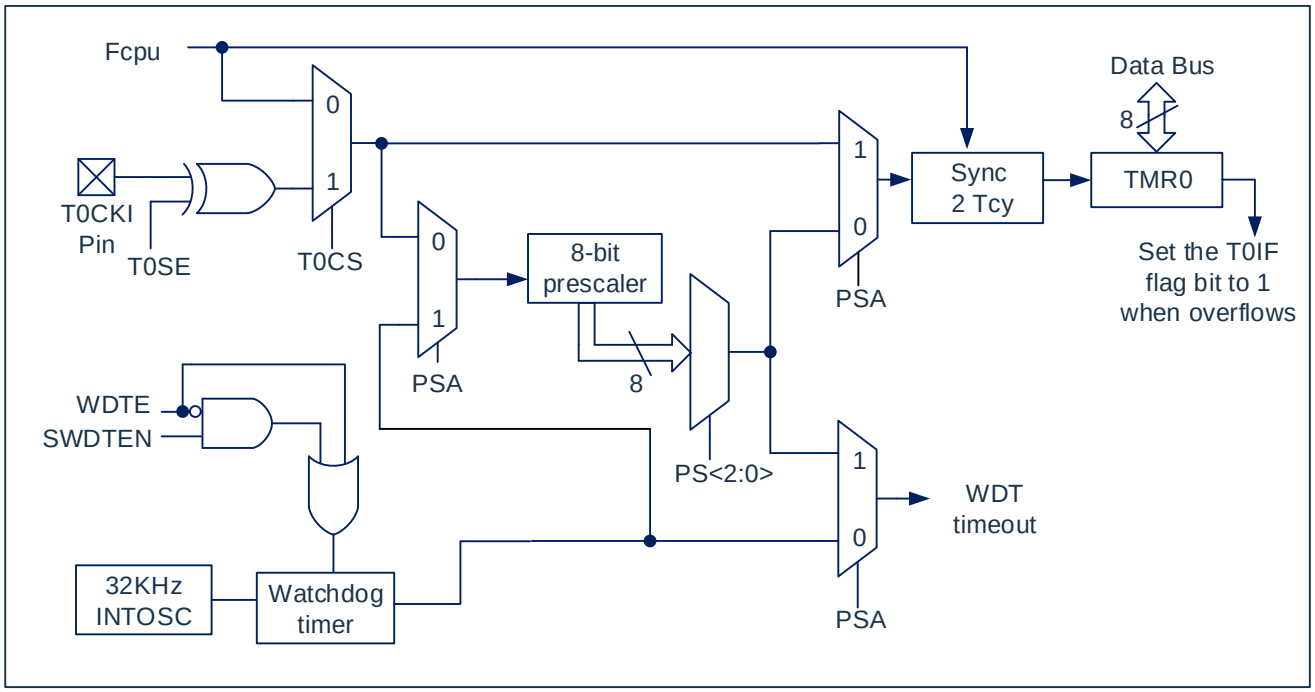


Figure 8-1: TIMER0/WDT mod structure

Note:

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

8.2 Working principle of 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 to TMR0.

8.2.2 8-bit counter mode

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

8.2.3 Software programmable pre-scaler

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

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

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

8.2.4 Switch prescaler between TIMER0 and WDT module

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

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

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

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

8.2.5 TIMER0 interrupt

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

Note: The TIMER0 interrupt cannot wake up the processor because the timer is off in the sleep state.

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 change the working mode of TIMER0, etc. Please refer to Section 2.6 Prescaler Register (OPTION_REG) Application.

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	---	---	---	---	---	---	---	---

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 Unused

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_{CPU})

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 PAS: Pre-scaler allocation bit
- 1= Pre-scaler allocated to WDT
- 0= Pre-scaler allocated toTIMER0 mod

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

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

9. TIMER2

9.1 TIMER2 overview

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

1. 8-bit timer register (TMR2)
2. 8-bit period register (PR2)
3. Interrupt when TMR2 matches PR2
4. Software programmable prescaler ratio (1:1, 1:4 and 1:16)
5. Software programmable postscaler ratio (1:1 to 1:16)

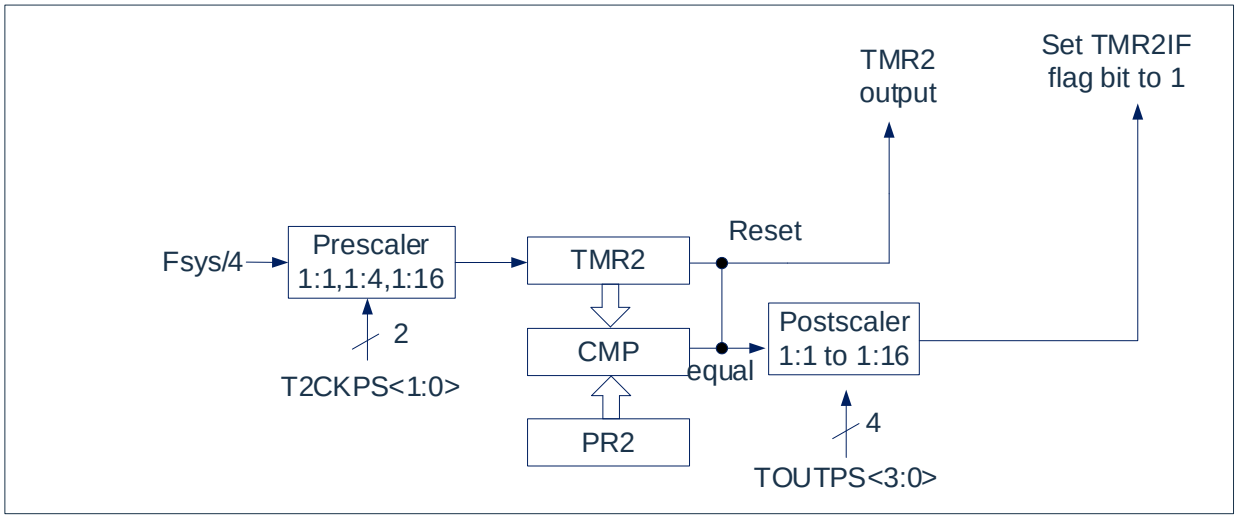


Figure 9-1: Block diagram of TIMER2

9.2 Working principle of TIMER2

The input clock to the TIMER2 module is the system instruction clock (Fsys/4). The clock is input to the TIMER2 prescaler, which is available in the following ratios: 1:1, 1:4 or 1:16. The output of the prescaler is then used to increment the TMR2 register.

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

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

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

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

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

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

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

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

Note: Writing T2CON does not reset TMR2 to zero, and when TMR2ON=0, the TMR2 register cannot be written.

9.3 TIMER2 related registers

There are three registers related to TIMER2, namely the data register TMR2, the period register PR2, and the 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	0	0	0	0	0	0	0	0

TIMER2 period register PR2（92H）

92H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PR2								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	1	1	1	1	1	1	1	1

TIMER2 control register T2CON(12H)

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

Bit7 Unused, read 0

Bit6~Bit3 TOUTPS<3:0>: TIMER2 output postscaler ratio select bit

0000= 1:1
0001= 1:2
0010= 1:3
0011= 1:4
0100= 1:5
0101= 1:6
0110= 1:7
0111= 1:8
1000= 1:9
1001= 1:10
1010= 1:11
1011= 1:12
1100= 1:13
1101= 1:14
1110= 1:15
1111= 1:16

Bit2 TMR2ON: TIMER2 enable bit

1= Enable TIMER2
0= Disable TIMER2

Bit1~Bit0 T2CKPS<1:0>: TIMER2 clock prescaler ratio select bit

00= 1
01= 4
1x= 16

10. Analog to Digital Conversion (ADC)

10.1 ADC overview

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

The ADC reference voltage is always generated internally, and the ADC can generate an interrupt after the conversion is complete, as shown in the block diagram below.

