JZ8FC003 Series MCU

User Guide

Built-in 16 Bit PWM / 12 Bit ADC / 1T 8051 18K Flash MCU

Table of Contents

1 Introduction

JZ8FC003 series chip is an 8-bit MCU based on 1T 8051 core, which usually runs 10 times faster than the traditional 8051 chip and has more superior performance. The built-in 18K Flash program memory, which can be repeatedly programmed many times, brings great convenience to users' development. It not only retains the basic features of traditional 8051 chip, but also integrates function modules such as 12 Bit ADC, 16 Bit PWM, UART, I^2C, OP-AMP and low voltage detection (LVD), and supports online simulation function. Three power-saving modes, IDLE, STOP and low-speed operation, are supported to accommodate applications with different power requirements. Widely used in consumer electronics and home appliances.

2 Basic Features

Core

- CPU: 1T 8051, up to 10 times faster than traditional 8051
- Compatible with 8051 instruction set, dual DPTR operation mode

Memory

- Flash: 18 bytes, supports multiple rewrites
- Flash can be divided into program space and data space, the data space can be used to store the power down need to save data, can omit the EEPROM
- RAM:256 bytes of internal RAM, 1024 bytes of external RAM

Operating Voltage

- Operating voltage: 1.8 - 5.5V

Clock System

- Internal Low Speed RC Oscillator: 131KHz
- Internal High Speed RC Oscillator: 32MHz, accuracy ±1% (3.3V@25°C) (default 2 divisions when used as system clock, i.e.16MHz)
- External High Speed Oscillator: 1 - 24MHz, External Clock Input: 1 -24MHz

TMC Functions

- The clock source is the Internal low Speed RC Oscillator, and the minimum unit of interrupt time is 512 Low Speed RC Oscillator clock cycles.
- Configurable interrupt time from 1 to 256 minimum units of time

Interruption System

- 15 effective interrupt sources
- Two levels of interrupt priority which also supports interrupt nesting
- 10 external interrupt sources, each external interrupt can be configured with any signal pin as an interrupt input pin

Timer

- Three 16-bit general-purpose timers: Timer 0, Timer 1, Timer 2

General Purpose IO (GPIO)

- Supports up to 18 GPIO ports, support push-pull, open-drain, strong pull-up, weak pull-up, strong pull-down, weak pull-down, high resistance mode

Analog/Digital Converter (ADC)

- Supports 12 channel 12-bit SAR ADC with the function of operational amplification and comparison embedded
- Supports 3 Reference Voltage : VDD, Internal Reference Voltage ,External Reference Voltage
- VDD voltage can be measured when internal voltage is selected as reference voltage
- Supports configurable comparator modes
- Supports the detection signal through the op-amp reduction and then conversion, reduction multiplier can be selected
- ADC can directly detect op-amp A output
- ADC can be used in conjunction with PWM, with the PWM interrupt initiating the ADC conversion

Operational Amplifier (AMP)

- Op-amp A has built-in correction mechanism, after correction, the offset voltage is less than 0.5mV under full temperature condition.

PWM

- Supports 6 channel 12-bit SAR ADC with the function of operational amplification and comparison embedded
- PWM0 ~ PWM5 can select any IO pins as PWM output pins
- Supports Complementary Mode and Deadtime Control which could be used to drive Burushless DC motor
- Supports center fixed mode or edge fixed mode
- Support software brake and hardware brake
- Supports PWM pause function
- Supports to output of internal clock directly
- Supports PWM interrupt

Low Voltage Detection (LVD)

- Configurable voltage detection range, four grades selectable: 2.0V/2.7V/3.7V/4.4V
- Low voltage reset /interrupt configurable

Reset mode

- The chip supports multiple reset sources: Hard Reset,Soft Reset, Watch Dog Reset, LVD Reset, Power On/Down

Watchdog

- 27-bit Watch Dog Timer, 16 bit precision configurable,with Watch Dog Reset and Interrupt configurable as well

UART (UART1/UART2)

- Supports 2 UART ports
- Supports 1-byte receive buffer

SPI

- Internal 1 x 4-wire SPI interface, supporting master-slave mode

I2C interface

- Internal 1-channel I^2C interface, support master-slave mode, support standard/fast/high-speed mode

Buzzer

- Internal 1-channel buzzer drive output

Multiplying /Dividing Unit (MDU)

- Supports 1 clock cycle 16-bit×16-bit multiplication
- Supports 8 clock cycles of 32 bit÷32-bit division
- Supports 1 clock cycle 32-bit data shift operation left and right

Program Download and Simulation

- Supports ISP and IAP
- Supports dual-line and single-line online simulation function

Low power consumption

- For STOP mode, current <7uA
- For IDLE mode, current <15uA
- For Low speed mode, current <25uA

Package Type：TSSOP20/QFN20/SOP20

3 Chip Model and Function Description

Table 3-1 JZ8FC003 Specific Models and Their Features

Models	Flash Storage [BYTE]]	External Ram[BYTE]	Internal high-speed RC oscillator	Internal low-speed RC oscillator	External high-speed crystal oscillator	GPIO	UART	I²C	16 bit PWM channels	SPI	12-bit ADC channels	General-purpose op-amps	Multiplier and Divider	ISP Upgrade	Simulation On Chip	Operating Voltage	Package Type
JZ8FC003T3	18K	1024	√	√	√	18	2	√	6	√	12	1	√	√	√	1.8-5.5	TSSOP20
JZ8FC003N2	18K	1024	√	√	√	18	2	√	6	√	12	1	√	√	√	1.8-5.5	QFN20
JZ8FC003S4	18K	1024	√	√	√	18	2	√	6	√	12	1	√	√	√	1.8-5.5	SOP20

4 Block Diagram

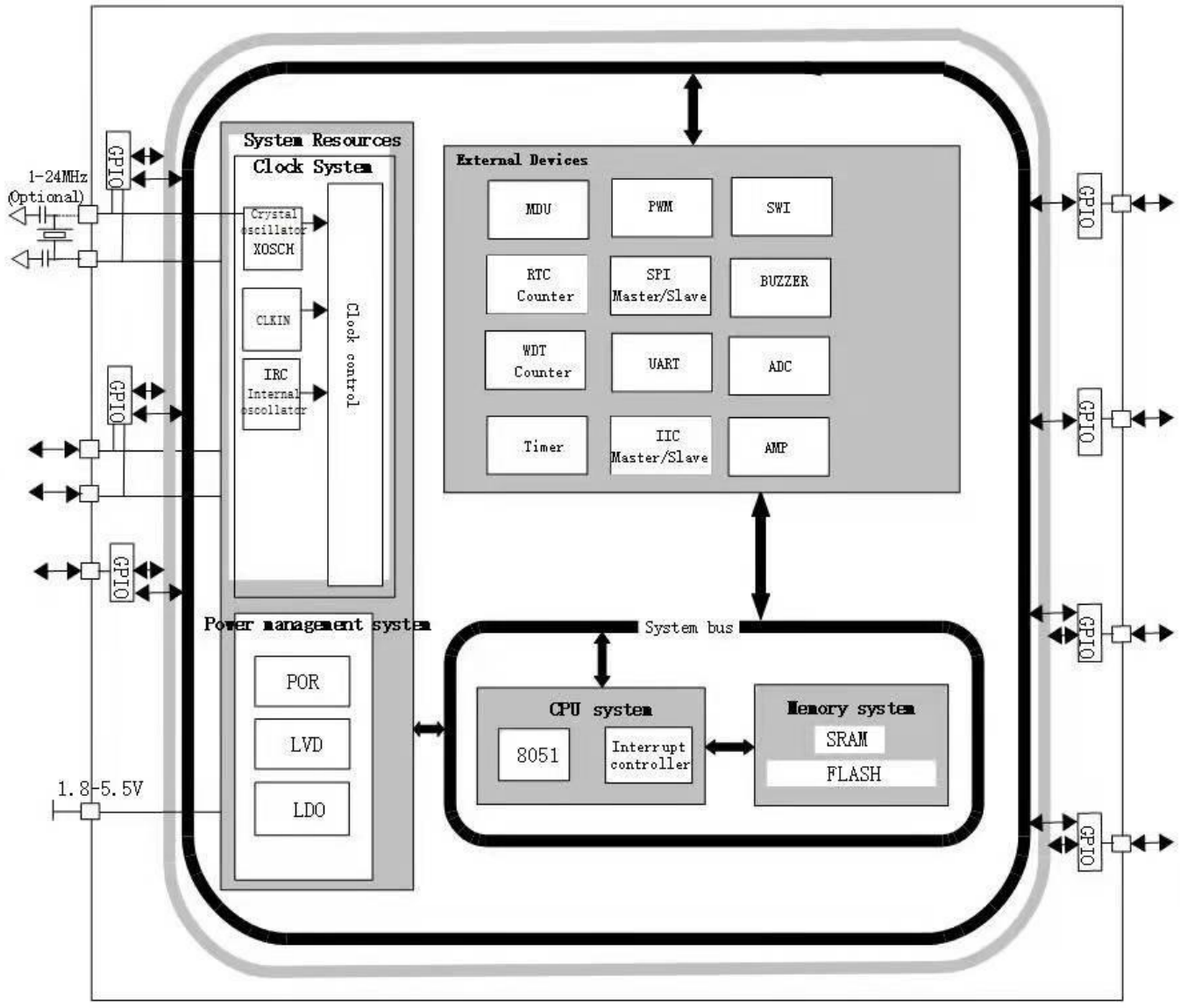


Figure 4-1-1 Chip Block Diagram

5 Pin Package and Description

5.1 Package Definition

Model：JZ8FC003T3/S4

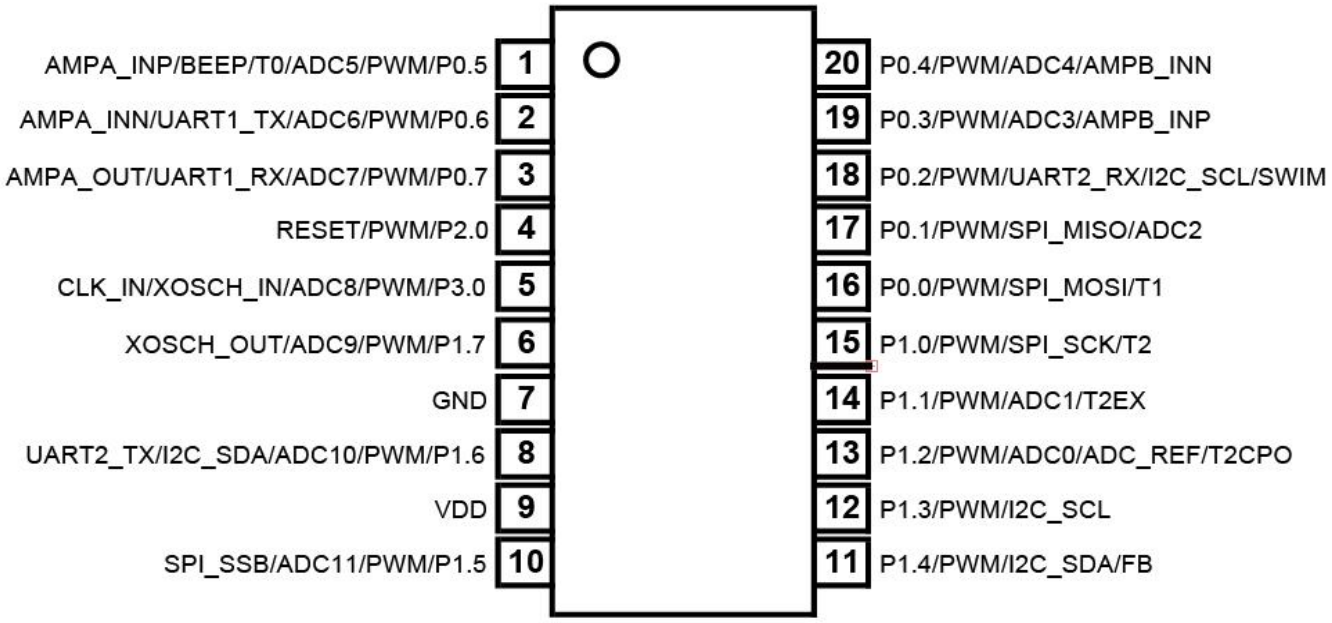


Figure 5-1-1 TSSOP20/SOP20 Package Diagram

Model：JZ8FC003N2

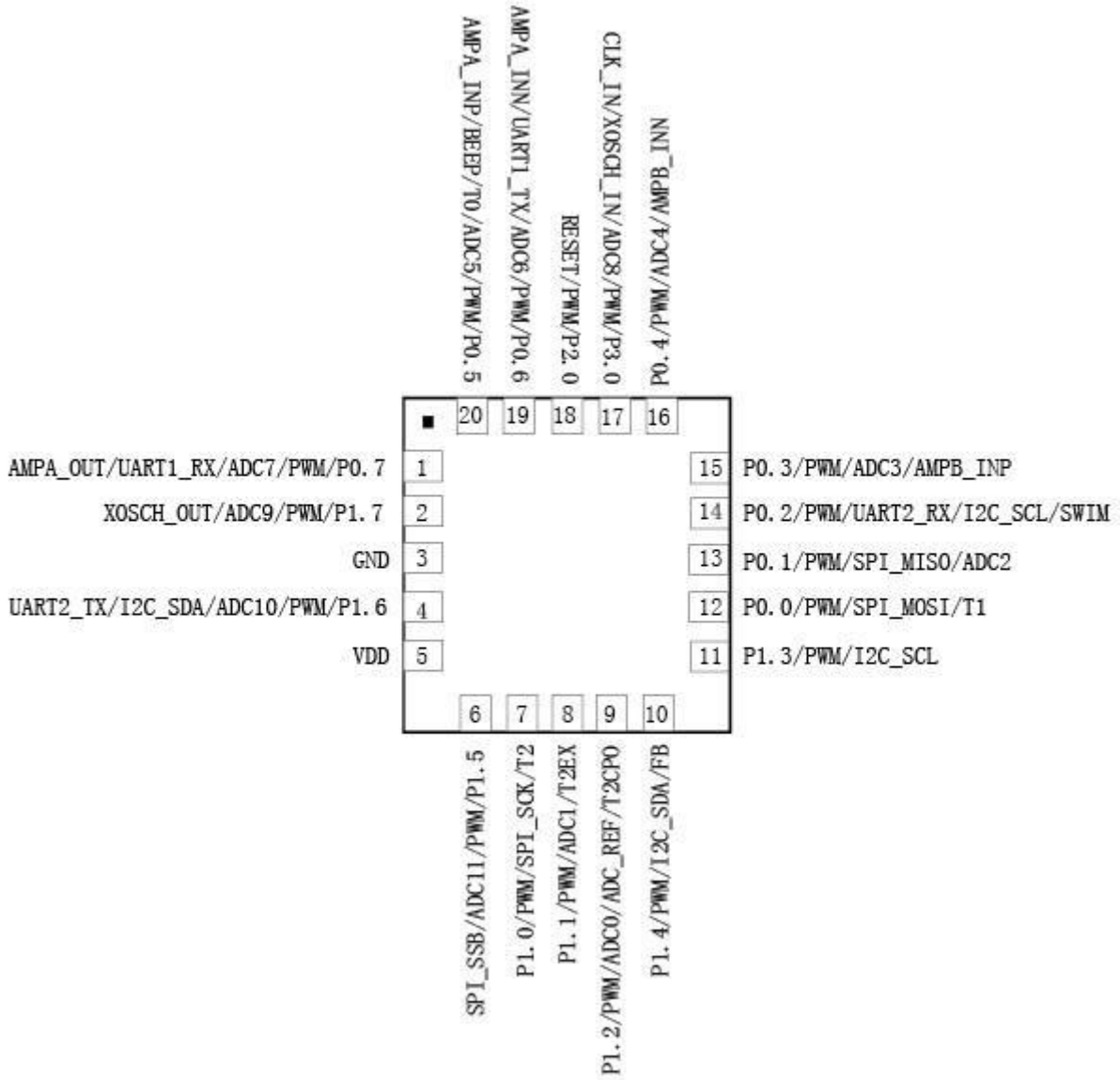


Figure 5-1-2 QFN20 package diagram

5.2 Pin Description

Table 5-2-1 Pin Description

Pin Sequence Number		Pin Name	Pin Function	Default Function
TSSOP20/ SOP20	QFN20			
1	20	P0.5/T0/ADC_CH[5]/BEEP/PWM/AMP_A_P/T2CP	General bi-directional I/O port Timer 0 T0 port ADC analog channel input Buzzer output PWM signal output Op-amp A input T2CP signal input	General bi-directional I/O port
2	19	P0.6/ADC_CH[6]/UART1_TX/PWM/AMP_A_N/T2CP	General bi-directional I/O port ADC analog channel input UART1 data sending port PWM signal output Op-amp A input T2CP signal input	General bi-directional I/O port
3	1	P0.7/ADC_CH[7]/UART1_RX/PWM/AMP_A_O/T2CP	General bi-directional I/O port ADC analog channel input UART1 data reception port PWM signal output Op-amp A output T2CP signal input	General bi-directional I/O port
4	18	P2.0/RESET/PWM/T2CP	General bi-directional I/O port Hard reset pins PWM signal output T2CP signal input	Hard reset
5	17	P3.0/ADC_CH[8]/XOSCH_IN/PWM/XCLK_IN/T2CP	General bi-directional I/O port ADC analog channel input External high-speed crystal input PWM signal output External high speed clock input port T2CP signal input	General bi-directional I/O port

