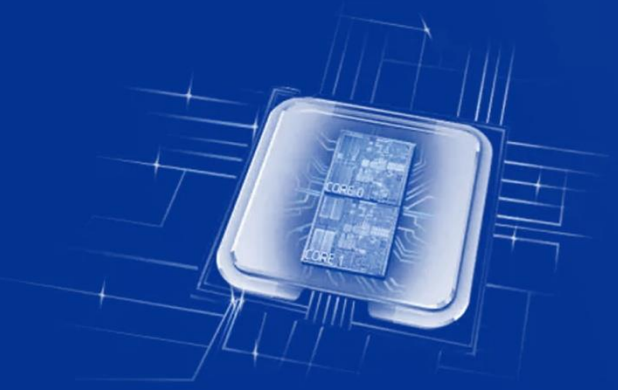
SCMCU



用户手册

SC8F2790 User Manual
Enhanced 8-bit CMOS Microcontroller with Flash Memory

V1.3.1

Please be reminded about following SCMCU's policies on intellectual property

(Ⅰ) SCMCU Limited has already applied for relative patents and entitled legal rights. Any patents related to SCMCU's MCU or other products is not authorized to use. Any individual, organization or company which infringes our company's intellectual property rights will be forbidden and stopped by our company through any legal actions, and our company will claim the lost and required for compensation of any damage to the company.

(Ⅱ) The name of SCMCU Limited and logo are both trademarks of our company

(Ⅲ) Our company preserve the rights to further elaborate on the improvements about product functions, reliability and design in this manual. However, our company is not responsible for any usage about this manual. The applications and their purposes in this manual are just for clarification, our company does not guarantee that these applications are feasible without further improvements and changes, and does not recommend any usage of the products in areas where people's safety is endangered during accident. Our company's products are not authorized to be used for life-saving or life support devices and systems. Our company has the right to change or improve the product without any notification.

Catalog

1. Product Overview

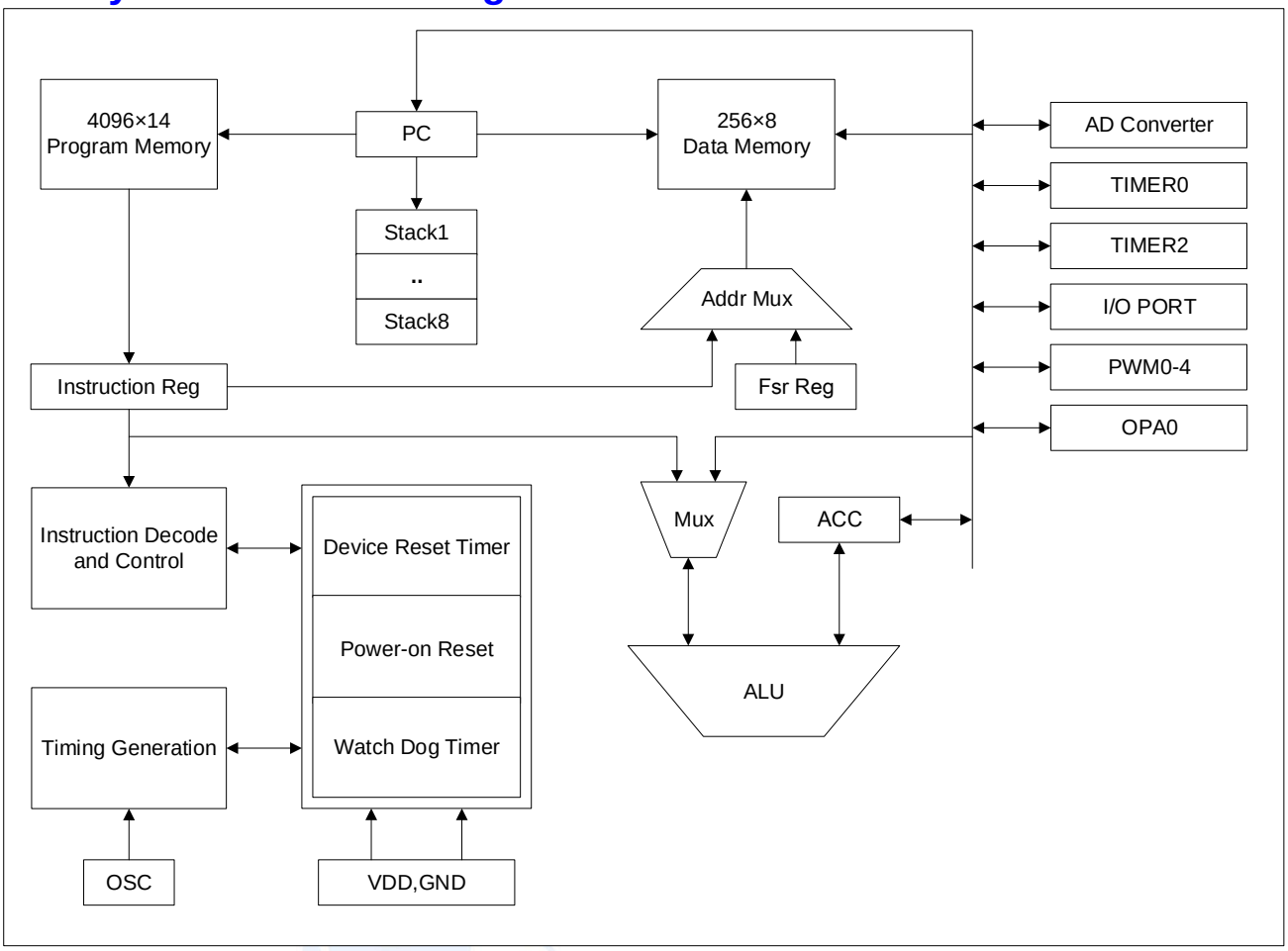
1.1 Functional Features

- Memory
  - Flash: 4K×14
  - Universal RAM: 256×8
- 8-level stack buffer
- Simple and practical instruction system (66 instructions)
- Built-in WDT timer
- Built-in low voltage detection circuit
- Interrupt source
  - 2 timer interrupts
  - RB port interrupt on change
  - Other peripheral interrupts
- Timer
  - 8-bit timer TIMER0, TIMER2
  - TIMER2 can select external 32.768 kHz oscillating clock source
- Built-in a high-performance OP-AMP module
- Operating voltage range: 3.3V~5.5V@16MHz
  2.5V~5.5V@8MHz
- Operating temperature range: -20℃~75℃
- An oscillation method
  - Internal RC oscillation: design frequency 8MHz/16MHz
- Instruction cycle (single or double instruction)
- PWM modules with complementary outputs
  - 10-bit PWM accuracy
  - 5-channel PWM, can be set to 2-channel complementary output
  - RB0/RB1, RB3/RB4 can enable 80mA heavy current to drive
  - 5-channel PWM with shared cycle and independent duty cycle
- 12-bit high-accuracy ADC
  - Selectable internal reference source: 1.2V/2.4V/3.0V
  - Built-in high-accuracy 0.6V reference voltage
    ±1.5% @VDD=2.5V~5.5V T_A =25°C
    ±2% @VDD=2.5V~5.5V T_A = -20°C~75°C

Model Description

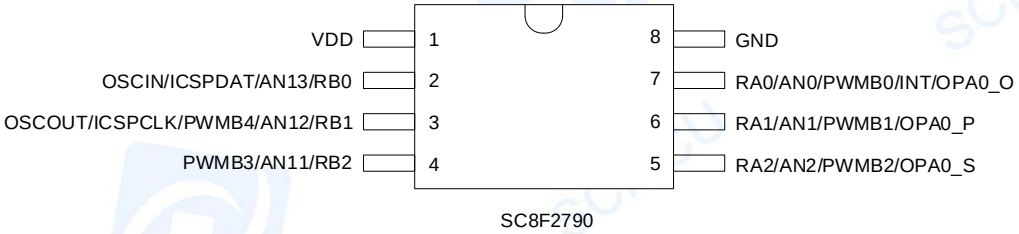
PRODUCT	Flash	RAM	I/O	PWM	OPA	ADC	PACKAGE
SC8F2790	4Kx14	256x8	6	5	1	12Bit×6	SOP8

1.2 System Structure Diagram



1.3 Pin Allocation

1.3.1 SC8F2790 Pinout Diagram



SC8F2790 Pin Description：

Pin Name	IO Type	Pin Description
VDD,GND	P	Voltage input pin and ground pin
OSCIN/OSCOUT	I/O	Crystal oscillator input/output pin
RA0-RA2	I/O	Programmable as input/push-pull output pin, with pull-up resistance function
RB0-RB2	I/O	Programmable as input/push-pull output pin, with pull-up resistance and interrupt on change function
ICSPCLK/ICSPDAT	I/O	Programmable as clock/data pin
AN0-AN2，AN11-AN13	I	12-bit ADC input pin
INT	I	External interrupt input pin
PWMB0-PWMB4	I/O	Group B PWM 0-4 output function
OPA0_P	I	Positive op-amp input
OPA0_S	I	Negative op-amp input
OPA0_O	O	Op-amp outputs

1.4 System Configuration Register

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

1. OSC (oscillation mode selection)
   - INTRC8M — F_{OSC} Selects internal 8MHz RC oscillation
   - INTRC16M — F_{OSC} Selects internal 16MHz RC oscillation (When $F_{SYS} = F_{OSC}/1$ is selected, the reset voltage needs to be selected as 3.3V)
2. WDT (watchdog selection)
   - ENABLE — Turn on the watchdog timer
   - DISABLE — Turn off the watchdog timer
3. PROTECT (encryption)
   - DISABLE — FLASH code is not encrypted
   - ENABLE — FLASH code encryption, the value read by the burner/emulator will be uncertain after encryption
4. LVR_SEL (low voltage detection voltage selection)
   - 2.5V
   - 3.3V
5. ICSPPORT_SEL (Emulation port function selection)
   - ICSP — ICSPCLK and DAT ports are always kept as emulation ports,all functions are not available
   - NORMAL — ICSPCLK and DAT ports are general function ports

1.5 Online Serial Programming

Serial programming of the microcontroller can be done in the final circuit application. Programming can be done simply with the following 4 wires:

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

This allows users to build boards with unprogrammed devices and program the microcontroller only before the product is delivered. Therefore, the latest version of firmware or custom firmware can be burned into the microcontroller.

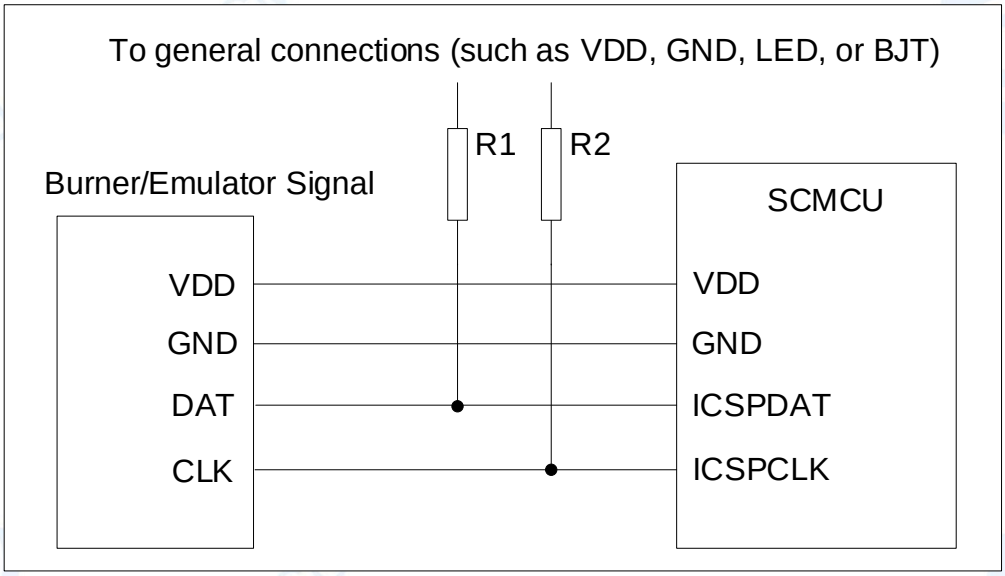


Figure 1-1: Typical Online Serial Programming Connection Method

In the above figure, R1 and R2 are electrically isolated devices, often replaced by resistors with the following resistance values: R1≥4.7K, R2≥4.7K.

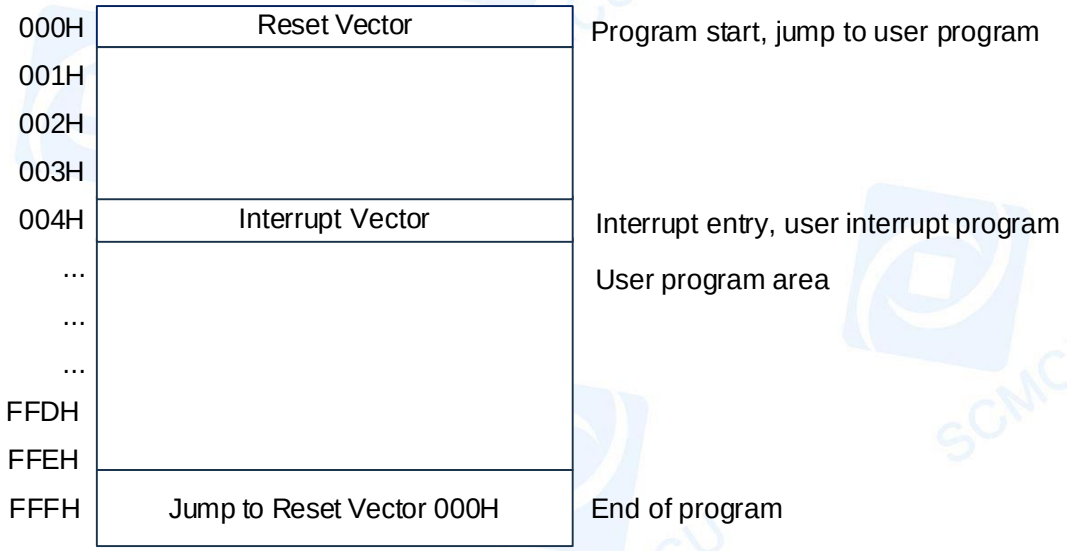
2. Central Processing Unit (CPU)

2.1 Memory

2.1.1 Program Memory

SC8F2790 Program Memory Space

FLASH:4K



2.1.1.1 Reset Vector (0000H)

The microcontroller has 1-bite long system reset vector (000H). It has the following 3 types of resets:

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

After either of these resets occurs, the program will restart execution from 000H and the system registers will both be recovered to their default values. The system reset method can be determined by the contents of the PD and TO flags in the STATUS register. The following program demonstrates how to define the reset vector in FLASH.

Example: Define the reset vector

```
        ORG     000H        ; System reset vector
        JP      START
        ORG     0010H       ; User program start
START:
        ...                 ; User Programs
        ...
        END                 ; End of program
```

2.1.1.2 Interrupt Vector

The interrupt vector address is 004H. Once there is an interrupt response, the current value of the program counter PC is stored in the stack buffer and jumps to 004H to start the execution of the interrupt service program. All interrupts will enter 004H, and it will be up to the user to decide which interrupt to execute based on the bits in the interrupt request flag bit register. The following example program illustrates how to write the interrupt service program.

Example: Define the interrupt vector and place the interrupt program after the user program

```
        ORG     000H        ; System reset vector
        JP      START
        ORG     004H        ; User program start
INT_START:
        CALL    PUSH        ; Save ACC and STATUS
        ...                 ; User interrupt program
        ...
INT_BACK:
        CALL    POP         ; Return ACC and STATUS
        RETI                ; Interrupt return
START:
        ...                 ; User Programs
        ...
        END                 ; End of program
```

Note: Since the microcontroller does not provide specific pop and push instructions, the user needs to protect the interrupt by himself.