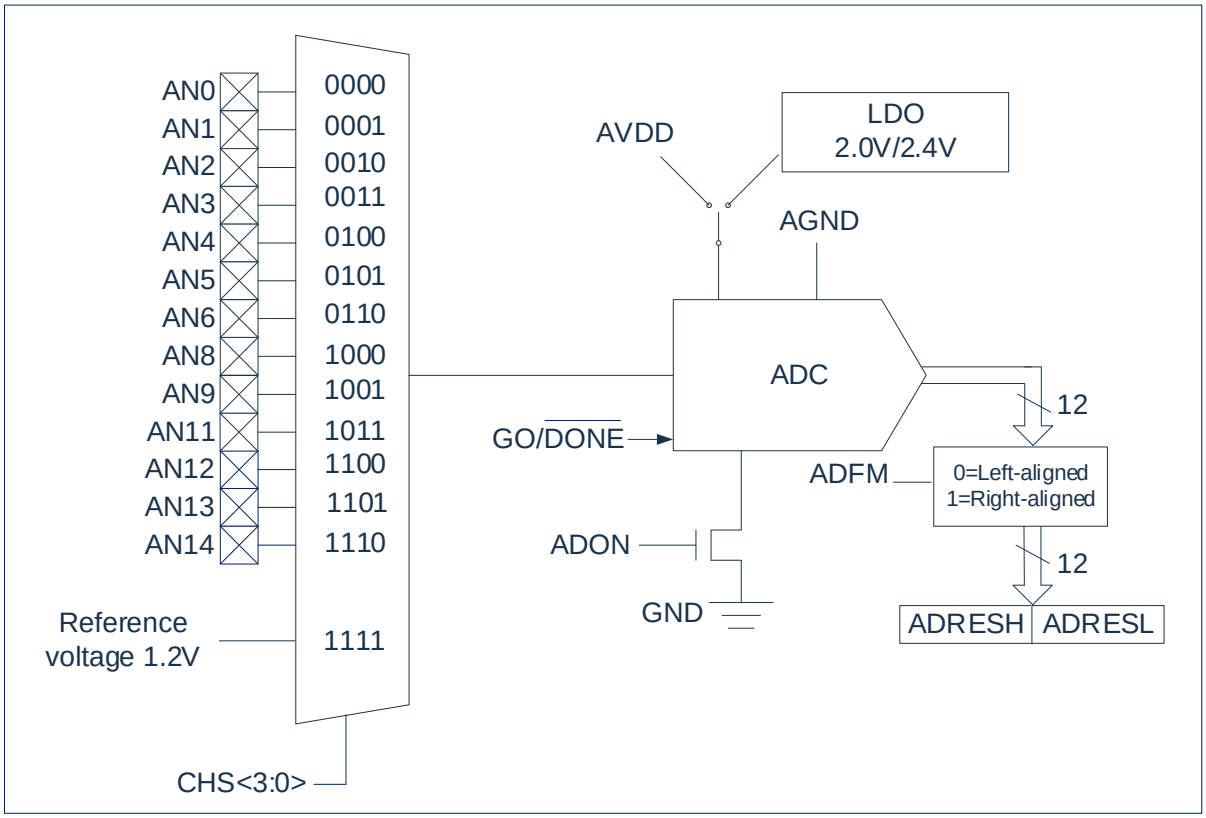


Figure 10-1: Block diagram of ADC

10.2 ADC configuration

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

- Port configuration;
- Channel selection;
- ADC conversion clock source;
- Interrupt control;
- Result storage format.

10.2.1 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 and ANS bits to 1. For more information, please refer to the corresponding ports section.

Note: Applying an analog voltage to a pin defined as a digital input may result in overcurrent in the input buffer.

10.2.2 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 delay is required before the next conversion starts. For more information see the Section “ADC working principle”.

10.2.3 ADC reference voltage

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

When the internal reference voltage is selected, the conversion clock division should be F_{SYS}/32 or slower.

10.2.4 Converter clock

The clock source and conversion time can be selected by software setting the ADCS bit in the ADCON0 register and the TADSEL bit in the ADCON1 register. 8 possible clock frequencies are available as follows.

TADSEL=1		TADSEL=0	
◆	F_{SYS}/8	◆	F_{SYS}/8
◆	F_{SYS}/16	◆	F_{SYS}/16
◆	F_{SYS}/32	◆	F_{SYS}/32
◆	F_{SYS}/128	◆	F_{LSI}(dedicated internal oscillator)

The time to complete a one-bit conversion is defined as TAD. When the TADSEL bit is 0, a complete 12-bit conversion requires 49 TAD cycles.

When the TADSEL bit is 1, a complete 12-bit conversion requires 16 TAD cycles.

The following table shows an example of the correct selection of the ADC clock, which must comply with the corresponding TAD specification in order to obtain correct conversion results.

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

ADC clock period (TAD) vs. Device operating frequency (VDD=5.0V, TADSEL=0)

ADC clock selection		One AD conversion time	
ADC clock source	ADCS<1:0>	F_{SYS} = 16MHz	F_{SYS} = 8MHz
F_{SYS}/8	00	24.5μs	49μs
F_{SYS}/16	01	49μs	98μs
F_{SYS}/32	10	98μs	196μs
F_{LSI}	11	1-3ms	1-3ms

DC clock period (TAD) vs. Device operating frequency (VDD=5.0V, TADSEL=1)

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

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

10.2.5 ADC interrupt

The ADC module allows an interrupt to be generated upon completion of the 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 allowed.

10.2.6 Output formatting

The result of 12-bit A/D conversion can be in two formats: left-aligned or right-aligned. The output format is controlled by the ADFM bit in ADCON0 register.

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

10.3 ADC working principle

10.3.1 Start conversion

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

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

10.3.2 Complete conversion

When the conversion is complete, the ADC mod will:

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

10.3.3 Stop conversion

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

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

10.3.4 Working principle of ADC in sleep mode

The ADC module can be operated in sleep mode. This operation requires the ADC clock source to be set to the F_{LSI}. If the F_{LSI} 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 the ADC interrupt is allowed, the device will wake up from sleep mode when the conversion is finished. If the ADC interrupt is disabled, the ADC module will be turned off after the conversion even if the ADON bit is kept set to 1. If the ADC clock source is not F_{LSI}, execution of the STOP instruction aborts the current conversion and shuts down the AD module, even though the ADON bit remains set to 1.