6	2	P1.7/ ADC_CH[9]/ XOSCH_OUT/PWM/T2CP	General bi-directional I/O port ADC analog channel input External high speed crystal output PWM signal output T2CP signal input	General bi-directional I/O port
7	3	VSS	Power ground pin	Power ground pin
8	4	P1.6/ADC_CH[10]/UART2_TX/PWM/I2C_SDA/T2CP	General bi-directional I/O port ADC analog channel input UART2 data sending port PWM signal output I2C data transfer port T2CP signal input	I2C data transfer port
9	5	VDD	Chip power supply pins	Chip power supply pins
10	6	P1.5/ADC_CH[11]/PWM/SPI_SSB/T2CP	General bi-directional I/O port ADC analog channel input PWM signal output SPI_SSB port T2CP signal input	General bi-directional I/O port
11	10	P1.4/FB/PWM/I2C_SDA/T2CP	General bi-directional I/O port FB port PWM signal output I2C data transfer port T2CP signal input	I2C data transfer port
12	11	P1.3/PWM/I2C_SCL/T2CP	General bi-directional I/O port PWM signal output I2C clock transfer port T2CP signal input	I2C clock transmission port
13	9	P1.2/ ADC_CH[0]/ADC_REF/PWM/T2CPO/T2CP	General bi-directional I/O port ADC analog channel input ADC external reference voltage input PWM signal output T2CPO signal output T2CP signal input	General bi-directional I/O port

14	8	P1.1/ADC_CH[1]/ PWM /T2CP	General bi-directional I/O port ADC analog channel input PWM signal output T2CP signal input	General bi-directional I/O port
15	7	P1.0/T2/PWM/SPI_SCK/T2CP	General bi-directional I/O port Timer 2 T2 port PWM signal output SPI clock transfer port T2CP signal input	General bi-directional I/O port
16	12	P0.0/T1/PWM/SPI_MOSI/T2CP	General bi-directional I/O port Timer 1 T1 port PWM signal output SPI_MOSI port T2CP signal input	General purpose bi-directional IO port
17	13	P0.1/ADC_CH[2]/PWM/SPI_MISO/T2CP	General bi-directional I/O port ADC analog channel input PWM signal output SPI_MISO port T2CP signal input	General bi-directional I/O port
18	14	P0.2/SWIM/UART2_RX/PWM/I2C_SCL/T2CP	General bi-directional I/O port Single-wire communication data port UART2 data reception port PWM signal output I2C clock transmission port T2CP signal input	Single wire communication data port
19	15	P0.3/ADC_CH[3]/PWM/AMP_B_P/T2CP	General bi-directional I/O port ADC analog channel input PWM signal output Op-amp B input port T2CP signal input	General bi-directional I/O port
20	16	P0.4/ ADC_CH[4]/PWM/AMP_B_N/T2CP	General bi-directional I/O port ADC analog channel input PWM signal output Op-amp B input port T2CP signal input	General bi-directional I/O port

Note: See Table 15-2-5 and Table 15-2-7 for details on how to set the signal pin multiplexing function

6 Central Processing Unit (CPU)

6.1 CPU Introduction

The JZ8FC003 series chips use a single-cycle 8051 CPU, which is fully compatible with the original MCS-51 instruction set. the CPU uses a pipelined architecture, and typically, the single-cycle 8051 CPU runs 10 times faster than a standard 8051 processor.

The features of this CPU are：

1T 8051 CPU

Compatible with 8051instruction set, for more you may refer to instruction set in Appendix

Double DPTR, so that the data could be moved quickly

6.2 Register Description

Program Counter (PC)

Program Counter (PC) is a 16-bit register without register address which is used to control the sequence of instructions. It is set to 0 after reset/power on and the machine will execute the program from zero address.

Accumulator (ACC)

Accumulator (ACC) is a special register and ‘A’ is used as its instruction mnemonic. It is often used to store the operand and result of logical/arithmetic computing.

General Register B

Register B cannot to be used without ACC in multiplying/dividing computing. Instruction MUL AB multiplies 8-bit unsigned number in ACC and B. The lower bytes (16 bit) and higher bytes(16 bit) of the computing result will be stored in A and B respectively. Furthermore, instruction DIV AB divides B by A, and the integer quotient will be stored in A with remainder stored in B. In addition, register B can also be used as general temporary storage register.

Stack Pointer (SP)

Stack Pointer(SP) is a 8 bit special register and indicates where the top of stack is in the internal RAM. It is initialized to 07H after a reset which makes stack actually starts from 08H. Since 08H~1FH belongs to working register group 1~3 , if they are used in program development, SP is recommended to be set to 80H or even higher.

Data Pointer (DPTR)

Data pointer DPTR0/DPTR1 are two 16-bit special register with their higher stored in register DP0H/DP1H respectively and lower bytes stored in register DP0H/DP1H respectively. By setting DPS(PSW.1) either of them can be used. For each DPTR, it can be seen as one 16-bit register or two independent 8-bit registers DP0H/DP1H and DP0L/DP1L.

Program Status Word (PSW)

Program Status Word(PSW) is a register indicates the statues of the CPU. The status bit of it will change correspondingly when the CPU is doing arithmetic or logical operations.

Table 6-2-1 Accumulator ACC

E0H	7	6	5	4	3	2	1	0
ACC	ACC[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Table 6-2-2 General Register B

F0H	7	6	5	4	3	2	1	0
B	B[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Table 6-2-3 Stack Pointer SP

81H	7	6	5	4	3	2	1	0
SP	SP[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	1	1	1

Table 6-2-4 Data Pointer DP0L

82H	7	6	5	4	3	2	1	0
DP0L	DP0L[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Table 6-2-5 Data Pointer DP0H

83H	7	6	5	4	3	2	1	0
DP0H	DP0H[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Table 6-2-6 Data Pointer DP1L

84H	7	6	5	4	3	2	1	0
DP1L	DP1L[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Table 6-2-7 Data Pointer DP1H

85H	7	6	5	4	3	2	1	0
DP1H	DP1H[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Table 6-2-8 Status Register PSW

D0H	7	6	5	4	3	2	1	0
PSW	CY	AC	F0	RS[1:0]		OV	DPS	P
R/W	R/W	R/W	R/W	R/W		R/W	R	R
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7	CY	Carry flag 0：There is no carry or borrow happened in arithmetic/logical operation 1： There is carry or borrow happened in arithmetic/logical operation
6	AC	Auxiliary Carry Flag 0：There is no auxiliary carry or borrow happened in arithmetic/logical operation 1： There is auxiliary carry or borrow happened in arithmetic/logical operation
5	F0	F0 flag It is defined by the user
4~3	RS	R0~R7 registers’ page selection 00：page 0(mapping to 00H-07H) 01：page 1(mapping to 08H-0FH) 10：page 2(mapping to 10H-17H) 11：page 3(mapping to 18H-1FH)
2	OV	Overflow flag 0： no overflow 1：overflow happened
1	DPS	DPTR selector, 0 for DPTR0, 1 for DPTR1
0	P	Parity flag 0：the number of 1 in ACC is even 1： the number of 1 in ACC in odd

Table 6-2-9 Register SPMAX

8100H	7	6	5	4	3	2	1	0
SPMAX	SPMAX[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7~0	SPMAX	SPMAX is used to record the maximum value of SP. Users can check this register using software to decide whether there is a risk that the stack may overflow

7 Memory Architecture

7.1 Random Access Memory (RAM)

The JZ8FC003 series chips provide 256 bytes of internal RAM and 1024 bytes of external RAM with the following memory address assignments：

- The low 128 bytes of internal RAM (address: 00H ~ 7FH) can be addressed directly or indirectly.
- The high 128 bytes of internal RAM (address: 80H ~ FFH) can only be indirectly addressed.

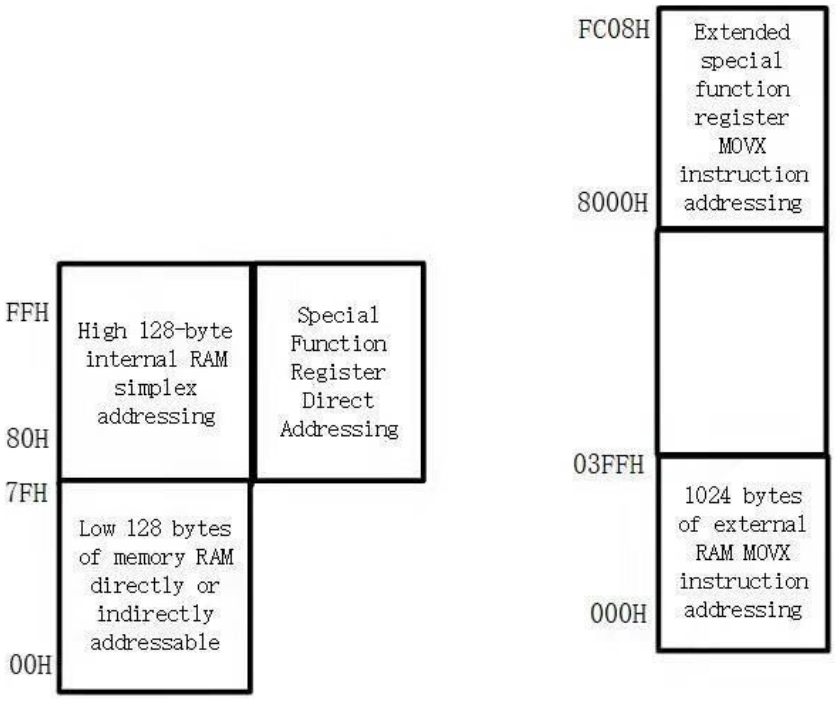


Figure 7-1-1 RAM Architecture

7.2 Special Function Register (SFR)

JZ8FC003 series chips provide SFR distribution compatible with traditional 8051, SFR and high 128 bytes of

internal RAM share the address 80H ~ FFH, can only be directly addressed, SFR mapping as shown in Table 7-2-1.

Table 7-2-1 Special Function Register (SFR) Mapping Table

	Bit-addressable	Non-bit addressable						
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8H	EXIP	EPIE	EPIF	EPCON	IDLSTL	IDLSTH	STPSTL	STPSTH
F0H	B	-	-	-	-	-	-	INDEX
E8H	EXIE	-	-	-	-	-	-	LVDCON
E0H	ACC	-	-	-	-	-	MDUCON	MDUDAT
D8H	-	-	PWMEN	PWMUPD	PWMCMX	PWMCON	PWMCFG	PWMDIVL
D0H	PSW	PWMDIVH	PWMDUTL	PWMDUTH	PWMAIF	PWMBIF	PWMCIF	PWMDIF
C8H	T2CON	T2MOD	T2CL	T2CH	TL2	TH2	-	-
C0H	-	-	-	-	-	-	-	-
B8H	IP	ADCON	ADCFGL	ADCFGH	ADCDL	ADCDH		
B0H	P3	I2CCON	I2CADR	I2CADM	I2CCCR	I2CDAT	I2CSTA	I2CFLG
A8H	IE	-	WDCON	WDFLG	WDVTHL	WDVTHH	-	-
A0H	P2	S2CON	S2BUF	S2RELL	S2RELH	SPCON	SPDAT	SPSTA
98H	-	-	S1CON	S1BUF	S1RELL	S1RELH	-	-
90H	P1	-	-	-	-	-	-	-
88H	TCON	TMOD	TL0	TL1	TH0	TH1	IT1CON	IT0CON
80H	P0	SP	DP0L	DP0H	DP1L	DP1H	PWCON	PCON

Due to the limited SFR address space, the JZ8FC003 series chips add extended special function registers to the external RAM address space, and the extended special function register mapping is shown in Figure 7-2-2.

Table 7-2-2 Extended Special Function Register Mapping Table

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
8000H	P00F	P01F	P02F	P03F	P04F	P05F	P06F	P07F
8008H	P10F	P11F	P12F	P13F	P14F	P15F	P16F	P17F
8010H	P20F	-	-	-	-	-	-	-
8018H	P30F	-	-	-	-	-	-	-
8080H	CKCON	CKSEL	CKDIV	IHCFGL	IHCFGH	ILCFG[L]	XHISEL	-
8088H	ADCALL	ADCALH	ACPDLL	ACPDLH	ACPDHL	ACPDHH	ADPWMTRG	ADOPC
8098H	PWMPS	PWMHS	PWMFBC	PWMFBS	PWMSBC	PWMBD		
80A0H	T2CPPS	T2CPCHS	T2CPF	T2CPL	T2CPH			
80A8H	TMCON	TMSNU						
80B0H	SWICON	SWIDAT	SWISTA	SWIOVT				
80B8H	AMPCON	AMPAOS						
80C0H	BZCON	BZDIVL	BZDIVH	BZDUTL	BZDUTH			
8100H	SPMAX	I2CIOS	-	-	-	-	-	-
8118H	UDCKS1	UDCKS2	-	-	-	-	FTCTL	TPCTL

8120H	P00C	P01C	P02C	P03C	P04C	P05C	P06C	P07C
8128H	P10C	P11C	P12C	P13C	P14C	P15C	P16C	P17C
8130H	P20C	-	-	-	-	-	-	-
8138H	P30C	-	-	-	-	-	-	-
FC00H	MECON	FSCMD	FSDAT	LOCK	PADRD	PTSL	PTSH	REPSET
FC08H								

7.3 Flash

7.3.1 Function Introduction

Flash memory contains 18K bytes of Flash main data area, Flash memory is rewritable, Flash memory is controlled by a specific set of registers, users can use these registers to read, write and erase, set the write protection and other operations.

7.3.2 Flash Architecture

- Flash consists of several sectors which are the smallest units for erasure. Each sector is 128 bytes.
- Flash can be divided into DATA area and PROGRAM area and the division unit is 128 byte. PROGRAM area is used to store use’ s program and DATA area is used to store data that needs to be protected during power off period.