Example: interrupt-in protection

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

Example: interrupt-out restore

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

2.1.1.3 Jump Table

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

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

Example: Example of a correct multi-address jump program

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

Example: Example of an incorrect multi-address jump program

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

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

2.1.2 Data Memory

SC8F2790 Data Memory List

Address		Address		Address		Address	
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
WPUB	08H	OSCCON	88H	----	108H	----	188H
IOCB	09H	----	89H	----	109H	----	189H
PCLATH	0AH	PCLATH	8AH	----	10AH	----	18AH
INTCON	0BH	INTCON	8BH	----	10BH	----	18BH
PIR1	0CH	----	8CH	----	10CH	----	18CH
PIE1	0DH	----	8DH	----	10DH	----	18DH
PWMD23H	0EH	----	8EH	----	10EH	----	18EH
PWM01DT	0FH	----	8FH	----	10FH	----	18FH
PWM23DT	10H	----	90H	----	110H	----	190H
TMR2	11H	PR2	91H	----	111H	----	191H
T2CON	12H	----	92H	----	112H	----	192H
PWMCON0	13H	----	93H	----	113H	----	193H
PWMCON1	14H	----	94H	----	114H	----	194H
PWMTL	15H	----	95H	----	115H	----	195H
PWMTH	16H	----	96H	----	116H	----	196H
PWMD0L	17H	----	97H	----	117H	----	197H
PWMD1L	18H	OPA0CON	98H	----	118H		198H
PWMD2L	19H	OPA0ADJ	99H	----	119H		199H
PWMD3L	1AH	----	9AH	----	11AH		19AH
PWMD4L	1BH	----	9BH	----	11BH		19BH
PWMD01H	1CH	ADCON1	9CH	----	11CH	- - -	19CH
PWMCON2	1DH	ADCON0	9DH	----	11DH		19DH
----	1EH	ADRESL	9EH	----	11EH		19EH
----	1FH	ADRESH	9FH	----	11FH		19FH
General-Purpose Registers 96 bytes	20H 6FH	General-Purpose Registers 80 bytes	A0H EFH	General-Purpose Registers 80 bytes	120H 16FH	----	1A0H 1EFH
	70H -- 7FH	Fast memory space 70H-7FH	F0H -- FFH	Fast memory space 70H-7FH	170H -- 17FH	Fast memory space 70H-7FH	1F0H -- 1FFH
BANK0		BANK1		BANK2		BANK3	

The data memory consists of 512 x 8 bits and is divided into two functional compartments: special function registers and general-purpose data memory. Most of the data memory are readable/writeable, but some are read-only. The special function registers are addressed from 00H-1FH and 80H-9FH.

SC8F2790 Special Function Register Summary Bank0

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
00H	INDF	Look-up for this unit will use FSR, not physical register.								xxxxxxxx
01H	TMR0	TIMER0 data register								xxxxxxxx
02H	PCL	Lower bit of program counter								00000000
03H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
04H	FSR	Indirect data memory address pointer								xxxxxxxx
05H	PORTA	----	----	----	----	----	RA2	RA1	RA0	-----xxx
06H	PORTB	----	----	----	----	----	RB2	RB1	RB0	-----xxx
07H	WPUA	----	----	----	----	----	WPUA2	WPUA1	WPUA0	-----000
08H	WPUB	----	----	----	----	----	WPUB2	WPUB1	WPUB0	-----000
09H	IOCB	----	----	----	----	----	IOCB2	IOCB1	IOCB0	-----000
0AH	PCLATH	----	----	----	----	Write buffer for the high 4 bits of the program counter				-----0000
0BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
0CH	PIR1	----	----	----	----	----	PWMIF	TMR2IF	ADIF	-----000
0DH	PIE1	----	----	----	----	----	PWMIE	TMR2IE	ADIE	-----000
0EH	PWMD23H	----	----	PWMD3[9:8]		----	----	PWMD2[9:8]		---00--00
0FH	PWM01DT	----	----	PWM01 dead-time						--00000
10H	PWM23DT	----	----	PWM23 dead-time						--00000
11H	TMR2	TIMER2 module register								00000000
12H	T2CON	CLK_SEL	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	00000000
13H	PWMCON0	CLKDIV[2:0]			PWM4EN	PWM3EN	PWM2EN	PWM1EN	PWM0EN	00000000
14H	PWMCON1	PWMIO_SEL[1:0]		PWM2DTEN	PWM0DTEN	----	----	DT_DIV[1:0]		0000--00
15H	PWMTL	Lower bit of PWM period register								00000000
16H	PWMTH	Higher bit of PWM period register								00000000
17H	PWMD0L	Lower bit of PWM0 duty register								00000000
18H	PWMD1L	Lower bit of PWM1 duty register								00000000
19H	PWMD2L	Lower bit of PWM2 duty register								00000000
1AH	PWMD3L	Lower bit of PWM3 duty register								00000000
1BH	PWMD4L	Lower bit of PWM4 duty register								00000000
1CH	PWMD01H	----		PWMD1[9:8]		----		PWMD0[9:8]		---00--00
1DH	PWMCON2	----	----	----	PWM4DIR	PWM3DIR	PWM2DIR	PWM1DIR	PWM0DIR	----00000
1EH	----	----	----	----	----	----	----	----	----	--------
1FH	----	----	----	----	----	----	----	----	----	--------

SC8F2790 Special Function Register Summary Bank1

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
80H	INDF	Look-up for this unit will use FSR, not physical register.								xxxxxxxx
81H	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	11111011
82H	PCL	Low byte of the program counter (PC)								00000000
83H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
84H	FSR	Indirect data memory address pointer								xxxxxxxx
85H	TRISA	----	----	----	----	----	TRISA2	TRISA1	TRISA0	-----111
86H	TRISB	----	----	----	----	----	TRISB2	TRISB1	TRISB0	-----111
87H	WPDB	----	----	----	----	----	WPDB2	WPDB1	WPDB0	-----000
88H	OSCCON	----	IRCF2	IRCF1	IRCF0	----	----	SWDTEN	----	-101-0-
89H	----	----	----	----	----	----	----	----	----	--------
8AH	PCLATH	----	----	----	----	Write buffer for the high 4 bits of the program counter				-----0000
8BH	INTCON	GIE	PEIE	T01E	INTE	RBIE	T0IF	INTF	RBIF	00000000
91H	PR2	TIMER2 cycle register								11111111
97H	----	----	----	----	----	----	----	----	----	--------
98H	OPA0CON	OPA0EN	OPA0O	OPA0_CMP	OPA0_ADC	----	----	----	OPA0FT	0000 - 1
99H	OPA0ADJ	OPA0OUT	OPA0COFM	OPA0CRS	OPA0ADJ[4:0]					00010000
9CH	ADCON1	ADFM	----	----	----	----	LDO_EN	LDO_SEL[1:0]		0----000
9DH	ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/ $\overline{\text{DONE}}$	ADON	00000000
9EH	ADRESL	Lower bit of A/D result register								xxxxxxxx
9FH	ADRESH	Higher bit of A/D result register								xxxxxxxx

2.2 Addressing Mode

2.2.1 Direct Addressing

The RAM is operated through the accumulator (ACC).

Example: The value of ACC is given to 30H register

```
LD      30H,A
```

Example: The value of 30H register is given to ACC

```
LD      A,30H
```

2.2.2 Immediate Addressing

Passes the immediate count to the accumulator (ACC).

Example: Immediately count 12H to ACC

```
LDIA    12H
```

2.2.3 Indirect Addressing

The data memory can be addressed directly or indirectly. Indirect addressing is available through the INDF register, which is not a physical register. When INDF is accessed, it will be addressed based on the value in the FSR register (lower 8 bits) and the IRP bit of the STATUS register (9th bit) and will point to the register at that address, so with the IRP bits of the FSR register and STATUS register set, the INDF register can be accessed as a target register. An indirect read of INDF (FSR=0) will result in 00H. An indirect write to the INDF register will result in a null operation. The following example illustrates the use of indirect addressing in a program.

Example: Application of FSR and INDF

```
        LDIA    30H
        LD      FSR,A           ;Indirect address pointer to 30H
        CLRB    STATUS,IRP      ;Pointer 9th bit is cleared to zero
        CLR     INDF            ;Clear INDF is actually clearing the 30H address
                                RAM pointed by FSR
```

Example: Example of indirectly addressed clear RAM (20H-7FH).

```
        LDIA    1FH
        LD      FSR,A           ;Indirect address pointer to 1FH
        CLRB    STATUS,IRP
LOOP:
        INCR    FSR             ;Address plus 1, initial address is 30H
        CLR     INDF            ;Clear the address pointed to by FSR
        LDIA    7FH
        SUBA    FSR
        SNZB    STATUS,C        ;Keep clearing until FSR address is 7FH
        JP      LOOP
```

2.3 Stacks

The stack buffer of the chip has 8 levels. The stack buffer is neither part of the data memory nor part of the program memory, and it can neither be read nor written. Operations on it are performed via the stack pointer (SP), which also cannot be read out or written to. When the system is reset, the stack pointer will point to the top of the stack. When the program counter (PC) value is added into the stack buffer when subroutine calls and interrupts occur, it is necessary to return the value to the program counter (PC) when the interrupt or subroutine returns, the diagram below illustrates how this works:

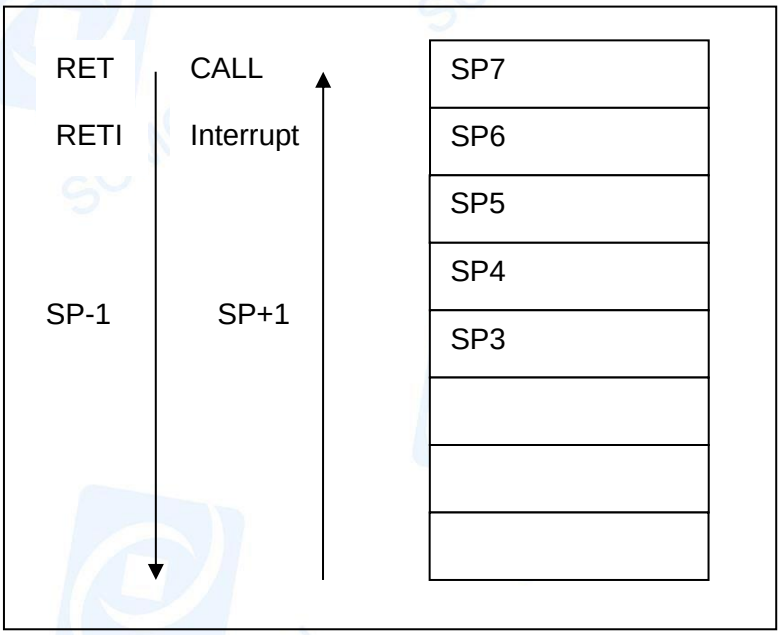


Figure 2-1: How the stack buffer works

The use of the stack buffer will follow the principle of "first in, first out".

Note: The stack buffer has only 8 levels, if the stack is full and a non-maskable interrupt occurs then only the interrupt flag bit will be logged and the interrupt response will be suppressed. The interrupt will not be responded to until the stack pointer is decremented. This feature prevents interrupts from overflowing the stack. Similarly, if the stack is full and a subroutine call occurs then the stack will overflow. The first contents of the stack will be lost and only the last 8 return addresses will be retained, so users should be aware of this when writing programs to prevent them from derailing.

2.4 Accumulator (ACC)

2.4.1 Overview

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 also controls the status bits (in STATUS register), which are used to indicate the status of the operation result.

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

2.4.2 ACC Applications

Example: Using ACC for data transfer

```
        LD      A,R01       ;Assign the value of register R01 to ACC
        LD      R02,A       ;Assign the value of ACC to register R02
```

Example: Using ACC to immediately address the target operand

```
        LDIA    30H         ;Assign 30H to ACC
        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: Using 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: Using 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)

The STATUS register is shown in the following table and contains:

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

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

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

Program status register STATUS (03H)

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

Bit7 IRP: Register memory select bit (for indirect addressing).Bank2 and
1= Bank3 (100h-1FFh).
0= Bank0 and Bank1 (00h-FFh).

Bit6~Bit5 RP[1:0]: The memory area selection bit.
00: Select Bank 0.
01: Select Bank 1.
10: Select Bank 2.
11: Select Bank 3.

Bit4 TO: Timeout bits.
1= Power on or CLRWDT instructions or STOP instructions.
0= A WDT timeout occurred.

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

Bit2 Z: Results in zero bits.
1= Result is zero.
0= Result is not zero.

Bit1 DC: Carry bit.
1= When carry happens to higher bits or no borrow happens in Lower 4 bits in the result.
0= When no carry happens to higher bits or borrow happens in Lower 4 bits in the result.

Bit0 C: Carry/borrow bit.
1= When carry happens at the highest bit or no borrow happens;
0= When no carry happens at the highest bit or borrow happens

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

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

Table of events affectingTO/PD

TO	PD	Reason for resetting
0	0	WDT overflow wakes upMCU
0	1	WDT overflow non-sleep state
1	1	Power on

Status of TO/PD after reset

2.6 Pre-scaler (OPTION_REG)

The OPTION_REG register is a read/write register that includes various control bits for configuring.

- TIMER0/WDT pre-scaler.
- TIMER0.
- PORTB pull-up resistor control.

Pre-scaler OPTION_REG(81H)

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

Bit7 RBPU: PORTB pull-up enable bit.
1= Prohibit PORTB pull-up.
0= The PORTB pull-up is enabled by the respective latch value of the port.

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

Bit5 T0CS: TIMER0 clock source select bit.
0= Internal instruction cycle clock (F_{SYS} /4).
1= The jump edge on the T0CKI pin.

Bit4 T0SE: TIMER0 clock source edge select bit.
0= Increments when the T0CKI pin signal jumps from low to high.
1= Increments when the T0CKI pin signal jumps from high to low.

Bit3 PSA: Pre-scaler assignment bit.
0= The pre-scaler is assigned to the TIMER0 module.
1= The pre-scaler is assigned to the WDT.

Bit2~Bit0 PS2~PS0: Pre-assigned parameter configuration bits.

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

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

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

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

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

```
CLRB    INTCON,GIE          Turn off the general interrupt enable bit toavoid
                            entering the interrupt program whenexecuting the
                            following specific timings
CLRB    STATUS,6
SETB    STATUS,5            Select BANK1
LDIA    B'00000111'
ORR     OPTION_REG,A        ;Pre-scaler value set to maximum
CLRB    STATUS,5            Select BANK0
CLR     TMR0                ;TMR0 cleared
SETB    STATUS,5            Select BANK1
SETB    OPTION_REG,PSA      ;Set pre-scaler assignment to WDT
CLRWDT                      ;Clear WDT
LDIA    B'xxxx1xxx'         ;Set the new pre-scaler
LD      OPTION_REG,A
CLRWDT                      ; Clear WDT
SETB    INTCON,GIE          If the program needs to use interrupts, turnthe
                            general enable bit back on here
```

To switch the pre-scaler from being assigned to the WDT to being assigned to the TIMER0 module, the following instruction should be executed

```
CLRWDT                      ; Clear WDT
LDIA    B'00xx0xxx'         ;Set the new pre-scaler
LD      OPTION_REG,A
```

Note: To enable TIMER0 to obtain a 1:1 pre-scaler ratio configuration, assign the pre-scaler to the WDTby setting PSA position 1 of the selection register.

2.7 Program Counter (PC)

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

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

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

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

The PC values for several special cases are given, and the diagram is shown in Figure 2-2.

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

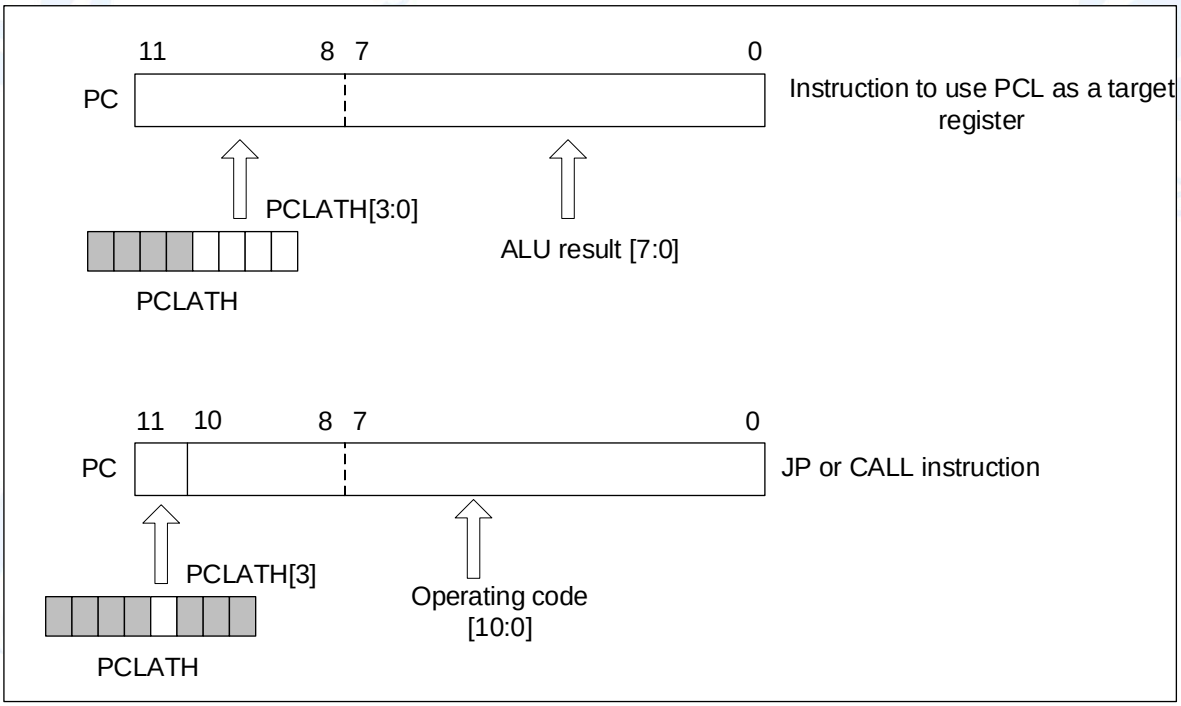


Figure 2-2: Principle of operating the PC in different situations

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