10.3.5 A/D conversion steps

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 (If switching from VDD to internal 2.0V/2.4V, it is necessary to wait at least 200us before starting to detect AD);
   - Select ADC conversion clock;
   - Select ADC input channel;
   - Choose result format;
   - Start ADC mod.
3. ADC interrupt configuration (optional):
   - Clear ADC interrupt flag bit;
   - Allow ADC interrupt;
   - Allow peripheral interrupt;
   - Allow global interrupt.
4. Wait for the required sampling time.
5. Set GO/$\overline{\text{DONE}}$ to 1 to start conversion.
6. Wait for the ADC conversion to end by one of the following methods:
   - Query GO/$\overline{\text{DONE}}$ bit
   - Wait for ADC interrupt (allow interrupt).
7. Read ADC results.
8. Clear the ADC interrupt flag bit (if interrupt is allowed, this operation is required).

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

Example: AD conversion

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

10.4 ADC related registers

There are mainly 4 registers related to AD conversion, namely control registers ADCON0 and ADCON1, 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/$\overline{DONE}$	ADON
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit6 ADCS<1:0>: A/D conversion clock selection bit

	TADSEL=1	TADSEL=0
00=	F_{SYS}/8	F_{SYS}/8
01=	F_{SYS}/16	F_{SYS}/16
10=	F_{SYS}/32	F_{SYS}/32
11=	F_{SYS}/128	F_{LSI} (32KHz generated by internal low-speed RC oscillator)

Bit5~Bit2 CHS<3:0>: Analog channel select bit

0000= AN0
0001= AN1
0010= AN2
0011= AN3
0100= AN4
0101= AN5
0110= AN6
0111= Reserved
1000= AN8
1001= AN9
1010= Reserved
1011= AN11
1100= AN12
1101= AN13
1110= AN14
1111= Fixed reference voltage (1.2V fixed reference voltage)

Bit1 GO/$\overline{DONE}$: AD conversion status bit

1= AD conversion in progress. Setting this bit to 1 starts the AD conversion. This bit is automatically cleared by the hardware when the AD conversion is complete
0= AD conversion complete/ or not in progress

Bit0 ADON: ADC enable bit

1= Enable ADC
0= Disable ADC, does not consume operating current

AD control register ADCON1(9FH)

9FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON1	ADFM	----	----	----	TADSEL	LDOEN	----	LDOSEL
R/W	R/W	----	----	----	R/W	R/W	----	R/W
Reset value	0	----	----	----	0	0	----	0

Bit7 ADFM: AD conversion result format selection bit
1= Right-aligned
0= Left-aligned

Bit6~Bit4 Unused, read 0

Bit3 TADSEL ADC conversion clock number selection bit
1= 16*TAD
0= 49*TAD

Bit2 LDOEN ADC internal reference LDO enable bit
1= Enable, the ADC's VREF input is LDO
0= Disable, the ADC's VREF input is VDD

Bit1 Unused, read 0

Bit0 LDOSEL AD reference voltage selection bit
1= 2.0V
0= 2.4V

AD data register high 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<9:2>: ADC result register bit;
Higher 8 bits of the 12-bit conversion result

AD data register low 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 Unused

AD data register high 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 Unused
Bit1~Bit0 ADRES<11:10>: ADC result register bit
The higher 2 bits of the 12-bit conversion result

AD data register low 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 9th to 2nd bits of the 12-bit conversion result

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

11. PWM Module

The chip contains a 10-bit PWM module, which can be configured as 4 common period, independent duty cycle outputs + 1 independent output.

The PWM1 and PWM0 outputs can be selected as RB1-RB0 or RB1-RB3 or RB4-RB3 or RA2-RA1 or RB6-RA6 or RB6-RB5 via CONFIG.

11.1 Pin configuration

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

11.2 PWM related registers

PWMCON PWM control register (1BH)

1BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMCON	CLKDIV[2:0]			---	PWM1DIR	PWM0DIR	PWM1EN	PWM0EN
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~Bit5	CLKDIV[2:0]:	PWM clock frequency division
	111=	F_{HSI} /128
	110=	F_{HSI} /64
	101=	F_{HSI} /32
	100=	F_{HSI} /16
	011=	F_{HSI} /8
	010=	F_{HSI} /4
	001=	F_{HSI} /2
	000=	F_{HSI} /1
Bit4	Unused	
Bit3~Bit2	PWMxDIR	PWM output polarity control bit
	1=	PWMx reverse output
	0=	PWMx normal output
Bit1~Bit0	PWMxEN:	PWMx enable bit
	1=	Enable PWMx
	0=	Disable PWMx

PWM control register PWMCON1(8EH)

8EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMCON1	---	---	PWM4DIR	PWM3DIR	PWM2DIR	PWM4EN	PWM3EN	PWM2EN
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	---	0	0	0	0	0	0

Bit7~Bit6	Unused	
Bit5~Bit3	PWMxDIR	PWM output polarity control bit
	1=	PWMx reverse output
	0=	PWMx normal output
Bit1~Bit0	PWMxEN:	PWMx enable bit
	1=	Enable PWMx
	0=	Disable PWMx

PWM0~PWM3 period low bit register PWMTL (19H)

19H	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]: PWM period low 8 bits

PWM4 period low bit register PWM4TL(1DH)

1DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM4TL	PWM4T[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWM4T[7:0]: PWM4 period low 8 bits

PWM period high bit register PWMTH (1AH)

1AH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMTH	---	---	PWMD4[9:8]		PWM4T[9:8]		PWMT[9:8]	
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	---	0	0	0	0	0	0

Bit7~Bit6 Unused.
Bit5~Bit4 PWMD4[9:8]: 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

Note: Writing to PWM4T[9:8] does not take effect immediately, but requires a write operation to PWM4TL.
Writing PWMD4[9:8] does not take effect immediately, but requires a write operation to PWMD4L.

PWM0 duty cycle low register PWMD0L (16H)

16H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD0L	PWMD0[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD0[7:0]: PWM0 duty cycle low 8 bits

PWMD1L PWM1 duty cycle low register (17H)

17H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD1L	PWMD1[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD1[7:0]: PWM1 duty cycle low 8 bits

PWM2 duty cycle low register PWMD2L (0EH)

0EH	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]: PWM2 duty cycle low 8 bits

PWM3 duty cycle low register PWMD3L (0FH)

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

Bit7~Bit0 PWMD3[7:0]: PWM3 duty cycle low 8 bits

PWM4 duty cycle low register PWMD4L (10H)

10H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD4L	PWMD4[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD4[7:0]: PWM4 duty cycle low 8 bits

PWM0 and PWM1 duty cycle high register PWMD01H (18H)

18H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD01H	---	---	---	---	PWMD1[9:8]		PWMD0[9:8]	
R/W	---	---	---	---	R/W	R/W	R/W	R/W
Reset value	---	---	---	---	0	0	0	0

Bit7~Bit4 Unused
Bit3~Bit2 PWMD1[9:8]: PWM1 duty cycle high 2 bits
Bit1~Bit0 PWMD0[9:8]: PWM0 duty cycle high 2 bits

PWM2 and PWM3 duty cycle high register PWMD23H (1CH)

1CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD23H	---	---	---	---	PWMD3[9:8]		PWMD2[9:8]	
R/W	---	---	---	---	R/W	R/W	R/W	R/W
Reset value	---	---	---	---	0	0	0	0

Bit7~Bit4 Unused
Bit3~Bit2 PWMD3[9:8]: PWM3 duty cycle high 2 bits
Bit1~Bit0 PWMD2[9:8]: PWM2 duty cycle high 2 bits

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

11.3 PWM register write sequence

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

1) Write the higher 2-bit value, the high 2-bit value is just written to the internal cache;
2) Write the lower 8 bits, then the full 10-bit duty cycle value is latched;
3) **The above operation sequence is only for PWM2-PWM4 duty cycle registers, PWM0-PWM1 duty cycle registers are not subject to this constraint.**

11.4 PWM period

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

In order to ensure the correctness of the cycle value, the chip is designed with an internal cache loading function. To operate the 10-bit cycle value, the following sequence should be strictly followed:

1) Writes the higher 2 bits of the value, the higher 2 bits of the value are just written to the internal cache;
2) Write the lower 8 bits, at which point the full 10-bit duty cycle value is latched;
3) **The above operation sequence is only for PWM4 cycle register, PWM0-PWM3 cycle registers are not subject to this constraint.**

Formula 1: PWM period calculation formula:

PWM period=[*PWMT+1*]***Tosc**(*CLKDIV* prescaler value)

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

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

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

11.5 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 It is latched into internal latch.

