SC8P062B User Manual

Enhanced OTP 8-bit CMOS microcontrollers

Rev.1.0.1

Please note the following CMS IP policy

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

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

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

Table of Contents

1. PRODUCT DESCRIPTION 5
1.1 FEATURES 5
1.2 PRODUCT MODEL LIST 6
1.3 SYSTEM STRUCTURE DIAGRAM 7
1.4 TOP VIEW 8
1.4.1 SC8P062BD606ST 8
1.4.2 SC8P062BD608SP 8
1.4.3 SC8P062BD610SP 8
1.4.4 SC8P062BD614SP 9
1.4.5 SC8P062BD616SP 9
1.4.6 SC8P062BD616NPR 10
1.5 SYSTEM CONFIGURATION REGISTER 12
1.6 ONLINE SERIAL PROGRAMMING 13
2. CENTRAL PROCESSING UNIT (CPU) 14
2.1 MEMORY 14
2.1.1 Program memory 14
2.1.2 Data memory 17
2.2 ADDRESSING 20
2.2.1 Direct addressing 20
2.2.2 Immediate addressing 20
2.2.3 Indirect addressing 20
2.3 STACKING 21
2.4 ACCUMULATOR (ACC) 22
2.4.1 Overview 22
2.4.2 ACC application 22
2.5 PROGRAM STATUS REGISTER (STATUS) 23
2.6 PRE-SCALER (OPTION_REG) 25
2.7 PROGRAM COUNTER (PC) 26
2.8 WATCHDOG TIMER (WDT) 27
2.8.1 WDT period 27
2.8.2 Registers related to watchdog control 28
3. SYSTEM CLOCK 29
3.1 OVERVIEW 29
3.2 SYSTEM OSCILLATOR 31
3.2.1 Internal RC oscillation 31
3.3 RESET TIME 31
3.4 OSCILLATOR CONTROL REGISTER 31
3.5 CLOCK BLOCK DIAGRAM 32
4. RESET 33
4.1 POWER ON RESET 33
4.2 EXTERNAL RESET 33
4.3 BROWN-OUT RESET 34
4.3.1 Overview 34
4.3.2 Improvements for brown-out reset 36
4.4 WATCHDOG RESET 37
5. SLEEP MODE 38
5.1 ENTER SLEEP MODE 38
5.2 WAKE UP FROM SLEEP MODE 38
5.3 INTERRUPT WAKEUP 39
5.4 SLEEP MODE APPLICATION 40
5.5 SLEEP MODE WAKE-UP TIME 40

6. I/O PORTS 41
6.1 I/O PORT STRUCTURE 42
6.1.1 PORTA I/O port 42
6.1.2 PORTB I/O port 43
6.2 PORTA 44
6.2.1 PORTA data and direction 44
6.2.2 PORTA open-drain output control 45
6.2.3 PORTA analog selection control 45
6.2.4 PORTA pull-up resistor 46
6.2.5 PORTA pull-down resistor 46
6.2.6 PORTA interrupt on change 47
6.3 PORTB 48
6.3.1 PORTB data and direction 48
6.3.2 PORTB open-drain output control 49
6.3.3 PORTB analog selection control 49
6.3.4 PORTB pull-up resistor 50
6.3.5 PORTB pull-down resistor 50
6.3.6 PORTB interrupt on change 51
6.4 I/O USAGE 52
6.4.1 Write to I/O port 52
6.4.2 Read from I/O port 52
6.5 CAUTIONS ON I/O PORT USAGE 53
7. INTERRUPT 54
7.1 OVERVIEW 54
7.2 INTERRUPT CONTROL REGISTER 55
7.2.1 Interrupt control register 55
7.2.2 Peripheral interrupt enable register 56
7.2.3 Peripheral interrupt request register 57
7.3 PROTECTION METHODS FOR INTERRUPT 58
7.4 INTERRUPT PRIORITY AND MULTI-INTERRUPT NESTING 58
8. TIMER0 59
8.1 TIMER0 OVERVIEW 59
8.2 WORKING PRINCIPLE OF TIMER0 60
8.2.1 8-bit timer mode 60
8.2.2 8-bit counter mode 60
8.2.3 Software programmable pre-scaler 60
8.2.4 Switch prescaler between TIMER0 and WDT module 60
8.2.5 TIMER0 interrupt 61
8.3 TIMER0 RELATED REGISTERS 62
9. TIMER2 63
9.1 TIMER2 OVERVIEW 63
9.2 WORKING PRINCIPLE OF TIMER2 64
9.3 TIMER2 RELATED REGISTERS 65
10. 10-BIT PWM MODULE 66
10.1 PIN CONFIGURATION 66
10.2 RELEVANT REGISTERS DESCRIPTION 66
10.3 10-BIT PWM REGISTER WRITE SEQUENCE 71
10.4 10-BIT PWM PERIOD 71
10.5 10-BIT PWM DUTY CYCLE 71
10.6 SYSTEM CLOCK FREQUENCY CHANGE 71
10.7 PROGRAMMABLE DEAD TIME DELAY MODE 72
10.8 10-BIT PWM CONFIGURATION 72

11. COMPARATOR (COMP) 73
11.1 BLOCK DIAGRAM 73
11.2 FEATURES 73
11.3 COMPARATOR RELATED FUNCTIONS 74
11.3.1 Comparator functions description 74
11.3.2 Internal resistor voltage divider output 74
11.3.3 Monitoring supply voltage 75
11.3.4 Interrupt usage 75
11.3.5 Interrupt sleep wake-up 76
11.3.6 Result output pin configuration 76
11.4 RELATED REGISTERS 77
12. ANALOG TO DIGITAL CONVERSION (ADC) 78
12.1 ADC OVERVIEW 78
12.2 ADC CONFIGURATION 79
12.2.1 Port configuration 79
12.2.2 Channel selection 79
12.2.3 ADC internal reference voltage 79
12.2.4 ADC reference voltage 79
12.2.5 Conversion clock 80
12.2.6 ADC interrupt 80
12.2.7 Result formatting 80
12.3 WORKING PRINCIPLE OF ADC 81
12.3.1 Start conversion 81
12.3.2 Complete conversion 81
12.3.3 Stop conversion 81
12.3.4 Working principle of ADC in sleep mode 81
12.3.5 A/D conversion procedure 82
12.4 ADC RELATED REGISTERS 83
13. ELECTRICAL PARAMETERS 86
13.1 LIMIT PARAMETERS 86
13.2 DC CHARACTERISTICS 87
13.3 COMPARATOR CHARACTERISTICS 88
13.4 ADC ELECTRICAL CHARACTERISTICS 88
13.5 POWER-ON RESET CHARACTERISTICS 89
13.6 AC ELECTRICAL CHARACTERISTICS 89
13.7 LSE CHARACTERISTICS 90
13.8 EMC CHARACTERISTICS 91
13.8.1 EFT electrical characteristics 91
13.8.2 ESD electrical characteristics 91
13.8.3 Latch-up electrical characteristics 91
14. INSTRUCTION 92
14.1 INSTRUCTION SET 92
14.2 INSTRUCTION DESCRIPTION 94
15. PACKAGE DIMENSIONS 109
15.1 SOT23-6 109
15.2 SOP8 110
15.3 MSOP10 111
15.4 SOP14 112
15.5 SOP16 113
15.6 QFN16(3*3*0.75-0.50) 114
16. REVISION HISTORY 115

1. Product Description

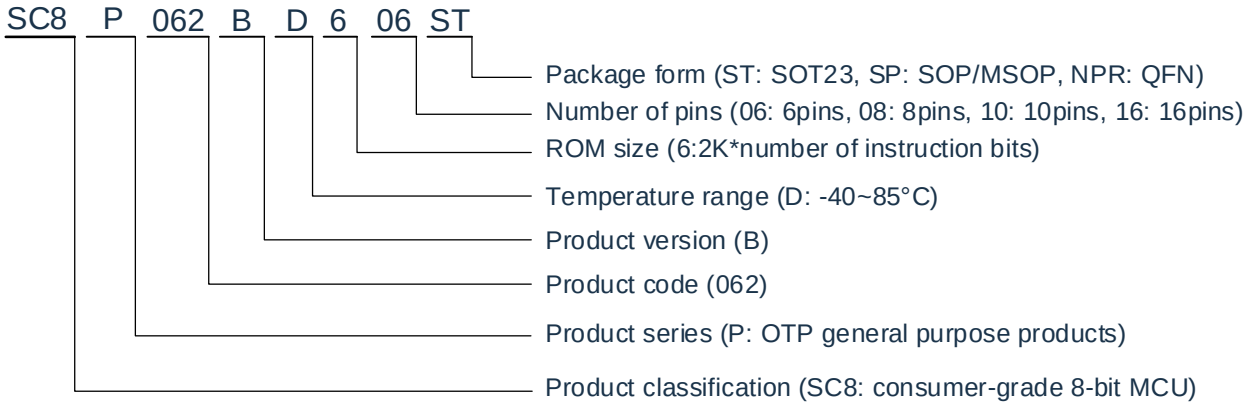
1.1 Features

- Memory
  - OTP: 2K×16Bit
  - Universal RAM: 176×8Bit
- 8-level stack buffer
- Short and clear instruction system (66 instructions)
- Low voltage detection circuit
- Watchdog timer
- Interrupt sources
  - 2 timer interrupts
  - RA, RB ports interrupt on change
  - Other peripheral interrupts
- Timer

  8-bit timers: TIMER0 and TIMER2

  TIMER0 and TIMER2 can select the external 32.768Khz oscillation clock source.
- Comparator module
  - Positive: RA1/resistor divider output
  - Negative: RA1/RA2/RB0/RB1/BG/ resistor divider output
- Operating voltage: V_{LVR4}~5.5V@16MHz/2T
  V_{LVR1}~5.5V@16MHz/4T
  Operating temperature:-40℃~85℃
- Internal RC oscillation: designed frequency of 16MHz
- Instruction period (single or dual instruction)
- PWM module
  - 5-channel PWM with 2 complementary outputs and selectable polarity
  - 4-channel PWM common period, independent duty cycle
  - 1-channel PWM independent period, independent duty cycle
  - 10-bit PWM precision
- LVR can be configured to 1.8V/2V/2.5V/3V
- High-precision 12-bit ADC
  - High-precision 1.2V reference voltage
  - Internal reference sources: 2.0V/2.4V/3.0V

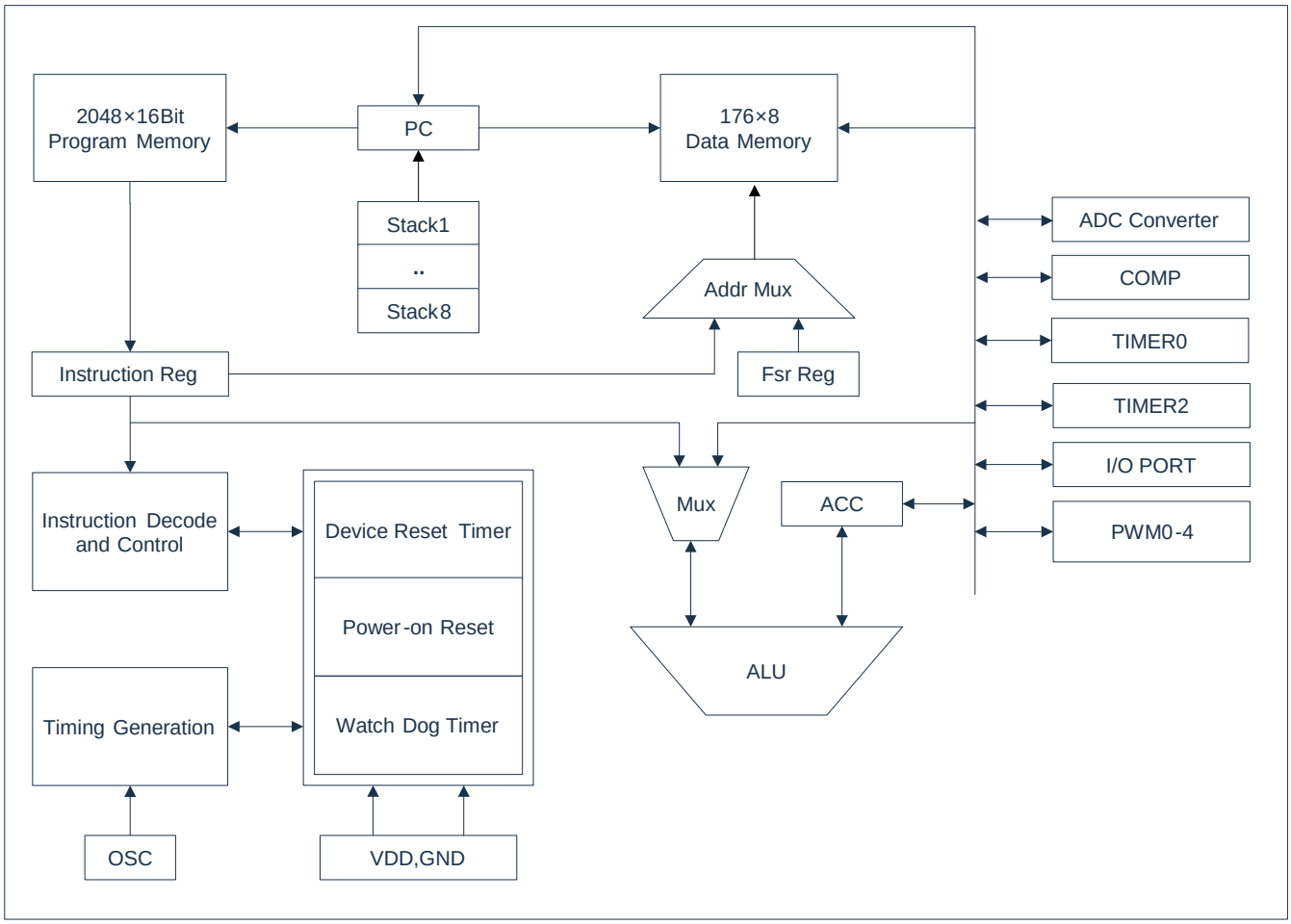
1.2 Product model list



Model description

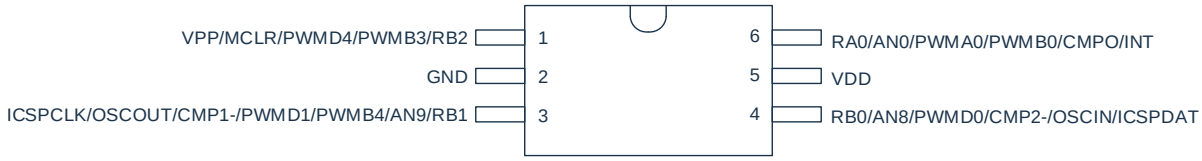
PRODUCT	ROM	RAM	PWM	ACOMP	I/O	ADC	TIMER	PACKAGE
SC8P062BD606ST	2K×16	176×8	3	1	4	12Bit×3	2	SOT23-6
SC8P062BD608SP	2K×16	176×8	5	1	6	12Bit×5	2	SOP8
SC8P062BD610SP	2K×16	176×8	5	1	8	12Bit×7	2	MSOP10
SC8P062BD614SP	2K×16	176×8	5	1	12	12Bit×11	2	SOP14
SC8P062BD616SP	2K×16	176×8	5	1	14	12Bit×13	2	SOP16
SC8P062BD616NPR	2K×16	176×8	5	1	14	12Bit×13	2	QFN16