```
        ORG00H
                                        The target address LABEL1 is located at address 300H, and the
        JP      LABEL1                  current PCLATH value is 00H, which is within thesame 2K
                                        range, so there is no need to change the
                                        PCLATH value before executing JP instruction
        ...
    ORG300H
LABEL1:
        LDIA    08H                     The target address LABEL2 is located at address 900H, and the
        LD      PCLATH,A                current PCLATH value is 00H, which is not in thesame 2K
        JP      LABEL2                  range, so you need to assign a value to PCLATH before
                                        executing JP instruction
        ...
ORG     7FEH
LABLE4:
        NOP                             ;7FEH
        NOP                             ;7FFH
        NOP                             ;800H
        LDIA    08H                     The target address LABEL5 is located at address 880H, the
        LD      PCLATH,A                current PCLATH value is 00H (the program runs normally, when
        JP      LABLE5                  the PC changes from 7FFH to 800H, the PCLATH value will not
                                        change with it), which is not in thesame 2K range, so before
                                        executing JP instruction, you
                                        need to assign a value to PCLATH first
        ...
    ORG880H
LABLE5:
        NOP
        RET
        ...
    ORG900H
LABLE2:
        NOP
                                        The target address LABEL3 is located at address E00H and the
        CALL    LABLE3                  current PCLATH value is 08H, which is within thesame 2K
                                        range, so there is no need to change the
                                        PCLATH value before executing the CALL instruction
        LDIA    00H                     The target address LABEL4 is located at address 7FEH, the
        LD      PCLATH,A                current PCLATH value is 08H, which is not in the same 2K
        CALL    LABLE4                  range, so before executing the CALL instruction,
                                        you need to assign a value to PCLATH first
        NOP
        ...
        ...
    ORG0E00H
LABLE3:
        NOP
        RET
        ...
```

2.8 Watchdog Timer (WDT)

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

2.8.1 WDT Cycle

WDT and TIMER0 share the same 8-bit pre-scaler. After all resets, the WDT overflow period is 18ms, if you need to change the WDT period, you can set the OPTION_REG register. the WDT overflow period will be affected by the ambient temperature, supply voltage and other parameters.

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

Notes:

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

2.8.2 Watchdog Timer Control

SWDTEN: Software enables or disables the watchdog timer bit.
1= Enables WDT.
0= WDT (reset value) is disabled.

Notes.

1. SWDTEN is located in OSCCON register Bit1.
2. If the WDT configuration bit = 1 in CONFIG, WDT is always enabled, independent of the state of the SWDTEN control bit. If the WDT configuration bit in CONFIG = 0, WDT can be enabled or disabled using the SWDTEN control bit.

3. System Clock

3.1 Overview

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

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

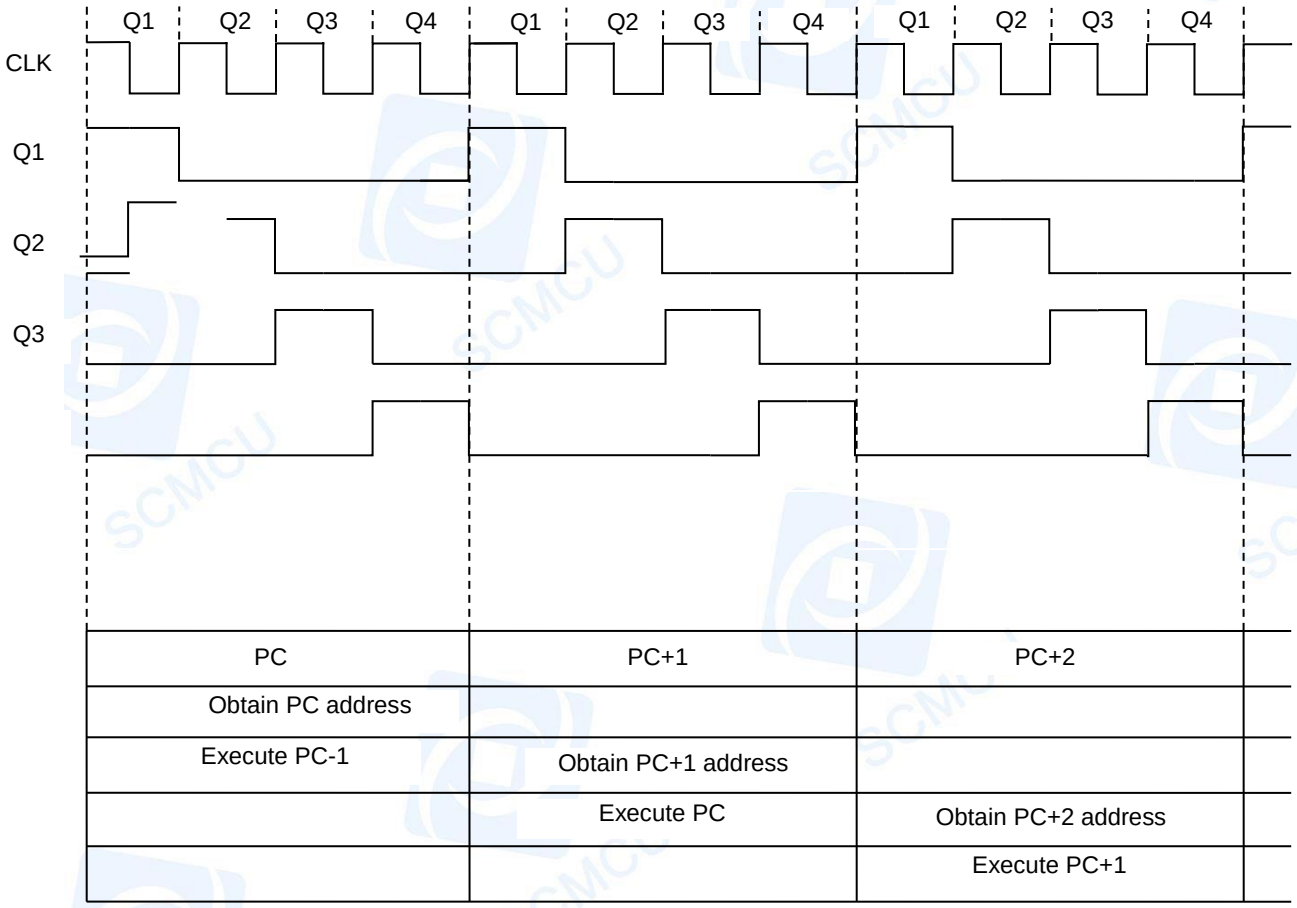


Figure 3-1: Clock and Instruction Cycle Timing Diagram

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

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

3.2 System Oscillator

The chip has only internal RC oscillation.

3.2.1 Internal RC oscillation

The default oscillation mode of the chip is internal RC oscillation with an oscillation frequency of 8MHz, which can be set via the OSCCON register.

3.3 Reset Time

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

3.4 Oscillator Control Register

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

Oscillator control register OSCCON(88H)

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

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

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

4. Reset

The chip can be reset in 3 ways as follows.

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

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

Any kind of reset situation requires a certain response time, and the system provides a cmpleted 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 provide the system clock.
- Execute the program: the power-on ends and the program starts to run.

4.2 Power Off Reset

4.2.1 Power-Off Reset Overview

A power-off reset addresses situations in which the system voltage dips due to external factors (e.g., disturbances or changes in external loads). Voltage dips may enter the system dead zone, which means that the power supply cannot meet the minimum operating voltage requirements of the system.

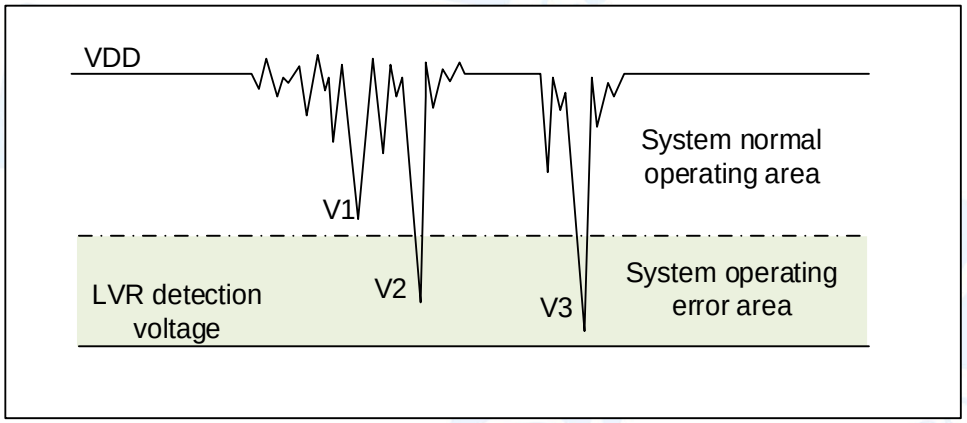


Figure 4-1: Power-off Reset

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

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

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

As shown above, the system normal operating voltage is generally higher than the system reset voltage, while the reset voltage is determined by the low voltage detection (LVR) level. When the system

execution speed increases, the system minimum operating voltage also increases accordingly. However, as the system reset voltage is fixed, there will be a voltage region between the system minimum operating voltage and the system reset voltage where the system cannot work normally and will not reset. This area is known as the dead zone.

4.2.2 Improvement of Power-Off Reset

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

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

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 slope of voltage drop

This method is used under AC. Voltage drops slowly under AC and cause the system to stay longer at the dead zone. If the system is power on at this moment, errors 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 protection setting for the system. In the normal state, the watchdog timer is cleared to zero by the program. If something goes wrong, the system is in an unknown state and the watchdog timer overflows, at which point the system resets. After the watchdog reset, the system reboots into the normal state.

The timing of the watchdog reset is as follows:

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

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

5. Sleep Mode

5.1 Enter Sleep Mode

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

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

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

5.2 Awaken from Sleep Mode

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

1. Watchdog timer awake (WDT forced enable).
2. 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 allow position 1 (allowed) must be set. The awaken is not related to the GIE bit. If the GIE bit is cleared (forbidden), the device will continue to execute the instruction following the STOP instruction. If the GIE bit is set to 1 (allowed), the device executes the instruction following 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 place a NOP instruction after the STOP instruction. The WDT will all be cleared when the device awakens from sleep mode, regardless of the reason for awakening.

5.3 Interrupt Awakening

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

- If an interrupt is generated before the STOP instruction is executed, then the STOP instruction will be executed as a NOP instruction. Therefore, WDT and its pre-scaler and post-scaler (if enabled) will not be cleared. At the same time, the TO bit will not be set to 1 and the PD will not be cleared.
- If an interrupt is generated during or after the execution of the STOP instruction, the device will beimmediately 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 obtain a smaller sleep current, please check all I/O status at first. If suspended I/O port is required by user, set all suspended ports as output to make sure each I/O has a fixed status and avoid increasing sleep current when I/O is input; turn off AD and other peripherals modes; WDT functions can be turned off to decrease the sleep current.

Example: Procedures for entering sleep mode

```
SLEEP_MODE.
        CLR     INTCON          ;Shutdown interrupt enable
        LDIA    B'0000000000'
        LD      TRISA,A
        LD      TRISB,A         ;All I/O are set to output ports
        LD      TRISC,A
        LD      TRISE,A
        ...                     ;Turn off other functions
        LDIA    0A5H
        LD      SP_FLAG,A       ;Set sleep status memory register (user-defined)
        CLRWDT                  ;Zeroing WDT
        STOP                    ;Execute STOP command
```

5.5 Sleep Mode Wake-Up Time

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

System master frequency clock source	System clock division selection (IRCF<2:0>)	Sleep awaken waiting time T_{WAIT}
Internal high-speed RC oscillation (F_{OSC})	$F_{SYS}=F_{OSC}$	$T_{WAIT}=1024*1/F_{OSC}$
	$F_{SYS}= F_{OSC}$ /2	$T_{WAIT}=1024*2/F_{OSC}$
	…	…
	$F_{SYS}= F_{OSC}$ /64	$T_{WAIT}=1024*64/F_{OSC}$
Internal low-speed RC oscillation ($F_{LFINTOSC}$)	----	$T_{WAIT}=8/F_{LFINTOSC}$

6. I/O ports

The chip has two I/O ports: PORTA, PORTB (Max. of 6 I/O). The read/write port data registers can directly read/write these ports.

Port	Bit	Pin Description	I/O
PROTA	0	Schmitt trigger input, push-pull output, AN0, PWM output, external interrupt input, op-amp 0output	I/O
	1	Schmitt trigger input, push-pull output, AN1, PWM output, op-amp 0 positive input	I/O
	2	Schmitt trigger input, push-pull output, AN2, PWM output, op-amp 0 negative input	I/O
PORTB	0	Schmitt trigger input, push-pull output, AN13, programmed data in/out, oscillation input port, PWMoutput	I/O
	1	Schmitt trigger input, push-pull output, AN12, programmed clock input, oscillation output port,PWM output	I/O
	2	Schmitt Trigger Input, Push-Pull Output, AN11, PWM Output	I/O

<Table 6-1: General Overview of Port Configuration>

6.1 I/O Port Structure Diagram

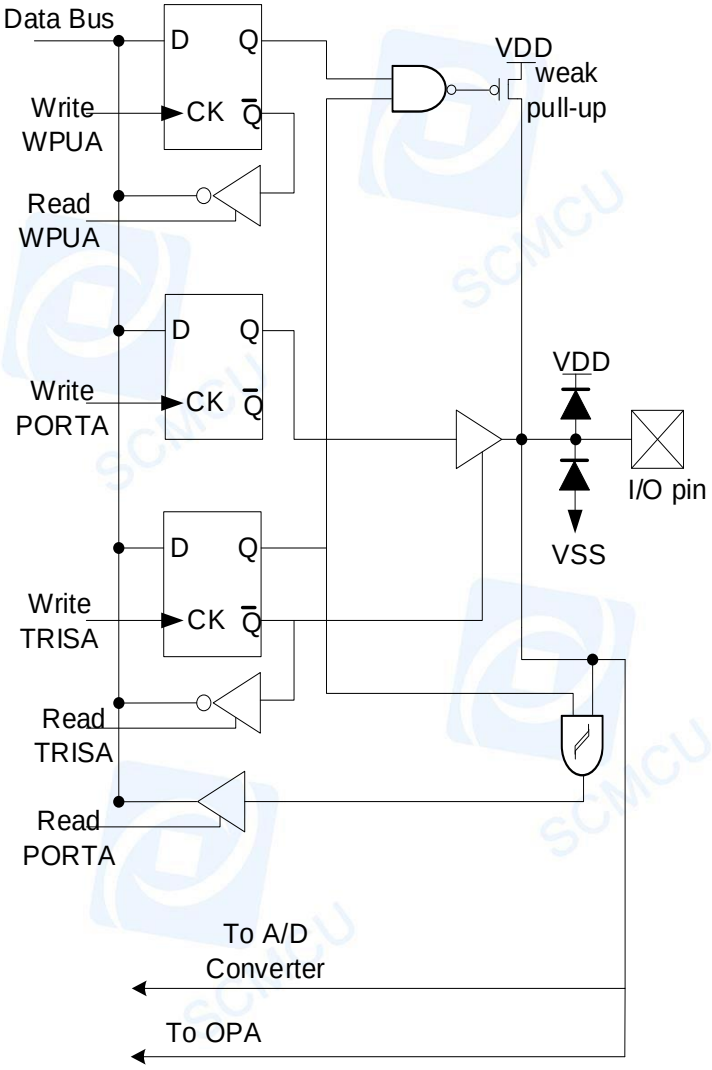


Figure 6-1: I/O Port Structure Diagram (1)

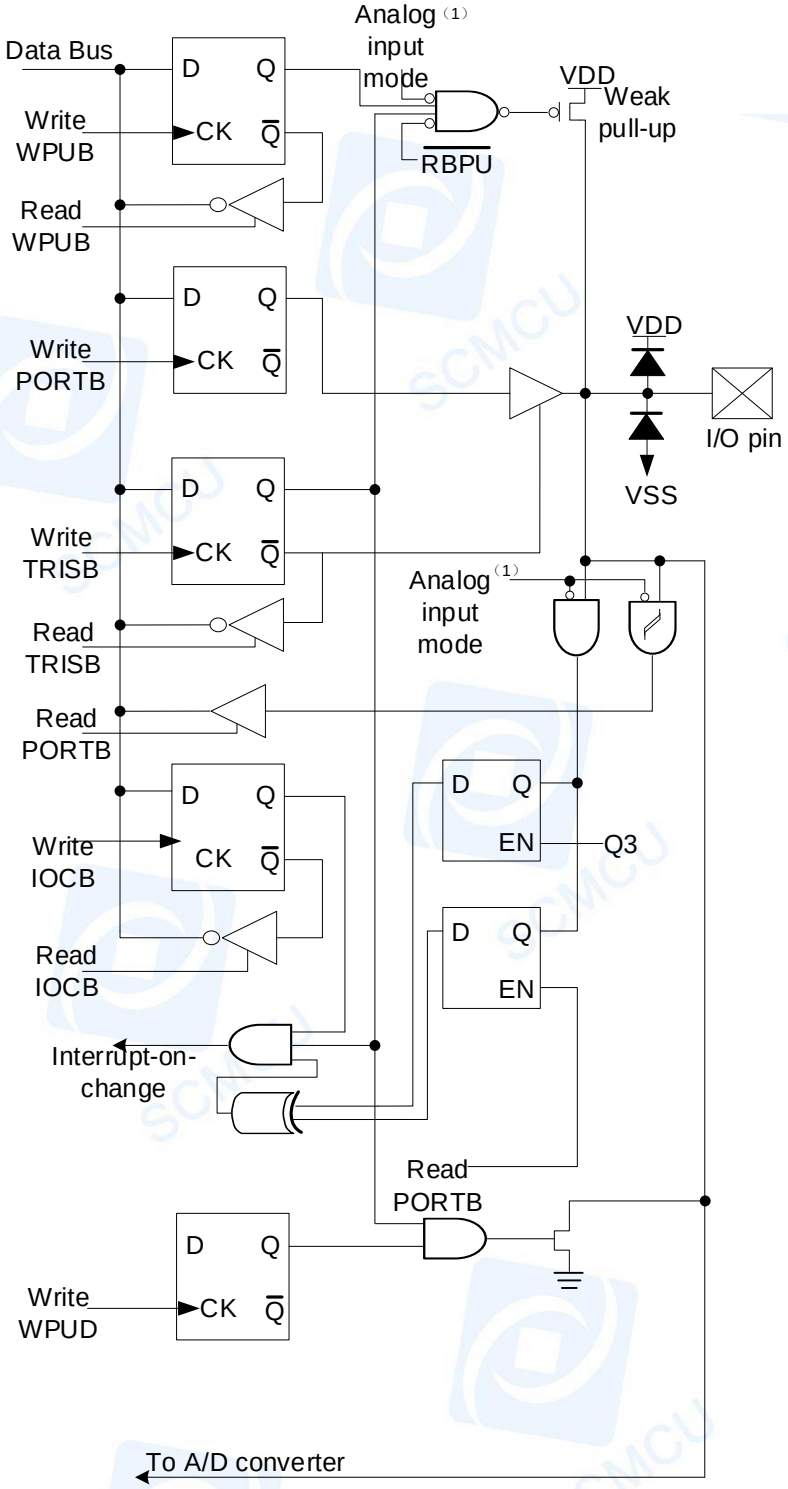


Figure 6-2: I/O Port Structure Diagram (2)

6.2 PORTA

6.2.1 PORTA Data and Direction Control

PORTA is a 6-bit bi-directional port. Its corresponding data direction register is TRISA. setting one bit of TRISA to position 1 (= 1) configures the corresponding pin as an input. Clearing a TRISA bit (= 0) configures the corresponding PORTA pin as an output.

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

The registers associated with the PORTA port are PORTA, TRISA, WPUA, etc.

PORTA Data Register PORTA (05H)

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

Bit7~Bit3 Not used.

Bit2~Bit0 PORTA<2:0>: 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	----	----	----	----	----	TRISA2	TRISA1	TRISA0
R/W	----	----	----	----	----	R/W	R/W	R/W
Reset value	----	----	----	----	----	1	1	1

Bit7~Bit3 Reserved Suggest writing 0

Bit2~Bit0 TRISA<2:0>: PORTA tri-state control bits.

1= PORTA pin is configured as an input (tri-state).

0= The PORTA pin is configured as an output.

Notes.

1. If the ADC channel on the pin is turned on, the input to the SMIT module for this IO port will be disabledand the value will always be 0.
2. To turn on AN0 channel, you need to set ADCON0.ADON=1 and ADCON0.CHS=0000. To turn on other ADC channels, simply set ADCON0.CHS to the corresponding channel

Example: Procedure for PORTA Port