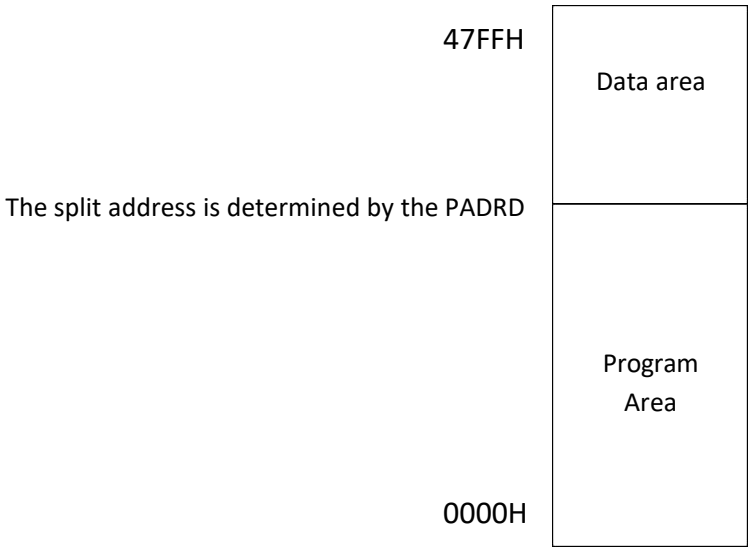


Figure 7-3-1 18K Flash Memory Structure

7.3.3 Flash Register Description

Table 7-3-3-1 Register MECON

FC00H	7	6	5	4	3	2	1	0
MECON	-	DPSTB	-	-	-	-	-	BOOT
R/W	-	R/W	-	-	-	-	-	R/W
Initial Value	-	0	-	-	-	-	-	0

Bit Number	Bit Symbol	Description
7	-	-
6	DPSTB	Flash SLEEP mode control in IDLE/STOP mode

		0：Flash in NORMAL mode while IDLE/STOP 1： Flash in SLEEP mode while IDLE/STOP Note：If DPSTB=1, when the chip enters IDLE/STOP mode, the Flash will enter SLEEP mode simultaneously and the power consumption of the Flash in SLEEP mode is 50nA. When the chip exits IDLE/STOP mode, Flash exits SLEEP mode as well.
5~1	-	-
0	BOOT	Set the program start space selection bit field after soft reset 0： Program runs from FLASH after soft reset 1：Program starts running from XRAM after soft reset

Table 7-3-3-2 Register FSCMD

FC01H	7	6	5	4	3	2	1	0
FSCMD	-	-	-	-	-	CMD[2:0]		
R/W	-	-	-	-	-	R/W		
Initial Value	-	-	-	-	-	0	0	0

Bit Number	Bit Symbol	Description
7~3	-	-
2~0	CMD	Command Register 000： No operations 100：Erase the whole Flash 001：Read Flash DATA area 010： Write Flash DATAarea 011：Erase one sector of the Flash DATA area 101：Read Flash program area 110：Write Flash program area 111：Erase one sector of the Flash program area *Note：* *1. The CMD is automatically cleared after the erase command is executed.* *2.The CMD remains unchanged after the read and write commands are written and then completed by reading and writing to FSDAT.*

Table 7-3-3-3 Register FSDAT

FC02H	7	6	5	4	3	2	1	0
FSDAT	FSDAT[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7~0	FSDAT	Flash Data Register

Table 7-3-3-4 Register LOCK

FC03H	7	6	5	4	3	2	1	0
LOCK								
R		REPE			FLKF	PLKF	DLKF	ILKF
W	LOCK[7:0]							
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
Write operation		
7~0	LOCK	28H：Unlock Flash programmable areas 29H：Unlock Flash program area 2AH： Unlock Flash DATA area AAH：Flash is locked, no write erase operation is possible
Read operation		
7~4	-	
3	FLKF	Programmable zone unlock flag, 1 means unlocked
2	PLKF	Program area unlock flag, 1 means unlocked
1	DLKF	Data area unlock flag, 1 means unlocked
0	-	-

Table 7-3-3-5 Register PADRD

FC04H	7	6	5	4	3	2	1	0
PADRD	PADRD[7:0]							
R/W	R/W							
Initial Value	0	1	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7~0	PADRD	Program area and DATA area division configuration register The unit for division is 128 bytes and when PADRD>0: The address space for PROGRAM area: 0 ~ (PADRD×128 - 1), The address space for DATA area : (PADRD×128) ~ 47FFH. *Note：* *1. PADRD=0 indicates the whole Flash is DATA area* *2. The maximum value of PADRD is 90H respectively, and the setting value of PADRD cannot exceed the maximum value.*

Table 7-3-3-6 Register PTS

FC05H	7	6	5	4	3	2	1	0

PTSL	PTS[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
FC06H	7	6	5	4	3	2	1	0
PTSH	-	-	-	PTS[12:8]				
R/W	-	-	-	R/W				
Initial Value	-	-	-	0	0	0	0	0

Bit Number	Bit Symbol	Description
15~13	-	-
12~0	PTS	target address pointer register

7.3.4 Flash Control Example

Divide Flash into DATA area and PROGRAM area

For instance, if the user wants to divide a 18K Flash (128 byte DATA area and the remains for PROGRAM area), the program may like this:

```
---------------------------------------------------------------------------------------- ----------------------------------
PADRD=0x8F;//Program area space address is: 0~0x477F,data area space address is: 0x4780~0x47FF
---------------------------------------------------------------------------------------- ----------------------------------
```

Note: The physical address of the above set data area in FLASH is 0x4780~0x47FF, but the logical address is 0x0000~0x007F, the logical address should be filled in when reading and writing data area.

Sector erasure of DATA area

Sector n of DATA area needs to be erased, for example, the program may as follows:

```
-----------------------------------------------------------------------------------------
FSCMD = 0;                                  //Set CMD = 0
LOCK = 0x2A;                                //unlock DATA area
PTSH = (unsigned char)((n*0x80)>>8); //set the higher bytes of the sector ’ s address
PTSL = (unsigned char)(n*0x80);             //set the lower bytes of the sector ’ s address
FSCMD = 3;                                  //set clear
LOCK = 0xAA;                                //lock FLASH
-----------------------------------------------------------------------------------------
```

Note: Sector serial number n=0, 1, 2 .....

Write data into DATA area

For instance, to write data 0xAA to the data space address n~(n+100), the program is as follows:

```
-----------------------------------------------------------------------------------------
unsigned char i;
FSCMD = 0;    //set CMD = 0
LOCK = 0x2A; //unlock DATA area
PTSH = (unsigned char)(n>>8); //set the higher 8 bits of data’s original address
PTSL = (unsigned char)n; //set the lower 8 bits of data’s original address FSCMD = 2; //set WRITE command
for(i=0;i<100;i++)
```

```
{
FSDAT = 0xAA;      //write data continuously
}
LOCK = 0xAA;       //lock FLASH
```

Note：

1． When data is written continuously, only original address has be set. PTS will increase automatically after writing FSDAT each time.

2． For DATA area R/W, only the logical address of the DATA area which starts from 0 needs to be set, instead of the physical address.

Read data from DATA area

For instance, to read data from data space address n~(n+100) to pointer pBuf, the program is as follows：

```
unsigned char i, pBuf;
FSCMD = 0;        // set CMD = 0
LOCK = 0x2A;      //unlock DATA area
PTSH = (unsigned char)(n>>8);      //set the higher 8 bits of data's original address
  PTSL = (unsigned char)n;             //set the lower 8 bits of data's original address FSCMD = 1; //set READ command
for(i=0;i<100;i++)
{
*pBuf++ = FSDAT ;//read data continuously
}
FSCMD=0;
LOCK = 0xAA;       //lock FLASH
```

Note：When data is read continuously, only original address has be set. PTS will increase automatically after writing FSDAT each time.

7.4 External RAM Mapped to Program Area

The 1024 bytes of external RAM can be mapped for use as program space with mapped addresses of 4800H~4BFFH, and the mapping diagram is shown in Figure 7-4-1. Users can download the program to the external RAM space, and when the program is running directly execute the jump instruction to jump to the mapped program area for execution. For the same effect, the value of BOOT (see register MECON for details) can also be set to 1, and then a soft reset is executed, and the program starts to execute from the external RAM space after the reset (the mapped address is 0000H~03FFH at this time). The mapped program area is particularly convenient for implementing functions such as IAP.

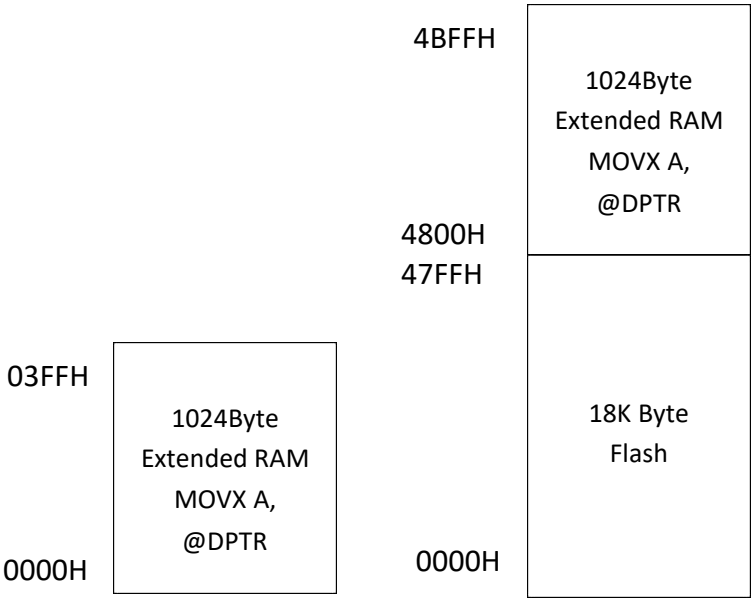


Figure 7-4-1 XRAM Address Mapping

8 Interruption System

8.1 Function Introduction

JZ8FC003 Series include a enhanced interrupt control system with 15 interrupt entries. For each interrupt entry, there are several interrupt sources with 2 level interrupt priorities for each source. Each interrupt source has its independent interrupt vector, priority setting, interrupt enable control and interrupt flag. CPU enters corresponding Interrupt Service Routine after responding to the interrupt. It will then returns to the former status after receiving RETI. If there are multiple valid sources requesting interrupts, CPU will respond sequentially according to the interrupt priority set before. If the sources share the same priority, CPU will respond according to their natural priority (from the smallest address to largest address of the interrupt entries).

8.2 Interrupt logic

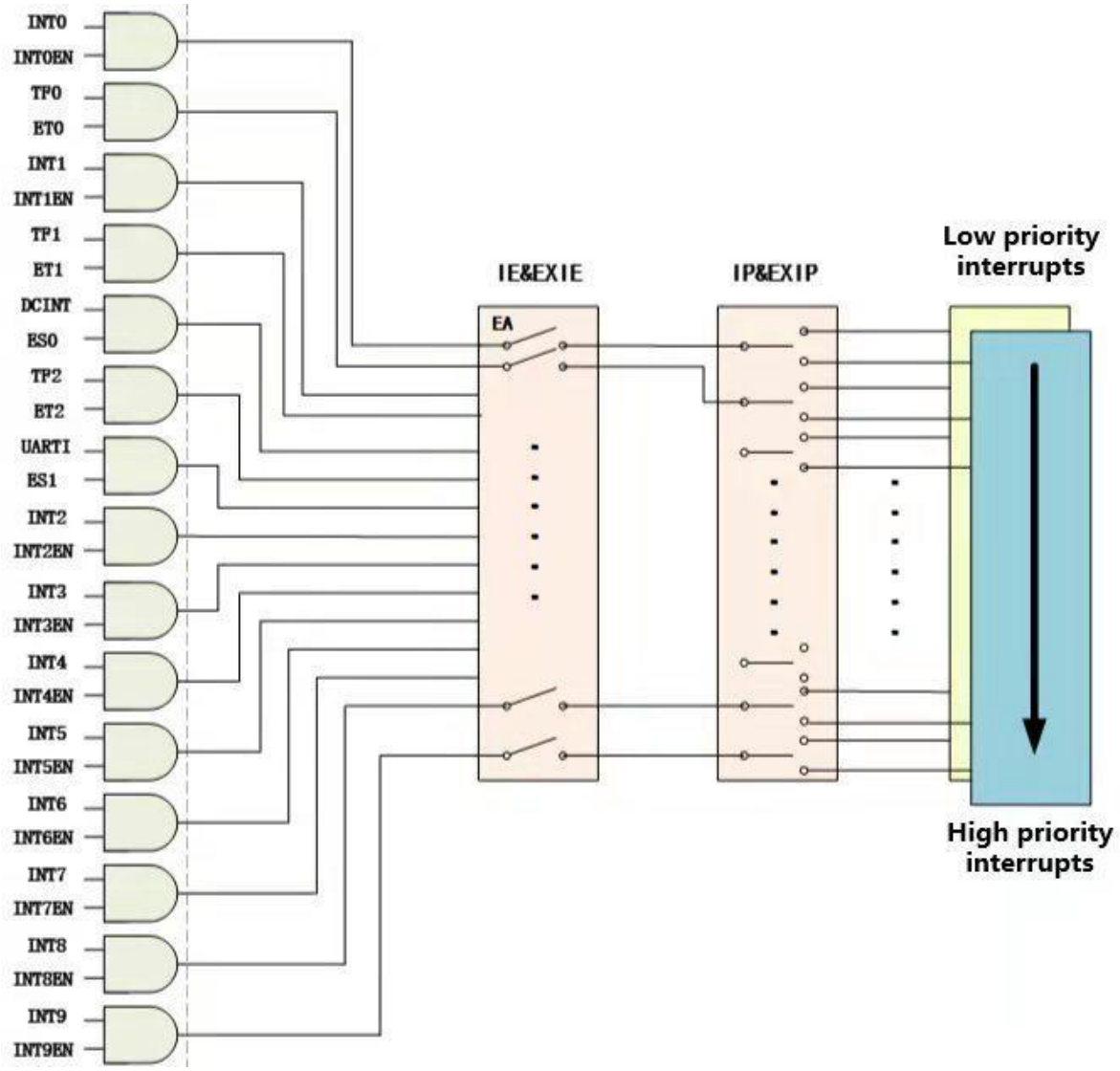


Table 8-2-1 Interrupt Logic

8.3 Interrupt vector table

Interruptions	Interrupt source	vector	Default Priority
INT0	INT0	03H	0
TF0	Timer 0	0BH	1
INT1	INT1	13H	2
TF1	Timer1	1BH	3
FSK	Wireless charger decoding	23H	4
TF2	Timer 2	2BH	5
UART1	UART1	33H	6
INT2	ADC/External Interrupt 2	3BH	7
INT3	UART2/External Interrupt 3	43H	8
INT4	LVD/External Interrupt 4	4BH	9

INT5	SPI/External Interrupts5	53H	10
INT6	I2C/SWI/External Interrupt 6	5BH	11
INT7	WDT/External Interrupt 7	63H	12
INT8	TMC/External Interrupt 8	6BH	13
INT9	PWM/External Interrupt 9	73H	14

8.4 Interrupt Control Register

Table 8-4-1 Register IE

A8H	7	6	5	4	3	2	1	0
IE	EA	ES1	ET2	ES0	ET1	EX1	ET0	EX0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7	EA	Global Interrupt enable control 0：disable Global Interrupt 1： enable Global Interrupt
6	ES1	UART1 Interrupt enable control 0：disable UART1 Interrupt 1： enable UART1 Interrupt
5	ET2	Timer 2 Interrupt enable control 0：disable Timer 2 Interrupt 1： enable Timer 2 Interrupt
4	ES0	Wireless decode interrupt enable control bit 0：Wireless decoding interrupt off 1：Wireless decoding interrupt on
3	ET1	Timer 1 Interrupt enable control 0：disable Timer 1 Interrupt 1： enable Timer 1 Interrupt
2	EX1	External interrupt 1 enable control bit 0： Timer 1 interrupt off 1：External interrupt 1 interrupt on
1	ET0	Timer 0 interrupt enable control
0	EX0	External interrupt 0 enable control 0：disable External Interrupt 0 1： enable External Interrupt 0

Table 8-4-2 Register EXIE

E8H	7	6	5	4	3	2	1	0
EXIE	INT9EN	INT8EN	INT7EN	INT6EN	INT5EN	INT4EN	INT3EN	INT2EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7	INT9EN	Interrupt 9 enable control bit (Interrupt 9 for PWM/External Interrupt 9) 0：Disable 1：Enable
6	INT8EN	Interrupt 8 enable control bit (Interrupt 8 for TMC/External Interrupt 8) 0：Disable 1：Enable
5	INT7EN	Interrupt 7 enable control bit (Interrupt 7 for WDT/External Interrupt 7) 0：Disable 1：Enable
4	INT6EN	Interrupt 6 enable control bit (Interrupt 6 for I2C/SWI/External Interrupt 6) 0：Disable 1：Enable
3	INT5EN	Interrupt 5 enable control bit (Interrupt 5 for SPI/External Interrupt 5) 0：Disable 1：Enable
2	INT4EN	Interrupt 4 enable control bit (Interrupt 4 for LVD/External Interrupt 4) 0：Disable 1：Enable
1	INT3EN	Interrupt 3 enable control bit (Interrupt 3 for UART2/External Interrupt 3) 0：Disable 1：Enable
0	INT2EN	Interrupt 2 enable control bit (Interrupt 2 for ADC/External Interrupt 2) 0：Disable 1：Enable

Note：The enable controls of EXIE corresponds to Interrupt Vector which means the enable control for each interrupt source has to be set as well. For example, if External Interrupt 2 needs to be enabled, both INT2EN and EPIE2(External Interrupt 2 enable control) need to be set to 1.