1.3 System structure diagram



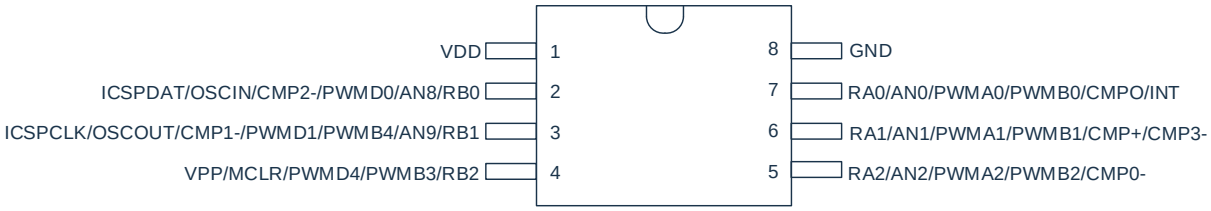
1.4 Top view

1.4.1 SC8P062BD606ST



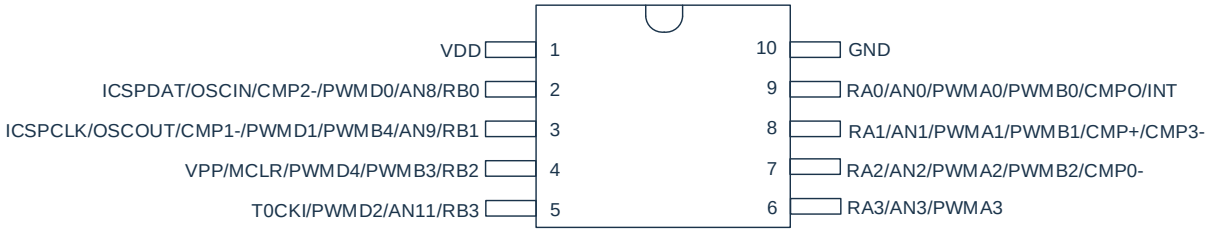
SC8P062BD606ST

1.4.2 SC8P062BD608SP



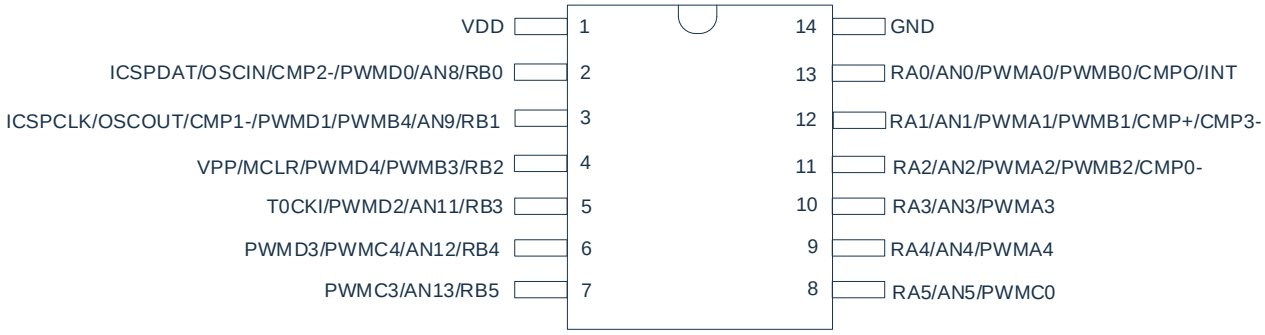
SC8P062BD608SP

1.4.3 SC8P062BD610SP



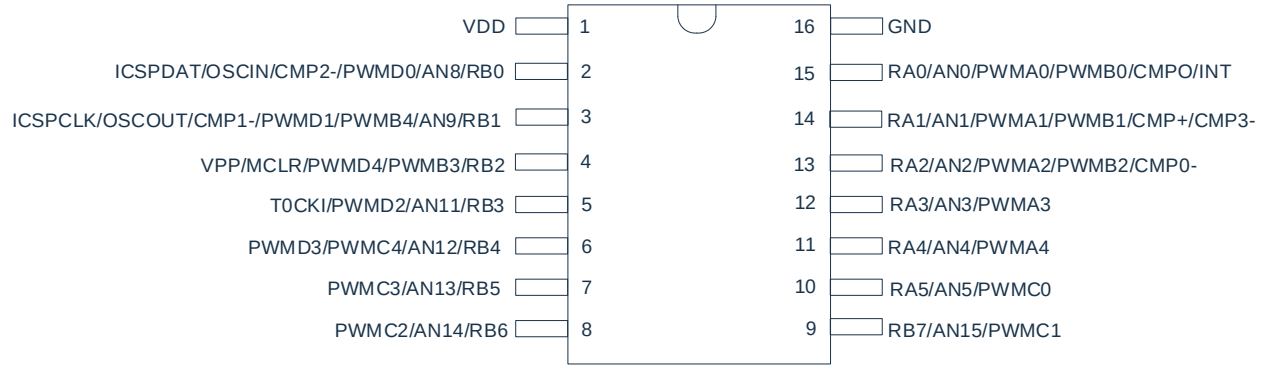
SC8P062BD610SP

1.4.4 SC8P062BD614SP



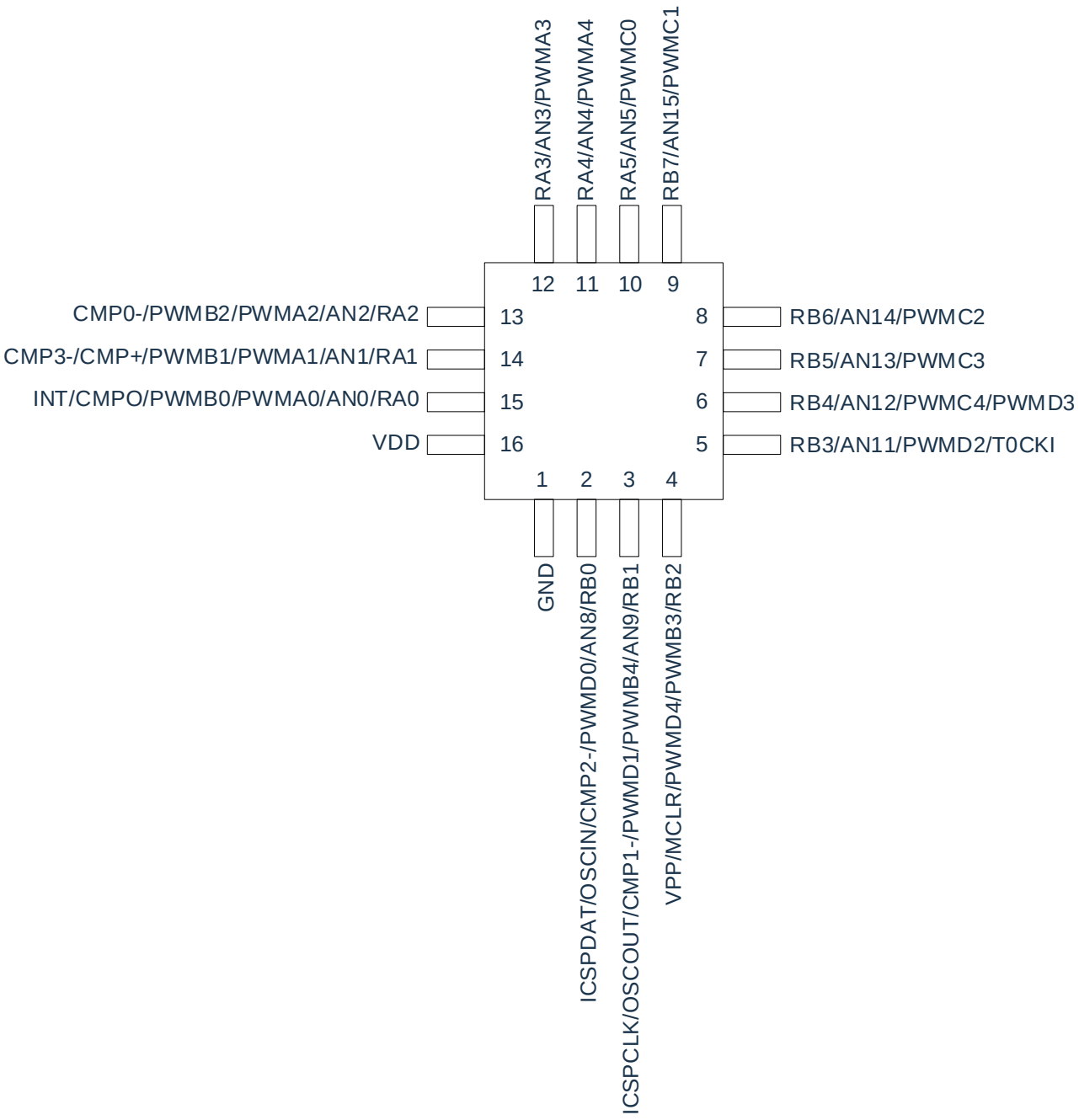
SC8P062BD614SP

1.4.5 SC8P062BD616SP



SC8P062BD616SP

1.4.6 SC8P062BD616NPR



SC8P062BD616NPR

SC8P062B pin description:

Pin name	IO type	Pin description
VDD,GND	P	Supply voltage input pin, ground pin
RA0-RA5	I/O	Programmable as input pin, push-pull or open-drain output pin, with pull-up and pull-down resistor function, and interrupt-on-change function.
RB0-RB1,RB3-RB7	I/O	Programmable as input pin, push-pull or open-drain output pin, with pull-up and pull-down resistor function, and interrupt-on-change function.
RB2	I/O	Programmable as input pin, open-drain output pin, with pull-up and pull-down resistor function, and interrupt-on-change function.
ICSPCLK/ICSPDAT	I/O	Programmable clock/data pin
VPP	I	Programmable high-voltage input pin
PWMA0-PWMA4	O	PWM output pin
PWMB0-PWMB4	O	PWM output pin
PWMC0-PWMC4	O	PWM output pin
PWMD0-PWMD4	O	PWM output pin
AN0-AN5, AN8, AN9, AN11-AN15	I	12-bit ADC input pin
INT	I	External interrupt input pin
CMP+	I	Comparator positive input pin
CMP0-, CMP1-, CMP2-, CMP3-	I	Comparator negative input pin
CMPO	O	Comparator result output pin
T0CKI	I	TIMER0 external clock input pin
OSCIN/OSCOUT	I/O	32.768K crystal oscillator input/output pin
MCLR	I	External reset input pin

1.5 System configuration register

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

1. WDT (watchdog selection)
   - ENABLE Enable WDT
   - DISABLE Disable WDT
2. PROTECT (encrypted)
   - DISABLE ROM code is not encrypted
   - ENABLE ROM code is encrypted, and the value read out by the programmed emulator will be uncertain after encryption
3. LVR_SEL (low-voltage detection selection)
   - 1.8V
   - 2.0V
   - 2.5V
   - 3.0V
4. F_{CPU}_DIV (instruction clock divison)
   - 4T Divided by 4, $F_{CPU}=F_{SYS}/4$
   - 2T Divided by 2, $F_{CPU}=F_{SYS}/2$
5. EXT_RESET (external reset port selection)
   - DISABLE Disable external reset function, RB2 is a normal IO port.
   - ENABLE Enable external reset function, RB2 is an external reset port.

1.6 Online serial programming

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

- Power line
- Ground line
- Data line
- Clock line
- High-voltage line

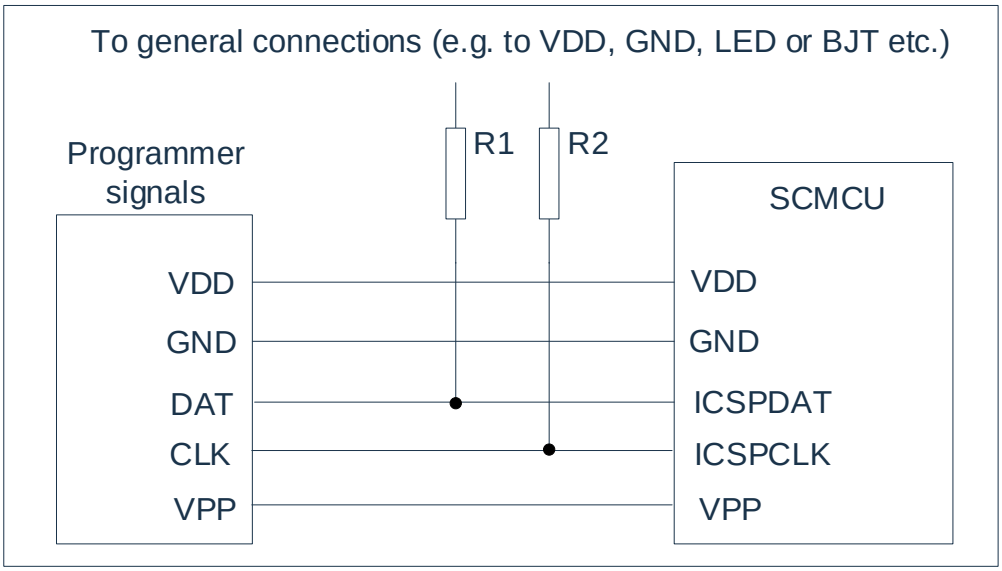


Figure 1-1: Typical online serial programming connection

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

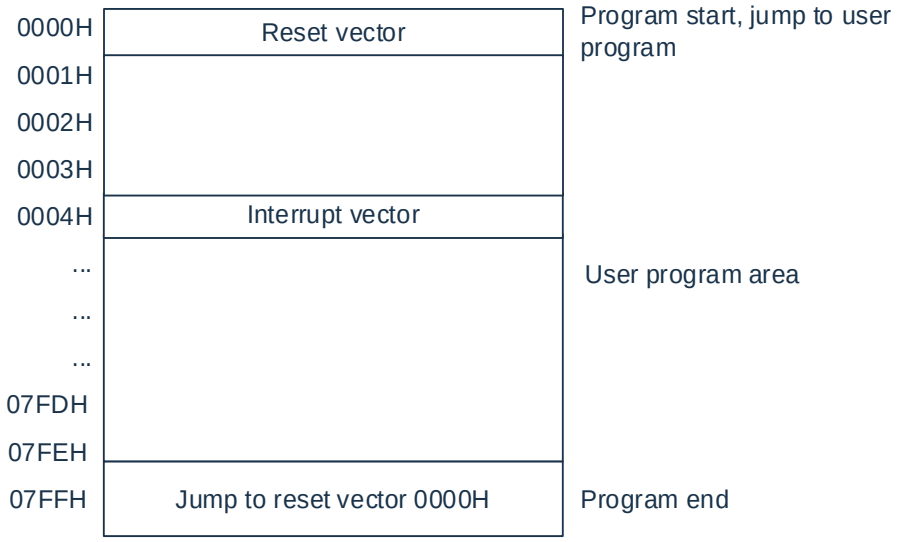
2. Central Processing Unit (CPU)

2.1 Memory

2.1.1 Program memory

SC8P062B program memory:

OTP: 2K



2.1.1.1Reset vector (0000H)

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

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

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

Example: define reset vector

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

2.1.1.2Interrupt vector

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

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

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

Note: MCU does not provide specific pop and push instructions, so users need to protect interrupt scene.

Example: interrupt-in protection

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

Example: interrupt-out restore

```
POP:
        SWAPA   STATUS_BAK      ;swap the half-byte data from STATUS_BAK to ACC
        LD      STATUS,A        ;load 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                     ;return
```

2.1.1.3Jump table

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

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

Example: correct illustration of multi-address jump

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

Example: wrong illustration of multi-address jump

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

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

2.1.2 Data memory

SC8P062B data memory list

Addr	BANK0	Addr	BANK1	Addr	BANK2	Addr	BANK3	Addr
00H	INDF	80H	INDF	100H	-----	180H	-----	180H
01H	OPTION_REG	81H	TMR0	101H	-----	181H	-----	181H
02H	PCL	82H	PCL	102H	-----	182H	-----	182H
03H	STATUS	83H	STATUS	103H	-----	183H	-----	183H
04H	FSR	84H	FSR	104H	-----	184H	-----	184H
05H	TRISB	85H	TRISA	105H	-----	185H	-----	185H
06H	PORTB	86H	PORTA	106H	-----	186H	-----	186H
07H	WPDB	87H	WPDA	107H	-----	187H	-----	187H
08H	WPUB	88H	WPUA	108H	-----	188H	-----	188H
09H	IOCB	89H	IOCA	109H	-----	189H	-----	189H
0AH	PCLATH	8AH	PCLATH	10AH	-----	18AH	-----	18AH
0BH	INTCON	8BH	INTCON	10BH	-----	18BH	-----	18BH
0CH	ODCONB	8CH	ODCONA	10CH	-----	18CH	-----	18CH
0DH	PIR1	8DH	-----	10DH	-----	18DH	-----	18DH
0EH	PIE1	8EH	-----	10EH	-----	18EH	-----	18EH
0FH	CMPCON0	8FH	-----	10FH	-----	18FH	-----	18FH
10H	CMPCON1	90H	-----	110H	-----	190H	-----	190H
11H	PR2	91H	-----	111H	-----	191H	-----	191H
12H	TMR2	92H	-----	112H	-----	192H	-----	192H
13H	T2CON	93H	ANSEL0	113H	-----	193H	-----	193H
14H	OSCCON	94H	ANSEL1	114H	-----	194H	-----	194H
15H	PWMCON0	95H	ADCON0	115H	-----	195H	-----	195H
16H	PWMCON1	96H	ADCON1	116H	-----	196H	-----	196H
17H	PWMTL	97H	----	117H	-----	197H	-----	197H
18H	PWMTH	98H	ADRESL	118H	-----	198H	-----	198H
19H	PWMD0L	99H	ADRESH	119H	-----	199H	-----	199H
1AH	PWMD1L	9AH	-----	11AH	-----	19AH	-----	19AH
1BH	PWMD4L	9BH	PWMD2L	11BH	-----	19BH	-----	19BH
1CH	PWMT4L	1CH	PWMD3L	9CH	-----	11CH	-----	19CH
1DH	PWMCON2	1DH	PWM23DT	9DH	-----	11DH	-----	19DH
1EH	PWMD01H	1EH	PWMD23H	9EH	-----	11EH	-----	19EH
1FH	PWM01DT	1FH	-----	9FH	-----	11FH	-----	19FH
20H	Universal register 96 bytes	A0H	Universal register 80 bytes	120H	-----	1A0H	-----	1A0H
BFH		BFH		16FH		1EFH		1EFH
F0H		F0H	Rapid storage area 70H-7FH	170H	-----	1F0H	-----	1F0H
--		--		--		--		--
7FH		FFH		17FH		1FFH		1FFH

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

SC8P062B special function register summary Bank0

Addr	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
00H	INDF	Addressing this unit will address data memory (not a physical register) using the contents of the FSR.								xxxxxxxx
01H	OPTION_REG	T0LSE_EN	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	01111011
02H	PCL	Program counter low bytes								00000000
03H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
04H	FSR	Indirect data memory address pointer								xxxxxxxx
05H	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	11111111
06H	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxxxxxx
07H	WPDB	WPDB7	WPDB6	WPDB5	WPDB4	WPDB3	----	WPDB1	WPDB0	00000-00
08H	WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0	00000000
09H	IOCB	IOCB7	IOCB6	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	IOCB0	00000000
0AH	PCLATH	----	----	----	----	----	Write buffer for the high 3 bits of the program counter			-----000
0BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
0CH	ODCONB	ODCONB7	ODCONB6	ODCONB5	ODCONB4	ODCONB3	----	ODCONB1	ODCONB0	00000-00
0DH	PIR1	----	----	CMPIF	PWMIF	RAIF	----	TMR2IF	ADIF	--000-00
0EH	PIE1	----	----	CMPIE	PWMIE	RAIE	----	TMR2IE	ADIE	--000-00
0FH	CMPCON0	CMPEN	CMPPS	CMPNS2	CMPNS1	CMPNS0	CMPNV	CMPOUT	CMPOEN	00000000
10H	CMPCON1	CMPIM	AN_EN	RBIAS_H	RBIAS_L	LVDS<3:0>				00000000
11H	PR2	TIMER2 period register								11111111
12H	TMR2	TIMER2 module register								00000000
13H	T2CON	CLK_SEL	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	00000000
14H	OSCCON	----	IRCF2	IRCF1	IRCF0	----	----	SWDTEN	-----	-101--1-
15H	PWMCON0	CLKDIV<2:0>			PWM4EN	PWM3EN	PWM2EN	PWM1EN	PWM0EN	00000000
16H	PWMCON1	PWMIO_SEL[1:0]		PWM2DTEN	PWM0DTEN	----	----	DT_DIV<1:0>		0000--00
17H	PWMTL	PWM0~PWM3 period low 8-bit register								00000000
18H	PWMTH	----	----	PWM4D<9:8>		PWM4T<9:8>		PWMT<9:8>		--000000
19H	PWMD0L	PWM0 duty cycle low 8 bits								00000000
1AH	PWMD1L	PWM1 duty cycle low 8 bits								00000000
1BH	PWMD4L	PWM4 duty cycle low 8 bits								00000000
1CH	PWMT4L	PWM4 period low 8-bit register								00000000
1DH	PWMCON2	----	----	----	PWM4DIR	PWM3DIR	PWM2DIR	PWM1DIR	PWM0DIR	---00000
1EH	PWMD01H	----	----	PWMD1<9:8>		----	----	PWMD0<9:8>		--00--00
1FH	PWM01DT	----	----	PWM01DT<5:0>						--000000

SC8P062B special function register summary Bank1

Addr	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
80H	INDF	Addressing this unit will address data memory (not a physical register) using the contents of the FSR.								xxxxxxxx
81H	TMR0	TIMER0 data register								xxxxxxxx
82H	PCL	Program counter low bytes								00000000
83H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
84H	FSR	Indirect data memory address pointer								xxxxxxxx
85H	TRISA	----	----	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	--111111
86H	PORTA	----	----	RA5	RA4	RA3	RA2	RA1	RA0	--xxxxxx
87H	WPDA	----	----	WPDA5	WPDA4	WPDA3	WPDA2	WPDA1	WPDA0	--000000
88H	WPUA	----	----	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0	--000000
89H	IOCA	----	----	IOCA5	IOCA4	IOCA3	IOCA2	IOCA1	IOCA0	--000000
8AH	PCLATH	----	----	----	----	----	Write buffer for the high 3 bits of the program counter			-----000
8BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
8CH	ODCONA	----	----	ODCONA5	ODCONA4	ODCONA3	ODCONA2	ODCONA1	ODCONA0	--000000
93H	ANSEL0	----	----	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0	--000000
94H	ANSEL1	ANS15	ANS14	ANS13	ANS12	ANS11	----	ANS9	ANS8	00000-00
95H	ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/ $\overline{\text{DONE}}$	ADON	00000000
96H	ADCON1	ADFM	CHS4	----	----	----	LDO_EN	LDO_SEL1	LDO_SEL0	00---000
98H	ADRESL	A/D result register low bytes								xxxxxxxx
99H	ADRESH	A/D result register high bytes								xxxxxxxx
9BH	PWMD2L	PWM2 duty cycle low 8 bits								00000000
9CH	PWMD3L	PWM3 duty cycle low 8 bits								00000000
9DH	PWM23DT	----	----	PWM23 dead delay time						--000000
9EH	PWMD23H	----	----	PWMD3<9:8>		----	----	PWMD2<9:8>		--00--00

2.2 Addressing

2.2.1 Direct addressing

The RAM is operated through the operation register (ACC).

Example: The value of ACC is loaded into the 30H register.

```
LD      30H,A
```

Example: The value of register 30H is loaded to ACC.

```
LD      A,30H
```

2.2.2 Immediate addressing

Load the immediate number to the operation register (ACC)

Example: Load the immediate number 12H to ACC

```
LDIA    12H
```

2.2.3 Indirect addressing

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

Example: application of FSR and INDF

```
        LDIA    30H
        LD      FSR,A       ;point to 30H for indirect addressing
        CLR     INDF        ;clearing INDF actually clears the RAM at the address
                            pointed to by FSR, which is address 30H.
```

Example: Indirect addressing to clear RAM (20H-7FH).

```
        LDIA    1FH
        LD      FSR,A       ;point to 1FH for indirect addressing
LOOP:
        INCR    FSR         ;address add 1, and initial address is 20H
        CLR     INDF        ;clear the address pointed to by the FSR.
        LDIA    7FH
        SUBA    FSR
        SNZB    STATUS,C    ;clear until the address of FSR is 7FH
        JP      LOOP
```

2.3 Stacking

The chip has an 8-level stack buffer. The stack buffer is neither part of the data memory nor the program memory, and it cannot be read or written. Operations on the stack are performed through the stack pointer (SP), which also cannot be read or written. After a system reset, the stack pointer points to the top of the stack. When a subroutine call or interrupt occurs, the program counter (PC) value is pushed onto the stack buffer. Upon returning from an interrupt or subroutine, the value is popped back to the program counter (PC). The following diagram illustrates its operation.