```
LDIA    B'0000000100'    Set PORTA<1:0> as output port and PORTA<2> as input port.
LD      TRISA,A
LDIA    03H              PORTA<1:0> output high level
LD      PORTA,A          Since PORTA<2> is an input port, 0 or 1 does not matter.
```

6.2.2 PORTA Pull-Up Resistance

Each PORTA pin has an individually configurable internal weak pull-up. Control bits WPUA<2:0> enable or disable each weak pull-up. When a port pin is configured as an output, its weak pull-up is automatically disconnected.

PORTA pull-up resistor register WPUA(07H)

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

Bit7~Bit3 Not used.

Bit2~Bit0 WPUA<2:0>: Weak pull-up register bit.

1= Enables pull-up.

0= Disable pull-up.

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

6.3 PORTB

6.3.1 PORTB Data and Direction

PORTB is a 3-bit bi-directional port. The corresponding data direction register is TRISB. Set a bit in TRISB to 1 (=1) to make the corresponding PORTB pin as the input pin. Clearing a bit in TRISB (=0) will make the corresponding PORTB pin as the output pin.

Reading the PORTB register reads the 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 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 bit in the TRISB register remains set to 1. I/O pins that are configured as analog inputs always read 0.

The registers related to PORTB port are PORTB, TRISB, WPUB, WPDB, IOCB, etc.

PORTB data register PORTB(06H)

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

Bit7~Bit3 Not used

Bit2~Bit0 PORTB<2:0>: 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	----	----	----	----	----	TRISB2	TRISB1	TRISB0
R/W	----	----	----	----	----	R/W	R/W	R/W
Reset value	----	----	----	----	----	1	1	1

Bit7~Bit3 Reserved Suggest writing 0

Bit2~Bit0 TRISB<2:0>: PORTB tri-state control bit.

1= The PORTB pin is configured as an input (tri-state).

0= The PORTB pin is configured as an output.

Notes.

1. If the ADC channel on the pin is turned on, the input to the SMIT module for this IO port will be disabled and the value will always be 0.
2. To turn on AN0 channel, you need to set ADCON0.ADON=1 and ADCON0.CHS=0000.
   To turn on other ADC channels, simply set ADCON0.CHS to the corresponding channel.

Example: PORTB port procedure

```
CLR     PORTB       ;Clear data register
LDIA    B'00000011' ;Set PORTB<1:0>as an input port, and set others as output ports
LD      TRISB,A
```

6.3.2 PORTB Pull-Down Resistance

Each PORTB pin has an individually configurable internal weak pull-down. Control bits WPDB<2:0> enable or disable each weak pull-down. When a port pin is configured as an output, its weak pull-down is automatically cut off.

PORTB pull-down resistance register WPDB(87H)

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

Bit7~Bit3 Not used

Bit2~Bit0 WPDB<2:0>: Weak drop-down register bit.

1= Enables the drop-down.

0= Disables the drop-down.

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

6.3.3 PORTB Pull-Up Resistance

Each PORTB pin has an individually configurable internal weak pull-up. Control bits WPUB<2:0> enable or disable each weak pull-up. When a port pin is configured as an output, its weak pull-up is automatically disconnected. On power-on reset, weak pull-ups are disabled by the RBPU bit of the OPTION_REG register.

PORTB pull-up resistor register WPUB(08H)

08H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUB	----	----	----	----	----	WPUB2	WPUB1	WPUB0
R/W	----	----	----	----	----	R/W	R/W	R/W
Resetvalue	----	----	----	----	----	0	0	0

Bit7~Bit3 Not used

Bit2~Bit0 WPUB<2:0>: Weak pull-up register bit.

1= Enables pull-up.

0= Disables pull-up.

Notes.

1. To enable either pull-up alone, the overall RBPU bit of the OPTION_REG register must be cleared to zero.
2. If the pin is configured as an output or analog input, weak pull-ups are automatically disabled.

6.3.4 PORTB Interrupt on Change

All PORTB pins can be individually configured as interrupt-on-change pins. Control bit IOCB<2:0> allows or disables this interrupt function for each pin. The interrupt-on-change pins are disabled on power-on reset.

For pins that have allowed interrupt on change, the value on the pin is compared with the old value latched when PORTB was read last time, and if the two values do not match, the corresponding pin level has changed and the RBIF bit in the INTCON register will be set to 1.

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

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

The mismatch state can be ended by reading or writing PORTB.

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

PORTB interrupt on change register IOCB(09H)

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

Bit7~Bit3	Reserved	Suggest writing 0
Bit2~Bit0	IOCB<2:0>	Interrupt-on-change control bit for PORTB.
	1=	Allow interrupt on changes.
	0=	Disable interrupt on change.

6.4 I/O Usage

6.4.1 Write I/O Port

The chip's I/O port registers, like general-purpose registers, can be written by data transmission instructions,bit operation instructions, etc.

Example: Write I/O port program

```
LD      PORTA,A     The ACC value is assigned to the PORTA port.
CLRB    PORTB,1     PORTB.1 port set to zero
SET     PORTA       PORTA all output ports set to 1
SETB    PORTB,1     PORTB.1 port set to 1
```

6.4.2 Read I/O port

Example: Read I/O port program

```
LD      A,PORTA     The value of PORTA is assigned to ACC.
SNZB    PORTA,1     ;Determine if PORTA,1 port is 1, if it is 1, skip the next statement
SZB     PORTA,1     ;Determine whether PORTA,1 port is 0 or not, if it is 0, skip the next statement
```

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

6.5 Precautions for I/O Port Usage

When operating the I/O ports, the following should be noted.

1. When I/O is converted from output to input, it waits for a few instruction-cycle times for the I/O port to stabilize.
2. If there is an internal pull-up resistor, then the stabilization time of the internal level when the I/O is converted from output to input is related to the capacitor connected to the I/O port. The user should set the wait time according to the actual situation to prevent the I/O port from scanning the level by mistake.
3. When the I/O port is an input port, its input level should be between "VDD+0.7V" and "GND-0.7V". If the input port voltage is not within this range, the method shown below can be used.

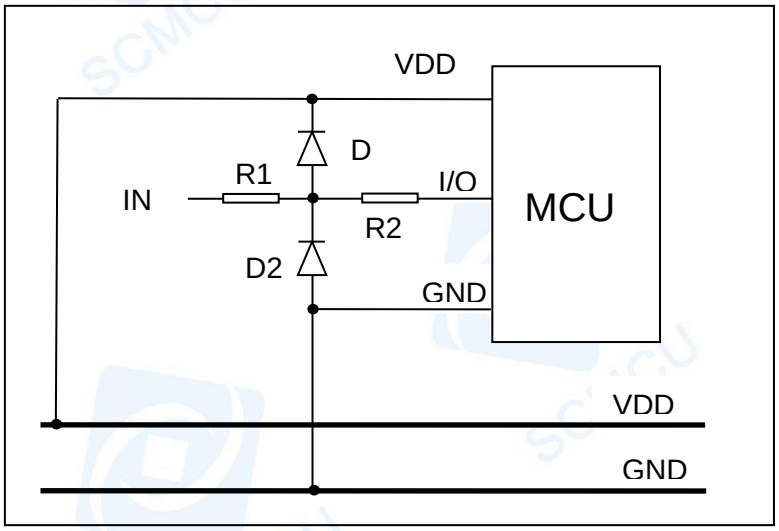


Figure 6-3: Input voltage Not in the Specified Range

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

7. Interrupt

7.1 Interrupt Overview

The chip has multiple interrupt sources as follows:

- TIMER0 overflow interrupt
- TIMER2 match interrupt
- PORTB interrupt on change
- INT interrupt
- AD Interrupt
- PWM 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 individual interrupt allow bits and global interrupt allow bits.

The global interrupt allows bit GIE (INTCON<7>) allows all unmasked interrupts when set to 1, and disables all interrupts when cleared to zero. Individual interrupts can be disabled by the corresponding allow bits in the INTCON and PIE1 registers. GIE is cleared to zero when reset.

Executing the "return from interrupt" instruction, RETI will exit the interrupt service program and set GIE position 1, thus re-allowing unmasked interrupts.

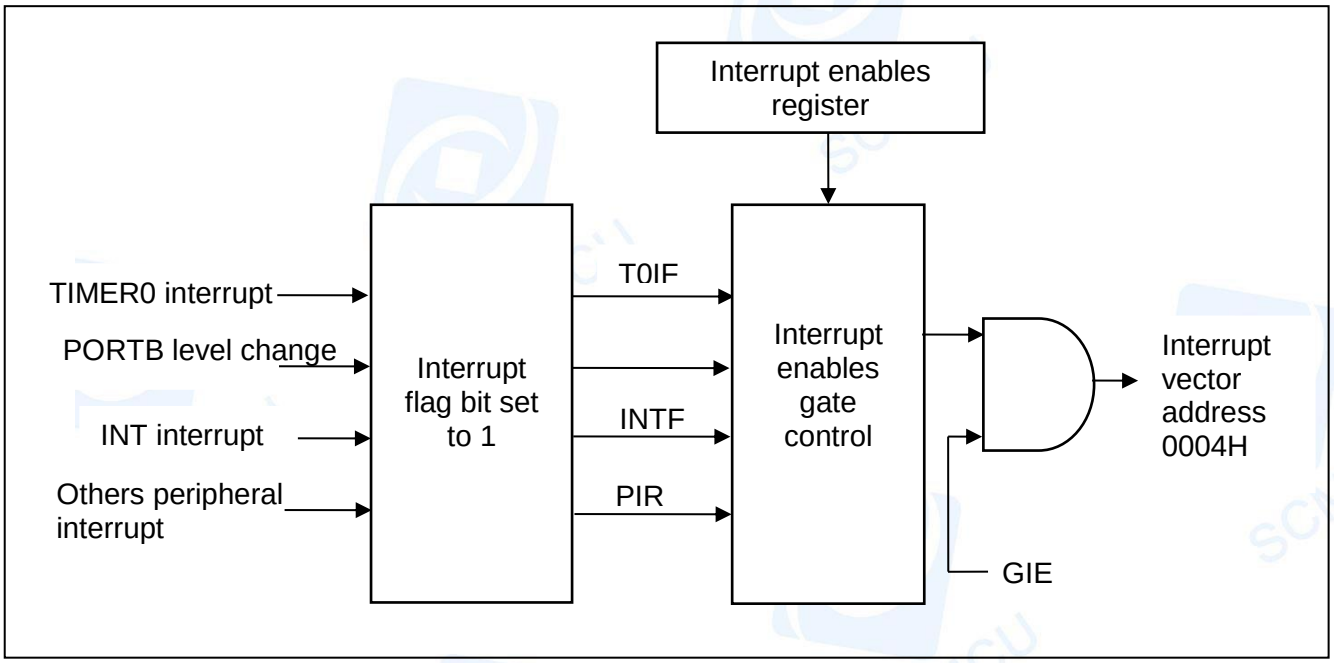


Figure 7-1: Diagram of Interruption

7.2 Interrupt Control Register

7.2.1 Interrupt Control Register

The interrupt control register INTCON is a readable/writeable register containing allowable and flag bits for TMR0 register overflow, interrupt on change PORTB port, etc.

When an interrupt condition is generated, the interrupt flag bit will be set to 1 regardless of the status of the corresponding interrupt allow bit or the global allow bit GIE (in the INTCON register.) The user software should ensure that the corresponding interrupt flag bit is cleared to zero 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
Resetvalue	0	0	0	0	0	0	0	0

Bit7 GIE: Global interrupt allow bit.
1= Allow all unmasked interrupts.
0= Disable all interruptions.

Bit6 PEIE: Peripheral interrupt allow bit.
1= Allow all unmasked peripheral interrupts.
0= Disable all peripheral interrupts.

Bit5 T0IE: TIMER0 overflow interrupt allow bit.
1= Allowing the TIMER0 interrupt.
0= Disable TIMER0 interrupt.

Bit4 INTE: INT external interrupt allow bit.
1= Allowing INT external interrupts.
0= Disable INT external interrupts.

Bit3 RBIE: PORTB interrupt on change allow bit (1).
1= Allowing PORTB interrupt on changes.
0= Disable PORTB interrupt on change.

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

Bit1 INTF: INT external interrupt flag bit.
1= The occurrence of an INT external interrupt (which must be cleared by software).
0= No INT external interrupt occurred.

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

Notes.
1. The IOCB register must also be enabled, and the corresponding port line must be set to the inputstate.
2. The T0IF bit is set to 1 when TMR0 is full to 0. Reset does not change TMR0 and should be initialized before clearing the T0IF bit to zero.

7.2.2 Peripheral Interrupt Enable Register

Peripheral interrupt allow register is PIE1. Before allowing any peripheral interrupt, the PEIE bit of the INTCON register must be set to 1.

Peripheral interrupt enables register PIE1(0DH)

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

Bit7~Bit3 Unused, read as 0.

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

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

7.2.3 Peripheral Interrupt Request Register

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

Peripheral interrupt request register PIR1(0CH)

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

Bit7~Bit3 Unused, read as 0.

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

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

Bit0 ADIF: AD converter interrupt flag bit.
1= Completion of AD conversion (must be cleared by software).
0= AD conversion is not completed or has not been started.

7.3 Protection Methods for Interrupt

After an interrupt request occurs and is responded to, 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 special in-stack save and out-stack restore instructions. Users need to protect the contents of ACC and STATUS by themselves to avoid possible program operation errors after the interrupt ends.

Example: Stack protection for ACC and STATUS