Table 8-4-3 Register IP

B8H	7	6	5	4	3	2	1	0
IP	-	PS1	PT2	PS0	PT1	PX1	PT0	PX0

R/W	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7	-	-
6	PS1	UART 1 priority control 0：low priority 1：high priority
5	PT2	Timer 2 priority control 0：low priority 1：high priority
4	PS0	Wireless charger decoding priority control bit 0： low priority 1：high priority
3	PT1	Timer 1 priority control bit 0： low priority 1：high priority
2	PX1	External interrupt 1 priority control bit 0： low priority 1：high priority
1	PT0	Timer 0 priority control bit 0： low priority 1：high priority
0	PX0	External interrupt 0 priority control bit 0： low priority 1：high priority

Table 8-4-4 Register EXIP

F8H	7	6	5	4	3	2	1	0
EXIP	PX9	PX8	PX7	PX6	PX5	PX4	PX3	PX2
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7	PX9	Interrupt INT9 priority control bit 0： low priority 1：high priority
6	PX8	Interrupt INT8 priority control bit 0： low priority 1：high priority
5	PX7	Interrupt INT7 priority control bit 0： low priority 1：high priority
4	PX6	Interrupt INT6 priority control bit 0： low priority 1：high priority
3	PX5	Interrupt INT5 priority control bit 0： low priority 1：high priority
2	PX4	Interrupt INT4 priority control bit 0： low priority 1：high priority
1	PX3	Interrupt INT3 priority control bit 0： low priority 1：high priority
0	PX2	Interrupt INT2 priority control bit 0： low priority 1：high priority

8.5 External Interrupt

8.5.1 External Interrupt Introduction

For INT0 and INT1, arbitrary input ports can be selected as interrupt sources. The system also includes 8 extended Interrupt Entries INT2~INT9 as External Interrupt. For each interrupt entry, any input ports can be selected as interrupt source as well. Either rising or falling edge trigger can be selected independently for each Extended External Interrupt. The External Interrupt can also be waken up in STOP mode. The status register

for INT2~INT9 External Interrupt is EPIF and the corresponding configuration register for INT2~INT9 which are EPCON0~EPCON7 also can be visited by setting the index of register (INDEX=0~7) .

Note: INT0 and INT1 can be triggered by rising or falling edge. The selection bits are IT0 and IT1 respectively. For details, please refer to the related description of register TCON.

8.5.2 External Interrupt Register

Table 8-5-2-1 Register IT0CON

8FH	7	6	5	4	3	2	1	0
IT0CON	-	-	-	IT0PS[4:0]				
R/W	-	-	-	R/W				
Initial Value	-	-	-	0	0	0	0	0

Bit number	Bit symbol	Description
7~5	-	-
4~0	IT0PS[4:0]	INT0 Interrupt pin selection The table for pin numbers and corresponding pins please refer to Table 8-5-2-6

Table 8-5-2-2 Register IT1CON

8EH	7	6	5	4	3	2	1	0
IT1CON	-	-	-	IT1PS[4:0]				
R/W	-	-	-	R/W				
Initial Value	-	-	-	0	0	0	0	0

Bit number	Bit symbol	Description
7~5	-	-
4~0	IT1PS[4:0]	INT1 interrupt pin selection bit Reference table 8-5-2-6 for number and pin correspondence

Table 8-5-2-3 Register EPIE

F9H	7	6	5	4	3	2	1	0
EPIE	EPIE9	EPIE8	EPIE7	EPIE6	EPIE5	EPIE4	EPIE3	EPIE2
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit symbol	Description
7	EPIE9	External Interrupt 9 enable control
6	EPIE8	External Interrupt 8 enable control
5	EPIE7	External Interrupt 7enable control
4	EPIE6	External Interrupt 6 enable control
3	EPIE5	External Interrupt 5 enable control
2	EPIE4	External Interrupt 4 enable control
1	EPIE3	External Interrupt 3enable control
0	EPIE2	External Interrupt 2enable control

Table 8-5-2-4 Register EPIF

FAH	7	6	5	4	3	2	1	0
EPIF	EPIF9	EPIF8	EPIF7	EPIF6	EPIF5	EPIF4	EPIF3	EPIF2
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit symbol	Description
7	EPIF9	External Interrupt 9 Interrupt Flag, cleared when 1 is written to it
6	EPIF8	External Interrupt 8 Interrupt Flag, cleared when 1 is written to it
5	EPIF7	External Interrupt 7 Interrupt Flag, cleared when 1 is written to it
4	EPIF6	External Interrupt 6 Interrupt Flag, cleared when 1 is written to it
3	EPIF5	External Interrupt 5 Interrupt Flag, cleared when 1 is written to it
2	EPIF4	External Interrupt 4 Interrupt Flag, cleared when 1 is written to it
1	EPIF3	External Interrupt 3 Interrupt Flag, cleared when 1 is written to it
0	EPIF2	External Interrupt 2 Interrupt Flag, cleared when 1 is written to it

Table 8-5-2-5 Register EPCON

FBH	7	6	5	4	3	2	1	0
EPCON	EPPL	-	-	EPPS[4:0]				
R/W	R/W	-	-	R/W				
Initial Value	0	-	-	0	0	0	0	0

Note：EPCON is a register with index, INDEX=0~7 indicates EPCON0~EPCON7 respectively

Bit number	Bit symbol	Description
7	EPPL	External Interrupt Trigger Edge Selection 0：Rising edge

		1：Falling edge
6~5	-	-
4~0	EPPS[4:0]	Interrupt Pin selection The table for pin numbers and corresponding pins please refer to Table 8-5-2-6

Table 8-5-2-6 Index for Interrupt Pin

Pin name	Number
P00	0
P01	1
P02	2
P03	3
P04	4
P05	5
P06	6
P07	7
P10	8
P11	9
P12	10
P13	11
P14	12
P15	13
P16	14
P17	15
P20	16
P30	17

8.5.3 External Interrupt Control Routines

External interrupt 0/1 control routine

For instance, to enable external interrupt 0, the procedure is as follows:

```
---------------------------------------------------------------------------------
void INT0_init(void)
{
      P10F = 1;              //set P10 as input pin
      IT0CON = 8;            //Set P10 as INT0 interrupt pin
      EX0 = 1;               //INT0 interrupt enable
      IE0 = 1;               //External interrupt 0 enable
      IT0 = 1;               //Set to falling edge interrupt
      PX0 = 1;               //Set INT0 to high priority
      EA = 1;                //Total interrupt enable}
void INT0_ISR (void) interrupt 0
{
         //External Interrupt0 Interrupt Service Routine
}
---------------------------------------------------------------------------------
```

For instance, to enable external interrupt 1, the program is as follows:

```
---------------------------------------------------------------------------------
void INT1_init(void)
{
      P10F = 1;              //P10 set to input function
      IT1CON = 8;            //Set P10 as INT1 interrupt pin
      EX1 = 1;               //INT1 interrupt enable
      IE1 = 1;               //External interrupt 1 enable
      IT1 = 1;               //Set to falling edge interrupt
      PX1 = 1;               //Set INT1 as high priority
      EA = 1;                //Total interrupt enable
      }
void INT1_ISR (void) interrupt 2
{
         //External Interrupt 1Interrupt Service Routine
}

---------------------------------------------------------------------------------
```

External interrupt 2~9 control example

Taking External Interrupt 2 for example, if P10 is set as the input pin for External Interrupt 2 and External Interrupt 2 is enable, the program may like this:

```
---------------------------------------------------------------------------------------
void INT2_init(void)
{
        P10F = 1;                       //set P10 as input pin
    INDEX = 0;     //INDEX is the register with index, set INDEX=0 corresponding to
    INT2EPCON = (1<<7) | 8;       //Set P10 as INT2 external interrupt pin, falling edge triggered
    INT2EN = 1;                 //External interrupt 2 interrupt enable
    EPIE |= 0x01;               //INT2 interrupt enable
    EA = 1;                     //Total interrupt enable
}
void INT2_ISR (void) interrupt 7
{
    if(EPIF & 0x01)             //Determine the external interrupt 2 interrupt flag
    {
        EPIF = 0x01;           //write 1 to the Interrupt Flag to clear it
                //External Interrupt 2Interrupt Service Routine

          ......
    }
}
---------------------------------------------------------------------------------------
```

9 Clock System

9.1 Clock System Introduction

The JZ8FC003 family of chips supports the following clock sources in total：

- 32MHz Internal RC oscillator
- 131KHz Internal RC oscillator
- Supports 1 - 24MHz external Crystal Oscillators
- Supports 1 - 24MHz external clock input

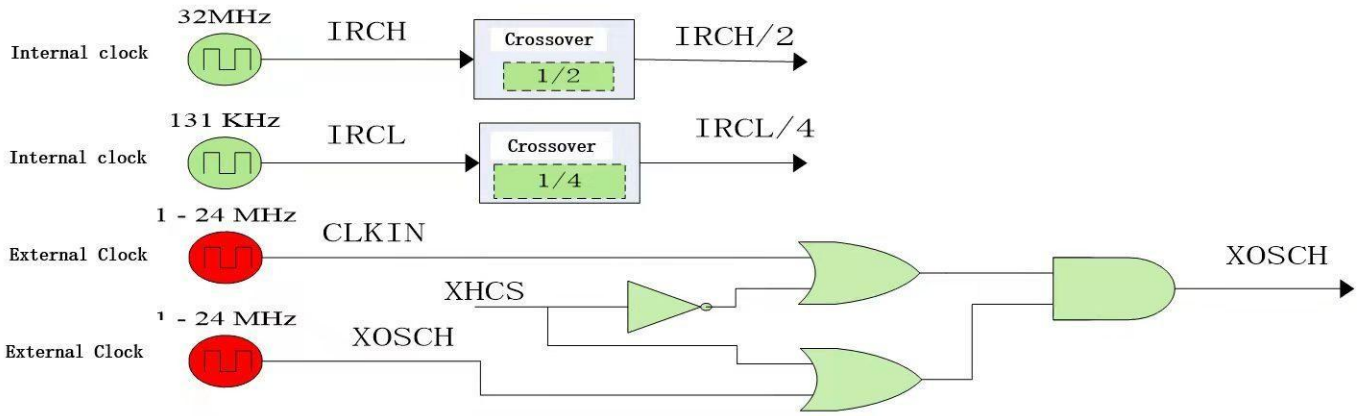


Figure 9-1-1 Clock source

Users can control the clock sources independently. They can disable or enable any of the clock sources in order to manage the power consumption flexibly.
All the clock sources can be set as system alarm clock and assigned to various peripherals as their clock sources. For more information you may refer to the Peripherals part.

9.1.1 Clock Specific Name Definition

Symbol	Description
IRCH	32MHz Internal RC oscillator
IRCL	131KHz Internal RC oscillator
XOSCH	1~24MHz External crystal oscillator
CLKIN	1~24MHz External clock input

9.1.2 32 MHz Internal RC Oscillator(IRCH)

IRCH is the default system clock after the chip is powered on, and can be turned on or off by the IHCKE bit of register CKCON. When IRCH is the system clock, the CPU clock frequency is 1/2 of the IRCH frequency, i.e., the default CPU clock is 16MHz. The chip is shipped from IRCH frequency correction for 32MHz@3.3V/25° C with a clock accuracy of ±1%.

9.1.3 131 kHz Internal RC Oscillator (IRCL)

IRCL can be turned on or off by the ILCKE bit in register CKCON. IRCL is set to the system clock to achieve low system power consumption. The chip is shipped from IRCL frequency is131KHz@3.3V/25° C with a clock accuracy of ±1%.

9.1.4 External high-speed crystal resonator (XOSCH) and external clock input (CLKIN)

XOSCH and CLKIN share the same connection pin, so only one of them can be selected by the XHCS bit in register CKCON. XOSCH can be turned on or off by the XHCKE bit in register CKCON, and the flag bit XHSTA indicates whether the XOSCH clock is stable.

Note: Hardware design of the crystal load capacitor ground as close as possible to the chip GND pin.

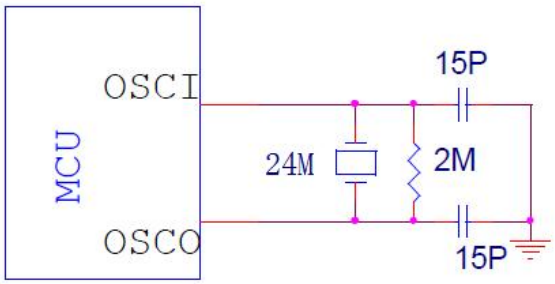


Figure 9-1-4-1 Typical circuit diagram of external high-speed crystal

9.2 Clock Control Register Description

Table 9-2-1 Register CKCON

8080H	7	6	5	4	3	2	1	0
CKCON	ILCKE	IHCKE	-	XHCS	-	-	XHCKE	XHSTA
R/W	R/W	R/W	-	R/W	-	-	R/W	R/W
Initial Value	0	0	-	0	-	-	0	0

Bit number	Bit Symbol	Description
7	ILCKE	IRCL enable control 1：enable 0： disable *Note：* *When it is 1, the clock is enabled;* *when it is 0, if the system or other modules selected this clock source, the clock is still enabled.*
6	IHCKE	IRCH enable control 1：enable 0： disable *Note：* *When it is 1, the clock is enabled;* *when it is 0, if the system or other modules selected this clock source, the clock is still enabled.*
5	-	-
4	XHCS	XOSCH clock source selection bit 0：XOSCH clock source is an external high-speed crystal 1： XOSCH clock source is external clock input *Note：* *Before rewriting the value of XHCS, you must ensure that the system clock is not selected as an external clock, otherwise the rewrite action will be invalid.*
3~2	-	-
1	XHCKE	XOSCH enable control bit 1： enable 0：disable *Note：* *1 When the bit is valid, the clock module is on, but when the bit is 0, the clock will still be on if the system or another module selects that clock source.* *2 Since XOSCH is an external clock, if you want to start XOSCH you also need to configure the corresponding pin function as XOSCH function.*
0	XHSTA	XOSCH clock stabilization signal flag, 1 valid

Table 9-2-2 Register IHCFG

8083H	7	6	5	4	3	2	1	0
IHCFGL	IHCFG[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
8084H	7	6	5	4	3	2	1	0
IHCFGH	-	-	-	-	-	-	-	IHCFG[8]
R/W	-	-	-	-	-	-	-	R/W
Initial Value	-	-	-	-	-	-	-	0

Bit number	Bit Symbol	Description
15~9	-	-
8~0	IHCFG	Internal 32MHz clock configuration register

Table 9-2-3 Register ILCFG

8085H	7	6	5	4	3	2	1	0
ILCFG	-	-	ILCFG[5:0]					
R/W	-	-	R/W					
Initial Value	-	-	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~6	-	-
5~0	ILCFG	Internal 131KHz clock configuration register

Table 9-2-4 Register XHISELL

8086H	7	6	5	4	3	2	1	0
ILCFG[L]	-	-				XHISEL		
R/W	-	-				R/W		
Initial Value	-	-				0	0	0

Bit number	Bit Symbol	Description
7~3	-	-
2~0	XHISEL	Change the high-speed crystal current: I =120*(1+XHISEL)uA

9.3 System Clock

System clock control is done by registers CKCON, CKSEL, and CKDIV. With these register sets, operations such as switching of each clock source, switching and dividing of the system clock can be set individually.

9.3.1 System clock Architecture

The system clock structure diagram is shown in Figure 9-3-1.