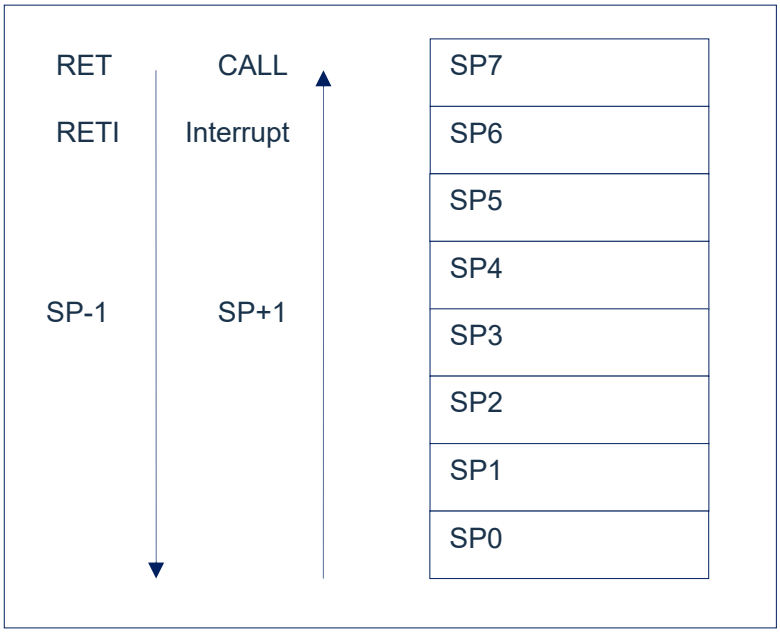


Figure 2-2: Stack buffer operation

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

Note: The stack buffer has only 8 levels. If the stack is full and an unmaskable interrupt occurs, only the interrupt flag will be recorded, and the interrupt response will be suppressed until the stack pointer decreases. The interrupt will then be acknowledged. This feature prevents stack overflow caused by interrupts. Similarly, if the stack is full and a subroutine call occurs, a stack overflow will happen. The content that entered the stack first will be lost, and only the last 8 return addresses will be retained. Therefore, users should be cautious when writing programs to avoid unexpected behavior or stack overflow.

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 can also control STATUS to represent the status of the calculation result.

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

2.4.2 ACC application

Example: use ACC for data transferring

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

Example: use ACC for immediate addressing

```
        LDIA    30H         ;load 30H to ACC
        ANDIA   30H         ;perform 'AND' on ACC and 30H
                            ;save the result to ACC
        XORIA   30H         ;perform 'XOR' on ACC and 30H
                            ;save the result to ACC
```

Example: use ACC as the first operand of a dual operand instruction

```
        HSUBA   R01         ;ACC-R01, save the result to ACC
        HSUBR   R01         ;ACC-R01, save the result to R01
```

Example: use ACC as the second operand of a dual operand instruction

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

2.5 Program status register (STATUS)

STATUS register includes:

- ALU arithmetic status
- Reset status

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

For example, CLRSTATUS will clear higher 3 bits and set the Z bit to 1. Hence the value of STATUS will be 000uu1uu (u will not change). It is recommended to only use CLRB, SETB, SWAPA and SWAPR instructions to change STATUS register because these will not affect any status bits.

Program status register STATUS (03H)

03H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C
R/W	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 (indirect addressing)
	1=	Unused
	0=	Bank0 and Bank1 (00h~FFh)
Bit6~Bit5	RP[1:0]:	Memory select bit
	00=	Select Bank0
	01=	Select Bank1
	10=	Unused
	11=	Unused
Bit4	TO:	Time out bit;
	1=	Power on or CLRWDT instruction or STOP instruction
	0=	WDT time out
Bit3	PD:	Power down bit
	1=	Power on or CLRWDT instruction
	0=	Execute STOP instruction
Bit2	Z:	Result bit
	1=	The result of an arithmetic or logical operation is zero
	0=	The result of an arithmetic or logical operation is not zero
Bit1	DC:	Half carry bit/borrow bit
	1=	Carry happens from the lower 4 bits to the higher bits, or no borrow from the lower 4 bits.
	0=	No carry from the lower 4 bits to the higher bits, or borrow from the lower 4 bits to the higher bits.
Bit0	C:	Carry/borrow bit
	1=	A carry from the highest bit, or no borrow.
	0=	No carry from the highest bit, or a borrow hanppens.

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

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

Table of events affecting PD and TO

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

Status of TO/PD after reset

2.6 Pre-scaler (OPTION_REG)

OPTION_REG register is a readable/writable register that contains various control bits for configuration.

- WDT pre-scaler
- External interrupt trigger edge

Prescaler control register OPTION_REG (01H)

01H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION_REG	T0LSE_EN	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	1	1	1	1	0	1	1

Bit7 T0LSE_EN: TIMER0 clock source selection F_{LSE} enable bit
0= The clock source for TIMER0 is determined by T0CS.
1= The TIMER0 clock source selects F_{LSE}.
Bit6 INTEDG: Trigger interrupt edge selection bit
0= INT pin falling edge trigger interrupt.
1= INT pin rising edge trigger interrupt.
Bit5 T0CS: TIMER0 clock source select bit
0= Internal instruction period clock (F_{CPU})
1= Transition edge on the T0CKI pin
Bit4 T0SE: TIMER0 clock source edge select bit
0= Increase when T0CKI pin signal transited from low to high
1= Increase when T0CKI pin signal transited from high to low
Bit3 PSA: Pre-scaler allocation bit
0= Allocate pre-scaler to TIMER0 module
1= Allocate pre-scaler to WDT
Bit2~Bit0 PS2~PS0: Pre-allocation parameter configure bit

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

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

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

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

2.7 Program counter (PC)

The Program Counter (PC) controls the execution sequence of instructions in the program memory (OTP). It can address the entire range of the OTP. After fetching an instruction, the PC automatically increments by one, pointing to the address of the next instruction. However, during operations such as jumps, conditional jumps, assigning a value to PCL, subroutine calls, reset initialization, interrupts, interrupt returns, or subroutine returns, the PC will load an address related to the instruction rather than the address of the next instruction.

When a conditional jump instruction is encountered and the jump condition is met, the next instruction fetched during the current instruction cycle will be discarded, and a no-op cycle will be inserted before the correct instruction can be fetched. Otherwise, the next instruction will be executed in sequence.

The Program Counter (PC) is 11 bits wide. The lower 8 bits can be accessed by the user through the PCL (02H) register, while the upper 3 bits are not user-accessible. It can address 2Kx16-bit program memory. Assigning a value to PCL will result in a short jump, with a jump range limited to 256 addresses within the current page.

Note: When the programmer uses PCL to make a short jump, the programmer must first load a value to the PC high bit buffer register PCLATH.

The PC values for several special cases are given below

Reset	PC=0000;
Interrupt	PC=0004 (original PC+1 will be pushed onto the stack automatically);
CALL	PC=Program specified address (original PC+1 will be pushed onto the stack automatically);
RET, RETI, RETI	PC=Value from stack;
PCL operation	PC[10:8] unchanged, PC[7:0]=user defined value;
JP	PC=Program specified value;
Other instructions	PC=PC+1;

2.8 Watchdog timer (WDT)

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

2.8.1 WDT period

The WDT uses an 8-bit prescaler. After any reset, the WDT overflow period is 128ms, and the overflow period is calculated as 16ms*the prescaler factor. To change the WDT period, the OPTION_REG register can be configured. The WDT overflow period is affected by factors such as environmental temperature, supply voltage, and other parameters.

The CLRWDT and STOP instructions will clear the WDT timer and the counter value in the prescaler. The WDT is typically used to prevent the system from malfunctioning, or more specifically, to prevent the microcontroller program from running out of control. Under normal conditions, the WDT should be cleared before it overflows using the CLRWDT instruction to prevent a reset. If the program malfunctions due to some interference and the CLRWDT instruction is not executed before the WDT overflows, the WDT will overflow, triggering a reset. This helps to restart the system and regain control. If the reset is caused by a WDT overflow, the TO bit in the STATUS register will be cleared. The user can check this bit to determine whether the reset was caused by a WDT overflow.

Note:

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

2.8.2 Registers related to watchdog control

Oscillation control register OSCCON (14H)

14H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OSCCON	---	IRCF2	IRCF1	IRCF0	---	---	SWDTEN	---
R/W	---	R/W	R/W	R/W	---	---	R/W	---
Reset value	---	1	0	1	---	---	1	---

Bit7 Unused

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

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

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

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

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

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

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

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

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

Bit3~Bit2 Unused

Bit1 SWDTEN: Software enable or disable watchdog timer bit

1= Enable WDT

0= Disable WDT

Bit0 Unused

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

3. System Clock

3.1 Overview

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

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

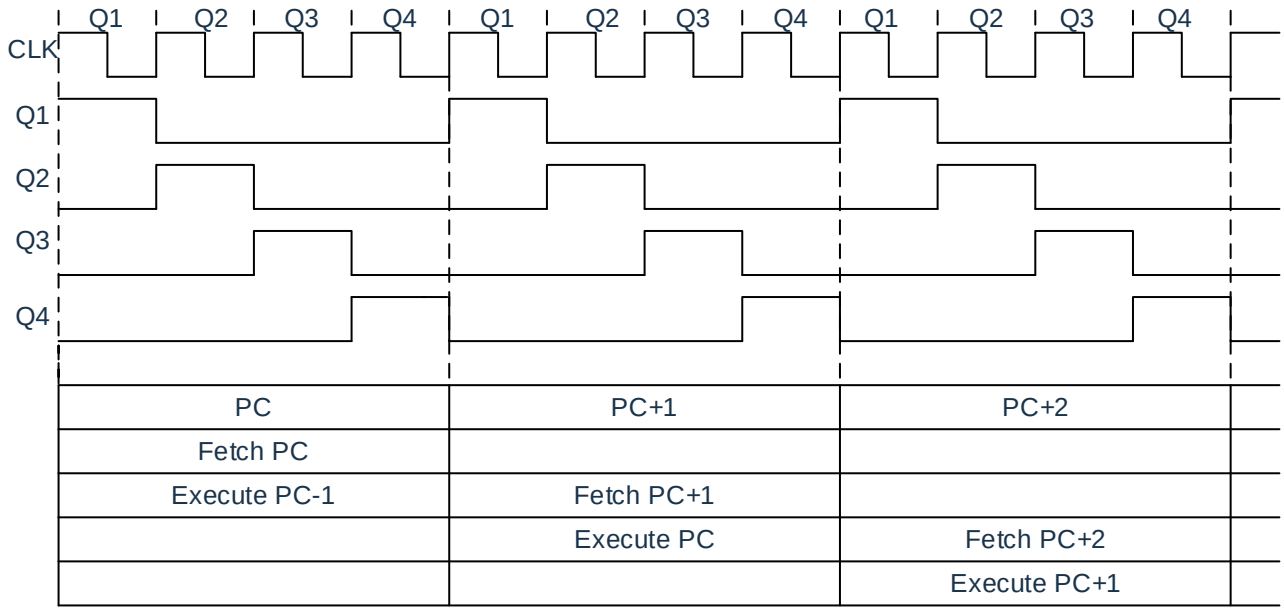


Figure 3-1: Clock and instruction cycle timing chart (F_{CPU}_DIV=4T)

The following lists the relationship between the system operating frequency and instruction speed when F_{CPU}_DIV = 4T:

System frequency (F_{SYS})	Dual instruction period	Single instruction period
1MHz	8μs	4μs
2MHz	4μs	2μs
4MHz	2μs	1μs
8MHz	1μs	500ns
16MHz	500ns	250ns

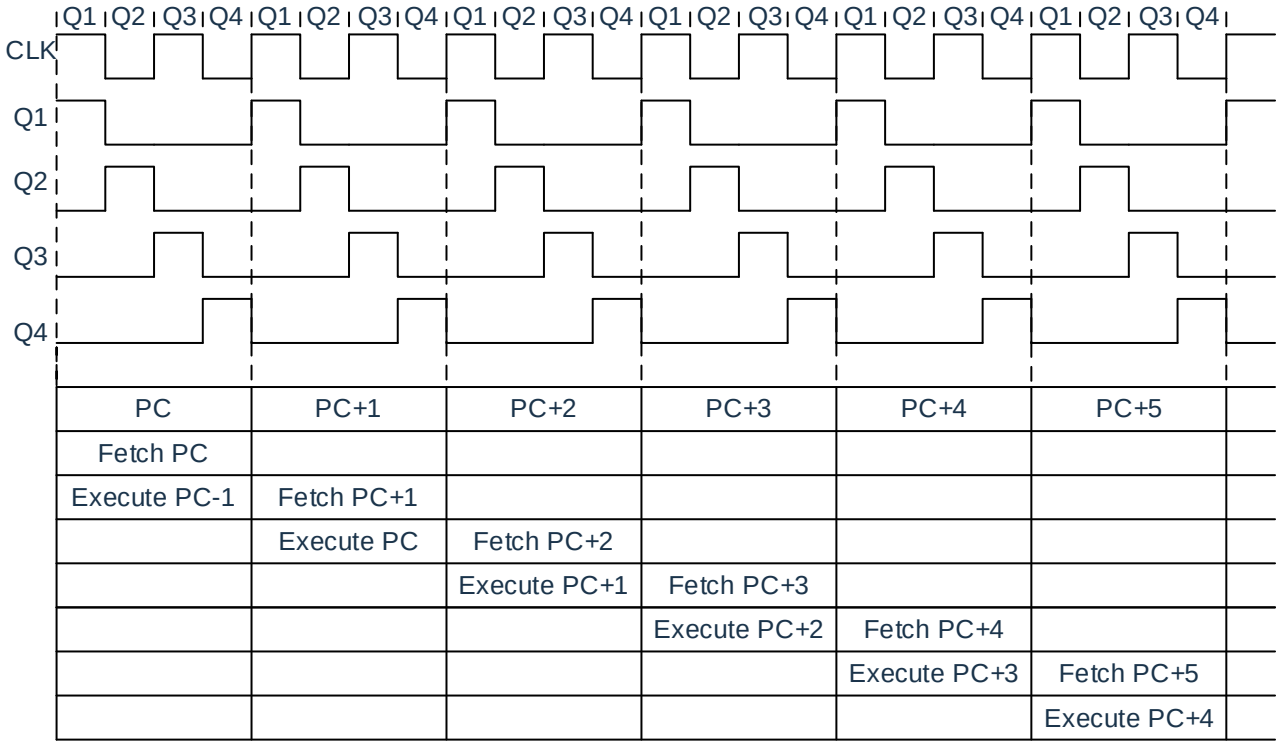


Figure 3-2: Clock and instruction cycle timing chart (F_{CPU}_DIV=2T)

The following lists the relationship between the system operating frequency and instruction speed when F_{CPU}_DIV = 2T:

System frequency (F_{SYS})	Dual instruction period	Single instruction period
1MHz	4μs	2μs
2MHz	2μs	1μs
4MHz	1μs	500ns
8MHz	500ns	250ns
16MHz	250ns	125ns

3.2 System oscillator

The chip has one type of oscillation: internal RC oscillation.

3.2.1 Internal RC oscillation

The default oscillation mode of the chip is internal RC oscillation, and the oscillation frequency is fixed at 16MHz. On this basis, the operating frequency of the chip can be set through the OSCCON register.

3.3 Reset time

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

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

3.4 Oscillator control register

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

Oscillator control register OSCCON (14FH)

14H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OSCCON	---	IRCF2	IRCF1	IRCF0	---	---	SWDTEN	---
R/W	---	R/W	R/W	R/W	---	---	R/W	---
Reset value	---	1	0	1	---	---	1	---

Bit7	Unused	
Bit6~Bit4	IRCF<2:0>:	Internal oscillator frequency selection bit
	111=	$F_{SYS} = F_{HSI}/1$
	110=	$F_{SYS} = F_{HSI}/2$
	101=	$F_{SYS} = F_{HSI}/4$ (default)
	100=	$F_{SYS} = F_{HSI}/8$
	011=	$F_{SYS} = F_{HSI}/16$
	010=	$F_{SYS} = F_{HSI}/32$
	001=	$F_{SYS} = F_{HSI}/64$
	000=	F_{SYS} = 32KHz (LFINTOSC)
Bit3~Bit2	Unused	
Bit1	SWDTEN:	Software enable or disable watchdog timer bit
	1=	Enable WDT
	0=	Disable WDT
Bit0	Unused	

3.5 Clock block diagram

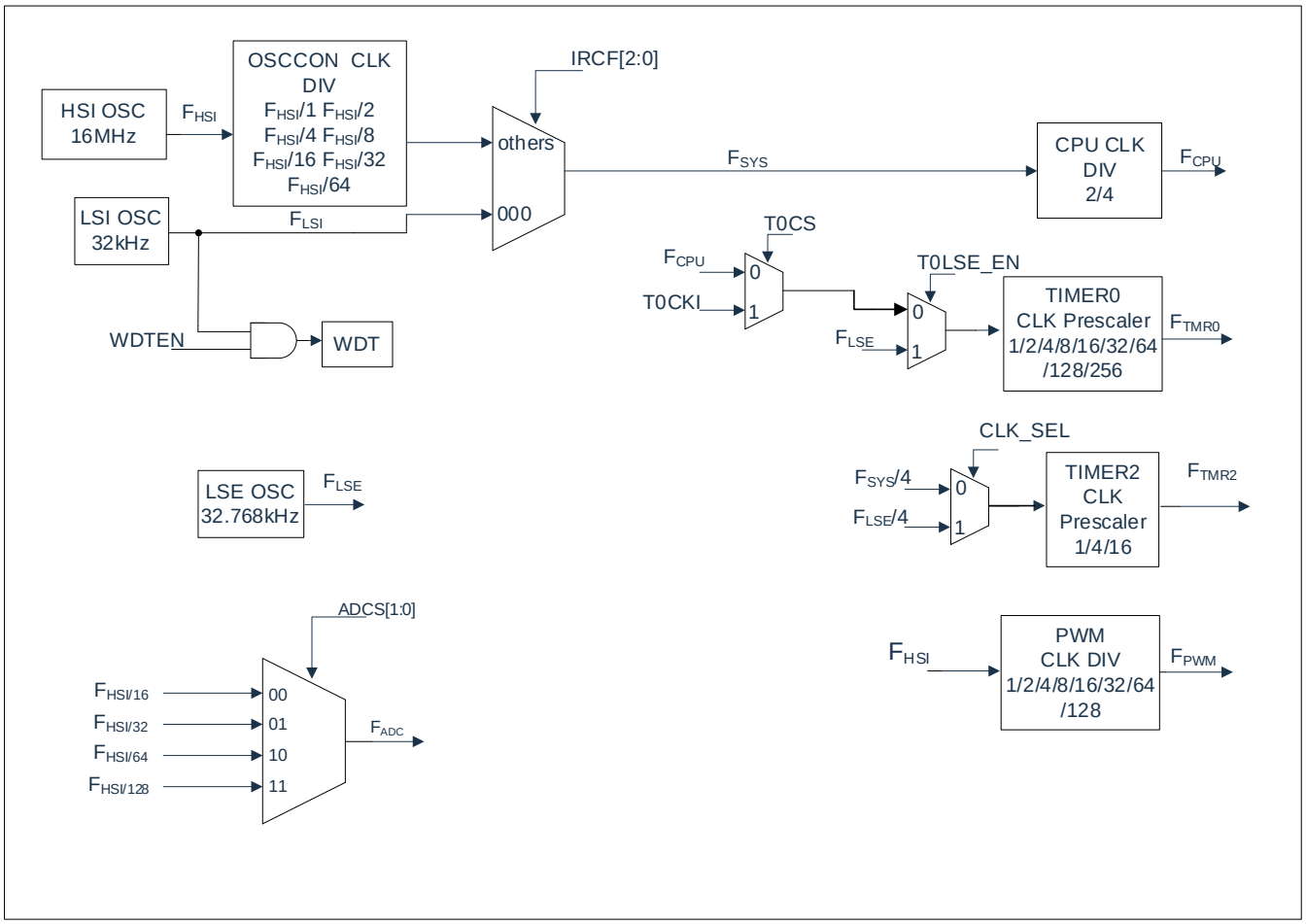


Figure 3-3: Clock block diagram

4. Reset

The chip can be reset in the following four ways:

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

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

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

4.1 Power on reset

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

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

4.2 External reset

The SC8P062B supports an external reset function, which can be configured through the CONFIG settings to use RB2 as the reset pin. When configured, RB2 automatically enables the internal weak pull-up. If RB2 is pulled low, the chip will undergo a reset.

4.3 Brown-out reset

4.3.1 Overview

Brown-out Reset is designed to handle situations where external factors cause a drop in system voltage (e.g., interference or changes in external load). A voltage drop may cause the system to enter a dead zone, where the power supply can no longer meet the minimum operating voltage requirements of the system.