```
                ORG     0000H
                JP      START           ;User program start address
                ORG     0004H
                JP      INT_SERVICE     ;Interrupt service program
                ORG     0008H
START.
                ...
                ...
INT_SERVICE.
PUSH.                                   Interrupt service program entry, save ACC and STATUS
                LD      ACC_BAK,A       ; Save the value of ACC, (ACC_BAK needs to be customized)
                SWAPA   STATUS
                LD      STATUS_BAK,A    ;Save the value of STATUS, (STATUS_BAK needs to be customized)
                ...
                ...
POP.                                    ;Interrupt service program exit, restore ACC and STATUS
                SWAPA   STATUS_BAK
                LD      STATUS,A        ;Restore the value of STATUS
                SWAPR   ACC_BAK         ; Restore the value of ACC
                SWAPA   ACC_BAK
                RETI
```

7.4 Interrupt Priority and Multiple Interrupt Nesting

The priority of each interrupt of the chip is equal. When an interrupt is in progress, it will not respond to another interrupt. Only after the execution of the "RETI" instruction, it will respond to the next interrupt.

When multiple interrupts occur simultaneously, the MCU does not have a preset interrupt priority. Firstly, the priority of each interrupt must be pre-set; Secondly, the interrupt enable and interrupt control bits are used to control whether the system responds to the interrupt. In the program, the interrupt control bit and interrupt request flag must be detected.

8. TIMER0

8.1 TIMER0 Overview

TIMER0 consists of the following functions:

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

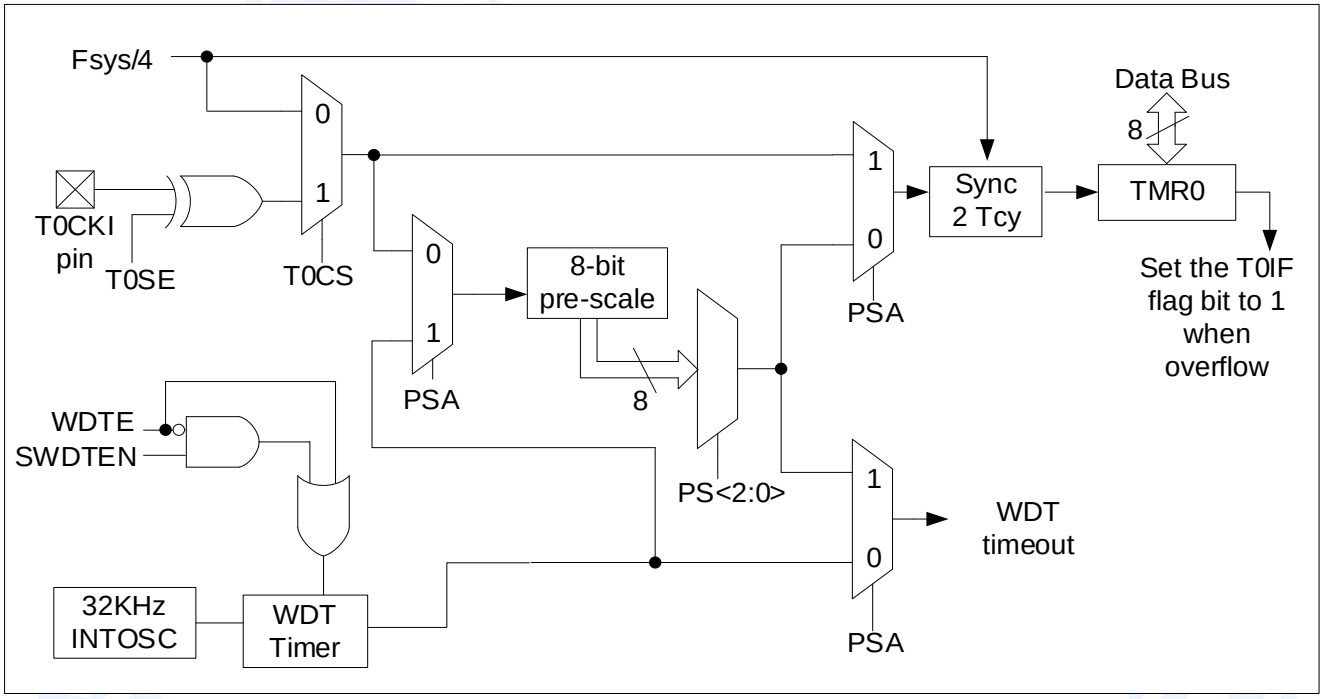


Figure 8-1: TIMER0/WDT Module Structure Diagram

Notes:

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

8.2 TIMER0 Operation Principle

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

8.2.1 8-bit Timer Mode

When used as a timer, the TIMER0 module will increment at each instruction cycle (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, incrementing will be disabled for the next two instruction cycles. The value of the write TMR0 register can be adjusted so that two instruction cycles of delay are counted when writing to TMR0.

8.2.2 8-bit Counter Mode

When used as a counter, the TIMER0 module will increment on each 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 position 1.

8.2.3 Software Programmable Pre-Scaler

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

The TIMER0 module has 8 pre-scaler options ranging from 1:2 to 1:256. The pre-scaler can be selected via the PS<2:0> bits of the OPTION_REG register. For the TIMER0 module to have a 1:1 pre-scaler ratio, the pre-scaler must be assigned to the WDT module.

The pre-scaler is not readable/writeable. When the pre-scaler is assigned to the TIMER0 module, all instructions written to the TMR0 register will clear the pre-scaler. When the pre-scaler is assigned to the WDT, the CLRWDT instruction will clear both the pre-scaler and the WDT.

8.2.4 Switching Pre-Scaler Between TIMER0 and WDT Modules

After assigning a pre-scaler to TIMER0 or WDT, an unintentional device reset may occur when switching the pre-scaler ratio. To change the pre-scaler from being assigned to TIMER0 to being assigned to a WDT module, the instruction sequence shown below must be executed.

Modify the pre-scaler (TMR0-WDT)

```
        CLRB    INTCON,GIE          Turn off the general interrupt enable bit to
                                    avoid entering the interrupt program when
                                    executing the following specific timings
        CLRB    STATUS,6
        SETB    STATUS,5            Select BANK1
        LDIA    B'0000000111'
        ORR     OPTION_REG,A        ;Pre-scaler set to maximum
        CLRB    STATUS,5            Select BANK0
        CLR     TMR0                ;TMR0 cleared
        SETB    STATUS,5            Select BANK1
        SETB    OPTION_REG,PSA      ;Set pre-scaler assignment to WDT
        CLRWDT                      ;WDT zeroing
        LDIA    B'xxxx1xxx'         ;Set the new pre-scaler
        LD      OPTION_REG,A
        CLRWDT                      ;WDT zeroing
        SETB    INTCON,GIE          If the program needs to use interrupts, turn
                                    the general enable bit back on here
```

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

Modify the pre-scaler (WDT-TMR0)