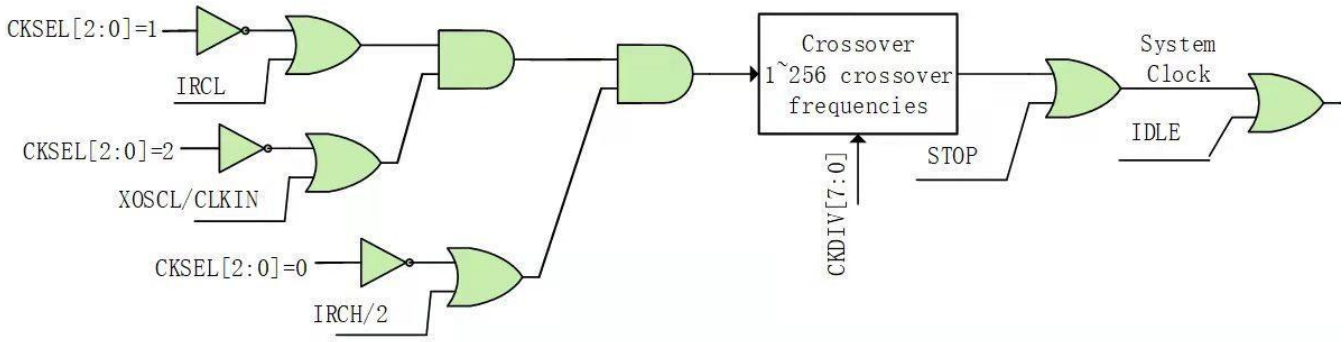


Figure 9-3-1 System clock Architecture

9.3.2 System Clock Control Register Description

Table 9-3-2-1 Register CKSEL

8081H	7	6	5	4	3	2	1	0
CKSEL	-	-	-	-	-	CKSEL[2:0]		
R/W	-	-	-	-	-	R/W		
Initial Value	-	-	-	-	-	0	0	0

Bit number	Bit Symbol	Description
7~3	-	-
2~0	CKSEL	System clock selection bit 000： IRCH's two-way frequency 001： IRCL 010：XOSCH/CLKIN Others: IRCH's two-way frequency *Note: If you set IRCL as system clock, you must wait for about 1ms after enabling IRCL clock and then switch to system clock, otherwise an exception may occur.*

Table 9-3-2-2 Register CKDIV

8082H	7	6	5	4	3	2	1	0
CKDIV	CKDIV[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~0	CKDIV	System clock frequency division： 00H：No division 01H： frequency divided by 2 02H： frequency divided by 3 03H： frequency divided by 4 „… FFH：frequency divided by 256

9.3.3 System Clock Control Method and Example

Set IRCH as the system clock

To set IRCH as the system clock. The program is as follows:

```
-------------------------------------------------------------------------------
#define IHCKE           (1<<6)
#define CKSEL_IRCH      0
void Sys_Clk_Set_IRCH(void)
{
    CKCON |= IHCKE;                     //Enable IRCH
    CKSEL = (CKSEL&0xF8) | CKSEL_IRCH;  //Set IRCH as system clock
}
-------------------------------------------------------------------------------
```

Set IRCL as the system clock

To set IRCL as the system clock. The program is as follows:

```
-------------------------------------------------------------------------------
#define ILCKE           (1<<7)
#define CKSEL_IRCL      1
void Sys_Clk_Set_IRCL(void)
{
    CKCON |= ILCKE;                     //Enable the IRCL clock
    Delay_ms(1);                        //Delay 1ms after enabling IRCL and wait for IRCL to stabilize
    CKSEL = (CKSEL&0xF8) | CKSEL_IRCL;  //Set the system clock to IRCL}
-------------------------------------------------------------------------------
```

Set XOSCL as the system clock

Set the system clock to XOSCH with the following procedure.

```
-------------------------------------------------------------------------------
#define XHCS            (1<<4)
#define XHCKE           (1<<1)
#define XHSTA           (1<<0)
#define CKSEL_XOSCH     2
void Sys_Clk_Set_XOSCH(void)
{
    P17F = 4;                           //Set P1.7 as external high-speed crystal output pin
    P30F = 4;                           //Set P3.0 as external high-speed crystal input pin
    CKCON &= ~XHCS;
    CKCON |= XHCKE;                     //Enable the XOSCH clock
    while(!(CKCON & XHSTA));            //Waiting for the clock to stabilize
    CKSEL = (CKSEL&0xF8) | CKSEL_XOSCH;//Set the system clock to XOSCH
}
-------------------------------------------------------------------------------
```

Set the system clock to CLKIN

Set the system clock to CLKIN and program as follows.

```
---------------------------------------------------------------------------------------
#define XHCS                    (1<<4)
#define CKSEL_XCLK              2
void Sys_Clk_Set_XCLK_IN(void)
{
    P30F = 4;                                   //Set P3.0 as external high-speed clock input pin
    CKCON |= XHCS;
    CKSEL = (CKSEL&0xF8) | CKSEL_XCLK;  //Set the system clock to CLKIN}
---------------------------------------------------------------------------------------
```

10 Power supply and reset system

10.1 Power Supply

There is 1.8V - 5.5V source between VDD pin and VSS pin for JZ8FC003 series which supplies the power for the chip. VDD and LDO supply power for the analog system and LDO supplies power for the digital system

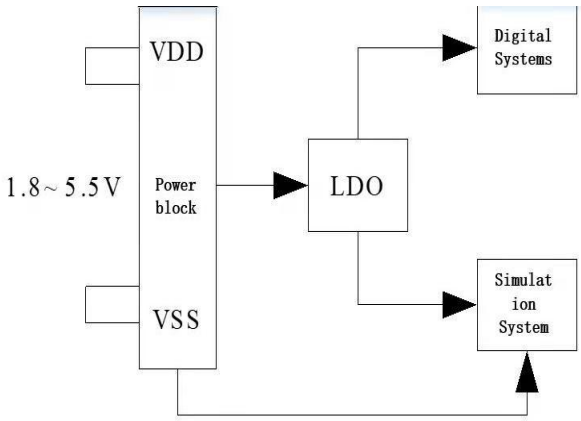


Figure 10-1-1 Power Supply Architecture

Figure 10-1-2 shows a typical circuit diagram of the chip power supply.

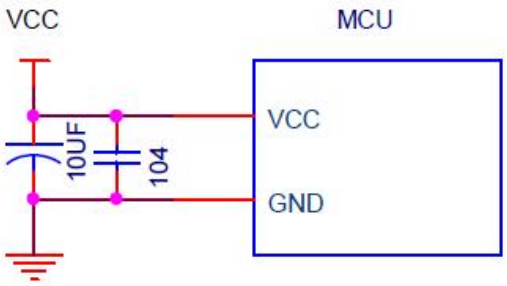


Figure 10-1-2 Typical Circuit for Power Supply

Important reminder:

1. In the above circuit, the filter capacitor 10uF and 104 are standard for the chip power supply circuit and cannot be omitted.This capacitor must be placed close to the chip power supply pins, otherwise it may cause the chip to work abnormally.

2. The above circuit and component parameters are for reference only, and may need to be modified according to the peripheral working environment and different voltage supply parameters.

10.1.1 LDO Function Introduction

There is an internal low dropout regulator (LDO) for JZ8FC003 Series chip. LDO module offers supply voltage for the chip. The output voltage of LDO is set by VLEVEL (PWCON[2:0]) and the default value for VLEVEL is 5, which implies the default voltage is 1.61V. When VDD/VSS is less than the output voltage set by VLEVEL, the output voltage will be VDD directly; when VDD/VSS is greater than the output voltage set by VLEVEL, LDO output the voltage set by VLEVE. High LDO voltage is benefits to clock module's rapid start while low LDO voltage will lower the chip's power consumption. There are two working modes for LDO: High Power mode and Low Power mode, which is selected by VHL (PWCON[3]). The load capacity is also different in different modes. The current is higher in High Power mode but with higher power consumption, while in Low Power mode it is vice versa. For the most time when the system is operating normally, LDO is usually High Power mode. The Low Power mode is usually used for Power Save Modes such as STOP, IDLE, Low Speed Mode etc.

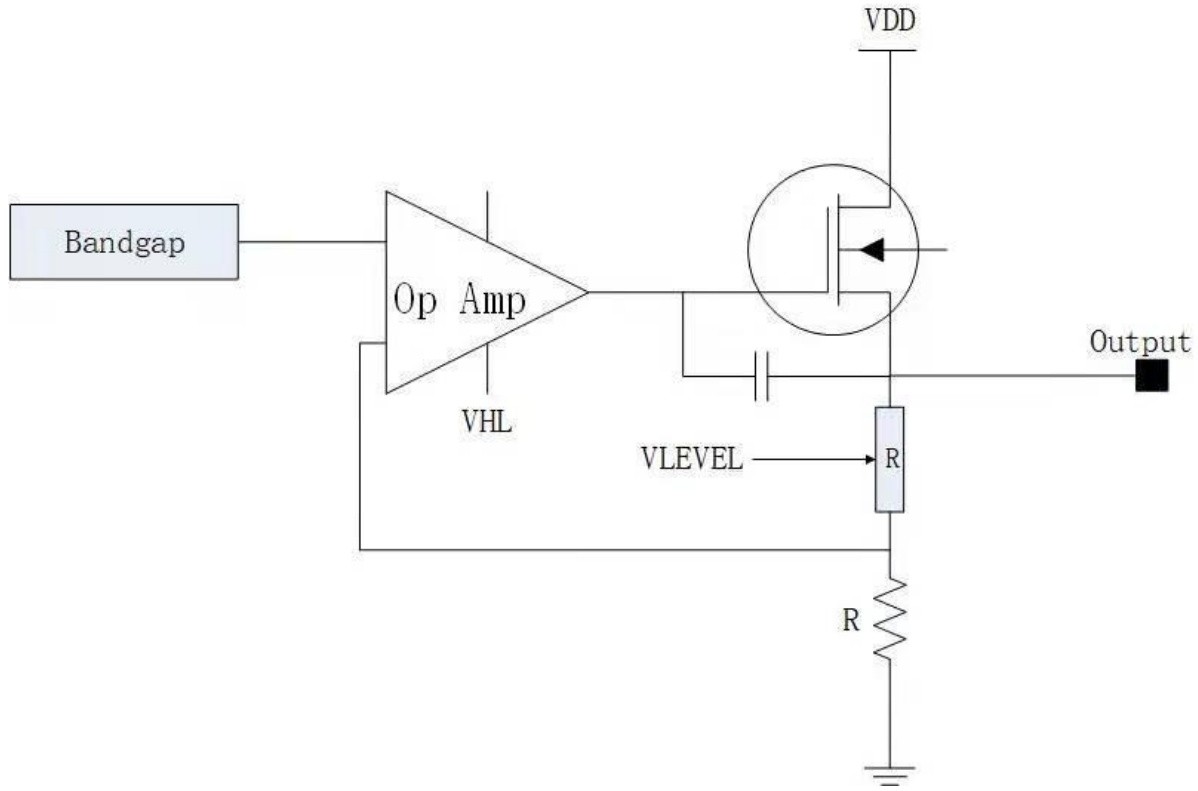


Figure 10-1-3 LDO Module Schematic

10.1.2 LDO Control Register

Table 10-1-2-1 Register PWCON

86H	7	6	5	4	3	2	1	0
PWCON	FLEVEL[3:0]				VHL	VLEVEL[2:0]		
R/W	R/W				R/W	R/W		
Initial Value	0	1	1	1	1	1	0	1

Bit Number	Bit Symbol	Description
7~4	FLEVEL	Internal reference voltage (Bandgap) output adjustment bit field0000：0.825V 0001：0.850V 0010：0.875V 0011：0.900V 0100：0.925V 0101：0.950V 0110：0.975V 0111：1.000V 1000：1.025V 1001：1.050V 1010：1.075V 1011：1.100V 1100：1.125V 1101：1.150V 1110：1.175V 1111：1.200V *Note: The internal reference voltage is automatically loaded by the system at power-up, and the user is not allowed to modify it.*
3	VHL	LDO operating mode control bit 1: High power mode 0: Low power mode
2~0	VLEVEL	LDO output voltage setting bit field 000：1.31V 001：1.37V 010：1.43V 011：1.49V 100：1.55V 101：1.61V 110：1.67V 111：1.73V *Note:* *1. The internal clock circuit is powered by LDO, changing the LDO output voltage will cause the internal clock frequency to change, in general, the LDO voltage can be kept at the default value, and it is not recommended to modify it.* *2. It is not allowed to set the LDO output voltage less than 1.5V, otherwise it may cause abnormalities.*

10.2 Reset System

There are multiple internal and external reset sources for JZ8FC2 Series chip as figure 10-2-1 shows.

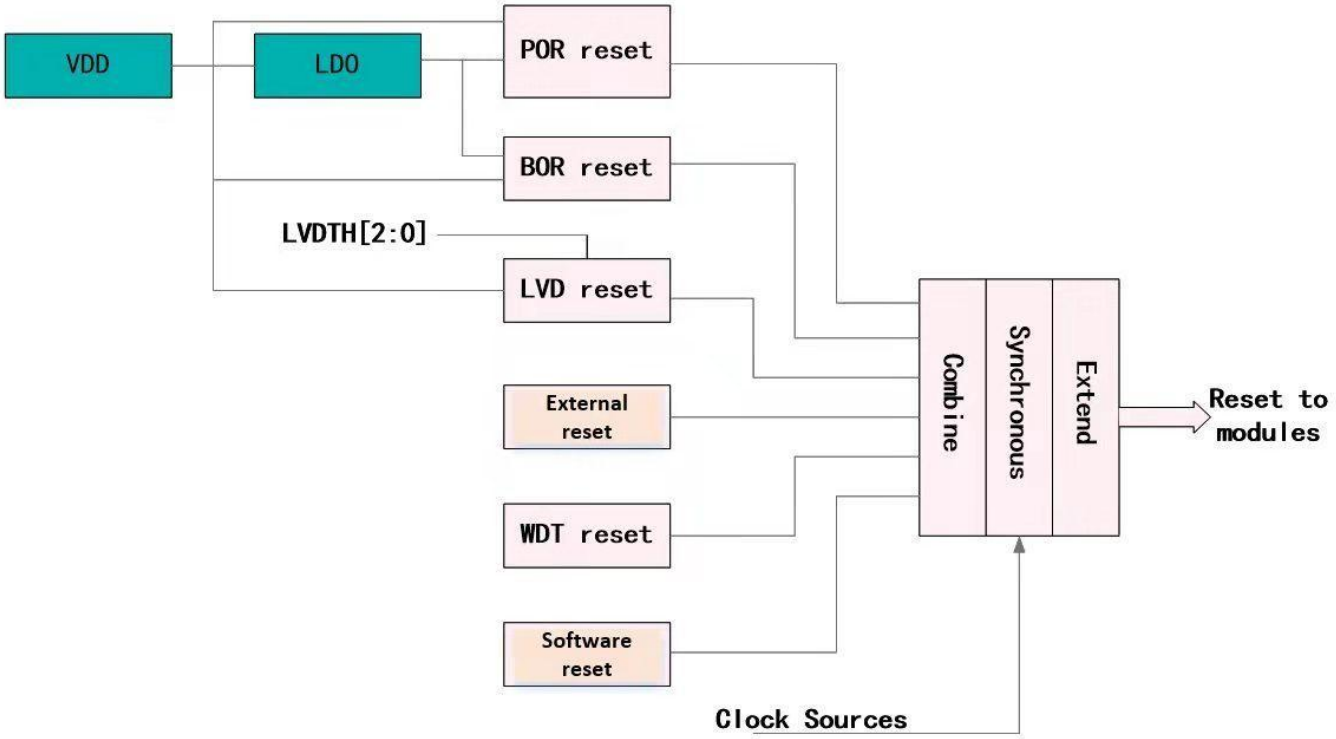
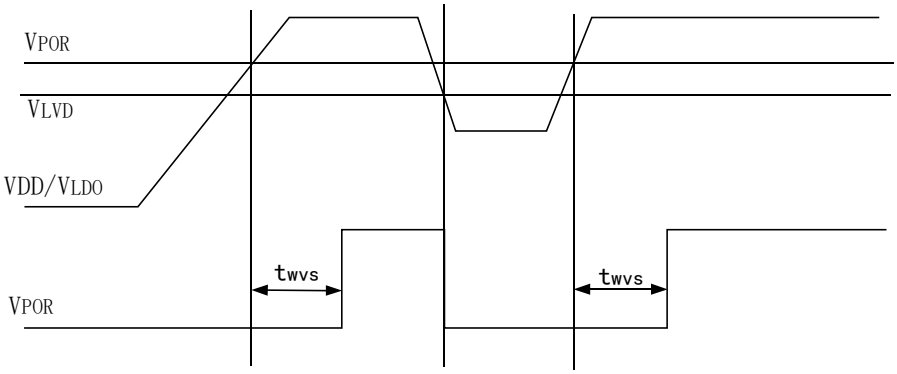


Figure 10-2-1 Reset System Architecture

- **Power On Reset (POR)**

It usually takes some time for the system to reach normal working voltage. The POR is mainly based on VDD and LDO. The POR signal is valid when the voltage is below the detection threshold.