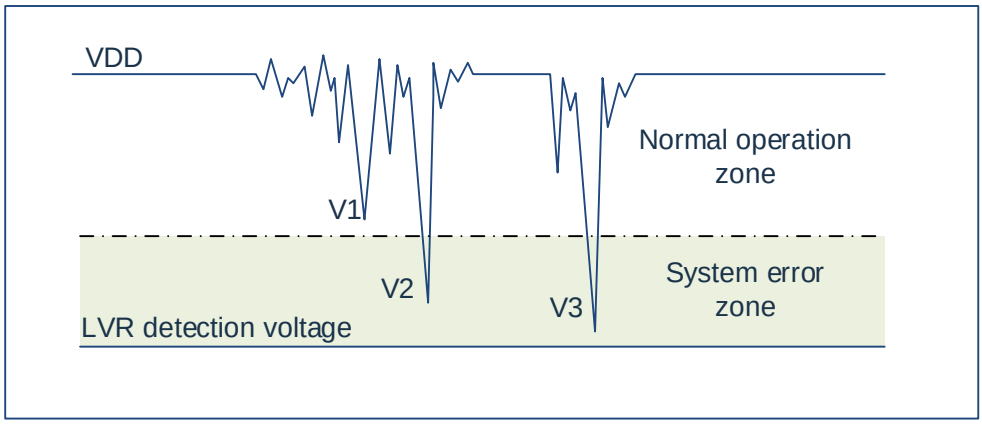


Figure 4-1: Brown-out reset

The diagram above illustrates a typical Brown-out Reset (BOR) scenario. In this diagram, VDD is subjected to severe interference, causing the voltage to drop significantly. The region above the dashed line represents the normal operating state of the system, while the region below the dashed line indicates an unstable or unknown working state, referred to as the dead zone. When VDD drops to V1, the system remains in a normal state. However, if VDD drops to V2 or V3, the system enters the dead zone, which can lead to errors.

The system may enter the dead zone under the following circumstances:

- DC applications:
  - In DC-powered applications, the system is generally powered by a battery. When the battery voltage is too low, or the microcontroller is driving a load, the system voltage may drop and enter the dead zone. In this case, the power supply will not fall further to the Low Voltage Detection (LVD) threshold, so the system stays in the dead zone.
- AC applications:
  - In AC-powered systems, the DC voltage is affected by noise from the AC power supply. When the external load is too high, such as when driving a motor, the interference generated by the load also impacts the DC power supply. If VDD drops below the minimum operating voltage due to interference, the system may enter an unstable operating state.
  - In AC applications, the system startup and power-down durations are typically longer. During power-up, the sequence protection ensures normal power-up, but during the power-down process, the situation is similar to that in DC applications. After the AC power is turned off, the VDD voltage slowly decreases, potentially entering the dead zone.

As shown in the diagram, the normal operating voltage range is typically higher than the system reset voltage. The reset voltage is determined by the Low Voltage Reset (LVR) threshold. When the system's

operating speed increases, the minimum operating voltage also increases. However, since the reset voltage is fixed, a voltage range exists between the minimum operating voltage and the reset voltage where the system cannot function properly, nor can it reset. This voltage range is called the dead zone.

4.3.2 Improvements for brown-out reset

Here are some suggestions for improving the system's Brown-out Reset (BOR) performance:

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

Watchdog timer

The watchdog timer ensures the program runs correctly. If the system enters a dead zone or the program encounters an error, the watchdog timer will overflow and trigger a system reset.

Lower the system's operating frequency

The faster the system operates, the higher the minimum operating voltage. This increases the likelihood of the system operating within the dead zone. By reducing the system's operating frequency, the minimum operating voltage is lowered, reducing the chances of the system running in the dead zone.

Increase the voltage descent slope

This method is useful for systems powered by AC. In typical AC-powered systems, the voltage during a power-down process decreases slowly, causing the chip to operate in the dead zone for a longer time. If the system is powered on again under such conditions, the chip may enter an incorrect state. To address this, a discharge resistor can be added between the chip's power and ground pins, allowing the MCU to quickly pass through the dead zone and enter the reset region, minimizing the chances of errors when the chip is powered on.

4.4 Watchdog reset

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

The timing of the watchdog reset is as follows:

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

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

5. Sleep Mode

5.1 Enter sleep mode

When the STOP instruction is executed to enter sleep mode, if the Watchdog Timer (WDT) is enabled, then:

- The WDT will be cleared and continue running.
- The PD bit in the STATUS register will be cleared.
- The TO bit will be set to 1.
- The oscillator driver will be turned off.
- The I/O ports will maintain the state they were in before the STOP instruction was executed (whether driven high, low, or in high-impedance state).

In sleep mode, to minimize current consumption, all I/O pins should be held at either VDD or GND, with no external circuits drawing current from the I/O pins. To avoid introducing switching currents due to floating input pins, external pull-up or pull-down resistors should be used to ensure high-impedance input I/O pins are pulled to either a high or low level. Additionally, the impact of the internal pull-up resistors of the chip should be considered to further reduce current consumption.

5.2 Wake up from sleep mode

The device can be woken from sleep by any of the following events.

1. Watchdog timer wakeup;
2. INT interrupt;
3. PORTB interrupt on change;
4. PORTA interrupt on change or peripheral interrupt.

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

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

5.3 Interrupt wakeup

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

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

5.4 Sleep mode application

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

Example: procedures for entering sleep mode

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

5.5 Sleep mode wake-up time

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

System main clock source	System clock frequency (IRCF<2:0>)	Sleep wakeup wait time T_{WAIT}
Internal high-speed RC oscillation (F_{HSI})	$F_{SYS}=F_{HSI}$	$T_{WAIT}=136*1/F_{HSI}+16*1/F_{HSI}$
	$F_{SYS}= F_{HSI} /2$	$T_{WAIT}=136*2/F_{HSI}+16*1/F_{HSI}$
	…	…
	$F_{SYS}= F_{HSI} /64$	$T_{WAIT}=136*64/F_{HSI}+16*1/F_{HSI}$
Internal low-speed RC oscillation ($F_{LFINTOSC}$)	----	$T_{WAIT}=11/F_{LSI}$

6. I/O Ports

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

Port	Bit	Pin description	I/O
PORTA	0	Schmitt trigger input, push-pull or open-drain output, AN0, PWMA0, PWMB0, CMP output, external interrupt input	I/O
	1	Schmitt trigger input, push-pull or open-drain output, AN1, PWMA1, PWMB1, CMP positive or negative input	I/O
	2	Schmitt trigger input, push-pull or open-drain output, AN2, PWMA2, PWMB2, CMP negative input	I/O
	3	Schmitt trigger input, push-pull or open-drain output, AN3, PWMA3	I/O
	4	Schmitt trigger input, push-pull or open-drain output, AN4, PWMA4	I/O
	5	Schmitt trigger input, push-pull or open-drain output, AN5, PWMC0	I/O
PORTB	0	Schmitt trigger input, push-pull or open-drain output, programming clock input, AN8, OSCIN, PWMD0, CMP negative input	I/O
	1	Schmitt trigger input, push-pull or open-drain output, programming data input/output, AN9, OSCOUT, PWMB4, PWMD1, CMP negative input	I/O
	2	Schmitt trigger input, open-drain output, programming high-voltage input, PWMB3, PWMD4	I/O
	3	Schmitt trigger input, push-pull or open-drain output, AN11, PWMD2, TMR0 external clock input	I/O
	4	Schmitt trigger input, push-pull or open-drain output, AN12, PWMC4, PWMD3	I/O
	5	Schmitt trigger input, push-pull or open-drain output, AN13, PWMC3	I/O
	6	Schmitt trigger input, push-pull or open-drain output, AN14, PWMC2	I/O
	7	Schmitt trigger input, push-pull or open-drain output, AN15, PWMC1	I/O

<Table 6-1: Port configuration overview>

6.1 I/O port structure

6.1.1 PORTA I/O port

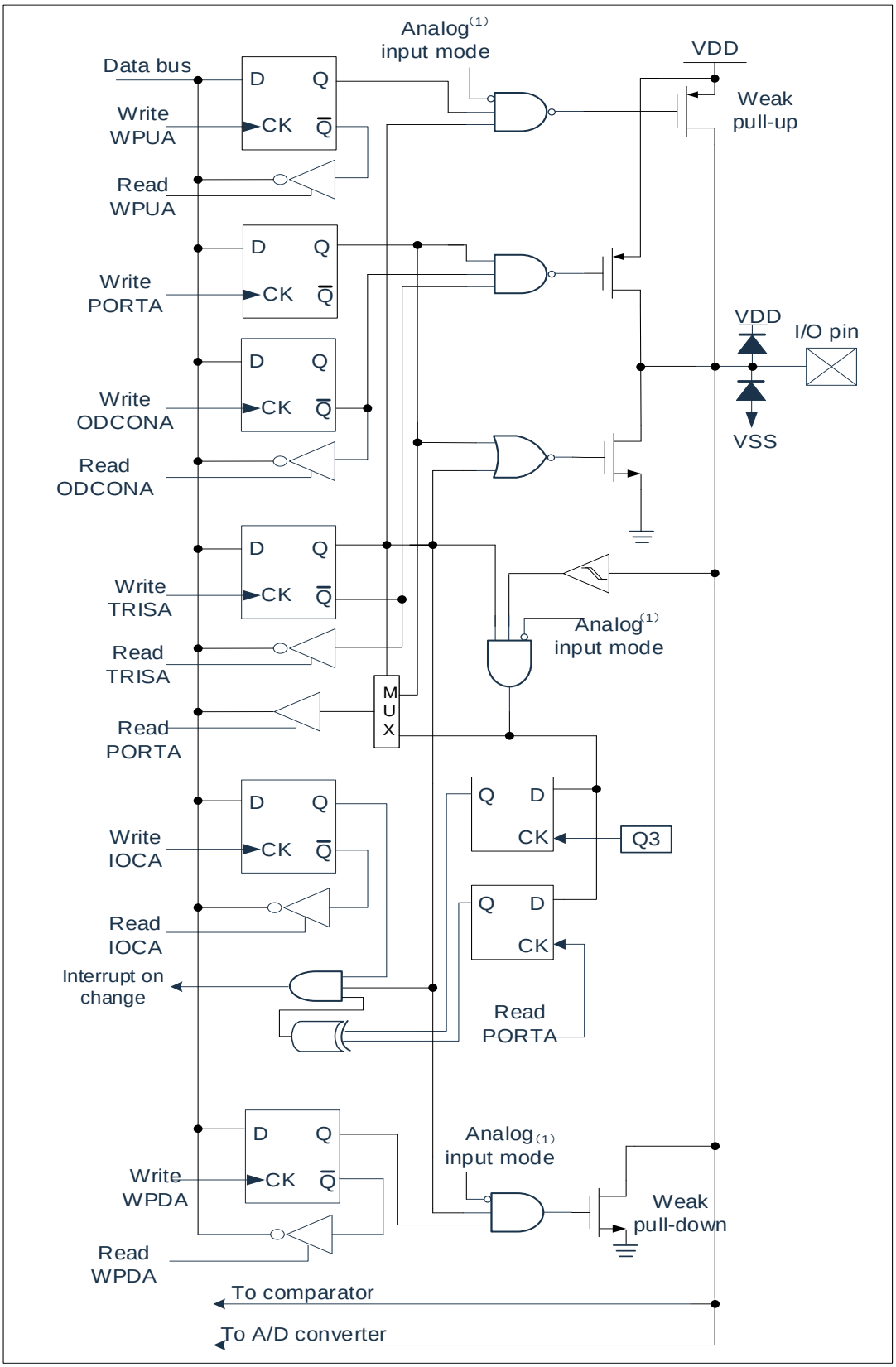


Figure 6-1: Diagram of PORTA I/O port stucture

6.1.2 PORTB I/O port

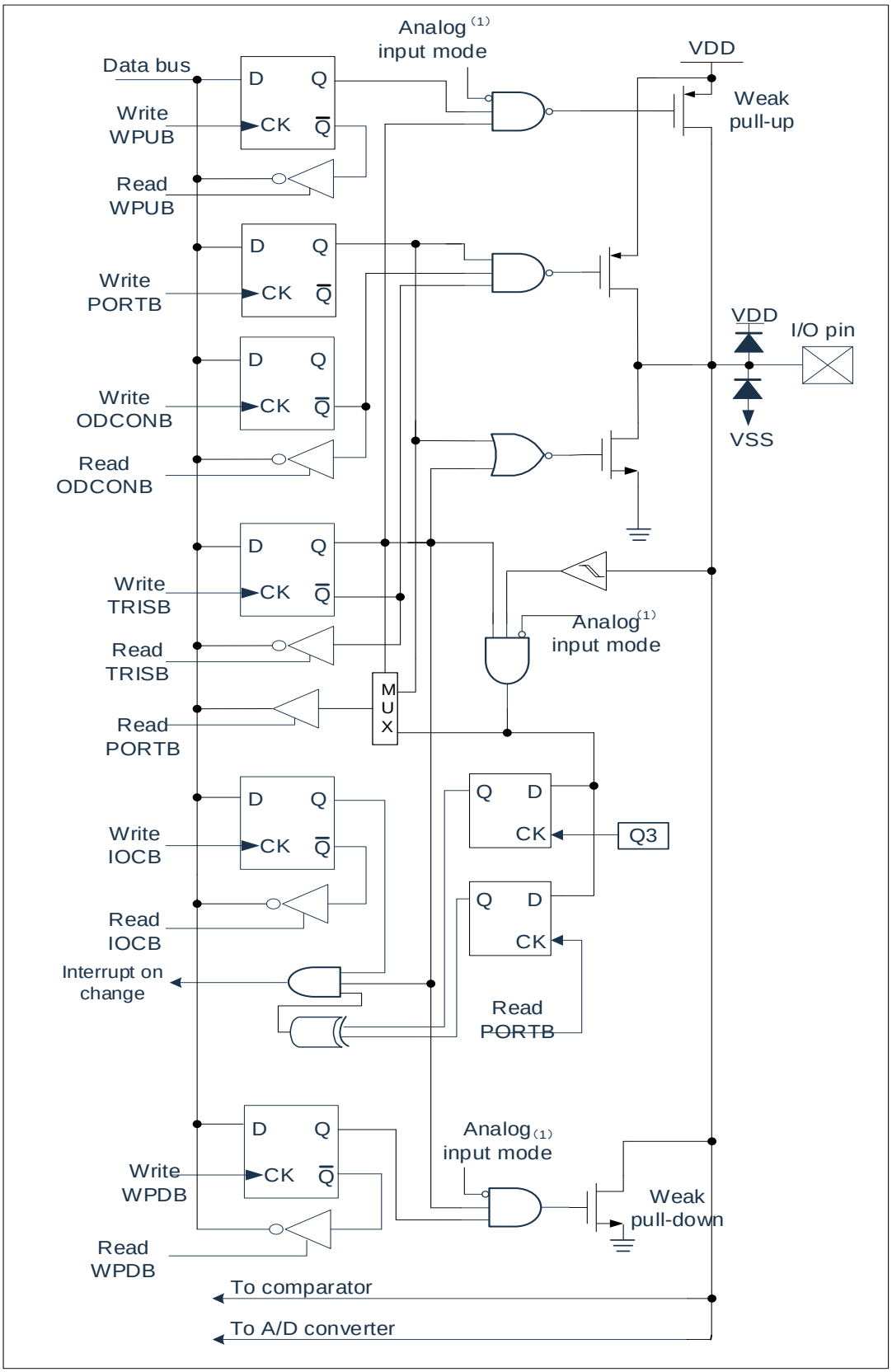


Figure 6-2: PORTB I/O port stucture

Note: AN_EN or ANSx determines the analog input mode.

6.2 PORTA

6.2.1 PORTA data and direction

PORTA is a 6-bit bi-directional port. Its corresponding data direction register is TRISA. Setting one bit of TRISA to 1 (=1) can configure the corresponding pin to be input. Setting one bit of TRISA to 0 (=0) can configure the corresponding PORTA pin to be output.

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

The registers related to the PORTA port are PORTA, TRISA, WPUA, WPDA, ODCONA, IOCA, ANSEL0 and so on.

PORTA data register PORTA(86H)

86H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTA	---	---	RA5	RA4	RA3	RA2	RA1	RA0
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	---	X	X	X	X	X	X

Bit7~Bit6	Unused	
Bit5~Bit0	PORTA<5:0>:	PORTA I/O pin bit
	1=	Port pin level>V_{IH}
	0=	Port pin level<V_{IL}

PORTA direction register TRISA(85H)

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

Bit7~Bit6	Unused	
Bit5~Bit0	TRISA<5:0>:	PORTA tristate control bit
	1=	PORTA pin is configured as an input (tristate)
	0=	PORTA pin is configured as an output

Example: PORTA procedure

```
LDIA    B'00110000'     ;set PORTA<3:0> as an output port, PORTA<5:4>as an input port
LD      TRISA,A
LDIA    03H             ;PORTA<1:0>output high level, PORTA<3:2>output low level
LD      PORTA,A         ;since PORTA<5:4> is an input port, loading 0 or 1 has no effect.
```

6.2.2 PORTA open-drain output control

Each PORTA pin has an individually configurable open-drain output enable control bit.

PORTA open-drain output enable register ODCONA(8CH)

8CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ODCONA	---	---	ODCONA5	ODCONA4	ODCONA3	ODCONA2	ODCONA1	ODCONA0
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	---	0	0	0	0	0	0

Bit7~Bit6 Unused

Bit5~Bit0 ODCONA<5:0>: PORTA open-drain output enable

1= Enable open-drain output

0= Disable open-drain output

6.2.3 PORTA analog selection control

The ANSEL0 register is used to configure the input mode of the I/O pins as analog mode. Setting the appropriate bits in ANSEL0 to 1 will disable digital read operations on the corresponding pins (they will always return 0) and enable the analog functionality of the pins. The state of the ANSEL0 bits does not affect the digital output functionality. Pins with TRIS cleared and ANSEL0 set to 1 will still function as digital outputs, but their input mode will switch to analog. This can lead to unpredictable results when performing read-modify-write operations on the affected ports.

PORTA analog selection register ANSEL0(93H)

93H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL0	----	----	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0
R/W	----	----	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	----	----	0	0	0	0	0	0

Bit7~Bit6 Unused

Bit5~Bit0 ANS<5:0>: Analogue select bit, selects the analog or digital function of pins AN<5:0> respectively

1= Analog input: The pin is configured as an analog input.

0= Digital I/O, pins are assigned to ports or special functions

6.2.4 PORTA pull-up resistor

Each PORTA pin has an individually configurable internal weak pull-up. Control bits WPUA<5:0> enable or disable each weak pull-up. When a port pin is configured as an output or analog input, its weak pull-up is automatically cut off.

PORTA pull-up resistor register WPUA(88H)

88H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUA	---	---	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	---	0	0	0	0	0	0

Bit7~Bit6 Unused
Bit5~Bit0 WPUA<5:0>: Weak pull-up register bit
1= Enable pull-up
0= Disable pull-up

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

6.2.5 PORTA pull-down resistor

Each PORTA pin has an internal weak pull-down that can be individually configured. The control bits WPDA<5:0> enable or disable each weak pull down. When a port pin is configured as an output or analog input, its weak pull-down is automatically cut off.

PORTA pull-down resistor register WPDA(87H)

87H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPDA	---	---	WPDA5	WPDA4	WPDA3	WPDA2	WPDA1	WPDA0
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	---	0	0	0	0	0	0

Bit7~Bit6 Unused
Bit5~Bit0 WPDA<5:0>: Weak pull-down register bit
1= Enable pull-down
0= Disable pull-down

Note: If the pin is configured as an output or analogue input, the weak pull-down is automatically disabled.

6.2.6 PORTA interrupt on change

All PORTA pins can be individually configured as interrupt on change pins. The control bit IOCA<5:0> enables or disables the interrupt function of each pin. Disable pin level change interrupt function when power on reset.

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

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

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

The mismatch status will continuously set the RAIF flag bit as 1. Reading or writing PORTA will end the mismatch state and allow the RAIF flag to be cleared.

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

PORTA interrupt-on-change register IOCA(89H)

89H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOCA	---	---	IOCA5	IOCA4	IOCA3	IOCA2	IOCA1	IOCA0
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	---	0	0	0	0	0	0

Bit7~Bit6	Unused	
Bit5~Bit0	IOCA<5:0>	PORTA interrupt-on-change control bit.
	1=	Enable interrupt on change.
	0=	Disable interrupt on change.

6.3 PORTB

6.3.1 PORTB data and direction

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

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

The registers related to the PORTB port are PORTB, TRISB, WUPB, WDPB, IOCB, ODCONB, ANSEL1 and so on.

PORTB data register PORTB(06H)

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

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

Note: RB2 is fixed as an open-drain output and can only output a low level or high resistance state.

PORTB direction register TRISB (05H)

05H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	1	1	1	1	1	1	1	1

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

Example: PORTB procedure

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

6.3.2 PORTB open-drain output control

Each PORTB pin has an individually configurable open-drain output enable control bit (except RB2).

PORTB open drain output enable register ODCONB(0CH)

0CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ODCONB	ODCONB7	ODCONB6	ODCONB5	ODCONB4	ODCONB3	---	ODCONB1	ODCONB0
R/W	R/W	R/W	R/W	R/W	R/W	---	R/W	R/W
Reset value	0	0	0	0	0	---	0	0

Bit7~Bit3 ODCONB<7:3>: PORTB open-drain output enable
1= Enable open-drain output
0= Disable open-drain output

Bit1~Bit0 ODCONB<1:0>: PORTB open-drain output enable
1= Enable open-drain output
0= Disable open-drain output

Note: The RB2 port is fixed as an open-drain output, and can only output low level or high resistance state.