Formula 2: Pulse width calculation formula:

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

Formula 3: PWM duty cycle calculation formula:

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

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.

11.6 System clock frequency change

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

11.7 PWM configuration

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

1. Select the PWM output IO port.
2. Set the corresponding TRIS bit to 1 to make it as an input pin.
3. Set the PWM period by loading the PWMTH and PWMTL registers.
4. Set the PWM duty cycle by loading the PWMDxL and PWMDxxH registers.
5. Clear the PWMIF flag bit.
6. Set the PWMCN[1:0] bits to enable the corresponding PWM outputs.
7. After the new PWM period starts, enable PWM output:
   - Wait for PWMIF bit set to 1.
   - Enable the PWM pin output driver by clearing the corresponding TRIS bit.

12. Dedicated 1/2BiasLCD Driver

SC8P171XD is built-in pull-up and pull-down resistors by 13 I/O ports, which can be used to do 1/2Bias LCD driver, and its control flow as follows.

1. Set LCDCON.7 bit LCDEN to 1.
2. Set the LCDCON[1:0] bits to select the COM port output current.
3. Set the corresponding COMEN bit to "1" to allow the pin to be a COM port for 1/2Bias
4. Control the COM port output high, low or 1/2 level by software. When COMEN=0 and the corresponding bit of TRIS is 0, the corresponding bit of PORT register controls the COM port to output high or low level; when COMEN=1, the COM port outputs 1/2 level.

The relevant registers are as follows.

LCD control register LCDCON(15H)

15H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDCON	LCDEN	---	---	---	---	---	LCD_ISLE[1:0]	
R/W	R/W	---	---	---	---	---	R/W	R/W
Reset value	0	---	---	---	---	---	0	0

Bit7	LCDEN:	LCD module enable
	0:	Disable LCD module
	1:	Enable LCD module
Bit6~Bit2	Unused	
Bit1~Bit0	LCD_ISLE[1:0]	LCD output current selection bit
	00:	100μA@5V
	01:	200μA@5V
	10:	400μA@5V
	11:	800μA@5V

LCD function COM port control register LCDCON0 (13H)

13H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDCON0	---	COM6EN	COM5EN	COM4EN	COM3EN	COM2EN	COM1EN	COM0EN
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	Unused	
Bit6~Bit0	COMxEN:	COM port function setting
	0:	The corresponding COMx port is a normal I/O port
	1:	The corresponding COMx port is the COM port for LCD function

LCD function COM port control register LCDCON1 (14H)

14H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDCON1	---	COM14EN	COM13EN	COM12EN	COM11EN	---	COM9EN	COM8EN
R/W	---	R/W	R/W	R/W	R/W	---	R/W	R/W
Reset value	---	0	0	0	0	---	0	0

Bit7 Unused

Bit6~Bit3 COMxEN: COM port function setting

0: The corresponding COMx port is a normal I/O port

1: The corresponding COMx port is the COM port for LCD function

Bit2 Unused

Bit1~Bit0 COMxEN: COM port function setting

0: The corresponding COMx port is a normal I/O port

1: The corresponding COMx port is the COM port for LCD function

13. Low Voltage Detection (LVD)

13.1 LVD module overview

The SC8P171xE microcontroller has a low voltage detection function that can be used to monitor the supply voltage. If the supply voltage falls below a set value, an interrupt can be generated. The program can read the LVD output flag bit in real time.

13.2 LVD related registers

LVD control register LVDCON (91H)

91H	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	Unused	
Bit3~Bit1	LVD_SEL[2:0]:	LVD voltage selection
	000=	2.2V
	001=	2.4V
	010=	2.7V
	011=	3.0V
	100=	3.3V
	101=	3.7V
	110=	4.0V
	111=	4.3V
Bit0	LVDEN:	LVD enable bit
	0=	Disable
	1=	Enable

13.3 LVD operation

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

LVD module 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, an LVD interrupt will be generated, the interrupt flag bit LVDIF will be set to 1, and the interrupt will be generated. LVD can also be used for interrupt wake-up mode.

14. Electrical Parameters

14.1 Limit parameters

Supply voltage……GND-0.3V~GND+6.0V

Storage temperature……-50℃~125℃

Working temperature……-40℃~85℃

Port input voltage……GND-0.3V~VDD+0.3V

Maximum 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 value of the operating conditions. We do not recommend that the device operate outside the range specified in this specification. The stability of the device will be affected if it is operated for a long period of time under extreme conditions.

14.2 DC characteristics

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

Symbol	Item	Test condition		Min.	Typ.	Max.	Unit
		VDD	Condition				
VDD	Operating voltage	-	16MHz	2.6	-	5.5	V
		-	8MHz	1.8	-	5.5	V
I_{DD}	Operating current	5V	F_{SYS}=16MHz, disable all analog modules.	-	2	-	mA
		5V	F_{SYS}=8MHz, disable all analog modules.	-	1	-	mA
		3V	F_{SYS}=16MHz, disable all analog modules.	-	1.2	-	mA
		3V	F_{SYS}=8MHz, disable all analog modules.	-	0.8	-	mA
I_{STB}	Static current	5V	----	-	0.1	2	uA
		3V	----	-	0.1	2	uA
V_{IL}	Low level input voltage	-	----	-	-	0.3VDD	V
V_{IH}	High level input voltage	-	----	0.7VDD	-	-	V
V_{OH}	High level output voltage	-	No load	0.9VDD	-	-	V
V_{OL}	Low level output voltage	-	No load	-	-	0.1VDD	V
V_{REF}	Internal fixed reference voltage	-	----	1.188	1.2	1.212	V
R_{PH}	Pull-up resistor value (Normal I/O)	5V	Vo=0.5VDD	-	32	-	KΩ
		3V	Vo=0.5VDD	-	52	-	KΩ
	Pull-up resistor value (VPP/RB2)	5V	Vo=0.5VDD	-	34	-	KΩ
		3V	Vo=0.5VDD	-	35	-	KΩ
R_{PL}	Pull-down resistor value	5V	Vo=0.5VDD	-	36	-	KΩ
		3V	Vo=0.5VDD	-	50	-	KΩ
I_{OL}	Output port sink current (Normal I/O)	5V	V_{OL}=0.3VDD	-	50	-	mA
		3V	V_{OL}=0.3VDD	-	25	-	mA
	Output port sink current (VPP/RB2)	5V	V_{OL}=0.3VDD	-	33	-	mA
		3V	V_{OL}=0.3VDD	-	15	-	mA
I_{OH}	Output port source current	5V	V_{OH}=0.7VDD	-	18	-	mA
		3V	V_{OH}=0.7VDD	-	8	-	mA