The POR circuit ensures that the chip remains reset during the Power On period hence the chip can starts from certain stable status. The POR signal will also be expanded by the internal counter to makes sure that all the analog modules can enter stable working status after Power On stage.



twvs：Waiting for voltage stabilization time

Figure 10-2-2 POR Circuit Example and Power On Stage

- **Brown Out Reset(BOR)**

BOR offers alarm signal for the chip when the voltage drops (eg. Inference or load changes). Once the VDD or internal LDO output voltage is below a certain threshold, it will reset the chip to avoid program error or system abnormality.

- **Low Voltage Reset**

The Low Voltage Detection (LVD) can detect VDD in multiple working modes. When VDD voltage is below the threshold set by LVD for 20us it will generates reset signal (on the premise of that LVD is Reset mode).

- **External Reset**

By pulling down the reset pin(RESET), external device can reset the chip as well. RESET can reset the whole in normal working modes, while in STOP mode, the hard reset will awaken the chip first and then reset it. Usually, RESET is pulled up internally and will not influence the internal reset circuit.

- **Watchdog Reset**

The WDT (watchdog timer) is responsible for monitor the how processor do with instructions. With proper configuration, if the WDT is not refreshed in certain time, a reset signal will be generated. WDT is disabled after POR, but users can enable and configure it if necessary.

- **Soft Reset**

The program can soft reset the chip. When 1 is written to SWRST of register PCON, CPU sends out reset signal.

POR and external hard reset will reset all the circuits while LVD and WDT can reset other circuits but not reset themselves. (eg：After WDT reset, WDT registers remains former status while others are all reset) LVD/WDT and soft reset can not reset storage control circuit. Program starts from Mask ROM after POR and external hard reset. Program starts from where BOOT configuration points to after soft reset. PC will point to address 0 after any reset.

11 Power Consumption Management

There are 3 low power consumption modes for JZ8FC003 Series : IDLE, STOP and Low Speed mode. The system power consumption for IDLE, STOP and Low Speed mode is less than 15uA, 7uA and 25uA respectively.

11.1 IDLE mode

CPU stops working in this mode. All the clocks can be disabled to save power before entering IDLE mode except the main clock. Peripherals can also be enabled/disabled before entering IDLE mode according to user's needs. Those enabled peripherals will operates normally in IDLE mode.

Register IDLST(IDLSTH and IDLSTL) needs to be checked before entering IDLE mode. If all the bits are 0, CPU will enter IDLE mode normally when the mode is set as IDLE. However, if NOT all the bits are 0, CPU will not enter IDLE mode and remains in normal working mode although the mode is set as IDLE. To deal with this situation, users must complete the IDLST corresponding interrupt processing first and then set the mode as IDLE again.

Any reset or interrupt will awake the chip. The clock will resume first and then the chip responds to the interrupt and enters the interrupt service routine after the CPU awakening. After the chip exits interrupt service routine , it will executes the instructions after the instruction which set IDLE to 1. When it exits IDLE mode, IDLE will be cleared automatically.

What must be mentioned is that there should be two "nop" instructions after setting IDLE to 1 to avoid program error .

11.2 STOP mode

The STOP mode is deeper low power consumption mode than IDLE. STOP mode is able to stop all the clocks (include the main clock) and clock generation circuits. If WDT and RTC are enabled, their clock module will still works, hence users may disable them to save power.

Similar to IDLE mode, before entering STOP mode, register STPST(STPSTH and STPSTL) has to check if all the bits are 0. If there are any 1, then they should be processed first to ensure the chip will enter STOP mode successfully.

The STOP mode can be awoken by external interrupt, LVD reset or interrupt, hard reset, RTC interrupt, WDT interrupt or reset, clock monitor interrupt and touch key interrupt. If it is awoken by an interrupt, the chip will

resume clock first and respond to the interrupt, and then enters corresponding the interrupt service routine. After the chip exits the interrupt service routine, it will executes the instructions after the setting STOP to 1 instruction.The STOP will be cleared automatically when the chip exits STOP mode.To arouse the chip better, it is recommended to set the internal clock as system clock before entering STOP mode because it will take longer time waiting for stable status when using external clock.

When the chip enters STOP mode, the last clock edge will disable system clock and then the chip enters STOP mode entirely. What must be mentioned is that there should be three “nop” instructions after setting STOP to 1 avoid program error.

Notes:
1. When entering STOP/IDLE mode, setting LDO to low power mode can effectively reduce standby power consumption, but when exiting STOP/IDLE mode, LDO must be set back to high power mode, otherwise it will cause the chip to work abnormally.
2. if the system clock is selected as IRCL, when entering STOP, IRCL must not be closed, otherwise an abnormality may occur when waking up from STOP.

11.3 Low Speed Mode

Since the power consumption of the chip is directly related to the operating speed, switching the master clock to run at a low clock speed can also significantly reduce power consumption. The current when the system is set to IRCL (frequency of 131KHz) is less than 25uA.

11.4 Low Power Related Register Description

Table 11-4-1 Register PCON

87H	7	6	5	4	3	2	1	0
PCON	-	-	SWRST	-	-	TSMODE	STOP	IDLE
R/W	-	-	W	-	-	R	W	W
Initial Value	-	-	0	-	-	0	0	0

Bit Number	Bit Symbol	Description
6~7	-	-
5	SWRST	Soft reset control Setting SWRST=1 will generate soft reset signal, it will be cleared to 0 automatically after the reset
4~3	-	-
2	TSMODE	Online emulation mode flag bit, 1 means the chip is working in online emulation mode
1	STOP	STOP mode control, 1 enables STOP mode

		When STOP=1 and STPST=0, the chip will enter STOP mode. it will be cleared to 0 automatically after the chip exits STOP mode
0	IDLE	IDLE mode control, 1 enables IDLE mode When IDLE=1and IDLST=0, the chip will enter IDLE mode. it will be cleared to 0 automatically after the chip exits IDLE mode

Table 11-4-2 Registers IDLSTL, IDLSTH

FCH	7	6	5	4	3	2	1	0
IDLSTL	IDLST[7:0]							
R/W	R							
Initial Value	0	0	0	0	0	0	0	0
FDH	7	6	5	4	3	2	1	0
IDLSTH	-	IDLST[14:8]						
R/W	-	R						
Initial Value	-	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
15	-	-
14	PWMINT/EPIF[7]	Interrupt status of PWM/external interrupt 9 in IDLE mode
13	RTCINT/EPIF[6]	Interrupt status of RTC/external interrupt 8 in IDLE mode
12	WDFLG[1]/EPIF[5]	Interrupt status of WDT/external interrupt 7 in IDLE mode
11	I2CINT/EPIF[4]	Interrupt status of I2C/external interrupt 6 in IDLE mode
10	CKMINT/EPIF[3]	Interrupt status of external interrupt 5 in IDLE mode
9	LVDINT/EPIF[2]	Interrupt status of LVD/external interrupt 4 in IDLE mode
8	TKINT/EPIF[1]	Interrupt status of external interrupt 3 in IDLE mode
7	ADCINT/EPIF[0]	Interrupt status of ADC/external interrupt 2 in IDLE mode
6	U1INT	Interrupt status of UART1 interrupt in IDLE mode
5	T2INT	Interrupt status of timer 2 in IDLE mode
4	U0INT	Interrupt state of wireless charger decoding in IDLE mode
3	TCON[7]	Interrupt status of timer 1 in IDLE mode
2	PIF[1]	Interrupt status of external interrupt 1 in IDLE mode

1	TCON[5]	Interrupt status of timer 0 in IDLE mode
0	PIF[0]	Interrupt status of external interrupt 0 in IDLE mode

Table 11-4-3 Registers STPSTL,STPSTH

FEH	7	6	5	4	3	2	1	0
STPSTL	STPST[7:0]							
R/W	R							
Initial Value	0	0	0	0	0	0	0	0
FFH	7	6	5	4	3	2	1	0
STPSTH	STPST[15:8]							
R/W	R							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
15	RTCWKF	Interrupt status of RTC in STOP mode
14	WDTWKF	Interrupt status of WDT in STOP mode
13	I2CWKF	Interrupt status of I2C in STOP mode
12	CKMWKF	Interrupt status of Clock Monitor in STOP mode
11	LVDWKF	Interrupt status of LVD in STOP mode
10	TKWKF	Interrupt status of Touch Key in STOP mode
9	EPWKF[7]	Interrupt status of External Interrupt9 in STOP mode
8	EPWKF[6]	Interrupt status of External Interrupt8 in STOP mode
7	EPWKF[5]	Interrupt status of External Interrupt7 in STOP mode
6	EPWKF[4]	Interrupt status of External Interrupt6 in STOP mode
5	EPWKF[3]	Interrupt status of External Interrupt5 in STOP mode
4	EPWKF[2]	Interrupt status of External Interrupt4 in STOP mode
3	EPWKF[1]	Interrupt status of External Interrupt3 in STOP mode
2	EPWKF[0]	Interrupt status of External Interrupt2 in STOP mode
1	PWKF[1]	Interrupt status of External Interrupt1 in STOP mode
0	PWKF[0]	Interrupt status of External Interrupt0 in STOP mode

11.5 Low power Consumption Control Example

STOP Mode Example

The program is like：

```
---------------------------------------------------------------------------------------------
void Stop(void)
{
    I2CCON = 0;    //Disable the I2C function, because I2C is enabled by default, if I2C is not turned off will not
                     be able to disable the IRCH clock
    SWICON |= 0x01;    //Disable single line communication function, otherwise the master clock cannot be turned off
    CKCON = 0;              // Disable all clocks
    PWCON &= ~0x08;         //Setting the LDO into low power mode
    MECON |= (1<<6);        // Set FLASH into deep sleep state
    while(STPSTH|STPSTL); // If an interrupt is not responded to, wait for the interrupt to be responded to
    EA = 0;
    PCON |= 0x02;           // Enter STOP mode
    EA = 1;
    PWCON |= 0x08;          The LDO must be set back to high power mode after the // exit STOP
}
---------------------------------------------------------------------------------------------
```

IDLE Mode Example

The IDLE mode procedure is as follows.

```
------------------------------------------------------------------------------ ------------
#define ILCKE                (1<<7)
#define CKSEL_IRCL           1
void Idle(void)
{
    CKCON |= ILCKE;         //Open IRCL
    CKSEL = (CKSEL&0xF8) | CKSEL_IRCL;    // System clock is set to IRCL
    I2CCON = 0;             //Disable I2C module, because I2C is enabled by default, if I2C is not disabled, IRCH
                              clock will not be disabled
    SWICON |= 0x01;    //Disable single line communication function, otherwise the master clock cannot be turned
                         off
    CKCON = 0;              // Disable all clocks
    PWCON &= ~0x08;         //Setting the LDO into low power mode
    MECON |= (1<<6);
    while(IDLSTH|IDLSTL); // If an interrupt is not responded to, wait for the interrupt to be responded to
    PCON |= 0x01;           //Enter IDLE mode
    PWCON  / =0x08;         //After exiting IDLE, the LDO must be set back to high power mode
}
```

Note: Since the master clock is still open after entering IDLE, if the master clock is a high-speed clock before entering IDLE, the power consumption will still be very large after entering IDLE mode, so you need to switch the master clock to a low-speed clock before entering IDLE.

```
---------------------------------------------------------------------------------------------
```

Low Speed Mode Example

The low-speed operation mode procedure is as follows.

```
---------------------------------------------------------------------------- ----------
#define ILCKE                   (1<<7)
#define CKSEL_IRCL              1
void Low Speed Mode(void)
{
    CKCON |= ILCKE;         //Open IRCL
    CKSEL = (CKSEL&0xF8) | CKSEL_IRCL;    //System clock set to IRCL
    I2CCON = 0;                    //Disable the I2C module, because I2C is enabled by default, if I2C is not turned off
                                   will not be able to disable the IRCH clock
    SWICON |= 0x01;    //Disable the single line communication function, otherwise the master clock cannot be    turned
                                  off
    CKCON = 0;                  //Disable all clocks
    PWCON &= ~0x08;         //Setting the LDO into low power mode
}
```

Note: After exiting the low speed operation mode, the LDO must be set back to high power mode, refer to the STOP/IDLE routine.

12 General Timer (Timer0,Timer1,Timer2)

12.1 Timer0

12.1.1 Timer0 Introduction

The timer/counter function can be selected by CT0 (TMOD[2]). When CT0=0 it operates as a timer; when CT0=1, it functions as a counter. As a timer, its clock is the system clock with frequency divided by 12. As a counter, its clock is the input clock for T0. Because it takes 2 clock cycles to detect the T0 input signal edge change, so when it operates as a counter, the maximum input baud rate is 1/2 of the internal system clock frequency. There is no limit for T0 input signal's duty cycle. However, in order to identify the 0 and 1 clearly, the signal has to keep for at least one internal system clock cycle. There are for modes for Timer0 which are selected by T0M0 andT0M1 (TMOD[1:0]).

- **Mode0**

Timer0 is a 13 bit timer/counter in this mode. The higher 8 bits are stored in TH0 and the lower 5 bits are stored in TL0[4:0] with TL0[7:5] invalid. When Timer0 overflows, the interrupt flag TF0 (TCON[5]) will be set to 1. TF0 will be cleared automatically after the interrupt response. When GATE0 (TCON[3])=0, the timer/counter's is enabled/disabled by TR0 (TCON[4]). When GATE0=1, the timer/counter's is enabled/disabled by INT0. INT0 signal with high level with enable the counting and vice verse.

- **Mode1**

Timer0 is a 16 bit timer/counter in this mode. The function is the same as Mode0.

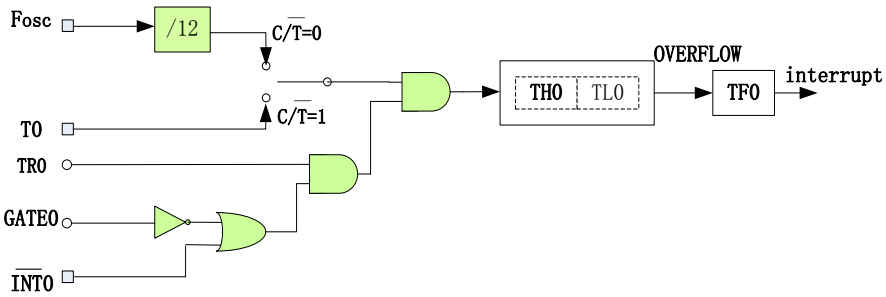


Figure 12-1-1-1 Timer0 Mode0/1

- **Mode2**

Timer0 is a 8 bit automatic reload counter/timer in this mode and only TL0 counts up automatically. When TL0 count overflows, there will be an interrupt flag TF0. The initial value for the count will be reloaded to TL0 from TH0 as well. The other settings are the same as mode0/1.