6.3.3 PORTB analog selection control

The ANSEL1 register is used to configure the input mode of the I/O pins as analog mode. Setting the appropriate bits in ANSEL1 to 1 will disable digital read operations on the corresponding pins (they will always return 0) and enable the analog functionality of the pins. The state of the ANSEL1 bits does not affect the digital output functionality. Pins with TRIS cleared and ANSEL1 set to 1 will still function as digital outputs, but their input mode will switch to analog. This can lead to unpredictable results when performing read-modify-write operations on the affected ports.

PORTB analog selection register ANSEL1 (94H)

94H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL1	ANS15	ANS14	ANS13	ANS12	ANS11	---	ANS9	ANS8
R/W	R/W	R/W	R/W	R/W	R/W	---	R/W	R/W
Reset value	0	0	0	0	0	---	0	0

Bit7~Bit3 ANS<15:11>: Analogue select bit, selects the analog or digital function of pins AN<15:11> respectively
1= Analog input: The pin is configured as an analog input.
0= Digital I/O, pins are assigned to ports or special functions

Bit2 Unused

Bit1~Bit0 ANS<9:8>: Analogue select bit, selects the analog or digital function of pins AN<9:8> respectively
1= Analog input: The pin is configured as an analog input.
0= Digital I/O, pins are assigned to ports or special functions

6.3.4 PORTB pull-up resistor

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

PORTB pull-up resistor register WPUB(08H)

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

Bit7~Bit0 WPUB<7:0>: PORTB weak pull-up enable bit
1= Enable pull-up
0= Disable pull-up

Note: When RB0, RB1, RB3~RB7 pins are configured as outputs or analog inputs, the weak pull-ups will be disabled automatically.
When RB2 is configured as an output, its weak pull-up will not be disabled.

6.3.5 PORTB pull-down resistor

Each PORTB pin has an individually configurable internal weak pull-down (except RB2). Control bits WPDB<7:0> enable or disable each weak pull-down. When a port pin is configured as an output or analog input, its weak pull-down is automatically cut off.

PORTB pull-down resistor register WPDB(07H)

07H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPDB	WPDB7	WPDB6	WPDB5	WPDB4	WPDB3	---	WPDB1	WPDB0
R/W	R/W	R/W	R/W	R/W	R/W	---	R/W	R/W
Reset value	0	0	0	0	0	---	0	0

Bit7~Bit3 WPDB<7:3>: PORTB weak pull-down enable bit
1= Enable pull-down
0= Disable pull-down
Bit2 Unused
Bit1~Bit0 WPDB<1:0>: PORTB weak pull-down enable bit
1= Enable pull-down
0= Disable pull-down

Note: RB0, RB1, RB3~RB7 pins are configured as outputs or analog inputs, the weak pull-down is automatically disabled.
There is no weak pull-down on pin RB2.

6.3.6 PORTB interrupt on change

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

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

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

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

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

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

PORTB interrupt-on-change register IOCB(09H)

09H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOCB	IOCB7	IOCB6	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	IOCB0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 IOCB<7:0> PORTB interrupt-on-change control bit.
1= Enable interrupt on change.
0= Disable interrupt on change.

6.4 I/O usage

6.4.1 Write to I/O port

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

Example: write to I/O port program

```
LD      PORTB,A     ;load ACC to PORTB
CLRB    PORTB,1     ;clear PORTB.1
SET     PORTB       ;set all output ports of PORTB to 1
SETB    PORTB,1     ;set PORTB.1 to 1
```

6.4.2 Read from I/O port

Example: read from I/O port program

```
LD      A,PORTB     ;load PORTB to ACC
SNZB    PORTB,1     ;check if PORTB,1 is 1, if it is 1, skip the next statement
SZB     PORTB,1     ;check if PORTB,1 is 0, if it is 0, skip the next statement
```

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

6.5 Cautions on I/O port usage

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

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

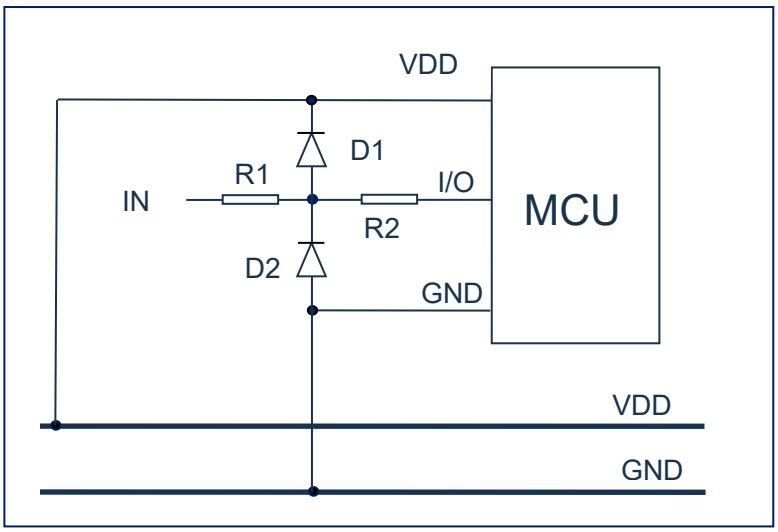


Figure 6-3: I/O port caution connection diagram

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

7. Interrupt

7.1 Overview

The chip has the following interrupt sources:

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

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

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

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

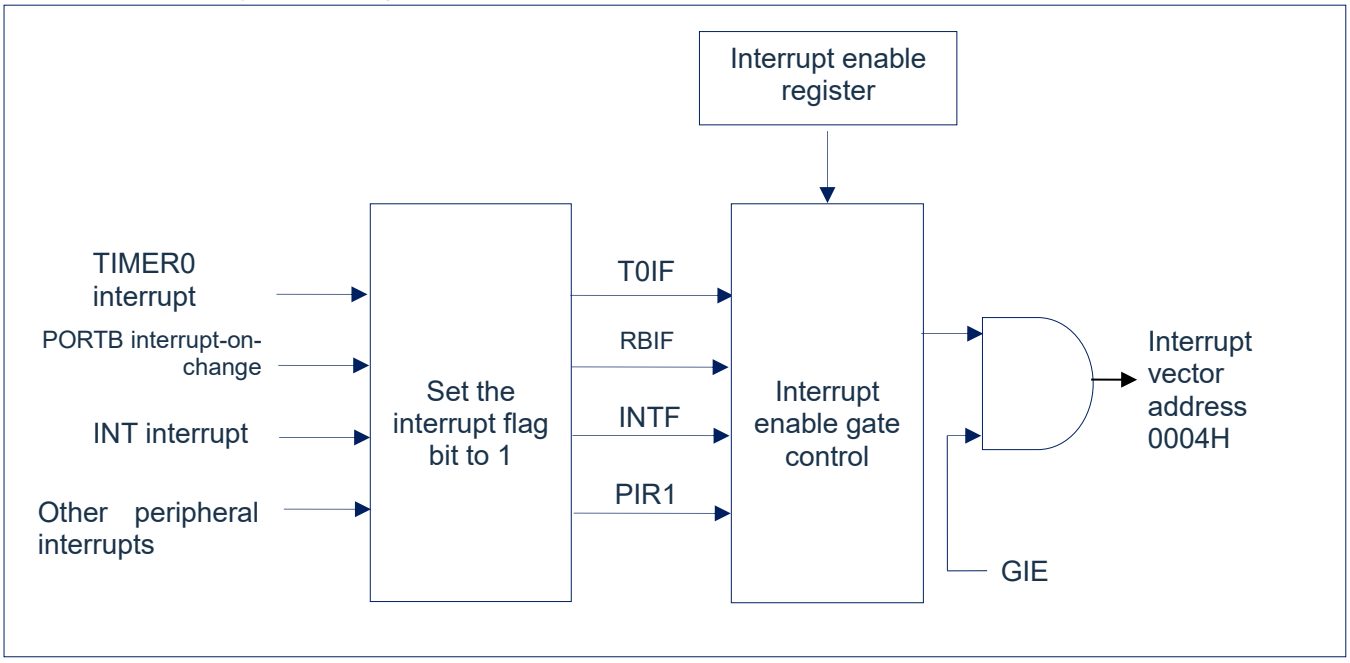


Figure 7-1: Schematic diagram of interrupt principle

7.2 Interrupt control register

7.2.1 Interrupt control register

Interrupt control register (INTCON) is a readable and writable register, including the enable and flag bits of Timer0 overflow interrupt, INT interrupt and PORTB port interrupt on change.

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

Interrupt control register INTCON (0BH)

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

Bit7 GIE: Global interrupt enable bit
1= Enable all unmask interrupts
0= Disable all interrupts

Bit6 PEIE: Peripheral interrupt enable bit
1= Enable all unmask peripheral interrupts
0= Disable all peripheral interrupts

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

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

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

Bit2 T0IF: TIMER0 overflow interrupt enable bit (2)
1= TMR0 register overflowed (cleared by software)
0= No TMR0 register overflow

Bit1 INTF: INT external interrupt flag bit
1= An INT external interrupt occurs (cleared by software)
0= No INT external interrupt occured

Bit0 RBIF: PORTB level change interrupt flag bit
1= At least one pin level status in the PORTB port has changed (cleared by software)
0= None of the PORTB universal I/O pin status has changed

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

7.2.2 Peripheral interrupt enable register

The peripheral interrupt enable register has PIE1, before allowing any peripheral interrupt, you must first set the PEIE bit of the INTCON register to 1.

Peripheral interrupt enable register PIE1 (0EH)

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

Bit7~Bit6 Unused

Bit5 CMPIE: Comparator interrupt enable bit
- 1= Enable comparator interrupt
- 0= Disable comparator interrupt

Bit4 PWMIE: PWM interrupt enable bit (PWM0/1/2/3)
- 1= Enable PWM interrupt
- 0= Disable PWM interrupt

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

Bit2 Unused

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: ADC interrupt enable bit
- 1= Enable ADC interrupt
- 0= Disable ADC interrupt

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 enable bit or the global enable bit GIE. The user software should ensure that the corresponding interrupt flag bit is cleared before allowing an interrupt.

Peripheral interrupt request register PIR1(0DH)

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

Bit7~Bit6 Unused

Bit5 CMPIF: Comparator interrupt flag bit (cleared by software)
- 1= A comparator interrupt occurs
- 0= No comparator interrupt occurred

Bit4 PWMIF: PWM interrupt flag bits (PWM0/1/2/3) (cleared by software)
- 1= A PWM interrupt occurs
- 0= No PWM interrupt occurred

Bit3 RAIF: PORTA interrupt-on-change flag bit
- 1= At least one pin level status in PORTA port has changed (cleared by software)
- 0= None of the PORTA universal I/O pin status has changed

Bit2 Unused

Bit1 TMR2IF: TIMER2 and PR2 match interrupt flag bit (cleared by software)
- 1= A TMR2 and PR2 match interrupt occurs
- 0= No TMR2 and PR2 match interrupt occured

Bit0 ADIF: AD converter interrupt flag bit
- 1= AD conversion is completed (cleared by software)
- 0= AD conversion is not completed or not started

7.3 Protection methods for interrupt

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

Example: Stack protection for ACC and STATUS

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

7.4 Interrupt priority and multi-interrupt nesting

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

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

8. TIMER0

8.1 TIMER0 overview

TIMER0 is composed of the following functions:

- 8-bit timer/counter register (TMR0);
- 8-bit pre-scaler (shared with watchdog timer);
- Programmable internal or external clock source;
- Programmable external clock edge selection;
- External 32.768K oscillating clock (F_{LSE}) can be selected
- Overflow interrupt.

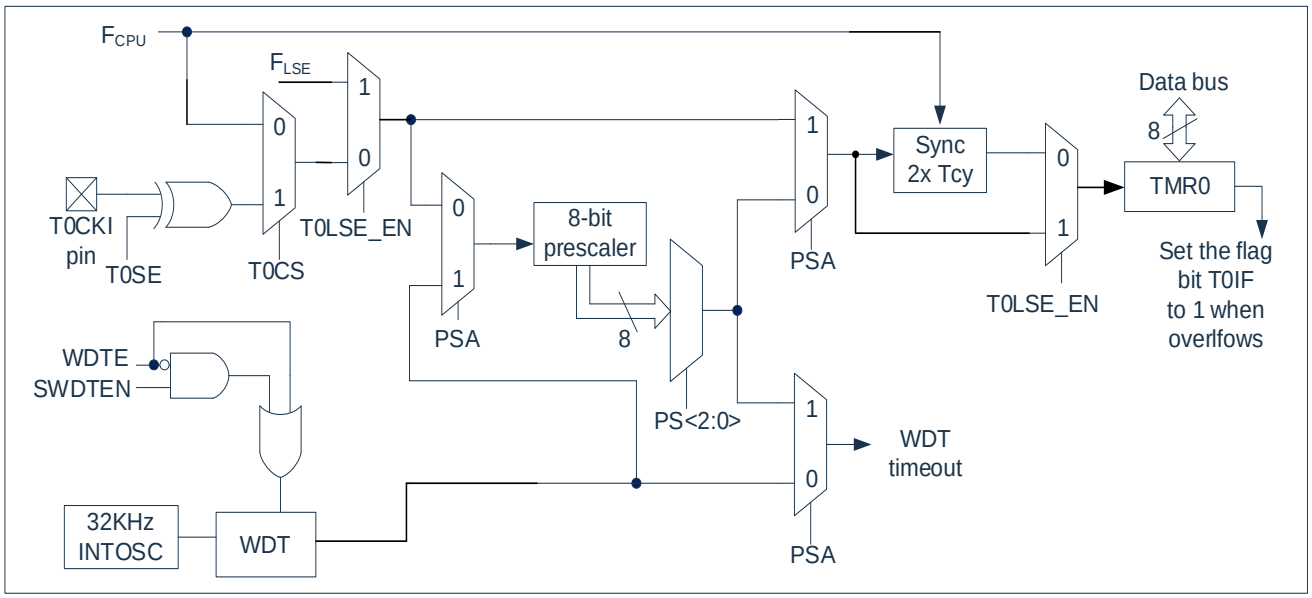


Figure 8-1: TIMER0/WDT structure diagram

Note:
1. T0SE, T0LSE_EN, T0CS, PSA, PS<2:0> are the bits in the OPTION_REG register.
2. SWDTEN is a bit in the OSCCON register.
3. WDTEN bit is in CONFIG.

8.2 Working principle of TIMER0

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

8.2.1 8-bit timer mode

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

8.2.2 8-bit counter mode

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

8.2.3 Software programmable pre-scaler

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

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

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

8.2.4 Switch prescaler between TIMER0 and WDT module

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

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

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

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

8.2.5 TIMER0 interrupt

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

Note: The TIMER0 interrupt wakes up the processor only if F_{LSE} is selected as the clock source.

8.3 TIMER0 related registers

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

TMR0 is an 8-bit readable and writable timer/counter, OPTION_REG is an 8-bit write-only register, the user can change the value of OPTION_REG to change the working mode of TIMER0, etc. Please refer to Section 2.6 Prescaler Register (OPTION_REG) Application.

8-bit timer/counter TMR0 (81H)

81H	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

Prescaler control register OPTION_REG (01H)

01H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION_REG	T0LSE_EN	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	1	1	1	1	0	1	1

Bit7 T0LSE_EN: TIMER0 clock source selection F_{LSE} enable bit
0= TIMER0 clock source is determined by T0CS
1= TIMER0clock source selects F_{LSE}

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

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

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

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

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

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

9. TIMER2

9.1 TIMER2 overview

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

1. 8-bit timer register (TMR2)
2. 8-bit period register (PR2)
3. Interrupt when TMR2 matches PR2
4. Software programmable prescaler ratio (1:1, 1:4 and 1:16)
5. Software programmable postscaler ratio (1:1 to 1:16)
6. External 32.768KHz oscillating clock (F_{LSE}) can be selected

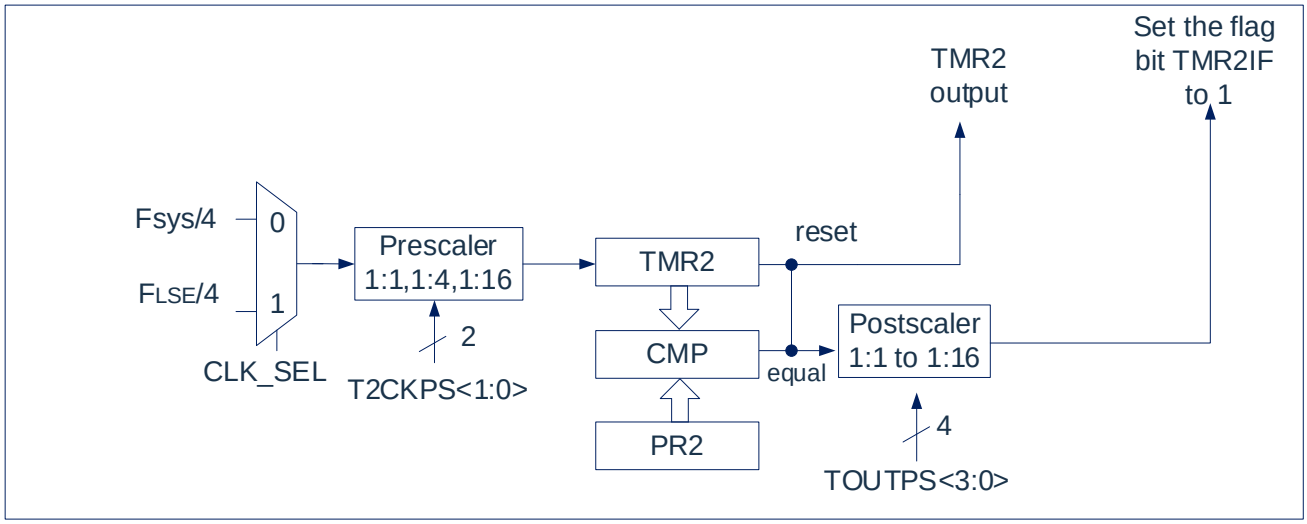


Figure 9-1: Block diagram of TIMER2

9.2 Working principle of TIMER2

The input clock to the TIMER2 module is the system command clock (F_{SYS}/4) or an external 32.768 kHz oscillation (F_{LSE}). The clock is input to the TIMER2 prescaler, which is available in the following ratios: 1:1, 1:4 or 1:16. The output of the prescaler is then used to increment the TMR2 register.

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

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

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

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

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

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

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

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

Note: Writing to T2CON does not clear TMR2. When TMR2ON=0, the TMR2 register cannot be written.

9.3 TIMER2 related registers

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

TIMER2 data register TMR2 (12H)

12H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR2								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

TIMER2 period register PR2 (11H)

11H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PR2								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	1	1	1	1	1	1	1	1

TIMER2 control register T2CON(13H)

13H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T2CON	CLK_SEL	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7 CLK_SEL: Clock source selection
1= Select external F_{LSE}/4 as TMR2 clock source (continue counting in sleep state)
0= Select internal F_{SYS}/4 as TMR2 clock source

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

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

Bit1~Bit0 T2CKPS<1:0>: TIMER2 clock prescaler ratio select bit
00= 1
01= 4
1x= 16

10. 10-Bit PWM Module

The chip contains a 10-bit PWM module, which can be configured as 4 common periods, independent duty cycle outputs or 1 group complementary output. The chip includes a 10-bit PWM module, which can be configured for 4 channels with a shared period and independent duty cycles, 1 channel with an independent period and independent duty cycle, or 2 pairs of complementary outputs.

10.1 Pin configuration

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

10.2 Relevant registers description

PWM control register PWMCON0(15H)

15H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMCON0	CLKDIV[2:0]			PWM4EN	PWM3EN	PWM2EN	PWM1EN	PWM0EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

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

Bit4~Bit0 PWM0/1/2/3/4EN: PWM0/1/2/3/4 enable bit
1= Enable PWM0/1/2/3/4
0= Disable PWM0/1/2/3/4

PWM control register PWMCON1(16H)

16H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMCON1	PWMIO_SEL[1:0]		PWM2DTEN	PWM0DTEN	---	---	DT_DIV<1:0>	
R/W	R/W	R/W	R/W	R/W	---	---	R/W	R/W
Reset value	0	0	0	0	---	---	0	0

Bit7~Bit6 PWMIO_SEL: PWM IO group selection
11= PWM in group A, PWM0-RA0,PWM1-RA1,PWM2-RA2,PWM3-RA3,PWM4-RA4
10= PWM in group B, PWM0-RA0,PWM1-RA1,PWM2-RA2,PWM3-RB2,PWM4-RB1
01= PWM in group C, PWM0-RA5,PWM1-RB7,PWM2-RB6,PWM3-RB5,PWM4-RB4
00= PWM in group D, PWM0-RB0,PWM1-RB1,PWM2-RB3,PWM3-RB4,PWM4-RB2

Bit5 PWM2DTEN: PWM2 dead zone enable bit
1= Enable the dead zone function for PWM2. PWM2 and PWM3 form a complementary output pair.
0= Disable the dead zone function for PWM2.