```
        CLRWDT                      ;WDT zeroing
        LDIA    B'00xx0xxx'         ;Set the new pre-scaler
        LD      OPTION_REG,A
```

8.2.5 TIMER0 Interrupt

The TIMER0 interrupt is generated when the TMR0 register overflows from FFh to 00h. The T0IF interrupt flag bit of the INTCON register is set to 1 each time the TMR0 register overflows, whether or not the TIMER0 interrupt is allowed. The T0IF bit must be cleared in software. The TIMER0 interrupt allowed bit is the T0IE bit of the INTCON register.

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

8.3 Registers Related to TIMER0

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

TMR0 is an 8-bit readable/writeable timer/counter and OPTION_REG is an 8-bit write-only register. The user can change the operating mode of TIMER0 by changing the value of OPTION_REG, etc. Please refer to 2.6 for the application of pre-scaler register (OPTION_REG).

8-bit Timer/Counter TMR0(01H)

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

OPTION_REG register (81H)

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

Bit7 RBPU: PORTB pull-up enable bit.
1= Prohibit PORTB pull-up.
0= The PORTB pull-up is enabled by the respective latch value of the port.

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

Bit5 T0CS: TMR0 clock source select bit. The
1= jump edge on the T0CKI pin.
0= Internal instruction cycle clock (F_{sys} /4).

Bit4 T0SE: TIMER0 clock source edge select bit.
1= Increments when the T0CKI pin signal jumps from high to low.
0= Increments when the T0CKI pin signal jumps from low to high.

Bit3 PSA: Pre-scaler assignment bit.
1= The pre-scaler is assigned to the WDT.
0= The pre-scaler is assigned to the TIMER0 module.

Bit2~Bit0 PS2~PS0: Pre-assigned 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

The TIMER2 module is an 8-bit timer/counter with the following features:

- 8-bit timer register (TMR2).
- 8-bit period register (PR2).
- Interrupt when TMR2 matches PR2.
- Software programmable pre-scaler ratios (1:1, 1:4 and 1:16).
- Software programmable post-scaler ratios (1:1 to 1:16).
- External 32.768KHz oscillation/4 can be selected as the clock source.

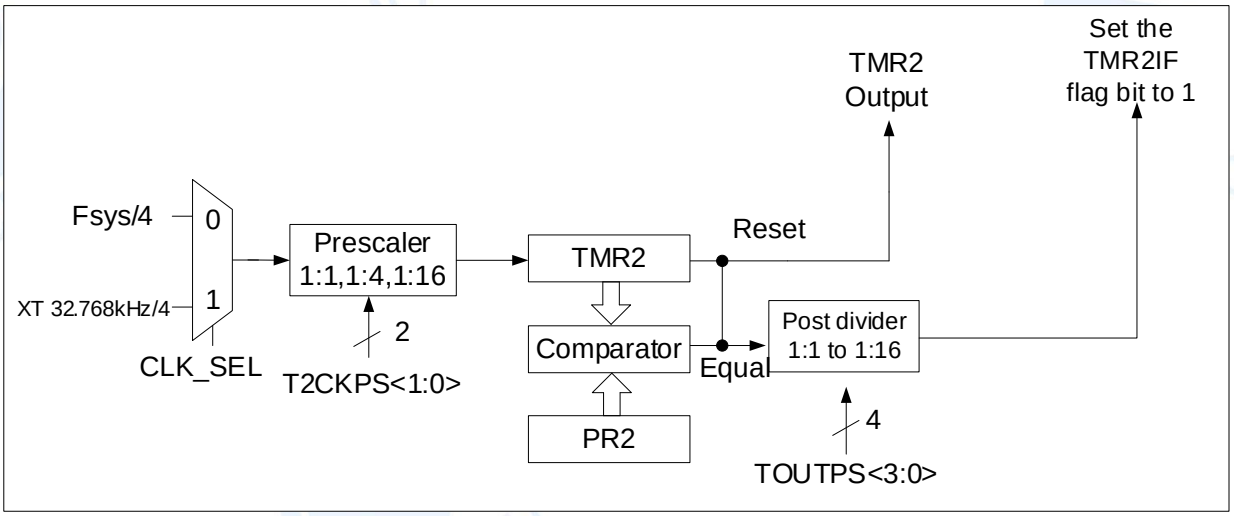


Figure 9-1: TIMER2 Block Diagram

9.2 TIMER2 Operation Principle

The input clock to the TIMER2 module is the system command clock (F_{sys}/4) or an external 32.768 kHz oscillation. The clock is input to the TIMER2 pre-scaler with the following dividing ratios available: 1:1, 1:4 or 1:16. The output of the pre-scaler will be used to increment the TMR2 register.

The values of TMR2 and PR2 are continuously compared to determine when they match. TMR2 will be incremented from 00h until it matches the value in PR2. When a match occurs, the following two events will occur.

- TMR2 is reset to 00h at the next incremental cycle.
- TIMER2 after the divider increment.

The matched output of TIMER2 and the PR2 comparator is then input to the TIMER2 post-scaler. The post-scaler has a choice of pre-scaler ratios from 1:1 to 1:16. The output of the TIMER2 post-scaler is used to make PIR1 The TMR2IF interrupt flag bit of the register be set to 1.

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

Enable TIMER2 by setting TMR2ON to 1 of T2CON register; and disable TIMER2 by clearing the TMR2ON bit to zero.

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

The pre-scaler and post-scaler counters are cleared when

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

Note: Writing T2CON will not clear TMR2 to zero.

9.4 Registers Related to TIMER2

There are 2 registers associated with TIMER2: the data memory TMR2 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
Resetvalue	X	X	X	X	X	X	X	X

TIMER2 control register T2CON(12H)

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

Bit7 CLK_SEL: Clock source selection.
1= Selection of external 32.768 kHz oscillation/4 as TMR2 clock source (sleep mode can continue counting).
0= Select the internal F_{SYS} /4 as the TMR2 clock source.

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

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

Bit1~Bit0 T2CKPS<1:0>: TIMER2 clock pre-scaler ratio selection bit.
00= A pre-scaler value of 1.
01= A pre-scaler value of 4.
1x= The pre-scaler value is 16.

10. Analog to Digital Conversion (ADC)

10.1 ADC Overview

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

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

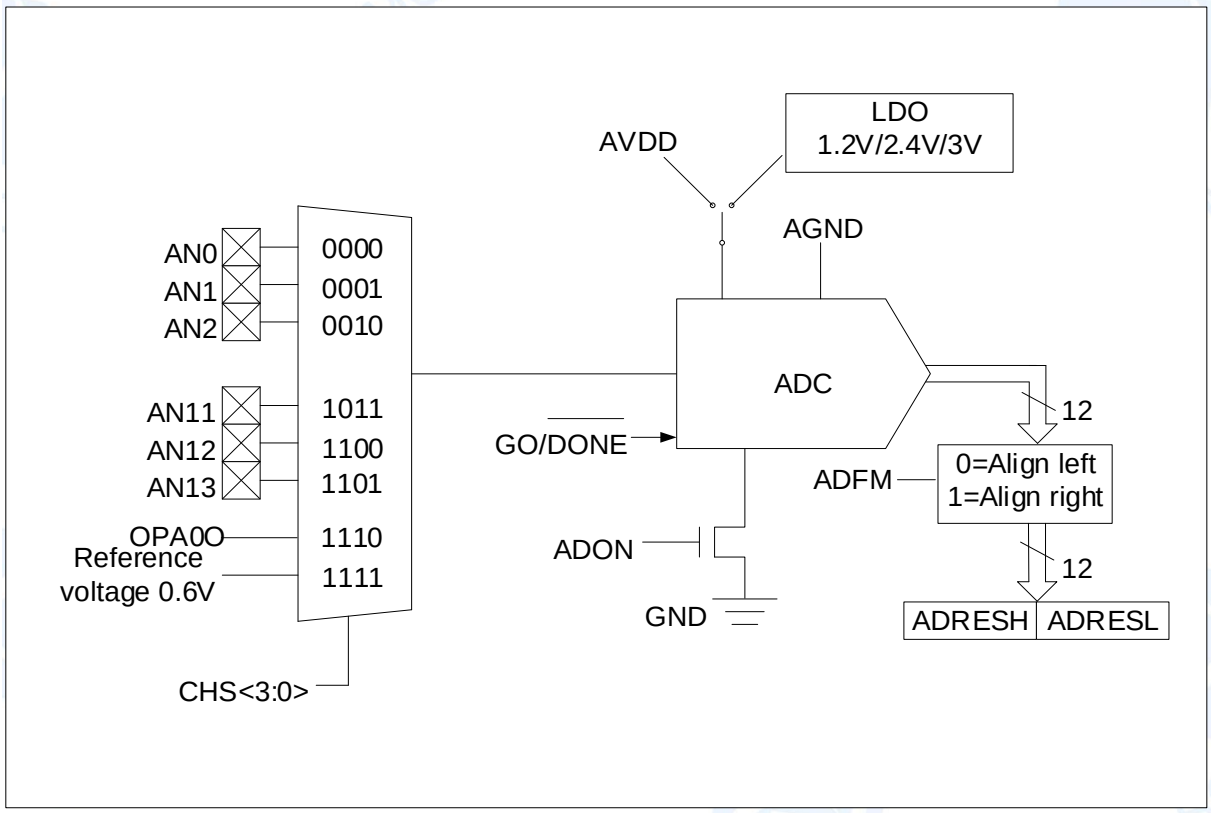


Figure 10-1: ADC Block Diagram

10.2 ADC Configuration

The following factors must be considered when configuring and using ADCs.

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

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 to 1. See the corresponding port section for more information.

Note: Applying analog voltages to pins defined as digital inputs may result in overcurrent in the input buffers.

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 Chapter "ADC Operation Principle ".

Notes.
1. If the ADC channel on the pin is turned on, the input to the SMIT module for this IO port will be disabled and the value will always be 0.
2. To turn on AN0 channel, you need to set ADCON0.ADON=1 and ADCON0.CHS=0000.
   To turn on other ADC channels, simply set ADCON0.CHS to the corresponding channel.

10.2.3 ADC Internal Reference Voltage

Chip built-in 0.6V reference voltage. When the reference voltage need to be detected, you should set ADCON0[5:2]bits to 1111, at the same time set ADCON1[2] bits to 0, and ADCON1[1] bits to 1.

10.2.4 ADC Reference Voltage

The reference voltage of the ADC can be provided by either the internal LDO output or the VDD and GND of the chip. The internal reference voltage is selectable from 1.2V/2.4V/3V.

When 1.2V is selected as the internal reference voltage, F_{RC} should be selected for the converter clock.

Note: When internal LDO is selected as the reference voltage, the maximum effective accuracy of ADC is only 8 bits. The lower the detection voltage, the higher the accuracy of the ADC obtained, and itis recommended that the input voltage be set to <1V.

10.2.5 Converter clock

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

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

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

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

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

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

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

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

10.2.6 ADC Interrupt

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

The interrupt can be generated regardless of whether the device is in operating mode or sleep mode. This interrupt can wake up the device if it is in sleep mode. When waking the device from sleep, the next instruction after the STOP instruction is always executed. If the user attempts to wake the device from hibernate mode and resume code execution in sequence, the global interrupt must be disabled. If global interrupts are allowed, the program will jump to the interrupt service program for execution.

10.2.7 Result Formatting

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

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

10.3 ADC Operation Principle

10.3.1 Start Conversion

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

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

10.3.2 Complete Conversion

When the conversion is complete, the ADC module will:

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

10.3.3 Termination of Conversion

If the conversion must be terminated before it is completed, the GO/$\overline{DONE}$ bits can be cleared by software. The ADRESH:ADRESL registers will not be updated with the results of the outstanding analog-to-digital conversion. Therefore, the ADRESH:ADRESL registers will retain the value obtained from the last conversion. In addition, after the AD conversion is terminated, a delay of 2 TADs must pass before the next acquisition can begin. After the delay, the input signal of the selected channelwill be started automatically.

Note: A device reset will force all registers into a reset state. Therefore, a reset will shut down the ADC module and terminate any pending conversions.

10.3.4 ADC Operation Principle in Sleep Mode

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

10.3.5 AD Conversion Steps

An example of analog-to-digital conversion using an ADC is given in the following steps:

1. Port configuration.
   - Prohibit pin-out drivers (see TRIS register).
   - Configure the pins as analog input pins.
2. Configure the ADC module.
   - Selection of ADC reference voltage.
     (When the reference voltage is switched from VDD to the internal LDO, a delay of 100us or more is required before ADC conversion can be performed)
   - Selection of the ADC conversion clock.
   - Selection of ADC input channels.
   - Select the format of the results.
   - Start the ADC module.
3. Configure ADC interrupts (optional).
   - Clear the ADC interrupt flag bit.
   - Allow ADC interrupts.
   - Allowing peripheral interrupts.
   - Allows global interrupts.
4. Wait for the required acquisition time.
5. Set GO/$\overline{DONE}$ to 1 to start the conversion.
6. Wait for the ADC conversion to finish by one of the following methods.
   - Query GO/$\overline{DONE}$ bits.
   - Wait for ADC interrupt (allow interrupt).
7. Read ADC results.
8. Clear the ADC interrupt flag bit to zero (this operation is required if interrupts are allowed).

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

Example: AD conversion

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

10.4 Related Registers to ADC

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

AD control register ADCON0(9DH)

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

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

Bit5~Bit2 CHS<3:0>: Analog channel selection bit
0000= AN0
0001= AN1
0010= AN2
0011= Not used
0100= Not used
0101= Not used
0110= Not used
0111= Not used
1000= Not used
1001= Not used
1010= Not used
1011= AN11
1100= AN12
1101= AN13
1110= OPA0O
1111= Internal reference voltage 0.6V **(Note: To use the internal reference voltage, the ADCON1[2] bit should be 0 and the ADCON1[1] bit should be 1)**

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

Bit0 ADON: ADC enable bit
1= Enable ADC
0= Disable ADC

AD data register higher bit ADCON1(9CH)

9CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON1	ADFM	----	----	----	----	LDO_EN	LDO_SEL[1:0]	
Read/Write	R/W	----	----	----	----	R/W	R/W	R/W
Reset value	0	----	----	----	----	0	0	0

Bit7 ADFM: The AD conversion result format selection bit.

1= Right alignment.

0= Left alignment.

Bit6~Bit3 Unused, read as 0.

Bit2 LDO_EN: Internal reference voltage enable bit.

1= Enabling the ADC internal LDO reference voltage.
(When internal LDO is selected as the reference voltage, the ADC hasa maximum effective accuracy of 8 bits)

0= VDD is used as the ADC reference voltage.

Bit1~Bit0 LDO_SEL: LDO reference voltage selection bit.

00= 1.2V **(Note: When selecting this reference voltage, theconversion clock needs to be selected from F_{RC})**

01= Reserved

10= 2.4V

11= 3.0V

AD data register higher bit ADRESH(9FH), ADFM=0

9FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESH	ADRES11	ADRES10	ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4
Read /Write	R	R	R	R	R	R	R	R
Resetvalue	X	X	X	X	X	X	X	X

Bit7~Bit0 ADRES<11:4>: ADC result register bits.

The higher 8 bits of the 12-bit conversion result.

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

9EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESL	ADRES3	ADRES2	ADRES1	ADRES0	----	----	----	----
Read/Write	R	R	R	R	----	----	----	----
Resetvalue	X	X	X	X	----	----	----	----

Bit7~Bit4 ADRES<3:0>: ADC result register bits.

The lower 4 bits of the 12-bit conversion result.

Bit3~Bit0 Not used.

AD data register higher bit ADRESH(9FH), ADFM=1

9FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESH	----	----	----	----	----	----	ADRES11	ADRES10
Read/Write	----	----	----	----	----	----	R	R
Resetvalue	----	----	----	----	----	----	X	X

Bit7~Bit2 Not used.

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

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

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

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

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

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

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

11. PWM Module

The chip contains a 10-bit PWM module that can be configured as 5 channels of a common period and independent duty cycle, or 2 sets of complementary outputs + 1 independent output.

The PWM outputs can be selected by software as RA0-RA2, RB1, RB2. Among them, PWM0/PWM1 and PWM2/PWM3 can be configured as complementary outputs.

11.1 Pin Configuration

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

11.2 Related Register Description

PWM control register 0 PWMCON0(13H)

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

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

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

PWM control register 1 PWMCON1(14H)

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

Bit7~6 PWMIO_SEL: PWM IO selection.
11= Reserved
10= PWM assigned in group B,
PWM0-RA0,PWM1-RA1,PWM2-RA2,PWM3-RB2,PWM4-RB1
01= Reserved
00= Reserved
Bit5 PWM2DTEN: PWM2 dead-time enable bit.
1= Enable PWM2 dead-time function, PWM2 and PWM3 form a complementary output pair.
0= Disable PWM2 dead-time function.
Bit4 PWM0DTEN: PWM0 dead-time enable bit.
1= Enable the PWM0 dead-time function, PWM0 and PWM1 form a complementary output pair.
0= Disable PWM0 dead-time function.
Bit3~Bit2 Not used.
Bit1~Bit0 DT_DIV[1:0] Dead time clock source dividing frequency.
11= F_{OSC} /8
10= F_{OSC} /4
01= F_{OSC} /2
00= F_{OSC} /1

PWM control register 2 PWMCON2(1DH)

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

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

PWM Cycle Low Register PWMTL(15H)

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

Bit7~Bit0 PWMT [7:0]: The PWM cycle is 8 bits lower.

PWM Cycle Higher Register PWMTH(16H)

16H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMTH	----	----	PWM4D[9:8]		----	----	PWMT [9:8]	
Read /Write	----	----	R/W	R/W	----	----	R/W	R/W
Reset value	----	----	0	0	----	----	0	0

Bit7~Bit6 Not used.
Bit5~Bit4 PWM4D [9:8]: PWM4 duty cycle is 2 bits higher.
Bit3~Bit2 Not used.
Bit1~Bit0 PWMT [9:8]: The PWM cycle is 2 bits higher.

PWM0 Duty Cycle Lower Register PWMD0L(17H)

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

Bit7~Bit0 PWMD0[7:0]: PWM0 duty cycle is 8 bits lower.

PWM1 Duty Cycle Lower Register PWMD1L(18H)

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

Bit7~Bit0 PWMD1[7:0]: PWM1 duty cycle is 8 bits lower.

PWM2 Duty Cycle Low Register PWMD2L(19H).

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

Bit7~Bit0 PWMD2[7:0]: PWM2 duty cycle is 8 bits lower.

PWM3 Duty Cycle Lower Register PWMD3L(1AH)

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

Bit7~Bit0 PWMD3[7:0]: PWM3 duty cycle is 8 bits lower.

PWM4 Duty Cycle Lower Register PWMD4L(1BH)

1BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD4L	PWMD4[7:0]							
Read /Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD4[7:0]: PWM4 duty cycle is 8 bits lower.

PWM0 and PWM1 Duty Cycle Higher Register PWMD01H(1CH)

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

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

PWM2 and PWM3 Duty Cycle Higher Register PWMD23H(0EH)

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

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

PWM0 and PWM1 Dead Time Register PWM01DT(0FH)

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

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

PWM2 and PWM3 dead time register PWM23DT(10H)

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

Bit7~Bit6 Not used.

Bit5~Bit0 PWM23DT[5:0]: PWM2 and PWM3 dead time.

11.3 PWM Cycle

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

Equation 1: PWM period calculation formula.

$$\text{PWM cycle}=[PWMT+1]*Tosc*(CLKDIV\ pre\text{-}scaler\ value)$$

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

When the PWM cycle counter is equal to PWMT, the following 3 events occur during the next incremental counting cycle:

- The PWM cycle counter is cleared to zero.
- The PWMx pin is set to 1.
- The PWM new cycle values are latched.
- The PWM new duty cycle value is latched.
- Generating the PWM interrupt flag bit.

11.4 PWM Duty Cycle

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

The PWMDxL and PWMDxxH registers can be written at any time, but the duty cycle value is not updated to the internal latch until the PWM cycle counter equals PWMT (i.e., end of cycle).

Equation 2: Pulse width calculation formula.

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

Equation 3: PWM duty cycle calculation formula.

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

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

11.5 Change of System Clock Frequency

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

11.6 Programmable Dead Time Delay Mode

Complementary output mode can be enabled by setting PWMxDT_EN to automatically enable the dead time delay function after enabling the complementary output.

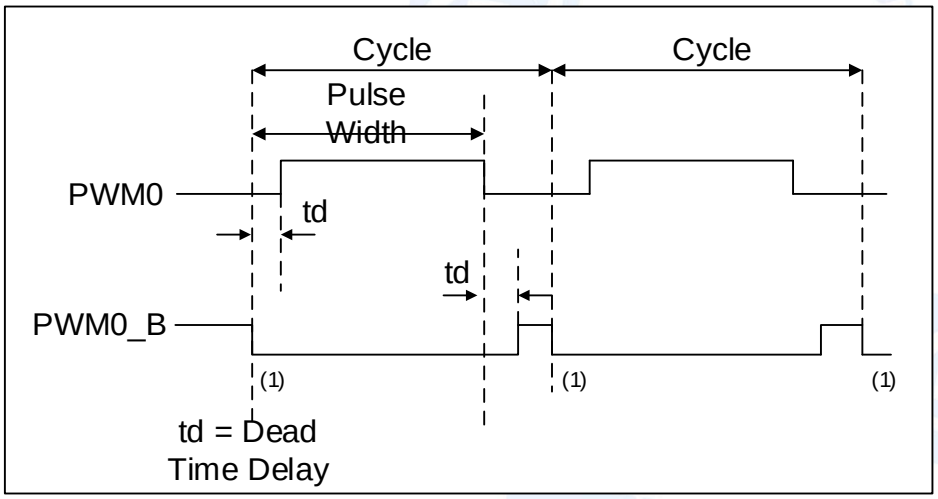


Figure 11-1: Example of PWM Dead Time Delay Output

The equation for calculating the dead time is:

td=(PWMxxDT[5:0]+1)*T_{OSC}*(DT_DIV pre-scaler value)

11.7 PWM Settings

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

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

12. Operational amplifier (OPA0)

The chip has built-in a group operational amplifier OPA0.

12.1 OPA0

OPA0 has the following functions:

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

12.1.1 OPA0 Enabled

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

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

12.1.2 OPA0 Port Selection

12.1.2.1 OPA0 Positive Input

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

12.1.2.2 OPA0 Negative Input

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

12.1.2.3 OPA0 Output

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

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

12.1.2.4 Port direction Setting When OPA0 Is Used

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

12.1.3 OPA0 Operation Mode

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

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

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

Regulation mode workflow:

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

12.1.4 Registers Related to OPA0

There are 2 registers related to OPA0, namely the control register OPA0CON and the offset-voltage regulating register OPA0ADJ.

OPA0 control register OPA0CON

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPA0CON	OPA0EN	OPA0O	OPA0_CMP	OPA0_ADC	----	----	----	OPA0FT
Read /write	R/W	R/W	R/W	R/W	----	----	----	R/W
Reset value	0	0	0	0	----	----	----	1

Bit7 OPA0EN: OPA0 enable bit.
1= Enabling OPA0.
0= Prohibit OPA0.
Bit6 OPA0O Op-amp output selection.
1= OPA0 output connected to the I/O port (OPA0O pin).
0= The OPA0 output is not connected to the I/O port.
Bit5 OPA0_CMP: Comparator mode
1: Comparator mode, comparator output can be read via OPA0ADJ[7]
0: Op-amp mode
Bit4 OPA0_ADC: The op-amp output to the ADC control bit.
1= Op-amp output to channel ADC14.
0= The op-amp output is not connected to the ADC.
Bit3~Bit1 Not used.
Bit0 OPA0FT: Op-amp output internal filtering selection.
1= (a) The op-amp output is internally connected to a filter circuit.
0= The op-amp output is not internally connected to a filter circuit.

OPA0 offset-voltage regulating register OPA0ADJ

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

Bit7 OPA0OUT: OPA0 output results.
1= (a) The op-amp output is high and the voltage at the positive terminal is higher than the voltage at the negative terminal.
0= The op-amp output is low and the voltage at the positive terminal is lower than the voltage at the negative terminal.
Bit6 OPA0COFM: OPA0 operating mode selection bit.
1= OPA0 operating in regulation mode.
0= OPA0 is operating in normal mode.
Bit5 OPA0CRS: OPA0 regulation mode input selection bit.
1= OPA0 regulation mode positive input.
0= OPA0 regulation mode negative input.
Bit4~Bit0 OPA0ADJ[4:0]: OPA0 out-of-regulation voltage regulation bit.

13. Electrical Parameters

13.1 Limit parameters

Supply voltage……..GND-0.3V~GND+6.0V
Storage temperature……-50℃~125℃
Operating temperature……-20℃~75℃
Port input voltage……GND-0.3V~VDD+0.3V
Maximum fill current for all ports……200mA
Maximum pull current for all ports……-150mA

Note: If the operating conditions of the device exceed the above "limit parameters", permanent damage may be caused to the device. These values are only extreme values for operating conditions, and we do not recommend that devices operate the values which is out of the specification limits. The stability of the device will be affected if it is operated under extreme conditions for a long time.

13.2 DC Feature

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

Symbol	Item	Test condition		Min	Typ	Max	Unit
		VDD	Condition				
VDD	Operating Voltage	-	F_{SYS}=8MHz	3.0		5.5	V
		-	F_{SYS}=4MHz	2.5		5.5	V
I_{DD}	Operating current	5V	F_{SYS}=8MHz		3		mA
		3V	F_{SYS}=8MHz		2		mA
I_{STB}	Static current	5V	----		0.1	2	μA
		3V	----		0.1	1	μA
V_{IL}	Low level input voltage	-	----			0.3VDD	V
V_{IH}	High level input voltage	-	----	0.7VDD			V
V_{OH}	High level output voltage	-	Without load	0.9VDD			V
V_{OL}	Low level output voltage	-	Without load			0.1VDD	V
R_{PH}	Pull-up resistor resistance value	5V	V_O=0.5VDD		30		kΩ
		3V	V_O=0.5VDD		50		kΩ
R_{PD}	Pull-down resistor resistance value	5V	V_O=0.5VDD		30		kΩ
		3V	V_O=0.5VDD		50		kΩ
I_{OL1}	Output port filling current General I/O	5V	V_{OL}=0.3VDD		30		mA
		3V	V_{OL}=0.3VDD		13		mA
I_{OL2}	Output port filling current High-current PWM I/O	5V	V_{OL}=0.3VDD		75		mA
		3V	V_{OL}=0.3VDD		33		mA
I_{OH1}	Output port pull current general I/O	5V	V_{OH}=0.7VDD		-18		mA
		3V	V_{OH}=0.7VDD		-7		mA
I_{OH2}	Output port pull current high-current PWM I/O	5V	V_{OH}=0.7VDD		-80		mA
		3V	V_{OH}=0.7VDD		-30		mA
V_{BG}	Internal referencevoltage 0.6V	VDD=2.5~5.5V T_A=25℃		-1.5%	0.6	1.5%	V
		VDD=2.5~5.5V T_A=－40~85℃		-2.0%	0.6	2.0%	V

13.3 ADC Feature

(T_A = 25°C, unless otherwise noted)

Symbol	Item	Test condition	Min	Typ	Max	Unit
V_{ADC}	ADC operating voltage	V_{ADREF} =VDD,F_{ADC} =500kHz	2.5		5.5	V
		V_{ADREF} =1.2V,F_{ADC} =F_{RC}	2.5		5.5	V
		V_{ADREF} =2.4V,F_{ADC} =250kHz	2.5		5.5	V
		V_{ADREF} =3.0V,F_{ADC} =250kHz	3.3		5.5	V
I_{ADC}	ADC conversion current	V_{ADC} =5V,F_{ADC} =500kHz			500	uA
		V_{ADC} =3V,F_{ADC} =500kHz			200	uA
V_{ADI}	ADC input voltage	V_{ADC} =5V,F_{ADC} =250kHz	0		V_{ADC}	V
DNL	Differential nonlinear error	V_{ADC} =5V,F_{ADC} =250kHz		±3		LSB
INL	Integral nonlinear error	V_{ADC} =5V,F_{ADC} =250kHz		±4		LSB
T_{ADC}	ADC conversion time			49		TADCCLK

13.4 ADC Internal LDO Reference Voltage Feature

(T_A = 25°C, unless otherwise noted)

Symbol	Item	Test condition	Min	Typ	Max	Unit
V_1	LDO_OUT=1.2V	VDD=2.5~5.5V	-1.5%	1.2	+1.5%	V
V_2	LDO_OUT=2.4V	VDD=2.5~5.5V	-1.5%	2.4	+1.5%	V
V_3	LDO_OUT=3V	VDD=3.3~5.5V	-1.5%	3.0	+1.5%	V

13.5 OPA Electrical Feature

(T_A = 25°C, unless otherwise noted)

Symbol	Item	Test condition	Min	Typ	Max	Unit
DC Electrical Features						
VDD	Operating Voltage	VDD=2.5~5.5V	2.5		5.5	V
I_{DD}	Static current	VDD=5.0V		380		uA
V_{OPOS}	Input offset voltage	Default Value VDD = 5V V_{CM}	-20		20	mV
		After zeroing VDD = 5V V_{CM}	-5		5	mV
V_{CM}	Common mode voltage		0		VDD-1.5V	V
PSRR	Supply voltage rejection		60	70		dB
CMRR	Common mode rejection ratio*	VDD=5V V_{CM} =0~VDD-1.5V	90	100		dB
AC Electrical Characteristics						
A_{OL}	Open-loop gain*		90	100		dB
GBW	Gain bandwidth*	R_L =1MΩ, C_L =100pF	1.5	2		MHz

* It is guaranteed by design, and not batch tested.

13.6 Power-On Reset Feature

(T_A = 25°C, unless otherwise noted)

Symbol	Item	Test condition	Min	Typ	Max	Unit
t_{VDD}	VDD rise rate	-	0.05			V/ms
V_{LVR2}	LVR set voltage = 2.5V	VDD=2.0~5.5V	2.2	2.5	2.8	V
V_{LVR3}	LVR set voltage = 3.3V	VDD=2.5~5.5V	3.0	3.3	3.7	V

13.7 AC Electrical Characteristics

(T_A = 25°C, unless otherwise noted)

Symbol	Item	Test condition		Min	Typ	Max	Unit
		VDD	Condition				
T_{WDT}	WDT reset time	5V			18		ms
		3V			36		ms
F_{RC}	Internal vibration frequency stability	VDD=4.0~5.5V TA=-20~75℃		-3%	8	+3%	MHz
		VDD=2.5~5.5V TA=-20~75℃		-6%	8	+6%	MHz

14. Instruction

14.1 Instruction List

Mnemonic sign		Operation	Instruction	Symbol
Control class				
NOP		Empty operation	1	None
STOP		Enter sleep mode	1	TO,PD
CLRWDT		Clear watchdog counter	1	TO,PD
Data Transfer				
LD	[R],A	Transferring ACC content to R	1	NONE
LD	A,[R]	Transferring R content to ACC	1	Z
TESTZ	[R]	Transferring data memory contents to data memory	1	Z
LDIA	i	Immediate transfer number i to ACC	1	NONE
Logical operations				
CLRA		Clear ACC	1	Z
SET	[R]	Reset data memory R	1	NONE
CLR	[R]	Clear data memory R	1	Z
ORA	[R]	Perform ‘OR’ on R and ACC, save the result to ACC	1	Z
ORR	[R]	Perform ‘OR’ on R and ACC, save the result to R	1	Z
ANDA	[R]	Perform ‘AND’ on R and ACC, save the result to ACC	1	Z
ANDR	[R]	Perform ‘AND’ on R and ACC, save the result to R	1	Z
XORA	[R]	Perform ‘XOR’ on R and ACC, save the result to ACC	1	Z
XORR	[R]	Perform ‘XOR’ on R and ACC, save the result to R	1	Z
SWAPA	[R]	Swap R register high and low half byte, save the result to ACC	1	NONE
SWAPR	[R]	Swap R register high and low half byte, save the result to R	1	NONE
COMA	[R]	The content of R register is reversed, and the result is stored in ACC	1	Z
COMR	[R]	The content of R register is reversed and the result is stored in R	1	Z
XORIA	i	Perform ‘XOR’ on i and ACC, save the result to ACC	1	Z
ANDIA	i	Perform ‘AND’ on i and ACC, save the result to ACC	1	Z
ORIA	i	Perform ‘OR’ on i and ACC, save the result to ACC	1	Z
Shift operation				
RRCA	[R]	Data memory rotates one bit to the right with carry, the result is stored in ACC	1	C
RRCR	[R]	Data memory rotates one bit to the right with carry, the result is stored in R	1	C
RLCA	[R]	Data memory rotates one bit to the left with carry, the result is stored in ACC	1	C
RLCR	[R]	Data memory rotates one bit to the left with carry, the result is stored in R	1	C
RLA	[R]	Data memory rotates one bit to the left without carry, and the result is stored in ACC	1	NONE
RLR	[R]	Data memory rotates one bit to the left without carry, and the result is stored in R	1	NONE
RRA	[R]	Data memory does not take carry and rotates to the right by one bit, and the result is stored in ACC	1	NONE
RRR	[R]	Data memory does not take carry and rotates to the right by one bit, and the result is stored in R	1	NONE
Incremental and Decremental				
INCA	[R]	Increment the data memory R and put the result into ACC	1	Z
INCR	[R]	Increment the data memory R and put the result into R	1	Z
DECA	[R]	Decrement the data memory R and put the result into ACC	1	Z

Mnemonic sign		Operation	Instruction	Symbol
DECR	[R]	Decrement the data memory R and put the result into R	1	Z
Bit operation				
CLRB	[R],b	Clear a bit in data memory R to zero	1	NONE
SETB	[R],b	Set some bit in data memory R 1	1	NONE
Mathematical operations				
ADDA	[R]	ACC+[R]→ACC	1	C,DC,Z,OV
ADDR	[R]	ACC+[R]→R	1	C,DC,Z,OV
ADDCA	[R]	ACC+[R]+C→ACC	1	Z,C,DC,OV
ADDCR	[R]	ACC+[R]+C→R	1	Z,C,DC,OV
ADDIA	i	ACC+i→ACC	1	Z,C,DC,OV
SUBA	[R]	[R]-ACC→ACC	1	C,DC,Z,OV
SUBR	[R]	[R]-ACC→R	1	C,DC,Z,OV
SUBCA	[R]	[R]-ACC-C→ACC	1	Z,C,DC,OV
SUBCR	[R]	[R]-ACC-C→R	1	Z,C,DC,OV
SUBIA	i	i-ACC→ACC	1	Z,C,DC,OV
HSUBA	[R]	ACC-[R]→ACC	1	Z,C,DC,OV
HSUBR	[R]	ACC-[R]→R	1	Z,C,DC,OV
HSUBCA	[R]	ACC-[R]- $\overline{C}$ →ACC	1	Z,C,DC,OV
HSUBCR	[R]	ACC-[R]- $\overline{C}$ →R	1	Z,C,DC,OV
HSUBIA	i	ACC-i→ACC	1	Z,C,DC,OV
Unconditional transfer				
RET		Return from subroutine	2	NONE
RET	i	Return from the subroutine and store the immediate number I into ACC	2	NONE
RETI		Return from interrupt	2	NONE
CALL	ADD	Subroutine calls	2	NONE
JP	ADD	Unconditional jump	2	NONE
Condition Transfer				
SZB	[R],b	If bit b of data memory R is "0", the next instruction is skipped	1 or 2	NONE
SNZB	[R],b	If bit b of data memory R is "1", the next instruction is skipped	1 or 2	NONE
SZA	[R]	Data memory R is sent to ACC, and if the content is "0", the next instruction is skipped	1 or 2	NONE
SZR	[R]	If the content of data memory R is "0", the next instruction is skipped.	1 or 2	NONE
SZINCA	[R]	Add "1" to data memory R and put the result into ACC. If the result is "0", the next instruction is skipped	1 or 2	NONE
SZINCR	[R]	Add "1" to data memory R, put the result into R. If the result is "0", the next instruction is skipped	1 or 2	NONE
SZDECA	[R]	Data memory R minus "1", the result is put into ACC. If the result is "0", the next instruction is skipped	1 or 2	NONE
SZDECR	[R]	Data memory R minus "1", the result is put into R. If the result is "0", the next instruction is skipped	1 or 2	NONE

14.2 Instruction Description

ADDA **[R]**

Operation: Add R to ACC and put the result into ACC

Cycle: 1

Affected flag bits: C, DC, Z, OV

Example:

```
LDIA    09H     ;Assign the value 09H to ACC
LD      R01,A   Assign the value of ACC (09H) to the custom register R01
LDIA    077H    ;Assign 77H to ACC
ADDA    R01     Execution result: ACC = 09H + 77H = 80H
```

ADDR **[R]**

Operation: Add R to ACC and put the result into R

Cycle: 1

Affected flag bits: C, DC, Z, OV

Example:

```
LDIA    09H     ;Assign the value 09H to ACC
LD      R01,A   Assign the value of ACC (09H) to the custom register R01
LDIA    077H    ;Assign 77H to ACC
ADDR    R01     ;Execution result: R01 = 09H + 77H = 80H
```

ADDCA **[R]**

Operation: R plus ACC plus C bit and put the result into ACC

Cycle: 1

Affected flag bits: C, DC, Z, OV

Example:

```
LDIA    09H     ;Assign the value 09H to ACC
LD      R01,A   Assign the value of ACC (09H) to the custom register R01
LDIA    077H    ;Assign 77H to ACC
ADDCA   R01     Execution result: ACC= 09H + 77H + C=80H (C=0)ACC= 09H +
                77H + C=81H (C=1)
```

ADDCR **[R]**

Operation: R plus ACC plus C bit and put the result into R

Cycle: 1

Affected flag bits: C, DC, Z, OV

Example:

```
LDIA    09H     ;Assign the value 09H to ACC
LD      R01,A   Assign the value of ACC (09H) to the custom register R01
LDIA    077H    ;Assign 77H to ACC
ADDCR   R01     ;Execution result: R01 = 09H + 77H + C=80H (C=0)R01 = 09H +
                77H + C=81H (C=1)
```

ADDIA **i**

Operation: Add the immediate number i to ACC and put the result into ACC

Cycle: 1

Affected flag bits: C, DC, Z, OV

Example:

```
LDIA    09H     ;Assign the value 09H to ACC
ADDIA   077H    Execution result: ACC = ACC(09H) + i(77H) = 80H
```

ANDA **[R]**

Operation: Register R and ACC perform logical sum operation, and the result is put into ACC

Cycle: 1

Influence Z

Example:

```
LDIA    0FH     ;Assign 0FH to ACC
LD      R01,A   Assign the value of ACC (0FH) to register R01
LDIA    77H     ;Assign 77H to ACC
ANDA    R01     Execution result: ACC=(0FH and 77H)=07H
```

ANDR **[R]**

Operation: Register R and ACC perform logical sum operation, and the result is put into R

Cycle: 1

Affected flag bits: Z

Example:

```
LDIA    0FH     ;Assign 0FH to ACC
LD      R01,A   Assign the value of ACC (0FH) to register R01
LDIA    77H     ;Assign 77H to ACC
ANDR    R01     Execution result: R01=(0FH and 77H)=07H
```

ANDIA **i**

Operation: Logical sum operation of immediate i with ACC, and the result is put into ACC

Cycle: 1

Affected flag bits: Z

Example:

```
LDIA    0FH     ;Assign 0FH to ACC
ANDIA   77H     Execution result: ACC =(0FH and 77H)=07H
```

CALL **add**

Operation: Calling subroutines

Cycle: 2

Affected flag bits: None

Example:

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

CLRA

Operation: Clear ACC

Cycle: 1

Affected flag bits: Z

Example:

```
CLRA                ;Execution result: ACC=0
```

CLR [R]

Operation: Register R is cleared to zero

Cycle: 1

Affected flag bits: Z

Example:

```
CLR     R01         ;Execution result: R01=0
```

CLRB [R],b

Operation: Bit b of register R is cleared to zero

Cycle: 1

Affected flag bits: None

Example:

```
CLRB    R01,3       ;Execution result: bit 3 of R01 is zero
```

CLRWDT

Operation: Clear the watchdog counter

Cycle: 1

Affected flag bits: TO, PD

Example:

```
CLRWDT              ;Watchdog counter clear
```

COMA [R]

Operation: Register R is inverted and the result is put into ACC

Cycle: 1

Affected flag bits: Z

Example:

```
LDIA    0AH         ;ACC assignment 0AH
LD      R01,A       Assign the value of ACC (0AH) to register R01
COMA    R01         Execution result: ACC=0F5H
```

COMR **[R]**

Operation: The register R is inverted and the result is put into R

Cycle: 1

Affected flag bits: Z

Example:

```
LDIA    0AH     ;ACC assignment 0AH
LD      R01,A   Assign the value of ACC (0AH) to register R01
COMR    R01     ;Execution result: R01=0F5H
```

DECA **[R]**

Operation: Register R is decreases by 1 and the result is put into ACC

Cycle: 1

Affected flag bits: Z

Example:

```
LDIA    0AH     ;ACC assignment 0AH
LD      R01,A   Assign the value of ACC (0AH) to register R01
DECA    R01     Execution result: ACC=(0AH-1)=09H
```

DECR **[R]**

Operation: Register R decreases by 1 and the result is put into R

Cycle: 1

Affected flag bits: Z

Example:

```
LDIA    0AH     ;ACC assignment 0AH
LD      R01,A   Assign the value of ACC (0AH) to register R01
DECR    R01     ;Execution result: R01=(0AH-1)=09H
```

HSUBA **[R]**

Operation: ACC minus R, the result is put into ACC

Cycle: 1

Affected flag bits: C,DC,Z,OV

Example:

```
LDIA    077H    ;ACC assignment 077H
LD      R01,A   Assign the value of ACC (077H) to register R01
LDIA    080H    ;ACC assignment 080H
HSUBA   R01     Execution result: ACC=(80H-77H)=09H
```

HSUBR **[R]**

Operation: ACC minus R. The result is put into R

Cycle: 1

Affected flag bits: C,DC,Z,OV

Example:

```
LDIA    077H    ;ACC assignment 077H
LD      R01,A   Assign the value of ACC (077H) to register R01
LDIA    080H    ;ACC assignment 080H
HSUBR   R01     ;Execution result: R01=(80H-77H)=09H
```

HSUBCA **[R]**

Operation: ACC minus R minus $\overline{C}$ The result is put into ACC

Cycle: 1

Affected flag bits: C,DC,Z,OV

Example:

```
LDIA    077H    ;ACC assignment 077H
LD      R01,A   Assign the value of ACC (077H) to register R01
LDIA    080H    ;ACC assignment 080H
HSUBCA  R01     ;Execution result: ACC=(80H-77H- C )=08H(C=0)
                                   ACC=(80H-77H- C )=09H(C=1)
```

HSUBCR **[R]**

Operation: ACC minus R minus $\overline{C}$ and the result is put into R

Cycle: 1

Affected flag bits: C,DC,Z,OV

Example:

```
LDIA    077H    ;ACC assignment 077H
LD      R01,A   Assign the value of ACC (077H) to register R01
LDIA    080H    ;ACC assignment 080H
HSUBCR  R01     ;Execution result: R01=(80H-77H- C )=08H(C=0)
                                   R01=(80H-77H- C )=09H(C=1)
```

INCA **[R]**

Operation: Register R adds 1 and the result is put into ACC

Cycle: 1

Affected flag bits: Z

Example:

```
LDIA    0AH     ;ACC assignment 0AH
LD      R01,A   Assign the value of ACC (0AH) to register R01
INCA    R01     Execution result: ACC=(0AH+1)=0BH
```

INCR [R]

Operation: Register R adds 1 and the result is put into R

Cycle: 1

Affected flag bits: Z

Example:

```
LDIA    0AH     ;ACC assignment 0AH
LD      R01,A   Assign the value of ACC (0AH) to register R01
INCR    R01     ;Execution result: R01=(0AH+1)=0BH
```

JP add

Operation: Jump to add address

Cycle: 2

Affected flag bits: None

Example:

```
JP      LOOP    ;Jump to the address of the subroutine with the name defined as"LOOP"
```

LD A,[R]

Operation: Assign the value of R to ACC

Cycle: 1

Affected flag bits: Z

Example:

```
LD      A,R01   ;Assign the value of register R0 to ACC
LD      R02,A   Assign the value of ACC to register R02 to move the data from R01to R02.
```

LD [R],A

Operation: Assign the value of ACC to R

Cycle: 1

Affected flag bits: None

Example:

```
LDIA    09H     ;Assign the value 09H to ACC
LD      R01,A   ;Execution result: R01=09H
```

LDIA i

Operation: The immediate number i is assigned to ACC

Cycle: 1

Affected flag bits: None

Example:

```
LDIA    0AH     ;ACC assignment 0AH
```

NOP

Operation: Empty command

Cycle: 1

Affected flag bits: None

Example:

```
NOP
NOP
```

ORIA i

Operation: The immediate number is logically orthogonal to ACC and the result is assigned to ACC

Cycle: 1

Affected flag bits: Z

Example:

```
LDIA    0AH     ;ACC assignment 0AH
ORIA    030H    Execution result: ACC = (0AH or 30H) = 3AH
```

ORA [R]

Operation: Register R and ACC perform logical or operation, and the result is put into ACC

Cycle: 1

Affected flag bits: Z

Example:

```
LDIA    0AH     ;Assign 0AH to ACC
LD      R01,A   Assign ACC(0AH) to register R01
LDIA    30H     ;Assign 30H to ACC
ORA     R01     Execution result: ACC=(0AH or 30H)=3AH
```

ORR [R]

Operation: The register R performs a logical or operation with ACC, and the result is put into R

Cycle: 1

Affected flag bits: Z

Example:

```
LDIA    0AH     ;Assign 0AH to ACC
LD      R01,A   Assign ACC(0AH) to register R01
LDIA    30H     ;Assign 30H to ACC
ORR     R01     Execution result: R01=(0AH or 30H)=3AH
```

RET

Operation: Return from subroutine

Cycle: 2

Affected flag bits: None

Example:

```
        CALL    LOOP        ;Call subroutine LOOP
        NOP                 This statement will be executed after the return of the ;RETinstruction
        ...                 ;Other programs
LOOP:
        ...                 ; Subroutine
        RET                 ; The subroutine returns
```

RET i

Operation: Return from subroutine with parameters, and put parameters into ACC

Cycle: 2

Affected flag bits: None

Example:

```
        CALL    LOOP        ;Call subroutine LOOP
        NOP                 This statement will be executed after the return of the ;RETinstruction
        ...                 ;Other programs
LOOP:
        ...                 ; Subroutine
        RET     35H         Subroutine return,ACC=35H
```

RETI

Operation: Interrupt return

Cycle: 2

Affected flag bits: None

Example:

```
INT_START                   ;Interrupt program entry
        ...                 ;Interrupt Handler
        RETI                ;Interrupt return
```

RLCA [R]

Operation: The register R with C is cyclically shifted left by one bit and the result is put into ACC

Cycle: 1

Affected flag bits: C

Example:

```
        LDIA    03H         ;ACC assignment 03H
        LD      R01,A       Assign ACC value to R01,R01=03H
        RLCA    R01         The result of the operation: ACC=06H(C=0);ACC=07H(C=1)
                            C=0
```

RLCR **[R]**

Operation: The register R with C is cyclically shifted left by one bit and the result is put into R

Cycle: 1

Affected flag bits: C

Example:

```
LDIA    03H     ;ACC assignment 03H
LD      R01,A   ACC value is assigned to R01,R01=03H
RLCR    R01     ;Operation result: R01=06H(C=0);
                R01=07H(C=1);C=0
```

RLA **[R]**

Operation: The register R does not carry the C cycle left shift one bit, the result is put into ACC

Cycle: 1

Affected flag bits: None

Example:

```
LDIA    03H     ;ACC assignment 03H
LD      R01,A   ACC value is assigned to R01,R01=03H
RLA     R01     ;Operation result: ACC=06H
```

RLR **[R]**

Operation: The register R without C is cyclically shifted left by one bit and the result is put into R

Cycle: 1

Affected flag bits: None

Example:

```
LDIA    03H     ;ACC assignment 03H
LD      R01,A   ACC value is assigned to R01,R01=03H
RLR     R01     ;Operation result: R01=06H
```

RRCA **[R]**

Operation: The register R with C is cyclically shifted one bit to the right, and the result is put into ACC

Cycle: 1

Affected flag bits: C

Example:

```
LDIA    03H     ;ACC assignment 03H
LD      R01,A   ACC value is assigned to R01,R01=03H
RRCA    R01     The result of the operation: ACC=01H(C=0);ACC=081H (C=1);
                C=1
```

RRCR **[R]**

Operation: The register R with C is cyclically shifted one bit to the right and the result is put into R

Cycle: 1

Affected flag bits: C

Example:

```
LDIA    03H     ;ACC assignment 03H
LD      R01,A   ACC value is assigned to R01,R01=03H
RRCR    R01     ;Operation result: R01=01H(C=0);
                R01=81H(C=1);C=1
```

RRA **[R]**

Operation: The register R does not carry the C cycle right shift one bit, the result is put into ACC

Cycle: 1

Affected flag bits: None

Example:

```
LDIA    03H     ;ACC assignment 03H
LD      R01,A   ACC value is assigned to R01,R01=03H
RRA     R01     ;Operation result: ACC=81H
```

RRR **[R]**

Operation: The register R without C is cyclically shifted one bit to the right, and the result is put into R

Cycle: 1

Affected flag bits: None

Example:

```
LDIA    03H     ;ACC assignment 03H
LD      R01,A   ACC value is assigned to R01,R01=03H
RRR     R01     ;Operation result: R01=81H
```

SET **[R]**

Operation: Register R all set to 1

Cycle: 1

Affected flag bits: None

Example:

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

SETB **[R],b**

Operation: Bit b of register R set to 1

Cycle: 1

Affected flag bits: None

Example:

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

STOP

Operation: Enter sleep mode

Cycle: 1

Affected flag bits: TO, PD

Example:

STOP	The chip enters power saving mode, CPU and oscillator stopworking, IO ports keep the original state

SUBIA i

Operation: Immediate number i minus ACC, the result is put into ACC

Cycle: 1

Affected flag bits: C,DC,Z,OV

Example:

```
LDIA    077H    ;ACC assignment 77H
SUBIA   80H     ;Operation result: ACC=80H-77H=09H
```

SUBA [R]

Operation: Register R minus ACC, the result is put into ACC

Cycle: 1

Affected flag bits: C,DC,Z,OV

Example:

```
LDIA    080H    ;ACC assignment 80H
LD      R01,A   The value of ACC is assigned to R01, R01=80H
LDIA    77H     ;ACC assignment 77H
SUBA    R01     ;Operation result: ACC=80H-77H=09H
```

SUBR [R]

Operation: Register R minus ACC, and the result is put into R

Cycle: 1

Affected flag bits: C,DC,Z,OV

Example:

```
LDIA    080H    ;ACC assignment 80H
LD      R01,A   The value of ACC is assigned to R01, R01=80H
LDIA    77H     ;ACC assignment 77H
SUBR    R01     ;Operation result: R01=80H-77H=09H
```

SUBCA **[R]**

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

Cycle: 1

Affected flag bits: C,DC,Z,OV

Example:

```
LDIA    080H    ;ACC assignment 80H
LD      R01,A   The value of ACC is assigned to R01, R01=80H
LDIA    77H     ;ACC assignment 77H
SUBCA   R01     The result of the operation: 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

Cycle: 1

Affected flag bits: C,DC,Z,OV

Example:

```
LDIA    080H    ;ACC assignment 80H
LD      R01,A   The value of ACC is assigned to R01, R01=80H
LDIA    77H     ;ACC assignment 77H
SUBCR   R01     ;Operation result: R01=80H-77H-C=09H(C=0)R01=80H-77H-C=08H(C=1)
```

SWAPA **[R]**

Operation: The high and low half bytes of register R are swapped and the result is put into ACC

Cycle: 1

Affected flag bits: None

Example:

```
LDIA    035H    ;ACC assignment 35H
LD      R01,A   The value of ACC is assigned to R01, R01=35H
SWAPA   R01     ;Operation result: ACC=53H
```

SWAPR **[R]**

Operation: The high and low half bytes of register R are swapped and the result is put into R

Cycle: 1

Influence None

Example:

```
LDIA    035H    ;ACC assignment 35H
LD      R01,A   The value of ACC is assigned to R01, R01=35H
SWAPR   R01     ;Operation result: R01=53H
```

SZB **[R],b**

Operation: Determine the b-bit of register R, and jump between 0, otherwise execute sequentially

Cycle: 1 or 2

Affected flag bits: None

Example:

```
SZB     R01,3     Bit 3 of judgment register R01
JP      LOOP      ;R01 bit 3 is 1 then executing this statement, jump to LOOP
JP      LOOP1     ;R01 bit 3 is 0 time jump, execute this statement, jump to LOOP1
```

SNZB **[R],b**

Operation: Determine the b-bit of register R, and jump between 1, otherwise execute sequentially

Cycle: 1 or 2

Affected flag bits: None

Example:

```
SNZB    R01,3     Bit 3 of judgment register R01
JP      LOOP      ;R01 bit 3 is 0 before executing this statement and jumping to LOOP
JP      LOOP1     The 3rd bit of R01 is 1 time jump, execute this statement and jumpto
                  LOOP1
```

SZA **[R]**

Operation: Assign the value of register R to ACC, if R is 0 then inter-hop, otherwise execute sequentially

Cycle: 1 or 2

Affected flag bits: None

Example:

```
SZA     R01       ;R01→ACC
JP      LOOP      ;Execute this statement when R01 is not 0 and jump to LOOP
JP      LOOP1     ;R01 is 0 time jump, execute this statement and jump to LOOP1
```

SZR **[R]**

Operation: Assign the value of register R to R. If R is 0, inter-hop, otherwise execute sequentially

Cycle: 1 or 2

Affected flag bits: None

Example:

```
SZR     R01       ;R01→R01
JP      LOOP      ;Execute this statement when R01 is not 0 and jump to LOOP
JP      LOOP1     ;R01 is 0 time jump execution of this statement, jump to LOOP1
```

SZINCA **[R]**

Operation: Add 1 to register R and put the result into ACC, if the result is 0, skip the next statement, otherwise execute in order

Cycle: 1 or 2

Affected flag bits: None

Example:

```
SZINCA    R01     ;R01+1→ACC
JP        LOOP    Execute this statement when ACC is not 0 and jump to LOOP
JP        LOOP1   Execute this statement when ACC is 0 and jump to LOOP1
```

SZINCR **[R]**

Operation: Add 1 to register R and put the result into R. If the result is 0, skip the next statement, otherwise execute sequentially

Cycle: 1 or 2

Affected flag bits: None

Example:

```
SZINCR    R01     ;R01+1→R01
JP        LOOP    Execute this statement when R01 is not 0 and jump to LOOP
JP        LOOP1   Execute this statement when R01 is 0 and jump to LOOP1
```

SZDECA **[R]**

Operation: Subtract 1 from register R, put the result into ACC, if the result is 0, skip the next statement, otherwise execute sequentially

Cycle: 1 or 2

Affected flag bits: None

Example:

```
SZDECA    R01     ;R01-1→ACC
JP        LOOP    Execute this statement when ACC is not 0 and jump to LOOP
JP        LOOP1   Execute this statement when ACC is 0 and jump to LOOP1
```

SZDECR **[R]**

Operation: Decrease the register R by 1, and put the result into R. If the result is 0, skip the next statement, otherwise execute in order

Cycle: 1 or 2

Affected flag bits: None

Example:

```
SZDECR    R01     ;R01-1→R01
JP        LOOP    Execute this statement when R01 is not 0 and jump to LOOP
JP        LOOP1   Execute this statement when R01 is 0 and jump to LOOP1
```

TESTZ **[R]**

Operation: Assign the value of R to R to influence the Z flag bit

Cycle: 1

Affected flag bits: Z

Example:

```
TESTZ   R0          ;Assign the value of register R0 to R0 for affecting the Z flag bit
SZB     STATUS,Z    ;Determine the Z flag bit, jump between 0
JP      Add1        ;Jump to address Add1 when register R0 is 0
JP      Add2        ;Jump to address Add2 when register R0 is not 0
```

XORIA **i**

Operation: The immediate number is logically iso-or with ACC and the result is put into ACC

Cycle: 1

Affected flag bits: Z

Example:

```
LDIA    0AH         ;ACC assignment 0AH
XORIA   0FH         Execution result: ACC=05H
```

XORA **[R]**

Operation: Register R performs a logical iso-or operation with ACC, and the result is put into ACC

Cycle: 1

Affected flag bits: Z

Example:

```
LDIA    0AH         ;ACC assignment 0AH
LD      R01,A       ACC value is assigned to R01,R01=0AH
LDIA    0FH         ;ACC assignment 0FH
XORA    R01         Execution result: ACC=05H
```

XORR **[R]**

Operation: Register R performs a logical iso-or operation with ACC, and the result is put into R

Cycle: 1

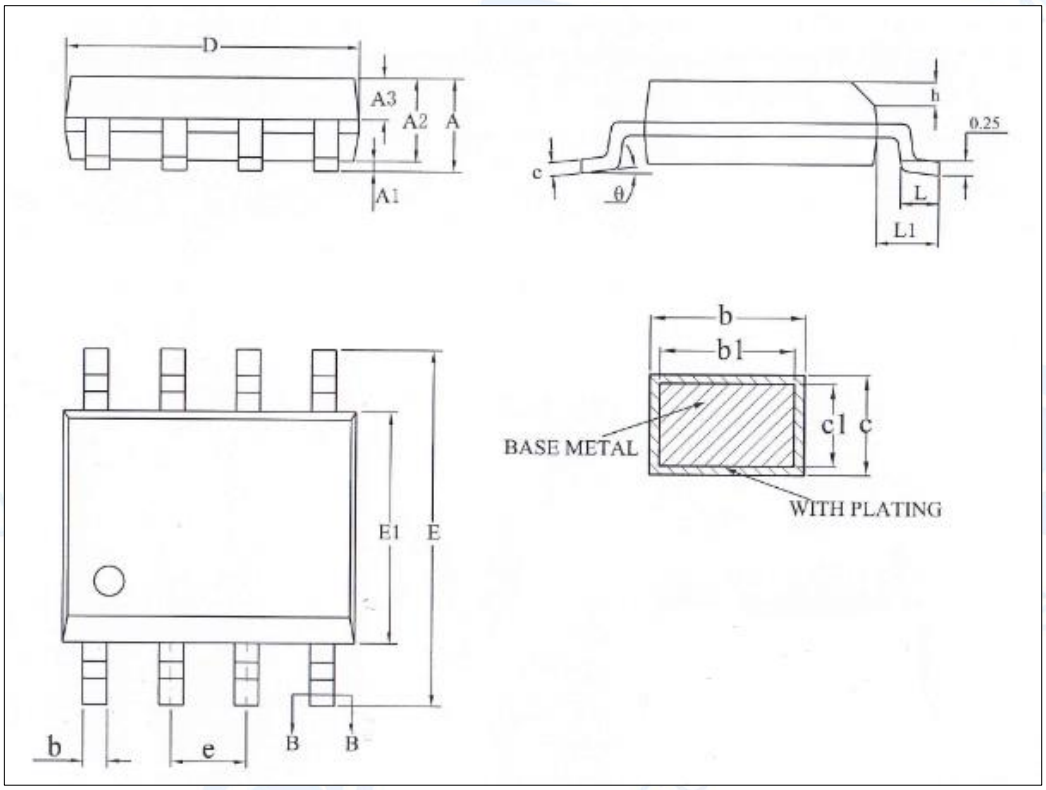
Affected flag bits: Z

Example:

```
LDIA    0AH         ;ACC assignment 0AH
LD      R01,A       ACC value is assigned to R01,R01=0AH
LDIA    0FH         ;ACC assignment 0FH
XORR    R01         ;Execution result: R01=05H
```

15. Package

15.1 SOP8



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

16. Version Revision

Version	Date	Revised Content
V1.0	July 2019	Initial Version
V1.1	August 2019	Corrected the notes on the operation of OPTION_REG when switching before TMR0 and WDT
V1.2	December 2019	Corrected some electrical parameters such as "quiescent current"
V1.3	April 2020	Corrected some errors in the package diagram
V1.3.1	March 2023	Corrected the content of 11 PWM Module