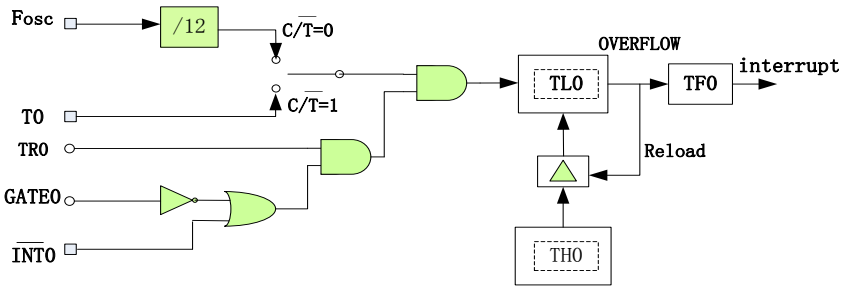


Figure 12-1-1-2 Timer0 Mode2

- **Mode3**

TL0 and TH0 are two independent 8 bit counter/timer in this mode. TL0 can be used as timer or counter while TH0 can only be used as counter. TL0 will be controlled by CT0,GATE0,TR0,TF0 and INT0 and TH0 will only be controlled by TR1 and TF1. The control method is the same as mode0/1. When Timer0 is working in mode3, Timer1 and TH0 both are controlled by TR1. Due to TF1 is used for TH0 already, at the same time, Timer1 can only be used when there is no need for interrupt.(eg, UART baud rate generation)

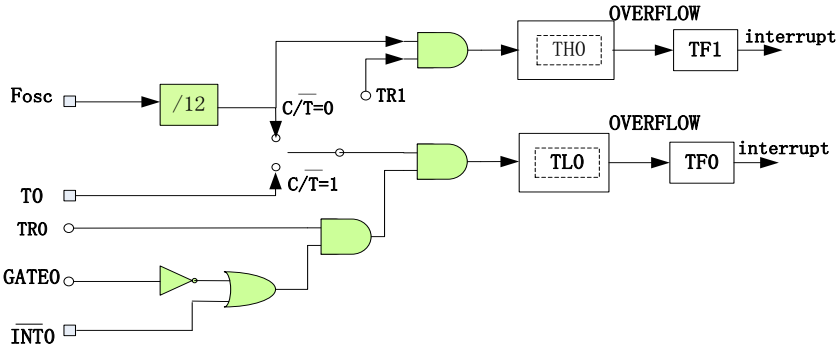


Figure 12-1-1-3 Timer0 Mode3

12.1.2 Timer0 Register Descriptions

Table 12-1-2-1 Register TCON

88H	7	6	5	4	3	2	1	0
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7	TF1	Timer0 TH0 overflow flag in mode3 /Timer1 overflow flag, it is cleared automatically after the interrupt response
6	TR1	Timer1 enable control, 1 enables it
5	TF0	Timer0 overflow flag, it is cleared automatically after the interrupt response
4	TR0	Timer0 enable control, 1 enables it
3	IE1	External Interrupt1 enable control, 1 enables it
2	IT1	External Interrupt1 trigger type control 0: External Interrupt1 is triggered when input pin signal comes to rising edge 1: External Interrupt1 is triggered when input pin signal comes to falling edge
1	IE0	External Interrupt0 enable control, 1 enables it
0	IT0	External Interrupt0 trigger type control 0: External Interrupt0 is triggered when input pin signal comes to rising edge 1: External Interrupt0 is triggered when input pin signal comes to falling edge

Table 12-1-2-2 Register TMOD

89H	7	6	5	4	3	2	1	0
TMOD	GATE1	CT1	T1M1	T1M0	GATE0	CT0	T0M1	T0M0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7	GATE1	Timer1 gating control. When it equals 1, Timer1 is enabled/disabled by INT1
6	CT1	Timer 1 counter/timer selection bit 0: Timer, the clock for it is the system clock with its frequency divided by 12 1: Counter, the clock for it is T1 input clock
5	T1M1	[T1M1,T1M0] for Timer1 mode selection
4	T1M0	00: Mode0, TL1 and TH1 make up a 13 bit Timer/Counter 01: Mode1, TL1 and TH1 make up a 16 bit Timer/Counter 10: Mode2, TL1 is a 8 bit Timer/Counter, TH1 is the automatic reload register 11: Mode3, TH1/TL1 locked in this mode, it is the same as TR1=0
3	GATE0	Timer0 gating control. When it equals 1, Timer0 is enabled/disabled by INT0
2	CT0	Timer0 Counter/Timer selection 0: Timer, the clock for it is the system clock with its frequency divided by 12 1: Counter, the clock for it is T0 input clock
1	T0M1	[T0M1,T0M0] Timer0 mode selection

0	T0M0	00: Mode0, TL0 and TH0 make up a 13 bit Timer/Counter 01: Mode1, TL0 and TH0 make up a 16 bit Timer/Counter 10: Mode2, TL0 is a 8 bit Timer/Counter, TH0 is the automatic reload register 11: Mode3, TL0 and TH0 are two independent 8 bit Timer/Counter

Table 12-1-2-3 Register TL0

8AH	7	6	5	4	3	2	1	0
TL0	TL0							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit Number	Bit Symbol	Description						
7~0	TL0	Lower byte of Timer0 count value in mode0/1, count value in mode2/3						

Table 12-1-2-4 Register TH0

8CH	7	6	5	4	3	2	1	0
TH0	TH0							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit Number	Bit Symbol	Description						
7~0	TH0	Higher byte of Timer0's count value in mode0/1, reload value in mode2, count value in mode3						

12.2 Timer1

12.2.1 Timer1 Introduction

The timer/counter function can be selected by CT1 (TMOD[6]). When CT1=0 it operates as a timer; when CT1=1, it functions as a counter.As a counter, its clock is the input clock for T1. Because it takes 2 clock cycles to detect the T1 input signal edge change, so when it operates as a counter, the maximum input baud rate is 1/2 of the internal system clock frequency. There is no limit for T1 input signal's duty cycle. However, in order to identify the 0 and 1 clearly, the signal has to keep for at least one internal system clock cycle time. There are for modes for Timer1 which are selected by T1M0 andT1M1 (TMOD[5:4]).

- **Mode 0**

In this mode, timer1 acts as a 13-bit timer/counter. TH1 holds the high 8 bits of the 13-bit timer/counter, TL1[4:0] holds the low 5 bits, and TL1[7:5] is invalid and should be ignored when read. When timer 1 overflows, interrupt flag bit TF1 (TCON[7]) will be set to 1. TF1 bit will be automatically cleared to 0 after the interrupt is responded. when GATE1 (TCON[7]) = 0, the timer/counter is enabled to count by TR1 (TCON[6]) bit, when GATE1 = 1, the timer/counter is enabled by pin INT1 control, when INT1 is high When GATE1=1, the timer/counter is enabled by pin INT1 and stops counting when INT1 is high.

- **Mode1**

Timer1 operates as a 16 bit timer/counter in this mode. TH1 stores the higher 8bits of the 16 bit timer/counter and TL1 stores the lower 8 bits. When Timer1 overflows, the interrupt flag TF1 (TCON[7]) will be set to 1. TF1 will be cleared automatically after the interrupt response. When GATE1 (TCON[7])=0, the Timer/Counter's is enabled/disabled by TR1 (TCON[6]). When GATE1=1, the timer/counter's is enabled/disabled by INT1. INT0 signal with high level with enable the counting and vice verse,

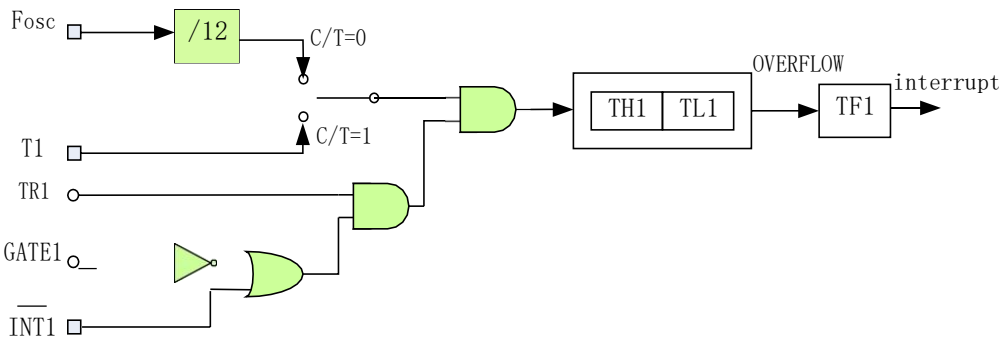


Figure 12-2-1 Mode0 and 1 of Timer1

- **Mode2**

In this mode, Timer 1 acts as an 8-bit auto-reload timer/counter and only TL1 is auto-accumulated. When TL1 counts overflow, not only the interrupt flag TF1 is generated, but also the count initial value is automatically loaded from TH1 to TL1. other setting methods are the same as modes 0 and 1.

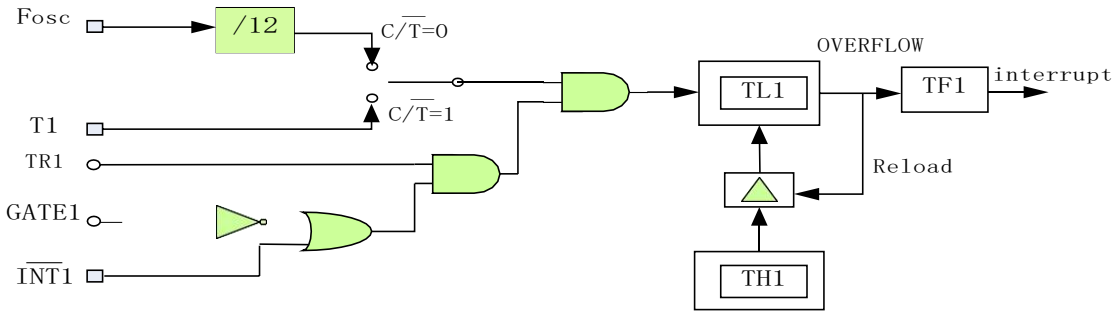


Figure 12-2-2 Mode2 of Timer1

- **Mode 3**

TH1 and TL1 are locked in this mode, which makes it the same as TR1=0.

12.2.2 Timer1 Register Description

For the the register TCON and TMOD please refer to Table12-1-2-1 and Table 12-1-2-2.

Table 12-2-2-1 Register TL1

8BH	7	6	5	4	3	2	1	0
TL1	TL1							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit Number	Bit Symbol	Description						
7~0	TL1	Lower byte of Timer1 count value in mode0/1, count value in mode2/3						

Table 12-2-2-2 Register TH1

8DH	7	6	5	4	3	2	1	0
TH1	TH1							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit Number	Bit Symbol	Description						
7~0	TH1	Higher byte of Timer1's count value in mode0/1, reload value in mode2, count value in mode3						

12.3 Timer2

12.3.1 Timer2 Introduction

Timer2 is a 16-bit (TH2 and TL2) timer/counter. T2P0 and T2P1 can be used to select different control methods or clock sources. When T2P=0 or 3, the system clock is directly selected as the clock for Timer2 (Unlike Timer0/1, the frequency of the system clock is not divided by 12).

When T2P=0, Timer2 is enabled/disabled by TR2； when T2P=2, it is electrical level gated by T2. When the level of T2 is high, the count is enabled, and when it is low, the count stops. When T2P=1 or 2, the input signal of T2 is selected as the count clock. It counts the falling edges when T2P=1 and rising edges when T2P=2.

The working modes of Timer2 can be selected by setting T2M0 and T2M1. When T2M=0, Timer2 operates as a counter/timer. TH2 and TL2 counts up as a 16 bit counter. Two reload modes can be selected or disabled by setting T2R0 and T2R1 in this mode. T2CH and T2CL stores the reload value in reload mode. If T2R=2, Timer2 will reload the initial count value from T2CH and T2CL to TH2 and TL2 when it overflows. If T2R=3, it reloads when pin T2EX comes to falling edge. The reload flag is set to 1 after the reload. If Timer2 interrupt enables reload interrupt, RF2 can be cleared by writing 1 to it.

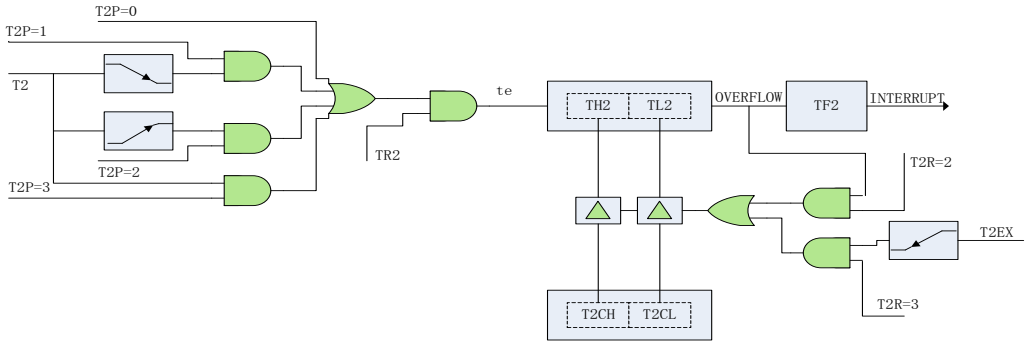


Figure 12-3-1-1 Timer2 Reload Mode

When T2M=1, Timer2 operates in compare mode. When TH2 and TL2 are greater than T2CH and T2CL, the pin T2CP output is high level, otherwise T2CP outputs low level signal.

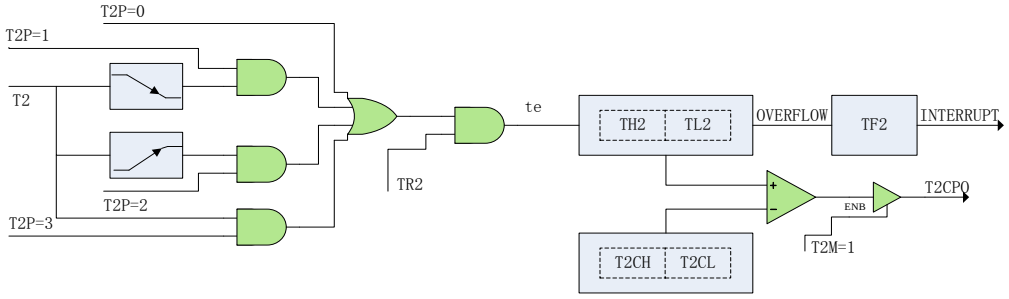


Figure 12-3-1-2 Timer2 Compare Mode Compare Mode

When T2M=2 or 3, Timer2 operates in capture mode. The T2CP pin can be selected arbitrarily when Timer 2 is operating in capture mode (see register T2CPPS description for details).

There are two modes to choose from: single-channel input capture mode and multi-channel input capture mode. When T2CS=0, it is single-channel input capture mode, otherwise it is multi-channel input capture mode.

In single channel input capture mode, when T2M=2, Timer2 count values TH2, TL2 are latched to T2CH, T2CL when pin T2CP trigger edge occurs, the trigger edge can be set by CCFG bit, when the capture event is generated, the capture interrupt flag CF2 is set to 1, if timer 2 interrupt enable will trigger the capture interrupt, CF2 is cleared to 0 by write 1. When T2M =3, the write register T2CL will generate the trigger event of latching, and the value of write T2CL is not saved, in this mode, the capture event will not set CF2.

Multi-channel input capture mode can support up to 3 channels of input, T2CPCH0E, T2CPCH1E and T2CPCH2E are enable bits for channels 0~2 respectively. 3 channels of trigger edge can be set independently, when INDEX=0~2, CCFG corresponds to CCFG0~CCFG2 respectively. channels 0~2 also have independent interrupt flags: CF20, CF21 and CF22, while CF2 no longer has a role. In multi-channel input capture mode, when the capture event is generated, the Timer 2 count values TH2, TL2 are latched to T2CPH, T2CPL (T2CPH, T2CPL are registers with index, INDEX=0~2 correspond to T2CPH0, T2CPL0~ T2CPH2, T2CPL2 respectively) instead of T2CH, T2CL. The T2CP pins of the three channels can be selected independently and arbitrarily, see the description of register T2CPPS for details.

12.3.2 Timer2 Register Description

Table 12-3-2-1 Register T2CON

C8H	7	6	5	4	3	2	1	0
T2CON	-	TR2	T2R1	T2R0	T2IE	UCKS	T2P1	T2P0
R/W	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	-	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7	-	-
6	TR2	Timer2 enable control, 1 enables it
5	T2R1	[T2R1,T2R0]Timer2 reload mode selection 10: mode 0 11: mode 1 Others: disable reload mode
4	T2R0	
3	T2IE	Timer2 interrupt enable control,1 enables it
2	UCKS	UART0 Clock selection 0: Timer1 overflow impulse used for UART0 clock 1: Timer2 overflow impulse used for UART0 clock

1	T2P1	[T2P1,T2P0] Timer2 pin T2 function selection 00: Timer2 uses internal system clock for count instead of T2 01: Timer2 counts T2 falling edges 10: Timer2 counts T2 rising edges 11: Timer2 uses internal system clock for count which is gated by T2
0	T2P0	

Table 12-3-2-2 Register T2MOD

C9H	7	6	5	4	3	2	1	0
T2MOD	TF2	CF2	RF2	CCFG[1:0]		-	T2M[1:0]	
R/W	R	R	R	R/W		-	R/W	
Initial Value	0	0	0	0	0	-	0	0

Bit Number	Bit Symbol	Description
7	TF2	Timer2 counter overflow interrupt flag, cleared by writing 1 to it
6	CF2	Capture interrupt flag for single channel capture mode, write 1 to clear 0 *Note:* *1. This bit is valid when T2CS=0. Refer to the T2CPCHS register description for details.* *2. When this bit is invalid, the value read out is always0.*
5	RF2	Automatic reload interrupt flag, cleared by writing 1 to it
4~3	CCFG	Capture mode trigger selection, valid when T2M=2 01: falling edge 10: rising or falling edge Others: rising edge *Note:* *1. When T2CS=0, CCFG is a general register and only points to the capture channel* *2. When T2CS=1, CCFG is the index register, which is pointed by INDEX=0~2 to grab channels 0~2 respectively.*
2	-	-
1~0	T2M	working mode selection 00: Timer/ counter mode 01: Compare mode 10: Capture mode 0 11: Capture mode 1

Table 12-3-2-3 Register T2C

CAH	7	6	5	4	3	2	1	0
T2CL	T2C[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
CBH	7	6	5	4	3	2	1	0
T2CH	T2C[15:8]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Note:

1. When T2CS=0, the grab function will only point to register T2C, and all functions operate normally on T2C.

2. When T2CS=1, the grab function cannot point to register T2C, but the operation of other functions on T2C is still normal.