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

Bit3~Bit2 Unused

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

PWM control register PWMCON2(1DH)

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

Bit7~Bit5 Unused
Bit4~Bit0 PWM0/1/2/3/4DIR PWM output reverse control bit
1= PWM0/1/2/3/4 reverse outputs
0= PWM0/1/2/3/4 normal outputs

PWM0~PWM3 period low bit register PWMTL (17H)

17H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMTL	PWMT<7:0>							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMT[7:0]: PWM0~PWM3 period low 8 bits

PWM4 period low bit register PWMT4L(1CH)

1CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMT4L	PWM4T<7:0>							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWM4T<7:0>: PWM4 period low 8 bits

Period high bit register PWMTH (18H)

18H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMTH	---	---	PWMD4<9:8>		PWM4T<9:8>		PWMT<9:8>	
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	---	0	0	0	0	0	0

Bit7~Bit6 Unused
Bit5~Bit4 PWMD4<9:8>: High 2 bits of PWM4 duty cycle
Bit3~Bit2 PWM4T<9:8>: High 2 bits of PWM4 period
Bit1~Bit0 PWMT<9:8>: High 2 bits of PWM0~PWM3 period

Note: Writing PWMD4<9:8> does not take effect immediately, it needs to be written to PWMD4L to take effect.

PWM0 duty cycle low bit register PWMD0L (19H)

19H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD0L	PWMD0<7:0>							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD0<7:0>: PWM0 duty cycle low 8 bits

PWM1 duty cycle low bit register PWMD1L (1AH)

1AH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD1L	PWMD1<7:0>							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD1<7:0>: PWM1 duty cycle low 8 bits

PWM2 duty cycle low bit register PWMD2L (9BH)

9BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD2L	PWMD2[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD2[7:0]: PWM0 duty cycle low 8 bits

PWM3 duty cycle low bit register PWMD3L (9CH)

9CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD3L	PWMD3[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

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

PWM4 duty cycle low bit register PWMD4L (1BH)

1BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD4L	PWMD4<7:0>							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD4<7:0>: PWM4 duty cycle low 8 bits

PWM0 and PWM1 duty cycle high bit register PWMD01H (1EH)

1EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD01H	---	---	PWMD1<9:8>		---	---	PWMD0<9:8>	
R/W	---	---	R/W	R/W	---	----	R/W	R/W
Reset value	---	---	0	0	---	----	0	0

Bit7~Bit6 Unused
Bit5~Bit4 PWMD1<9:8>: PWM1 duty cycle high 2 bits
Bit3~Bit2 Unused
Bit1~Bit0 PWMD0<9:8>: PWM0 duty cycle high 2 bits

Note: Writing PWMD0<9:8> does not take effect immediately, it needs to write PWMD0L to take effect.
Writing PWMD1<9:8> does not take effect immediately, it needs to write PWMD1L to take effect.

PWM2 and PWM3 duty cycle high bit register PWMD23H (9EH)

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

Bit7~Bit6 Unused
Bit5~Bit4 PWMD3[9:8]: PWM3 duty cycle high 2 bits
Bit3~Bit2 Unused
Bit1~Bit0 PWMD2[9:8]: PWM2 duty cycle high 2 bits

Note: Writing PWMD2<9:8> does not take effect immediately, it needs to write PWMD2L to take effect.
Writing PWMD3<9:8> does not take effect immediately, you need to write PWMD3L to take effect.

PWM0 and PWM1 dead time register PWM01DT(1FH)

1FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM01DT	---	---	PWM01DT<5:0>					
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	---	0	0	0	0	0	0

Bit7~Bit6 Unused
Bit5~Bit0 PWM01DT<5:0>: PWM0 and PWM1 dead time

PWM2 and PWM3 dead time register PWM23DT(9DH)

9DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM23DT	---	---	PWM23DT<5:0>					
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	---	0	0	0	0	0	0

Bit7~Bit6 Unused
Bit5~Bit0 PWM23DT<5:0>: PWM2 and PWM3 dead time

10.3 10-bit PWM register write sequence

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

1) Write the higher 2-bit value, the high 2-bit value is just written to the internal cache;

2) Write the lower 8 bits, then the full 10-bit duty cycle value is latched;

3) The above operations are only for PWM0, PWM1, PWM2, PWM3, PWM4 duty cycle registers.

10.4 10-bit PWM period

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

Formula 1: PWM cycle calculation formula.

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

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

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

- PWM period counter is cleared;
- PWMx pin is set to 1;
- New period of PWM is latched;
- New duty cycle of PWM is latched;
- Generate a PWM interrupt flag bit (No interrupt for PWM4).

10.5 10-bit PWM duty cycle

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

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

Formula 2: Pulse width calculation formula

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

Formula 3: PWM duty cycle calculation formula

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

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

10.6 System clock frequency change

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

10.7 Programmable dead time delay mode

The complementary output mode can be enabled by setting PWMxDT_EN, and the dead time delay function is automatically enabled after the complementary output is enabled.

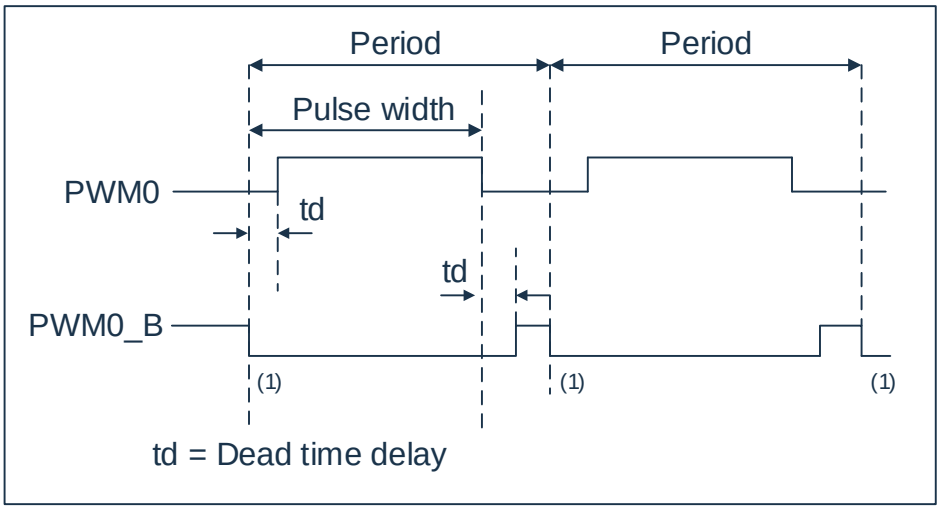


Figure 10-1: Example of PWM dead time delay output

The dead time calculation formula is:

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

10.8 10-bit PWM configuration

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

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

11. Comparator (COMP)

11.1 Block diagram

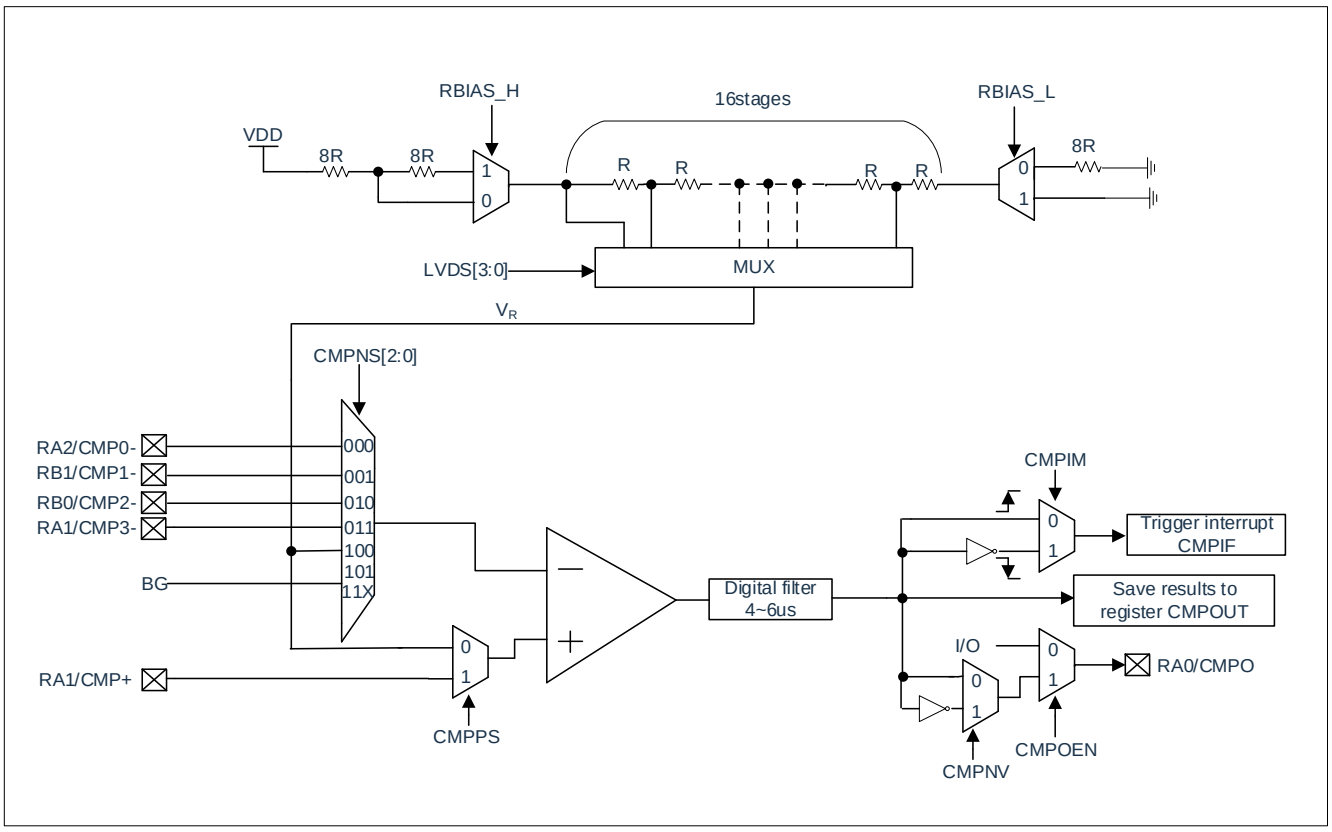


Figure 11-1: Functional block diagram of comparator

11.2 Features

- Internally integrated with one comparator.
- Comparator offset voltage is≤±13mV.
- Input common-mode voltage range: 0V~VDD-1.3V.
- Built-in resistor divider module with VDD as the reference voltage.
- The comparator result can be triggered by either rising or falling edge and generate an interrupt.
- The comparator result can be output from RA0 port and supports inverted output.

11.3 Comparator related functions

11.3.1 Comparator functions description

Figure 11-1 shows the functional block diagram of the comparator. The positive input of the comparator can be used for selecting the CMP+ port or the internal resistor voltage divider output V_R by configuring CMPPS bit of the CMPCON0 register. The negative input can be used for selecting the CMPx- port, the internal resistor voltage divider output V_R, or the 1.2V BG voltage by configuring the CMPNS<2:0> bits of the CMPCON0 register. When the voltage at the positive input of the comparator is greater than the voltage at the negative input, the comparator outputs 1 after digital filtering. Conversely, if the voltage at the positive input is less than the voltage at the negative input, the comparator outputs 0 after digital filtering.

11.3.2 Internal resistor voltage divider output

The comparator integrates an internal resistor voltage divider module with a reference voltage of VDD. Different resistor voltage divider outputs V_R can be obtained by configuring the values of the control bits RBIAS_H, RBIAS_L, and LVDS<3:0> of the CMPCON1 register. The four calculation formulas for V_R are as follows:

RBIAS_H	RBIAS_L	V_R calculation formula
0	0	$V_R=\frac{1}{4}*VDD+\frac{n+1}{32}*VDD$
0	1	$V_R=\frac{n+1}{24}*VDD$
1	0	$V_R=\frac{1}{5}*VDD+\frac{n+1}{40}*VDD$
1	1	$V_R=\frac{n+1}{32}*VDD$

Note: n is the value of LVDS<3:0>, i.e., n = 0, 1, 2, ..., 14, 15.

11.3.3 Monitoring supply voltage

According to the comparator structure block diagram in Figure 11-1 and the formula in section 11.3.2, when the negative end of the comparator selects BG 1.2V, and the positive end selects the internal resistor voltage divider output V_R, the power supply voltage can be monitored by the comparator. When the power supply voltage is lower than the set value, the comparator outputs 0, and when the power supply voltage is higher than the set value, the comparator outputs 1. By configuring the values of RBIAS_H, RBIAS_L, LVDS[3:0], different voltage monitoring points can be set as follows.

RBIAS_H	RBIAS_L	LVDS[3:0]	Monitor value(V)	RBIAS_H	RBIAS_L	LVDS[3:0]	Monitor value(V)	RBIAS_H	RBIAS_L	LVDS[3:0]	Monitor value(V)
0	1	0101	4.80	0	0	0100	2.95	1	0	1101	2.18
1	0	0010	4.36	0	1	1001	2.88	0	0	1001	2.13
0	0	0000	4.27	1	0	1000	2.82	1	0	1110	2.09
0	1	0110	4.11	0	0	0101	2.74	0	1	1101	2.06
1	0	0011	4.00	1	0	1001	2.67	0	0	1010	2.02
0	0	0001	3.84	0	1	1010	2.62	1	0	1111	2.00
1	0	0100	3.69	0	0	0110	2.56	-	-	-	-
0	1	0111	3.60	1	0	1010	2.53	-	-	-	-
0	0	0010	3.49	0	0	0111	2.40	-	-	-	-
1	0	0101	3.43	1	0	1100	2.29	-	-	-	-
0	0	0011	3.20	0	0	1000	2.26	-	-	-	-
1	0	0111	3.00	0	1	1100	2.22	-	-	-	-

11.3.4 Interrupt usage

To use the interrupt function of the comparator, the comparator interrupt can be enabled through the following configuration steps:

- Configure the CMPPS bit of the CMPCON0 register to select the positive input.
- Configure the CMPNS<2:0> bits of the CMPCON0 register to select the negative input.
- Configure the CMPIM bit of the CMPCON1 register to select the rising or falling edge trigger for the interrupt.
- Set the CMPEN bit of the CMPCON0 register to enable the comparator.
- Delay for 10us.
- Clear the CMPIF bit of the PIR1 register.
- Set the CMPIE bit of the PIE1 register to 1 to enable the comparator interrupt.
- Set the PEIE and GIE bits of the INTCON register to 1 to enable peripheral and global interrupts.

11.3.5 Interrupt sleep wake-up

The comparator interrupt can wake up the chip from sleep mode. The specific configuration can be found in the following program routine:

Example: Comparator interrupt sleep wake-up program

```
SLEEP_MODE:
        LDIA    B'00000110'
        LD      TRISA,A         ;Configure RA1/CMP+ and RA2/CMP0- as input ports
        ...                     ;Disable other functions
        LDIA    00H
        LD      CMPCON0,A       ;CMPNS<2:0>=000, select RA2/CMP0- as the negative input
        SETB    CMPCON0,6       ;CMPPS=1, select RA1/CMP+ as the positive input
        SETB    CMPCON1,6       ;ANSEL=1, set CMP+ and CMP0- as analog ports to reduce sleep power consumption
        SETB    CMPCON1,7       ;CMPIM=1, select falling edge trigger for the comparator interrupt
        SETB    CMPCON0,7       ;Enable the comparator
        CALL    DELAY10US       ;Delay to ensure stable output from the comparator after enabling it
        CLRB    PIR1,5          ;Clear CMPIF (necessary)
        SETB    PIE1,5          ;Enable the comparator interrupt
        SETB    INTCON,6        ;Enable peripheral interrupts
        SETB    INTCON,7        ;Enable global interrupts, the program will jump to the interrupt vector address 0004H after waking up
        CLRWDT                  ;Clear the WDT
        STOP                    ;Execute the STOP instruction
        NOP
        NOP
```

11.3.6 Result output pin configuration

After digital filtering, the result of the comparator can be obtained by reading the CMPOUT bit of the CMPCON0 register. It can also be output to the RA0/CMPO pin through the following configuration steps:

- Set TRISA0 to 0 to configure RA0/CMPO as an output pin.
- Configure the CMPNV bit of the CMPCON0 register to select normal or inverted output.
- Set the CMPOEN bit of the CMPCON0 register to 1 to enable the output of CMPOUT to the RA0/CMPO pin.

11.4 Related registers

Comparator control register CMPCON0(0FH)

0FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CMPCON0	CMPEN	CMPPS	CMPNS2	CMPNS1	CMPNS0	CMPNV	CMPOUT	CMPOEN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7 CMPEN: CMP enable bit
1= Enable CMP
0= Disable CMP
Bit6 CMPPS: CMP positive input select bit
1= CMP+ port voltage
0= VDD voltage after dividing by internal resistor
Bit5~Bit3 CMPNS<2:0>: CMP negative input select bit
000= CMP0- port voltage
001= CMP1- port voltage
010= CMP2- port voltage
011= CMP3- port voltage
100= VDD voltage after dividing by internal resistor
101= BG
11x= BG
Bit2 CMPNV: CMPO port output inverse control bit
1= Invert CMPOUT output at CMPO port
0= Normal CMPOUT output at CMPO port
Bit1 CMPOUT: CMP result bit
Bit0 CMPOEN: CMPO port output enable bit
1= Enable CMPOUT output at CMPO port
0= Disable CMPOUT output at CMPO port

Comparator control register CMPCON1(10H)

10H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CMPCON1	CMPIM	AN_EN	RBIAS_H	RBIAS_L	LVDS<3:0>			
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7 CMPIM: CMP interrupt trigger edge selection
1= Falling edge of the CMP output triggers an interrupt
0= Rising edge of CMP output triggers interrupt
Bit6 AN_EN: Analog port enable bit, enables analog functionality for CMP+ and CMPX-
1= Analog port
0= Digital port
Bit5 RBIAS_H: Specific usage refers to the comparator block diagram
Bit4 RBIAS_L: Specific usage refers to the comparator block diagram
Bit3~Bit0 LVDS<3:0>: Internal resistor divider ratio selection bit

Note: The AN_EN bit is only valid for I/O pins selected for comparator functionality. Additionally, the ANSEL0 and ANSEL1 registers can be used to enable the analog functions for CMP+ and CMPx- pins.

12. Analog to Digital Conversion (ADC)

12.1 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 circuit. The output of the sample 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 the internal LDO or VDD The ADC can generate an interrupt after the conversion is completed.