14.3 ADC electrical characteristics

(T_A= 25˚C, unless otherwise specified)

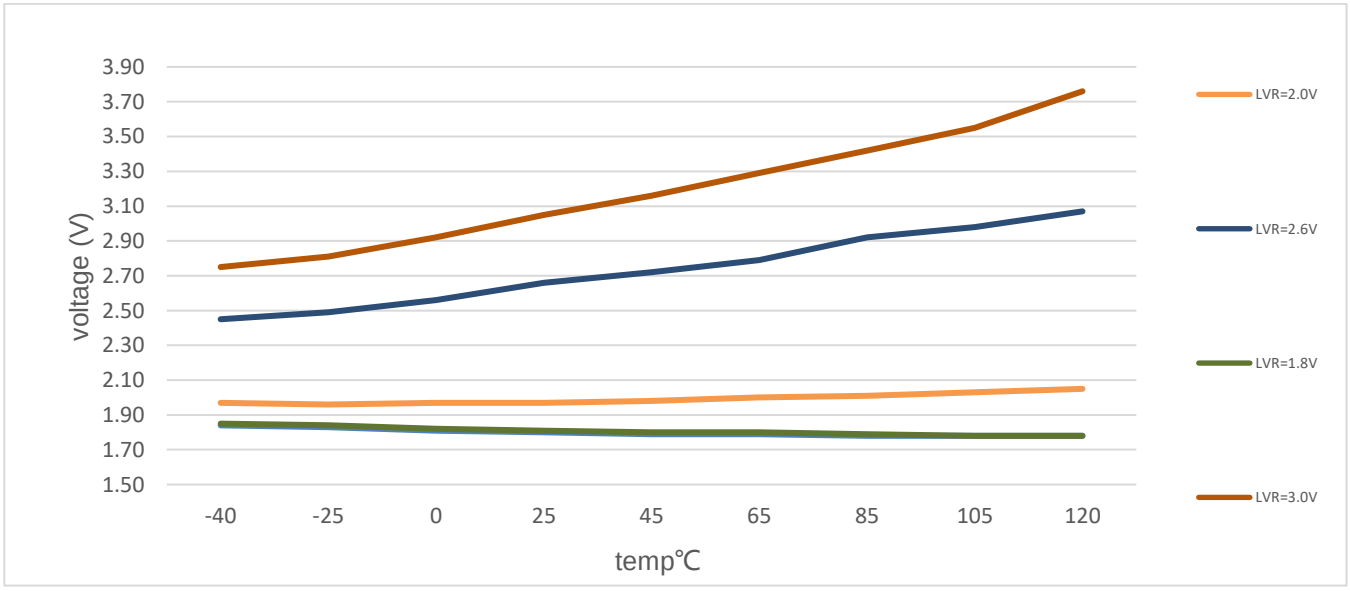
Symbol	Item	Test condition	Min.	Typ.	Max.	Unit
V_{ADC}	ADC operating voltage	AD_{VREF}=VDD, F_{ADC}=1MHz	3.0		5.5	V
		AD_{VREF}=VDD, F_{ADC}=500kHz	2.6		5.5	V
		AD_{VREF}=2.4V, F_{ADC}=250kHz	2.6		5.5	V
		AD_{VREF}=2.0V, F_{ADC}=250kHz	2.6		5.5	V
I_{ADC}	ADC conversion current	V_{ADC}=5V, AD_{VREF}=VDD, F_{ADC}=500kHz			500	uA
		V_{ADC} =3V, AD_{VREF}=VDD, F_{ADC}=500kHz			200	uA
V_{ADI}	ADC input voltage	V_{ADC}=5V, AD_{VREF}=VDD, F_{ADC}=250kHz	0		V_{ADC}	V
DNL	Differential nonlinear error	V_{ADC}=5V, AD_{VREF}=VDD, F_{ADC}=250kHz			±2	LSB
INL	Integral nonlinear error	V_{ADC}=5V, AD_{VREF}=VDD, F_{ADC}=250kHz			±2	LSB
T_{ADC}	ADC conversion time	-		16		T_{ADCCLK}
AD_{VREF1}	AD reference LDO accuracy1	VDD=2.6~5.5V, VREF=2.0V	-0.6%	2.0	+0.6%	V
AD_{VREF2}	AD reference LDO accuracy2	VDD=2.6~5.5V, VREF=2.4V	-0.6%	2.4	+0.6%	V

14.4 Power on reset characteristics

(T_A= 25˚C, unless otherwise specified)

Symbol	Item	Test condition	Min.	Typ.	Max.	Unit
t_{VDD}	VDD rising rate	-	0.05			V/ms
V_{LVR1}	LVR set voltage=1.8V	VDD=1.6~5.5V	1.7	1.8	1.9	V
V_{LVR2}	LVR set voltage=2.0V	VDD=1.8~5.5V	1.9	2.0	2.1	V
V_{LVR3}	LVR set voltage=2.6V	VDD=2.4~5.5V	2.5	2.6	2.7	V
V_{LVR3}	LVR set voltage=3.0V	VDD=2.8~5.5V	2.9	3.0	3.1	V

LVR temperature profile



14.5 LVD electrical characteristics

(TA= 25˚C, unless otherwise specified)

Symbol	Item	Test condition	Min.	Typ.	Max.	Unit
VLVD	Operating voltage	-	2.0		5.5	V
	Acccuracy	VDD=2.0~5.5V, T_A= -40~85˚C	-5%	V_{SET}	+5%	V

14.6 AC characteristics

(T_A= 25˚C, unless otherwise specified)

Symbol	Item	Test condition		Min.	Typ.	Max.	Unit
		VDD	Condition				
T_{WDT}	WDT reset time	5V	---	-	16	-	ms
		3V	---	-	16	-	ms
T_{AD}	AD conversion time	5V	TADSEL=0	-	49	-	CLK
		5V	TADSEL=1	-	16	-	CLK
F_{INTRC}	Internal frequency 8MHz	VDD=4.0~5.5V	TA=25˚C	-1.5%	8	+1.5%	MHz
		VDD=2.5~5.5V	TA=25˚C	-2.0%	8	+2.0%	MHz
		VDD=4.0~5.5V	TA= -40~85˚C	-2.5%	8	+2.5%	MHz
		VDD=2.5~5.5V	TA= -40~85˚C	-3.5%	8	+3.5%	MHz
		VDD=1.8~5.5V	TA= -40~85˚C	-5.0%	8	+5.0%	MHz
	Internal frequency 16MHz	VDD=4.0~5.5V	TA=25˚C	-1.5%	16	+1.5%	MHz
		VDD=2.5~5.5V	TA=25˚C	-2.0%	16	+2.0%	MHz
		VDD=4.0~5.5V	TA= -40~85˚C	-3.5%	16	+3.5%	MHz
		VDD=2.5~5.5V	TA= -40~85˚C	-5%	16	+5%	MHz

14.7 EMC characteristics

14.7.1 EFT electrical characteristics

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

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

14.7.2 ESD electrical characteristics

Symbol	Item	Test condition	Level
V_{ESD}	Electrostatic discharge (Human Body Discharge Mode HBM)	T_A = + 25℃, JEDEC EIA/JESD22- A114	Class 3A

14.7.3 Latch-Up electrical characteristics

Symbol	Item	Test condition	Test type
LU	Static latch-up class	JEDEC STANDARD NO.78D NOVEMBER 2011	Class I (T_A = +25℃)

15. Instructions

15.1 Instruction set

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

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

15.2 Instruction description

ADDA **[R]**

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

period: 1

Affected flag bit: C, DC, Z, OV

example:

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

ADDR **[R]**

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

period: 1

Affected flag bit: C, DC, Z, OV

example:

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

ADDCA **[R]**

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

period: 1

Affected flag bit: C, DC, Z, OV

example:

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

ADDCR **[R]**

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

period: 1

Affected flag bit: C, DC, Z, OV

example:

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

ADDIA **i**

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

period: 1

Affected flag bit: C, DC, Z, OV

example:

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

ANDA **[R]**

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

period: 1

Affected flag bit: Z

example:

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

ANDR **[R]**

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

period: 1

Affected flag bit: Z

example:

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

ANDIA **i**

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

period: 1

Affected flag bit: Z

example:

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

CALL **add**

operation: Call subroutine

period: 2

Affected flag bit: none

example:

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

CLRA

operation: ACC clear

period: 1

Affected flag bit: Z

example:

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

CLR [R]

operation: Register R clear

period: 1

Affected flag bit: Z

example:

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

CLRB [R],b

operation: Clear b bit on register R

period: 1

Affected flag bit: none

example:

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

CLRWDT

operation: Clear watchdog timer

period: 1

Affected flag bit: TO, PD

example:

```
CLRWDT                  ;watchdog timer clear
```

COMA [R]

operation: Reverse register R, save the result to ACC

period: 1

Affected flag bit: Z

example:

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

COMR **[R]**

operation: Reverse register R, save the result to R

period: 1

Affected flag bit: Z

example:

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

DECA **[R]**

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

period: 1

Affected flag bit: Z

example:

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

DECR **[R]**

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

period: 1

Affected flag bit: Z

example:

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

HSUBA **[R]**

operation: ACC subtract R, save the result to ACC

period: 1

Affected flag bit: C,DC,Z,OV

example:

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

HSUBR **[R]**

operation: ACC subtract R, save the result to R

period: 1

Affected flag bit: C,DC,Z,OV

example:

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

HSUBCA **[R]**

operation: ACC subtract C, save the result to ACC

period: 1

Affected flag bit: C,DC,Z,OV

example:

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

HSUBCR **[R]**

operation: ACC subtract C, save the result to R

period: 1

Affected flag bit: C,DC,Z,OV

example:

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

INCA **[R]**

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

period: 1

Affected flag bit: Z

example:

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

INCR **[R]**

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

period: 1

Affected flag bit: Z

example:

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

JP **add**

operation: Jump to add address

period: 2

Affected flag bit: None

example:

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

LD **A,[R]**

operation: Load the value of R to ACC

period: 1

Affected flag bit: Z

example:

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

LD **[R],A**

operation: Load the value of ACC to R

period: 1

Affected flag bit: none

example:

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

LDIA **i**

operation: Load i to ACC

period: 1

Affected flag bit: none

example:

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

NOP

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

```
NOP
NOP
```

ORIA **i**

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

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

ORA **[R]**

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

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

ORR **[R]**

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

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

RET

operation: Return from subroutine

period: 2

Affected flag bit: none

example:

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

RET i

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

period: 2

Affected flag bit: none

example:

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

RETI

operation: Interrupt return

period: 2

Affected flag bit: none

example:

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

RLCA [R]

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

period: 1

Affected flag bit: C

example:

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

RLCR **[R]**

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

period: 1

Affected flag bit: C

example:

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

RLA **[R]**

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

period: 1

Affected flag bit: none

example:

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

RLR **[R]**

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

period: 1

Affected flag bit: none

example:

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

RRCA **[R]**

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

period: 1

Affected flag bit: C

example:

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

RRCR **[R]**

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

period: 1

Affected flag bit: C

example:

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

RRA **[R]**

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

period: 1

Affected flag bit: none

example:

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

RRR **[R]**

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

period: 1

Affected flag bit: none

example:

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

SET **[R]**

operation: Set all bits in register R as 1

period: 1

Affected flag bit: none

example:

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

SETB **[R],b**

operation: Set b bit in register R to 1

period: 1

Affected flag bit: none

example:

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

STOP

operation: Enter sleep

period: 1

Affected flag bit: TO, PD

example:

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

SUBIA **i**

operation: I minus ACC, save the result to ACC

period: 1

Affected flag bit: C,DC,Z,OV

example:

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

SUBA **[R]**

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

period: 1

Affected flag bit: C,DC,Z,OV

example:

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

SUBR **[R]**

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

period: 1

Affected flag bit: C,DC,Z,OV

example:

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

SUBCA **[R]**

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

period: 1

Affected flag bit: C,DC,Z,OV

example:

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

SUBCR **[R]**

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

period: 1

Affected flag bit: C,DC,Z,OV

example:

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

SWAPA **[R]**

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

period: 1

Affected flag bit: none

example:

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

SWAPR **[R]**

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

period: 1

Affected flag bit: none

example:

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

SZB **[R],b**

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

period: 1 or 2

Affected flag bit: none

example:

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

SNZB **[R],b**

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

period: 1 or 2

Affected flag bit: none

example:

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

SZA **[R]**

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

period: 1 or 2

Affected flag bit: none

example:

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

SZR **[R]**

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

period: 1 or 2

Affected flag bit: none

example:

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

SZINCA **[R]**

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

period: 1 or 2

Affected flag bit: none

example:

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

SZINCR **[R]**

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

period: 1 or 2

Affected flag bit: none

example:

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

SZDECA **[R]**

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

period: 1 or 2

Affected flag bit: none

example:

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

SZDECR **[R]**

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

period: 1 or 2

Affected flag bit: none

example:

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

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

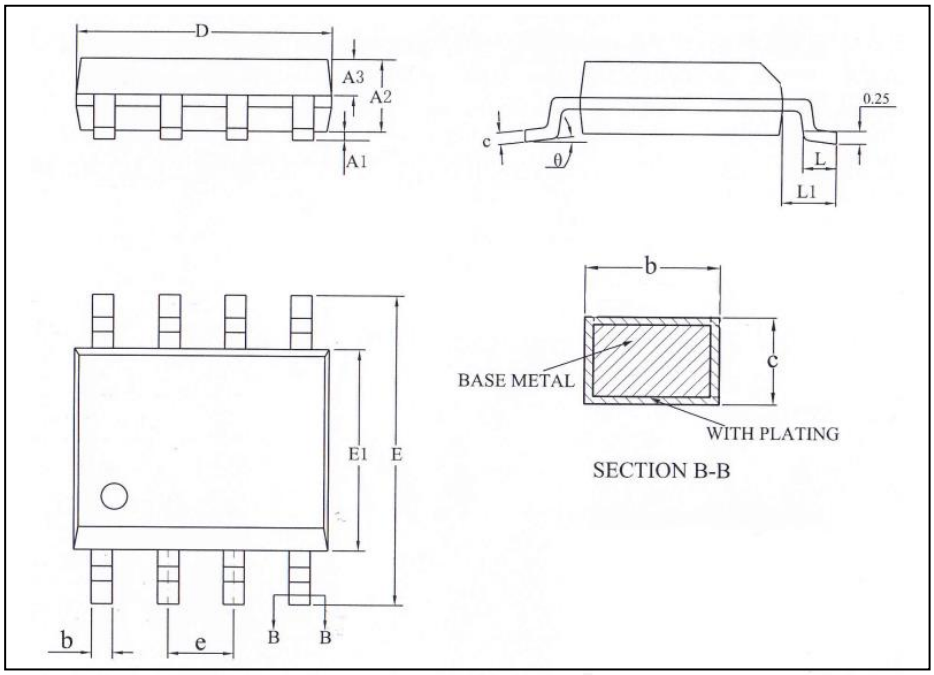
Affected flag bit: Z

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
```

16. Package

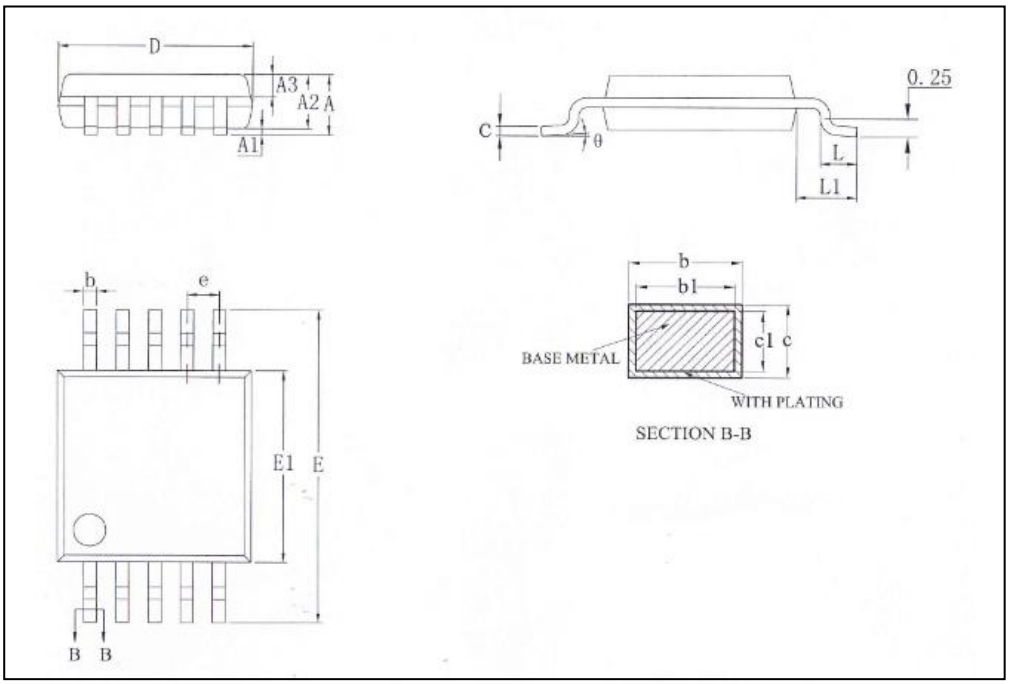
16.1 SOP8



Symbol	Millimeter		
	Min	Nom	Max
A1	0.05	-	0.25
A2	1.30	1.40	1.60
A3	0.55	-	0.70
b	0.33	-	0.51
c	0.17	-	0.26
D	4.70	-	5.10
E	5.80	6.00	6.20
E1	3.70	-	4.10
e	1.27BSC		
L	0.40	-	0.80
L1	1.05REF		
θ	0	-	8°

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

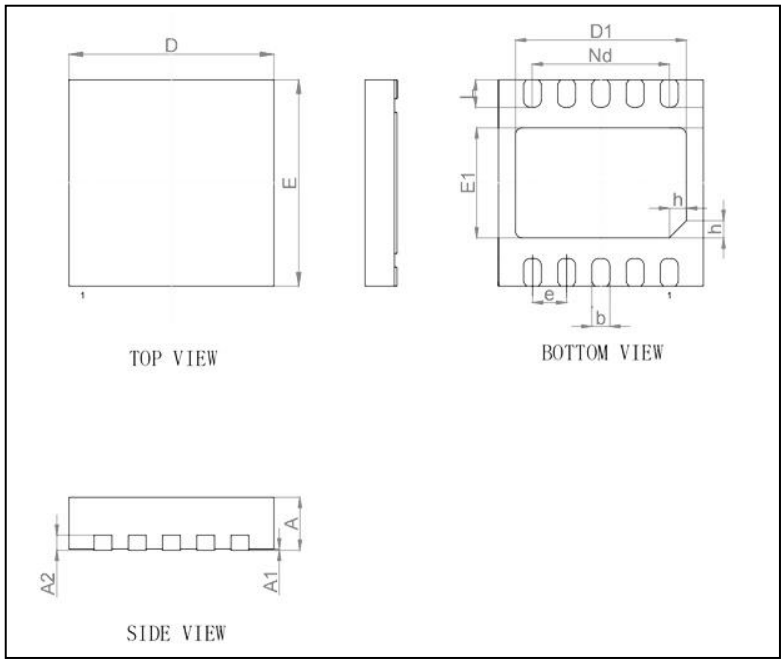
16.2 MSOP10



Symbol	Millimeter		
	Min	Nom	Max
A	-	-	1.10
A1	0.05	-	0.15
A2	0.75	0.85	0.95
A3	0.30	0.35	0.40
b	0.17	-	0.27
b1	0.17	0.20	0.23
c	0.08	-	0.23
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	-	0.80
L1	0.95REF		
θ	0	-	8°

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

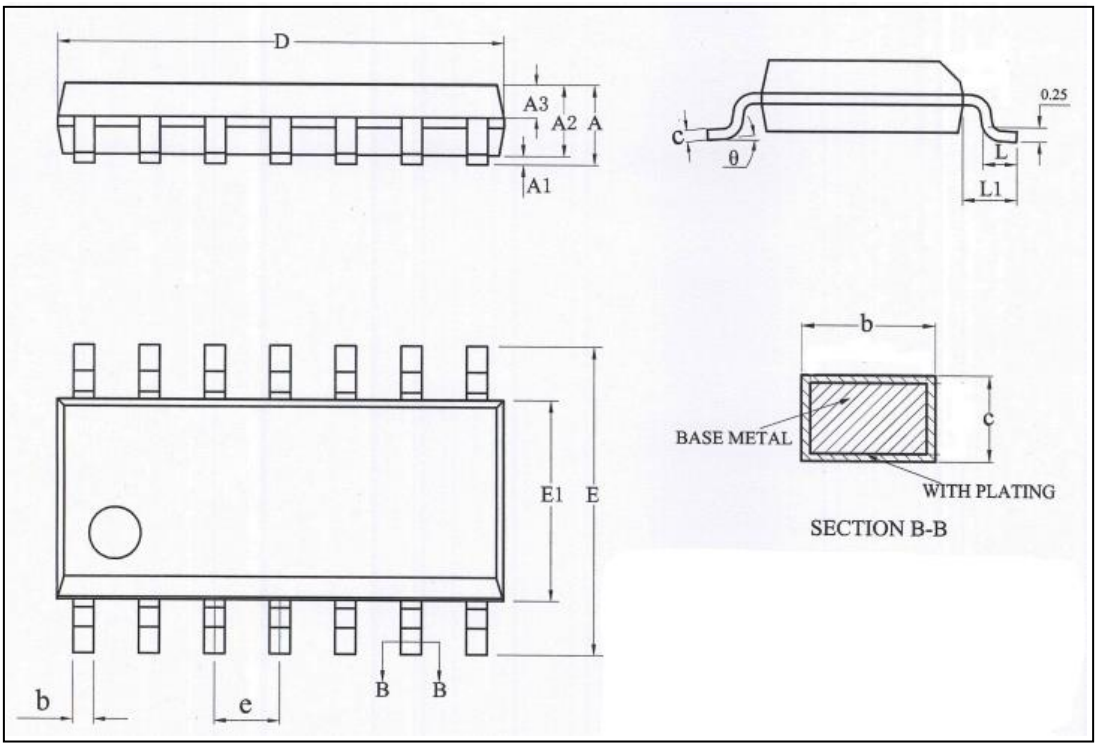
16.3 DFN10



Symbol	Millimeter		
	Min	Nom	Max
A	0.70	0.75	0.80
A1	0	-	0.05
b	0.18	-	0.30
A2	0.203REF		
D	2.90	3.00	3.10
E	2.90	3.00	3.10
D1	2.30	-	2.60
E1	1.45	-	1.80
e	0.50BSC		
h	0.20	0.25	0.30
Nd	2.00BSC		
L	0.30	0.40	0.50

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

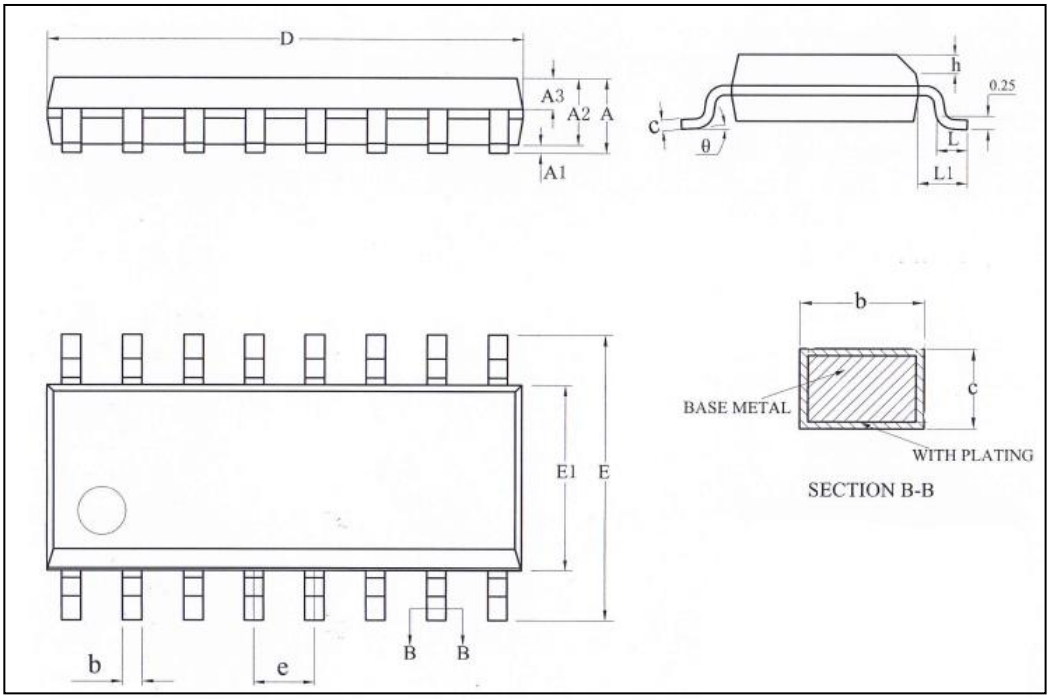
16.4 SOP14



Symbol	Millimeter		
	Min	Nom	Max
A	-	-	1.85
A1	0.05	-	0.25
A2	1.30	-	1.60
A3	0.60	0.65	0.70
b	0.356	-	0.47