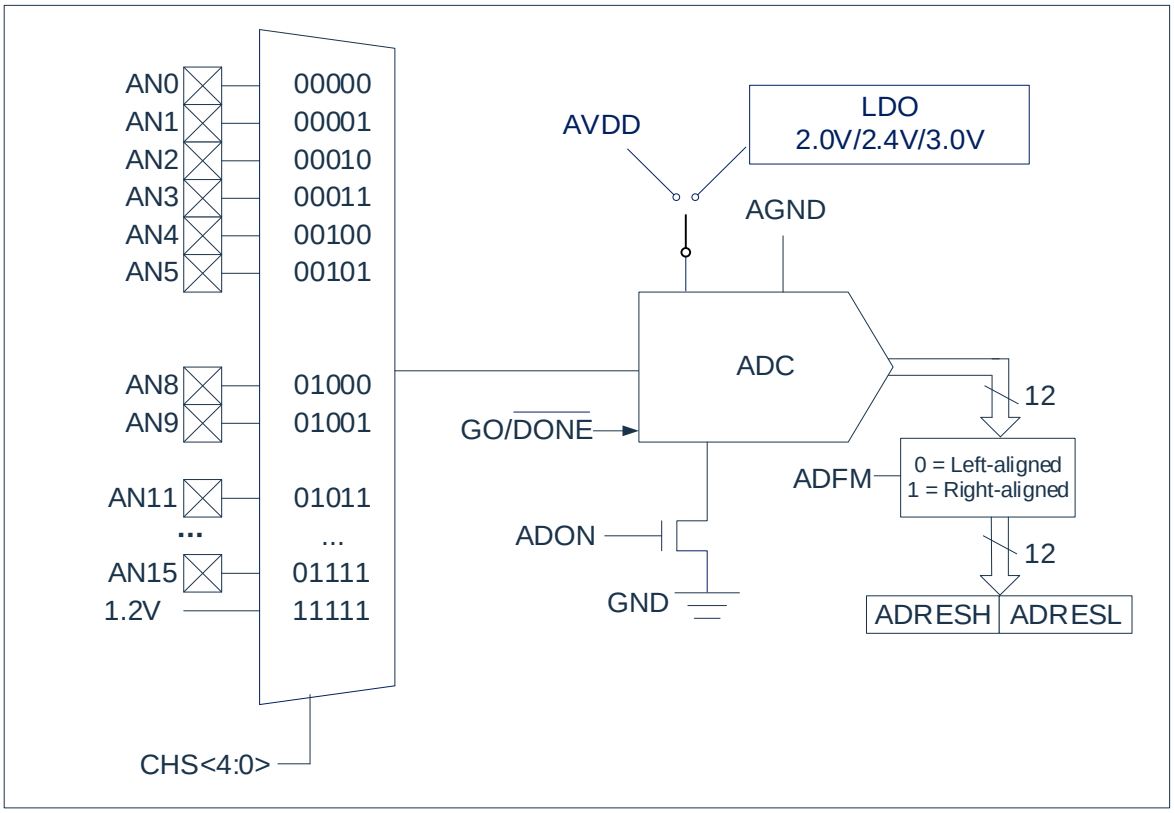


Figure 12-1: Block diagram of ADC

12.2 ADC configuration

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

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

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

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

12.2.2 Channel selection

The CHS bits of the ADCON0 and ADCON1 registers determine which channel is connected to the sample-and-hold circuit.

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

12.2.3 ADC internal reference voltage

The chip has a built-in reference voltage. To detect this reference voltage, the CHS[4:0] bits should be set to 11111.

12.2.4 ADC reference voltage

The ADC reference voltage can be selected from the internal LDO output or provided by the chip's VDD and GND. The internal reference voltage options are 2.0V, 2.4V, and 3.0V.

When selecting the internal reference voltage, a slower conversion clock should be chosen. Please refer to the conversion clock section for details.

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

12.2.5 Conversion clock

The clock source for the conversion can be selected by software by setting the ADCS bit in the ADCON0 register. There are 4 possible clock frequencies to choose from as follows.

- $F_{HSI}/16$
- $F_{HSI}/32$
- $F_{HSI}/64$
- $F_{HSI}/128$

The time to complete a one-bit conversion is defined as the TAD, and a complete 12-bit conversion requires 16 TAD cycles.

The corresponding TAD specification must be complied with in order to obtain the correct conversion result, the following table is an example of the correct ADC clock selection.

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

Reference voltage (V)	Operating voltage (V)	Fastest frequency division F_{HSI}=16M	Time for one AD conversion (us)
VDD	4.0~5.5	$F_{HSI}/16$	16
VDD	2.7~4.0	$F_{HSI}/32$	32
3.0	4.0~5.5	$F_{HSI}/32$	32
3.0	3.3~4.0	$F_{HSI}/64$	64
2.4	4.0~5.5	$F_{HSI}/32$	32
2.4	2.7~4.0	$F_{HSI}/64$	64
2.0	4.0~5.5	$F_{HSI}/64$	64
2.0	2.7~4.0	$F_{HSI}/128$	128

12.2.6 ADC interrupt

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

12.2.7 Result formatting

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

12.3 Working principle of ADC

12.3.1 Start conversion

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

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

12.3.2 Complete conversion

When the conversion is completed, the ADC module will:

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

12.3.3 Stop conversion

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

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

12.3.4 Working principle of ADC in sleep mode

The ADC module cannot operate in sleep mode.

12.3.5 A/D conversion procedure

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

1. Port configuration:
- Configures the pin as an input pin (see TRIS register).
2. ADC mod configuration:
- Select ADC conversion clock;
- Select ADC input channel;
- Choose the format of the result;
- Start the ADC mod.
3. ADC interrupt (optional) configuration:
- Clear ADC interrupt flag bit;
- Enable ADC interrupt;
- Enable peripheral interrupt;
- Enable global interrupt.
4. Wait for the required acquisition time.
5. Set GO/$\overline{\text{DONE}}$ to 1 to start the conversion.
6. Wait for the ADC conversion to finish by one of the following methods:
- Query the GO/$\overline{\text{DONE}}$ bit.
- Wait for ADC interrupt (enable interrupt).
7. Read ADC results.
8. Clear the ADC interrupt flag bit to zero (this operation is required if interrupts are enabled).

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           ;delay 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
```

12.4 ADC related registers

There are four main registers related to ADC conversion: the control registers ADCON0 and ADCON1, and the data registers ADRESH and ADRESL.

AD control register ADCON0(95H)

95H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	ADON
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit6 ADCS<1:0>: A/D conversion clock select bit
00= F_{HSI}/16
01= F_{HSI}/32
10= F_{HSI}/64
11= F_{HSI}/128

Bit5~Bit2 CHS<3:0>: The lower four bits of the analog channel selection and CHS4 form a five-bit channel selection.
CHS<4:0>:
00000= AN0
00001= AN1
00010= AN2
00011= AN3
00100= AN4
00101= AN5
00110= Reserved
00111= Reserved
01000= AN8
01001= AN9
01010= Reserved
01011= AN11
…
01110= AN14
01111= AN15
11111= 1.2V (fixed reference voltage)
Others= Reserved

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

Bit0 ADON: ADC enable bit
1= Enable ADC
0= Disable ADC, no operating current consumption.

AD data high bit register ADCON1(96H)

96H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON1	ADFM	CHS4	---	---	---	LDO_EN	LDO_SEL1	LDO_SEL0
R/W	R/W	R/W	---	---	---	R/W	R/W	R/W
Reset value	0	0	---	---	---	0	0	0

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

Bit6 CHS4: Channel select bit

Bit5~Bit3 Unused

Bit2 LDO_EN: Internal reference voltage enable bit
1= Enable ADC internal LDO reference voltage
Maximum effective ADC precision is 8 bits when internal LDO is selected as reference voltage.
0= VDD is used as the ADC reference voltage

Bit1~Bit0 LDO_SEL<1:0>: Reference voltage selection bit
11= 3.0V
10= 2.4V
0x= 2.0V

AD data high bit register ADRESH(99H), ADFM=0

99H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESH	ADRES11	ADRES10	ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4
R/W	R	R	R	R	R	R	R	R
Reset value	X	X	X	X	X	X	X	X

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

AD data low bit register ADRESL(98H), ADFM=0

98H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESL	ADRES3	ADRES2	ADRES1	ADRES0	----	----	----	----
R/W	R	R	R	R	----	----	----	----
Reset value	X	X	X	X	----	----	----	----

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

Bit3~Bit0 Unused

AD data high bit register ADRESH(99H), ADFM=1

99H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESH	----	----	----	----	----	----	ADRES11	ADRES10
R/W	----	----	----	----	----	----	R	R
Reset value	----	----	----	----	----	----	X	X

Bit7~Bit2 Unused

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

High 2 bits of the 12-bit conversion result

AD data low bit register ADRESL(98H), ADFM=1

98H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESL	ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4	ADRES3	ADRES2
R/W	R	R	R	R	R	R	R	R
Reset value	X	X	X	X	X	X	X	X

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

Bits 2-9 of the 12-bit conversion result

Note: When ADFM = 1, the ADC conversion result only stores the high 10 bits of the 12-bit result. The ADRESH register stores the high 2 bits, and the ADRESL register stores bits 2 to 9.

13. Electrical Parameters

13.1 Limit parameters

Supply voltage……GND-0.3V~GND+6.0V
Storage temperature……-50℃~125℃
Working temperature……-40℃~85℃
Port input voltage……GND-0.3V~VDD+0.3V
Maximum positive current for all ports……200mA
Maximum negative current for all ports……-150mA

Note: If the device operating conditions exceed the above “limit parameters”, it may cause permanent damage to the device. The above values are extreme values for the operating conditions, and we do not recommend that the device be operated outside of the range specified in this specification. The stability of the device will be affected if it is operated for a long period of time under extreme conditions.

13.2 DC characteristics

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

Symbol	Parameter	Test condition		Min.	Typ.	Max.	Unit
		VDD	Condition				
VDD	Operating voltage	-	F_{SYS}=16MHz/2T	V_{LVR4}	-	5.5	V
		-	F_{SYS}=16MHz/4T	V_{LVR1}	-	5.5	V
I_{DD}	Operating current	5V	F_{SYS}=16MHz, All analog modules are disabled	-	2.5	-	mA
		5V	F_{SYS}=8MHz, All analog modules are disabled	-	1	-	mA
		3V	F_{SYS}=16MHz, All analog modules are disabled	-	1.5	-	mA
		3V	F_{SYS}=8MHz, All analog modules are disabled	-	0.5	-	mA
I_{STB}	Static current	5V	LVR=DIS WDT=DIS	-	1.5	5	uA
		3V	LVR=DIS WDT=DIS	-	0.8	3	uA
		5V	LVR=DIS WDT=EN	-	4.8	12	uA
		3V	LVR=DIS WDT=EN	-	2.1	5.5	uA
V_{IL}	Low level input voltage	-	----	-	-	0.3VDD	V
V_{IH}	High level input voltage	-	----	0.7VDD	-	-	V
V_{OH}	High level output voltage	-	No load	0.9VDD	-	-	V
V_{OL}	Low level output voltage	-	No load	-	-	0.1VDD	V
R_{PH}	Pull-up resistor value	5V	Vo=0.5VDD	-	32	-	KΩ
		3V	Vo=0.5VDD	-	52	-	KΩ
R_{PL}	Pull-down resistor value	5V	Vo=0.5VDD	-	34	-	KΩ
		3V	Vo=0.5VDD	-	56	-	KΩ
I_{OL}	Output port positive current	5V	V_{OL}=0.3VDD	-	37	-	mA
		3V	V_{OL}=0.3VDD	-	17	-	mA
I_{OH}	Output port negative current	5V	V_{OH}=0.7VDD	-	-16	-	mA
		3V	V_{OH}=0.7VDD	-	-7	-	mA
V_{BG}	Internal reference voltage 1.2V	VDD=2.5~5.5V T_A=25°C		-1.5%	1.2	+1.5%	V
		VDD=2.0~5.5V T_A=25°C		-2.5%	1.2	+2.5%	V
		VDD=2.5~5.5V T_A=-40~85°C		-2.0%	1.2	+2.0%	V
		VDD=2.0~5.5V T_A=-40~85°C		-3.0%	1.2	+3.0%	V

13.3 Comparator characteristics

(TA= 25℃, unless otherwise specified)

Symbol	Parameter	Test condition	Min.	Typ.	Max.	Unit
VDD	Operating voltage range	-	2.0	-	5.5	V
Iwork	Operating current	VDD=5V COMP+=2V COMP-=2V	-	34	46	uA
		VDD=3V COMP+=1V COMP-=1V	-	20	26	uA
I_{BG}	BG operating current	VDD=5V	-	35	46	uA
		VDD=3V	-	33	44	uA
V_{IN}	Input common mode voltage range	-	0	-	VDD-1.3	V
Vos	Offset voltage	-	-	-	±13	mV
LSB	Minimum resolution	-	-	10	20	mV
Tr	Response time	-	-	-	6	us
-	Internal resistor divider erro	VDD=5V V_R>1V	-1%	-	+1%	-
		VDD=5V V_R<1V	-2%	-	+2%	-

Note: V_R is the internal resistor voltage divider output value.

13.4 ADC electrical characteristics

(T_A= 25℃, unless otherwise specified)

Symbol	Parameter	Test condition	Min.	Typ.	Max.	Unit
V_{ADC}	ADC operating voltage	V_{ADREF}= VDD, F_{ADCCLK}=1MHz	3.0	-	5.5	V
		V_{ADREF}= VDD, F_{ADCCLK}=500kHz	2.7	-	5.5	V
		V_{ADREF}=2.0V, F_{ADCCLK}=250kHz	2.7	-	5.5	V
		V_{ADREF}=2.4V, F_{ADCCLK}=250kHz	2.7	-	5.5	V
		V_{ADREF}=3.0V, F_{ADCCLK}=500kHz	3.3	-	5.5	V
I_{ADC}	ADC conversion current	V_{ADC}=5V, V_{ADREF}= VDD, F_{ADCCLK}=500kHz	-	-	500	uA
		V_{ADC} =3V, V_{ADREF}= VDD, F_{ADCCLK}=500kHz	-	-	200	uA
V_{AIN}	ADC input voltage	V_{ADC}=5V, V_{ADREF} = VDD, F_{ADCCLK}=500kHz	0	-	V_{ADC}	V
DNL1	Differential nonlinearity error	V_{ADC}=5V, V_{ADREF} = VDD, F_{ADCCLK}=1MHz	-	±4	-	LSB
INL1	Integral nonlinearity error	V_{ADC}=5V, V_{ADREF} = VDD, F_{ADCCLK}=1MHz	-	±8	-	LSB
DNL2	Differential nonlinearity error	V_{ADC}=5V, V_{ADREF} = 3.0V, F_{ADCCLK}=500kHz, V_{AIN}<1V	-	±4	-	LSB
INL2	Integral nonlinearity error	V_{ADC}=5V, V_{ADREF} = 3.0V, F_{ADCCLK}=500kHz, V_{AIN}<1V	-	±16	-	LSB
DNL3	Differential nonlinearity error	V_{ADC}=5V, V_{ADREF} = 2.4V, F_{ADCCLK}=250kHz, V_{AIN}<0.8V	-	±4	-	LSB
INL3	Integral nonlinearity error	V_{ADC}=5V, V_{ADREF} = 2.4V, F_{ADCCLK}=250kHz, V_{AIN}<0.8V	-	±16	-	LSB
DNL4	Differential nonlinearity error	V_{ADC}=5V, V_{ADREF} = 2.0V, F_{ADCCLK}=250kHz, V_{AIN}<0.7V	-	±4	-	LSB
INL4	Integral nonlinearity error	V_{ADC}=5V, V_{ADREF} = 2.0V, F_{ADCCLK}=250kHz, V_{AIN}<0.7V	-	±16	-	LSB
T_{ADC}	ADC conversion time	-	-	16	-	T_{ADCCLK}

Remark: Low-temperature specification values are guaranteed by the design, and are not tested in mass production.

13.5 Power-on reset characteristics

(T_A= 25˚C, unless otherwise specified)

Symbol	Parameter	Test condition	Min.	Typ.	Max.	Unit
t_{VDD}	VDD rising rate	-	0.05	-	-	V/ms
V_{LVR1}	LVR set voltage=1.8V	VDD=1.6~5.5V	1.7	1.8	1.9	V
V_{LVR2}	LVR set voltage=2.0V	VDD=1.8~5.5V	1.9	2.0	2.1	V
V_{LVR3}	LVR set voltage=2.5V	VDD=2.3~5.5V	2.4	2.5	2.6	V
V_{LVR4}	LVR set voltage=3.0V	VDD=2.8~5.5V	2.9	3.0	3.1	V

13.6 AC electrical characteristics

(T_A= 25˚C, unless otherwise specified)

Symbol	Parameter	Test condition	Min.	Typ.	Max.	Unit
F_{WDT}	WDT clock source	VDD=2.5~5.5V T_A=25˚C	-20%	32	+20%	KHz
		VDD=1.8~5.5V T_A=25˚C	-35%	32	+35%	KHz
		VDD=2.5~5.5V T_A=-40~85˚C	-30%	32	+30%	KHz
		VDD=1.8~5.5V T_A=-40~85˚C	-50%	32	+50%	KHz
F_{INTRC}	Internal frequency 16MHz	VDD=4.0~5.5V TA=25˚C	-1.5%	16	+1.5%	MHz
		VDD=2.5~5.5V TA=25˚C	-2.0%	16	+2.0%	MHz
		VDD=1.8~5.5V TA=25˚C	-3.2%	16	+3.2%	MHz
		VDD=4.0~5.5V TA=-40~85˚C	-2.5%	16	+2.5%	MHz
		VDD=2.5~5.5V TA=-40~85˚C	-3.5%	16	+3.5%	MHz
		VDD=1.8~5.5V TA=-40~85˚C	-5.0%	16	+5.0%	MHz

13.7 LSE characteristics

(TA= 25℃, unless otherwise specified)

Symbol	Parameter	Test condition	Min.	Typ.	Max.	Unit
VDD	Operating voltage range	-	1.8	-	5.5	V
F_{LSE}	LSE oscillator frequency	-	-	32.768	-	KHz
C_1	OSCIN pin matching capacitance	-	-	22	-	pF
C_2	OSCOUT pin matching capacitance	-	-	22	-	pF
I_{LSE}	LSE operating current	VDD=5V C1=22 pF C2=22pF	-	20	-	uA
		VDD=3V C1=22 pF C2=22pF	-	8	-	uA
T_{LSE}	LSE stabilization time	VDD=5V C1=22 pF C2=22pF	-	260	700	ms
		VDD=3V C1=22 pF C2=22pF	-	300	1000	ms

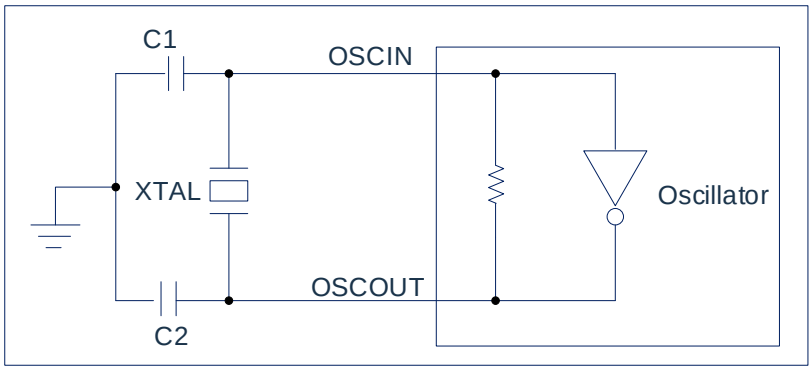


Figure 13-1: LSE typical application circuit

13.8 EMC characteristics

13.8.1 EFT electrical characteristics

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

Note: The immunity performance against Electrical Fast Transient (EFT) pulses is closely related to system design aspects, including power supply structure, circuit design, layout and wiring, chip configuration, program structure, and more. The EFT parameters listed in the table are results obtained from testing on CMS internal testing platforms and may not apply universally to all application environments. These test data serve as reference only. Various aspects of system design can influence EFT performance. In applications where high EFT immunity is required, it is advisable to design while minimizing the impact of interference sources on system operation. It is recommended to analyze interference paths and optimize designs to achieve the best immunity performance against EFT disturbances.

13.8.2 ESD electrical characteristics

Symbol	Parameter	Test condition	Classification
V_{ESD}	Electrostatic discharge (Human-Body Model: HBM)	T_A = + 25°C, JEDEC EIA/JESD22- A114	Class 2

13.8.3 Latch-up electrical characteristics

Symbol	Parameter	Test condition	Classification
LU	Static latch-up class	JEDEC STANDARD NO.78D NOVEMBER 2011	Class I A (T_A = +25°C)

14. Instruction

14.1 Instruction set

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

mnemonic		operation	period	symbol
bit operation				
CLRB	[R],b	Clear some bit in data memory R	1	NONE
SETB	[R],b	Set some bit in data memory R to 1	1	NONE
math operation				
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 subroutine, save I to ACC	2	NONE
RETI		Return from interrupt	2	NONE
CALL	ADD	Subroutine call	2	NONE
JP	ADD	Unconditional jump	2	NONE
conditional transfer				
SZB	[R],b	If the b bit of data memory R is “0”, skip the next instruction	1 or 2	NONE
SNZB	[R],b	If the b bit of data memory R is “1”, skip the next instruction	1 or 2	NONE
SZA	[R]	data memory R is sent to ACC, if the content is “0”, skip the next instruction	1 or 2	NONE
SZR	[R]	If the content of data memory R is “0”, skip the next instruction	1 or 2	NONE
SZINCA	[R]	Add “1” to data memory R and put the result into ACC, if the result is “0”, skip the next oneinstructions	1 or 2	NONE
SZINCR	[R]	Add “1” to data memory R, put the result into R, if the result is “0”, skip the next instruction	1 or 2	NONE
SZDECA	[R]	Data memory R minus “1”, the result is put into ACC, if the result is “0”, skip the next instruction	1 or 2	NONE
SZDECR	[R]	Data memory R minus “1”, put the result into R, if the result is “0”, skip the next instruction	1 or 2	NONE

14.2 Instruction description

ADDA **[R]**

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

period: 1

affected flag bit: C, DC, Z, OV

example:

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

ADDR **[R]**

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

period: 1

affected flag bit: C, DC, Z, OV

example:

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

ADDCA **[R]**

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

period: 1

affected flag bit: C, DC, Z, OV

example:

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

ADDCR **[R]**

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

period: 1

affected flag bit: C, DC, Z, OV

example:

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

ADDIA **i**

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

period: 1

affected flag bit: C, DC, Z, OV

example:

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

ANDA **[R]**

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

period: 1

affected flag bit: Z

example:

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

ANDR **[R]**

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

period: 1

affected flag bit: Z

example:

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

ANDIA **i**

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

period: 1

affected flag bit: Z

example:

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

CALL **add**

operation: Call subroutine

period: 2

affected flag bit: none

example:

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

CLRA

operation: ACC clear

period: 1

affected flag bit: Z

example:

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

CLR **[R]**

operation: Register R clear

period: 1

affected flag bit: Z

example:

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

CLRB **[R],b**

operation: Clear b bit on register R

period: 1

affected flag bit: None

example:

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

CLRWDT

operation: Clear watchdog timer

period: 1

affected flag bit: TO, PD

example:

```
CLRWDT          ;watchdog timer clear
```

COMA **[R]**

operation: Reverse register R, save the result to ACC

period: 1

affected flag bit: Z

example:

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

COMR **[R]**

operation: Reverse register R, save the result to R

period: 1

affected flag bit: Z

example:

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

DECA **[R]**

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

period: 1

affected flag bit: Z

example:

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

DECR **[R]**

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

period: 1

affected flag bit: Z

example:

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

HSUBA **[R]**

operation: ACC subtract R, save the result to ACC

period: 1

affected flag bit: C,DC,Z,OV

example:

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

HSUBR **[R]**

operation: ACC subtract R, save the result to R

period: 1

affected flag bit: C,DC,Z,OV

example:

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

HSUBCA **[R]**

operation: ACC subtract $\overline{C}$, save the result to ACC

period: 1

affected flag bit: C,DC,Z,OV

example:

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

HSUBCR **[R]**

operation: ACC subtract $\overline{C}$, save the result to R

period: 1

affected flag bit: C,DC,Z,OV

example:

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

INCA **[R]**

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

period: 1

affected flag bit: Z

example:

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

INCR **[R]**

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

period: 1

affected flag bit: Z

example:

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

JP **add**

operation: Jump to add address

period: 2

affected flag bit: None

example:

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

LD **A,[R]**

operation: Load the value of R to ACC

period: 1

affected flag bit: Z

example:

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

LD **[R],A**

operation: Load the value of ACC to R

period: 1

affected flag bit: None

example:

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

LDIA **i**

operation: Load i to ACC

period: 1

affected flag bit: None

example:

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

NOP

operation: Empty instructions

period: 1

affected flag bit: None

example:

```
NOP
NOP
```

ORIA **i**

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

period: 1

affected flag bit: Z

example:

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

ORA **[R]**

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

period: 1

affected flag bit: Z

example:

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

ORR **[R]**

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

period: 1

affected flag bit: Z

example:

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

RET

operation: Return from subroutine
period: 2
affected flag bit: None
example:

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

RET i

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

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

RETI

operation: Interrupt return
period: 2
affected flag bit: None
example:

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

RLCA [R]

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

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

RLCR **[R]**

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

period: 1

affected flag bit: C

example:

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

RLA **[R]**

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

period: 1

affected flag bit: None

example:

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

RLR **[R]**

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

period: 1

affected flag bit: None

example:

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

RRCA **[R]**

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

period: 1

affected flag bit: C

example:

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

RRCR **[R]**

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

period: 1

affected flag bit: C

example:

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

RRA **[R]**

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

period: 1

affected flag bit: None

example:

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

RRR **[R]**

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

period: 1

affected flag bit: None

example:

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

SET **[R]**

operation: Set all bits in register R as 1

period: 1

affected flag bit: None

example:

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

SETB **[R],b**

operation: Set b bit in register R to 1

period: 1

affected flag bit: None

example:

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

STOP

operation: Enter sleep
period: 1
affected flag bit: TO, PD
example:

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

SUBIA **i**

operation: I minus ACC, save the result to ACC
period: 1
affected flag bit: C,DC,Z,OV
example:

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

SUBA **[R]**

operation: Register R minus ACC, save the result to ACC
period: 1
affected flag bit: C,DC,Z,OV
example:

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

SUBR **[R]**

operation: Register R minus ACC, save the result to R
period: 1
affected flag bit: C,DC,Z,OV
example:

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

SUBCA **[R]**

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

period: 1

affected flag bit: C,DC,Z,OV

example:

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

SUBCR **[R]**

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

period: 1

affected flag bit: C,DC,Z,OV

example:

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

SWAPA **[R]**

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

period: 1

affected flag bit: None

example:

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

SWAPR **[R]**

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

period: 1

affected flag bit: None

example:

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

SZB **[R],b**

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

period: 1 or 2

affected flag bit: None

example:

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

SNZB **[R],b**

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

period: 1 or 2

affected flag bit: None

example:

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

SZA **[R]**

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

period: 1 or 2

affected flag bit: None

example:

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

SZR **[R]**

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

period: 1 or 2

affected flag bit: None

example:

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

SZINCA **[R]**

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

period: 1 or 2

affected flag bit: None

example:

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

SZINCR **[R]**

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

period: 1 or 2

affected flag bit: None

example:

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

SZDECA **[R]**

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

period: 1 or 2

affected flag bit: None

example:

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

SZDECR **[R]**

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

period: 1 or 2

affected flag bit: None

example:

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

TESTZ **[R]**

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

period: 1

affected flag bit: Z

example:

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

XORIA **i**

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

period: 1

affected flag bit: Z

example:

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

XORA **[R]**

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

period: 1

affected flag bit: Z

example:

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

XORR **[R]**

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

period: 1

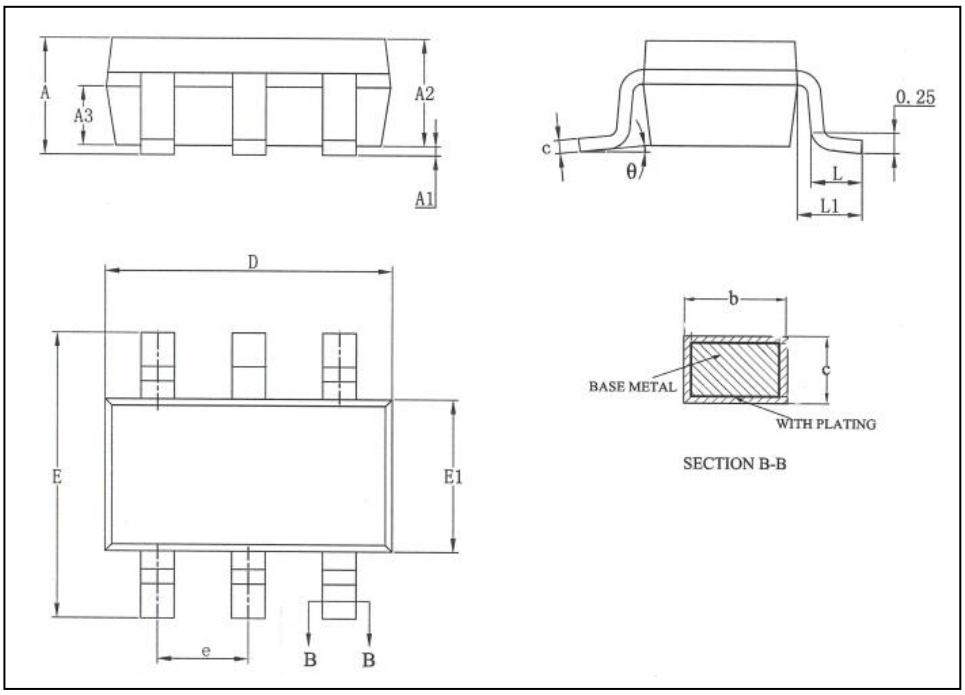
affected flag bit: Z

example:

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

15. Package Dimensions

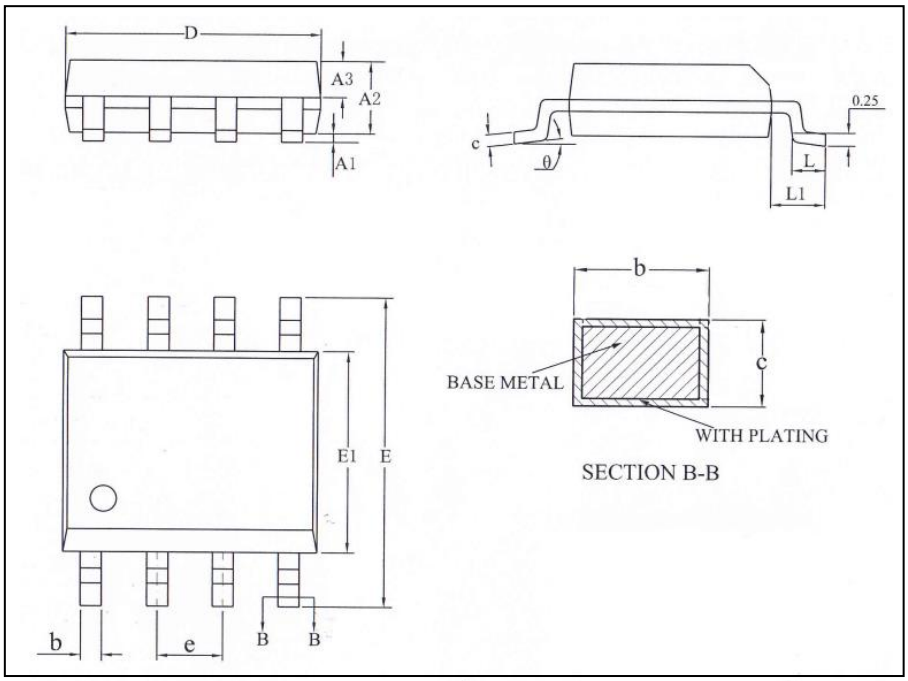
15.1 SOT23-6



Symbol	Millimeter		
	Min	Nom	Max
A	1	-	1.35
A1	0.01	-	0.15
A2	0.90	-	1.20
A3	0.55	-	0.80
b	0.30	-	0.50
c	0.12	-	0.21
D	2.77	-	3.07
E	2.65	2.80	3.00
E1	1.40	1.60	1.80
e	0.95BSC		
L	0.30	-	0.60
L1	0.52	-	0.75
θ	0	-	8°

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

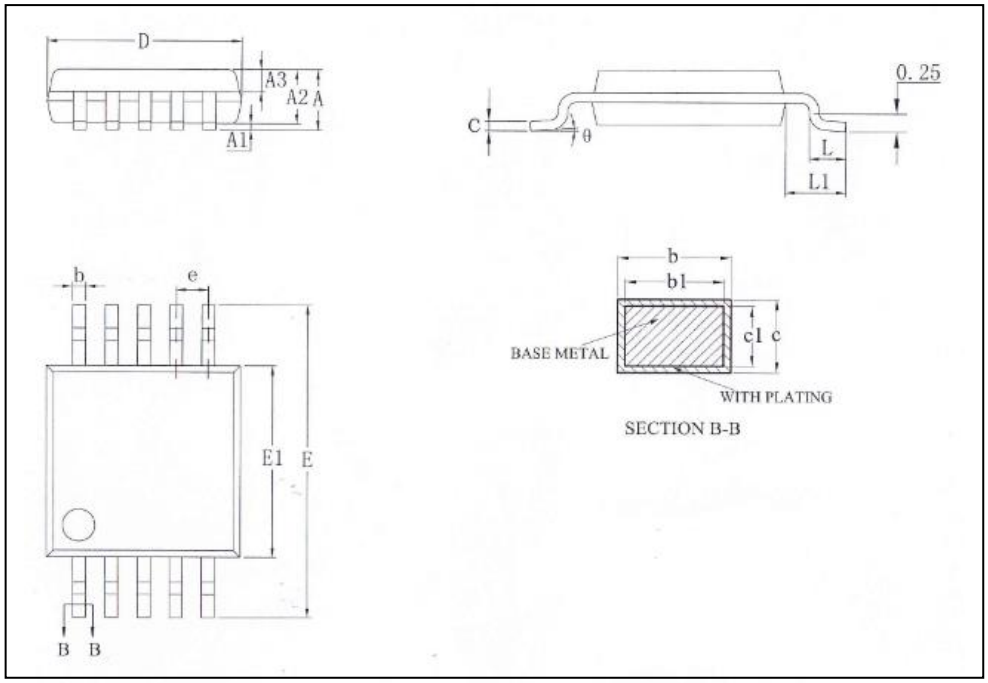
15.2 SOP8



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

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

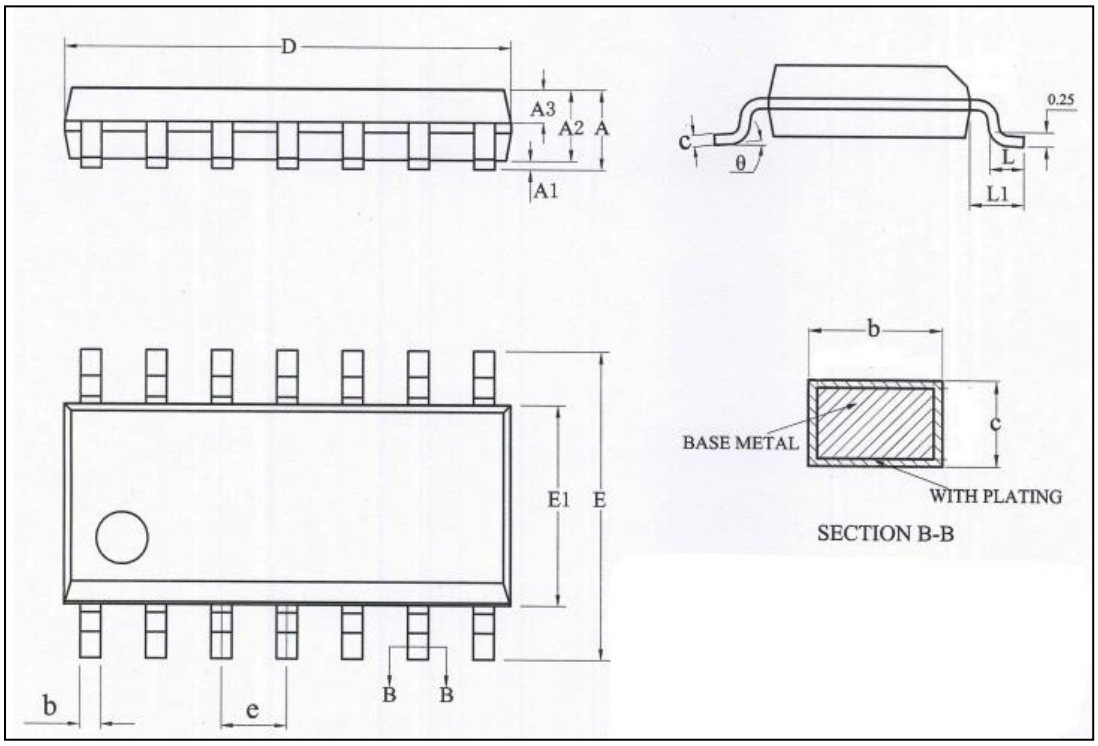
15.3 MSOP10



Symbol	Millimeter		
	Min	Nom	Max
A	-	-	1.10
A1	0.05	-	0.15
A2	0.75	0.85	0.95
A3	0.30	0.35	0.40
b	0.17	-	0.27
b1	0.17	0.20	0.23
c	0.08	-	0.23
c1	0.14	0.15	0.16
D	2.90	3.00	3.10
E	4.70	4.90	5.10
E1	2.90	3.00	3.10
e	0.50BSC		
L	0.40	-	0.80
L1	0.95REF		
θ	0	-	8°

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

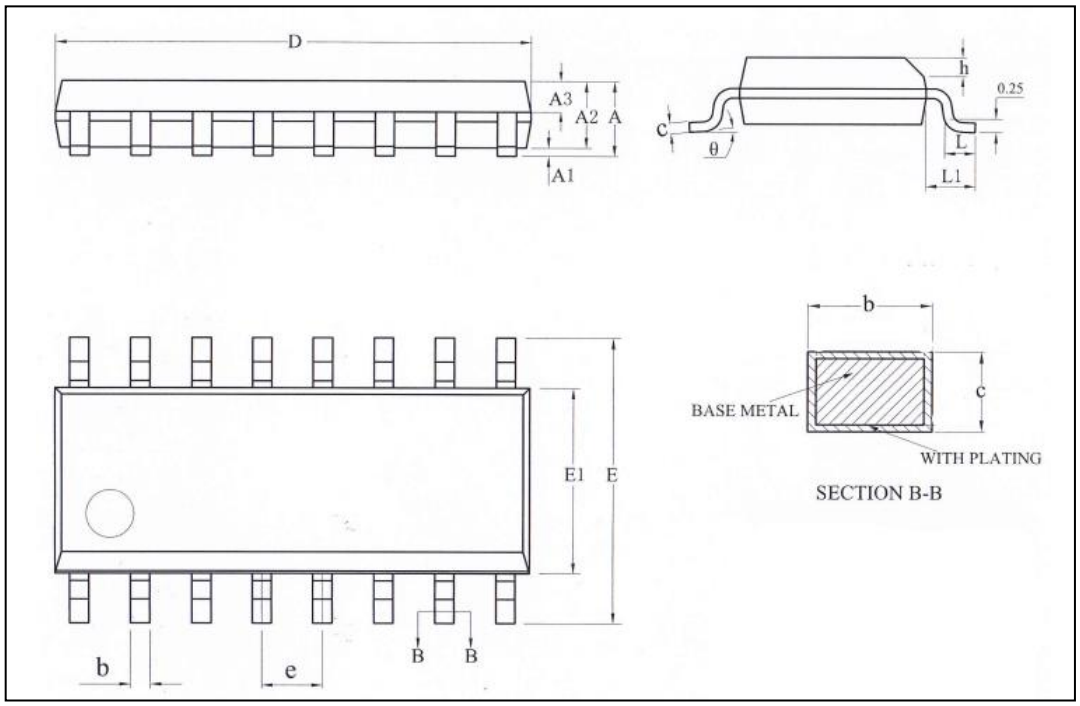
15.4 SOP14



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

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

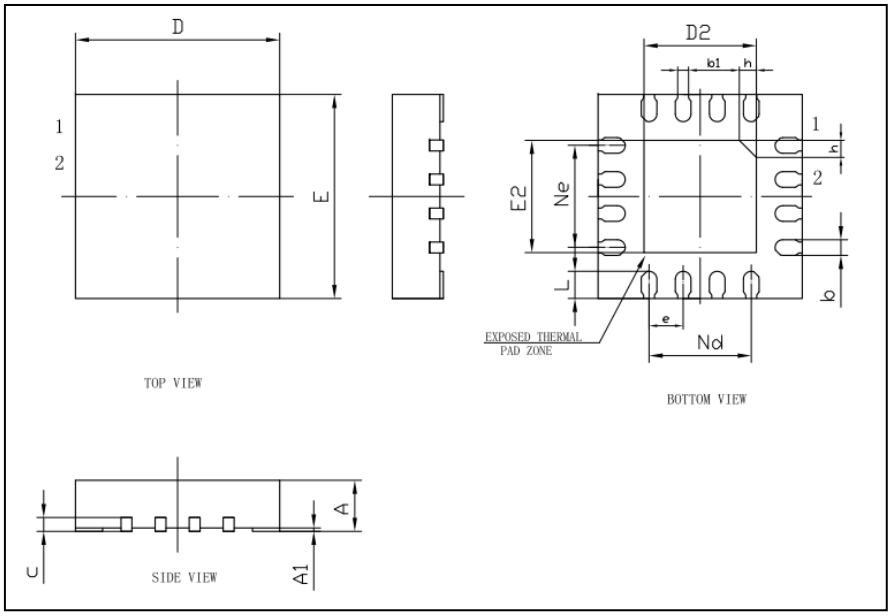
15.5 SOP16



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

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

15.6 QFN16(3*3*0.75-0.50)



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

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

16. Revision History

Version #	Date	Description of changes
V0.5.0	Nov. 2023	Initial release
V0.5.1	Jul. 2024	Revised model description tables
V0.5.2	Oct. 2024	Modified Chapter 15 package dimensions
	Nov.2024	Modified pin description
V1.0.0	Dec 2024	1) Deleted Section ADC internal LDO reference voltage characteristics 2) Modified the parameters of AC electrical characteristics
V1.0.1	Jul 2025	Modified the wording of the operational voltage range