c	0.193	-	0.26
D	8.45	-	8.85
E	5.80	6.00	6.20
E1	3.70	-	4.10
e	1.27BSC		
L	0.40	-	0.80
L1	1.05REF		
θ	0	-	8°

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

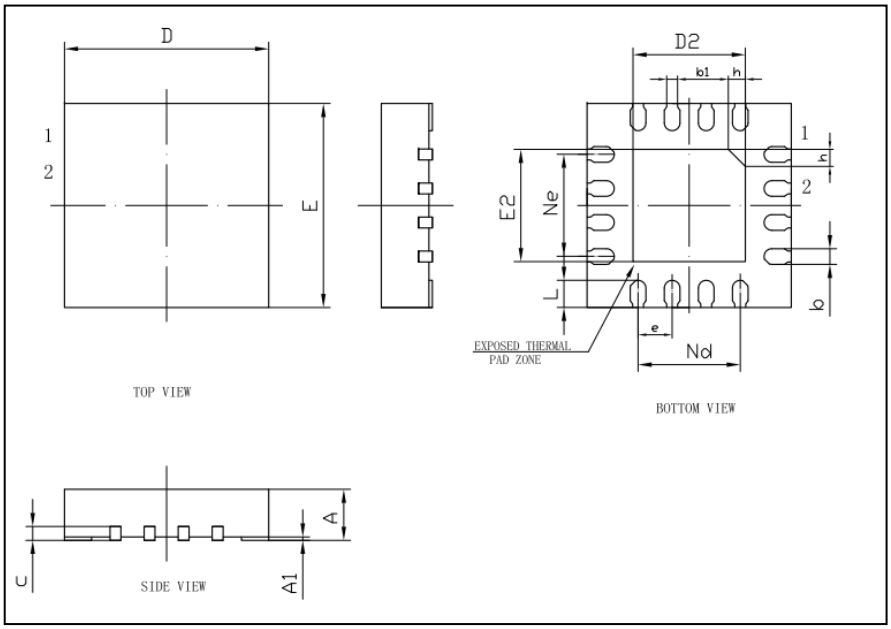
16.5 SOP16



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

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

16.6 QFN16



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

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

17. Revision History

Version	Date	Revised content
V1.0	May 2020	Initial version
V1.1	January 2021	Changed instructions for using PWM
V1.2	June 2021	Added some DFN10/QFN16 pin related contents
V1.3	May 2022	Added an LVR temperature profile
V1.4	June 2022	Added section 14.7 EMC characteristics
V1.4.1	October 2023	1) Added notes on pull-up resistors for RB2 2) Deleted error description about ADC interrupt wakeup 3) Update 14.7 EMC characteristics 4) Update the package information in 16.4/16.6 5) Add clock block diagram and modify relevant clock sources 6) Correct the input level range of the IO port 7) Correction of errors in TIMER2 working principle section and addition of PR2 register description 8) Unified Register Representation
V1.4.1	November 2023	1) ADC clock description in the revised clock block diagram
V1.4.2	September 2024	1) Revised the cover page 2) Modified SOP8/MSOP10/DFN10/SOP14/SOP16/QFN16 package dimensions