3. Refer to the description of register T2CP for detailed comparison.

Bit Number	Bit Symbol	Description
15~0	T2C	Capture value storage register

Table 12-3-2-4 Register TL2

CCH	7	6	5	4	3	2	1	0
TL2	TL2							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7~0	TL2	Lower byte of the count value in Timer2

Table 12-3-2-5 Register TH2

CDH	7	6	5	4	3	2	1	0
TH2	TH2							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7~0	TH2	Higher byte of the count value in Timer2

Table 12-3-2-6 Register T2CPPS

80A0H	7	6	5	4	3	2	1	0
T2CPPS	-	-	-	T2CPPS[4:0]				
R/W	-	-	-	R/W				
Initial Value	-	-	-	0	1	0	1	0

Notes:

1. When T2CS=0, T2CPS is a general register, which is only used as a pin selection control register for single capture channel.

2. When T2CS=1, T2CPPS is the index register, and INDEX=0~2 points to T2CPPS0~T2CPPS2 respectively to control the grabbing channels 0~2.

Bit Number	Bit Symbol	Description
7~5	-	-
4~0	T2CPPS	Timer2 Capture pin selection bit field 00000: Select P0.0 00001: Select P0.1 00010: Select P0.2 00011: Select P0.3 00100: Select P0.4 00101: Select P0.5 00110: Select P0.6 00111: Select P0.7 01000: Select P1.0 01001: Select P1.1 01010: Select P1.2 01011: Select P1.3 01100: Select P1.4 01101: Select P1.5 01110: Select P1.6 01111: Select P1.7 10000: Select P2.0 Other: Select P3.0

Table 12-3-2-7 Register T2CPCHS

80A1H	7	6	5	4	3	2	1	0
T2CPCHS	T2CS	-	-	-	-	T2CPCH2E	T2CPCH1E	T2CPCH0E
R/W	R/W	-	-	-	-	R/W	R/W	R/W
Initial Value	0	-	-	-	-	0	0	0

Bit Number	Bit Symbol	Description
7	T2CS	Capture mode selection 0: Single channel capture mode 1: Multi-channel capture mode

		Note: *1. Regardless of the value of T2CS, the operation of the affected registers is limited to the operation of the grab function, and the operation of other functions remains unchanged.* *2. When T2CS=0, T2CPCH0E~T2CPCH2E is invalid.*
6~3	-	-
2	T2CPCH2E	In multi-channel capture mode, capture channel enable control bit, 1 means capture channel 2 enable
1	T2CPCH1E	In multi-channel capture mode, capture channel enable control bit, 1 means capture channel 1 enable
0	T2CPCH0E	In multi-channel capture mode, capture channel enable control bit, 1 means capture channel 0 enable

Table 12-3-2-8 Register T2CPF

80A2H	7	6	5	4	3	2	1	0
T2CPF	-	-	-	-	-	CF22	CF21	CF20
R/W	-	-	-	-	-	R	R	R
Initial Value	-	-	-	-	-	0	0	0

Notes:

1. Registers CF20/CF21/CF22 are valid only when T2CS=1, refer to T2CPCHS description for details.

2. When register CF20/CF21/CF22 is invalid, the value read out is always 0.

Bit Number	Bit Symbol	Description
7~3	-	-
2	CF22	Capture interrupt flag for capture channel T2CP2, write 1 to clear 0
1	CF21	Capture interrupt flag for capture channel T2CP1, write 1 to clear 0
0	CF20	Capture interrupt flag for capture channel T2CP0, write 1 to clear 0

Table 12-3-2-9 Register T2CP

80A3H	7	6	5	4	3	2	1	0
T2CL	T2CP[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
80A4H	7	6	5	4	3	2	1	0
T2CH	T2CP[15:8]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Note:

1. T2CP is the index register, and INDEX=0~2 points to registers T2CP0~T2CP2 respectively.

2. When T2CS=0, write T2CP is invalid and read T2CP will always be a fixed value (0 or some other value, depending on its state when T2CS is cleared).

3. Refer to the description of register T2C for detailed comparison.		
Bit Number	Bit Symbol	Description
15~0	T2CP	Capture value storage register

13 Watchdog Timer(WDT)

13.1 Watchdog Timer (WDT) Function Introduction

The watchdog timer is a 27 bit backward counter with alternate clock sources. When the clock frequency is 16MHz, the count time can be 0.128ms – 8.389s with 16 bit adjustment precision. The watchdog is mainly used for monitoring the system so that CPU will not break down due to external interference. If the software can not refresh WDT before it overflows, the watchdog will generate internal reset or interrupt. Writing A5H to register WDFLG will refresh the watchdog and reading WDFLG will get the status of the watchdog. If the watchdog is enabled in STOP mode, then the clock selected by the watchdog will works normally. In addition, if the interrupt function is also enabled for watchdog, it will awaken CPU in STOP mode.

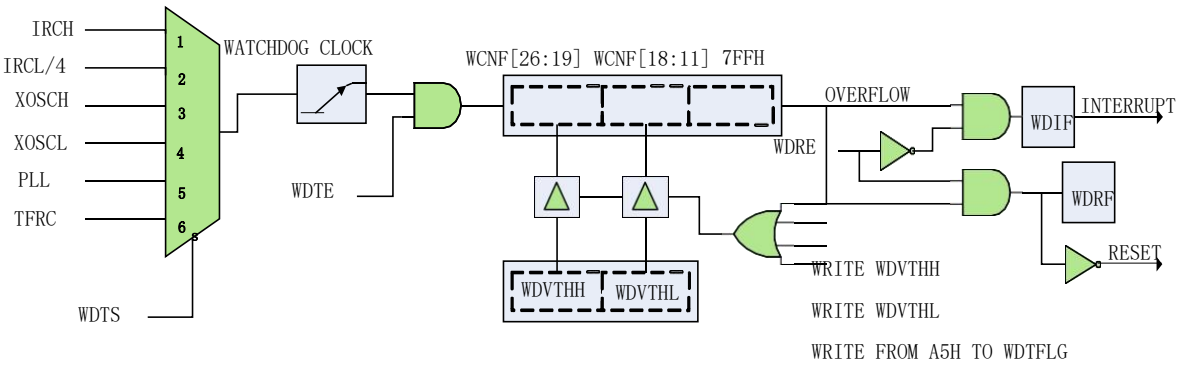


Figure 13-1-1 Watchdog Module Architecture

13.2 Watchdog Timer (WDT) Register Description

Table 13-2-1 Register WDCON

AAH	7	6	5	4	3	2	1	0
WDCON	WDTS[1:0]			-	-	-	-	WDRE
R/W	R/W			-	-	-	-	R/W
Initial Value	0	0	0	-	-	-	-	0

Bit Number	Bit Symbol	Description
7~5	WDTS	WDT Clock selection 001: IRCH 010: IRCL with frequency divided by 4 011: Invalid Others: WDT disabled

4~1	-	
0	WDRE	WDT function selection 0: interrupt happens when WDT overflows 1: reset happens when WDT overflows

Table 13-2-2 Register WDFLG

ABH	7	6	5	4	3	2	1	0
WDFLG							WDIF	WDRF
R/W	-						R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7~2	-	-
1	WDIF	WDT interrupt flag, writing A5H to the register will clear it
0	WDRF	WDT reset flag, writing A5H to the register will clear it

Table 13-2-3 Register WDVTHL、WDVTHH

ACH	7	6	5	4	3	2	1	0
WDVTHL	WDVTH[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
ADH	7	6	5	4	3	2	1	0
WDVTHH	WDVTH[15:8]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
15~0	WDVTH	WDT threshold setting, the equation is as follows: WDT trigger time ＝ (WDVTH * 800H+7FFH) * clock cycle

13.3 Watchdog Timer Control Example

Example for Watchdog interrupt mode

For instance, IRCH is set for the watchdog clock. The watchdog works in interrupt mode and the overflow time is one second, the program is like:

```
---------------------------------------------------------------------------------
#define WDTS_IRCH          (1<<5)
#define WDRE_int           (0<<0)
#define WDIF               (1<<1)
void WDT_init(void)
{
    WDCON = WDTS_IRCH | WDRE_int;      // Set the clock as IRCH and watchdog in interrupt mode
    WDVTHH = 0x1E;                     // Set one second as the time for watchdog
    WDVTHL = 0x84;
    INT7EN = 1;                        // Enable watchdog interrupt
    WDFLG = 0xA5;                      // Refresh the Watchdog
    EA = 1;                            // Enable total interruption
}
void WDT_ISR (void) interrupt 12
{
    if(WDFLG &WDIF)
    {
        // Watchdog interrupt service program
        WDFLG = 0xA5;                  // Refresh Watchdog
    }
}
```

Note: When the watchdog clock is set to IRCH, the watchdog clock is 1/2 of the IRCH frequency, i.e., the watchdog clock is 16MHz.

Example for watchdog reset mode

For instance, the watchdog clock is set to IRCH, the watchdog is set to reset mode, and the overflow time is 1 second, the program is as follows:

```
---------------------------------------------------------------------------------
#define WDTS_IRCH          (1<<5)
#define WDRE_reset         (1<<0)
void WDT_init(void)
{
    WDCON = WDTS_IRCH | WDRE_reset;    // Set watchdog clock to IRCH, watchdog reset mode
    WDVTHH = 0x1e;                     // Set one second as the time for watchdog
    WDVTHL = 0x84;
    WDFLG = 0xA5;                      // Refresh the Watchdog
```

}

Note: When the watchdog clock is set to IRCH, the watchdog clock is 1/2 of the IRCH frequency, i.e., the watchdog clock is 16MHz.

---

14 TMC Timer

14.1 TMC Function Introduction

The clock source of TMC timer is IRCL, and the minimum unit of interrupt is 512 IRCL clock cycles, and the interrupt time can be configured from 1 to 256 minimum units of time. In STOP/IDLE mode, the TMC interrupt can wake up the CPU.

14.2 TMC Register Description

Table 14-2-1 Register TMCON

F1H	7	6	5	4	3	2	1	0
TMCON	TME	-	-	-	-	-	-	TMF
R/W	R/W	-	-	-	-	-	-	R
Initial Value	0	-	-	-	-	-	-	0

Bit Number	Bit Symbol	Description
7	RTCE	TME module enable, 1 enables it
6~1	-	-
0	TMF	TMC interrupt flag, 1 enables it , write 1 to clear 0

Table 14-2-2 Register TMSNU

E9H	7	6	5	4	3	2	1	0
TMSNU	TMSNU[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	-

Bit Number	Bit Symbol	说明
7~0	TMSNU	TMC interrupt time configuration register, the interrupt time of TMC is (TMSNU+1) × 512 × Tircl *Note: Tircl is one cycle time of IRCL clock.*

14.3 TMC Control Routines

Set the TMC to the minimum unit time interrupt (i.e. 512 IRCL clock cycles) and program as follows.

```
---------------------------------------------------------------------------------------
#define TME(N)      (N<<7)    //N=0-1
#define TMF         (1<<0)
#define ILCKE       (1<<7)

void INT8_ISR (void) interrupt 13
{
    if(TMCON & TMF)             //Determine the TMC interrupt flag
    {
        TMCON |= TMF;           //Clear the TMC interrupt flag

    }
}

void TMC_init(void)
{
    CKCON |= ILCKE;     // Enable the IRCL clock
    TMCON = TME(1);     // TMC Enable
    TMSNU = 0;          // Set 1 minimum unit time ( 512 IRCL clock cycles) to generate interrupts
    INT8EN =1;          // Enable TMC interrupt
    EA = 1;             // Enable total interruption
}
---------------------------------------------------------------------------------------
```

15 General Purpose Input/Output(GPIO) and Alternate Functions

15.1 Function Introduction

General Purpose Input/Output is used for data transmission between the chip and external environment, There are at most 18 I/O pins for JZ8FC003 Series chip package and all of the pins are alternate function pins. They can not only be independently programmed as input/output port, be also be configured as pins for other functions. For each pin, there is a function register PnxF and PnxC corresponding to pin Pnx (n=0~7, stands for P0~P7, x=0~7, stands for Pn.0~Pn.7). Users can configure the main function and other options by setting register PnxF and PnxC.

Each I/O can be enabled by PnxPUP/PnxPDP(PnxF[7]/PnxF[6]) to enable pull-up/down resistors, strong/weak pull-up/down resistors are selected by PU_SEL/PD_SEL(PnxC[5]/PnxC[4]), when PU_SEL/PD_SEL is 1, it is selected as strong pull-up/down, otherwise it is weak pull-up/down, the default is strong pull-up/down.

When I/O is set to output mode, I/O is open-drain output mode when PnxOPR(PnxF[5]) is 1.

When I/O is push-pull output, DRV(PnxC[3:2]) can set IO push-pull output drive strength, SR(PnxC[1:0]) can set IO output flip slope, when I/O output level flip, due to inductance effect, it will generate overshoot signal in I/O port, this overshoot signal may cause certain impact on chip system, reduce I/O output strength and Reduce the I/O output strength and I/O speed can effectively reduce the amplitude of the overshoot signal, in the application, these two parameters can be flexibly configured.

When I/O is input mode, SMIT mode or Inverter mode can be selected by SMIT_EN (PnxC[6]) bit. When selected as Inverter mode, the trigger threshold value of high and low level is 1/2 VDD, and the default is Smit mode.

Main features of GPIO：

- High impedance mode configurable
- The strong pull-up, weak pull-up, strong pull-down, and weak pull-down resistors can be set independently for the I/O structure
- Open-drain or push-pull output can be selected for the output mode
- The data output latch can be read/written/revised
- Supports 1.8~5.5V voltage
- When set to push-pull output, the IO drive intensity can be set independently
- When set to push-pull output, the IO output speed can be set independently

Important reminder: All GPIO pin input voltages must not be higher than the VDD pin voltage, otherwise it may cause the chip to work abnormally.

The GPIO push-pull mode structure diagram is shown in Figure 15-1-1.

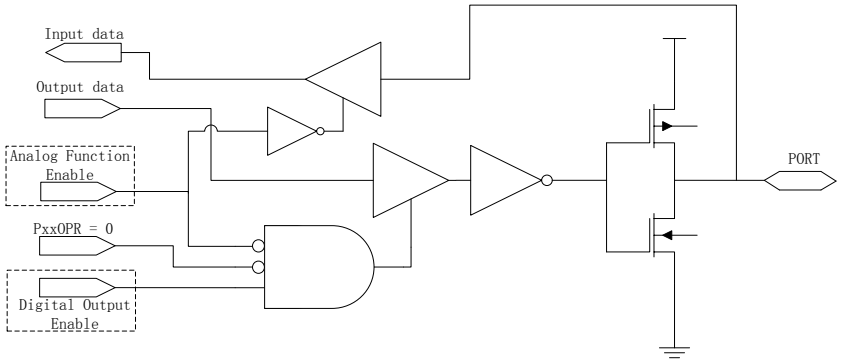


Figure 15-1-1 I/O Push-pull Mode Structure

The Figure 15-1-2 shows GPIO Open-drain Mode Structure。

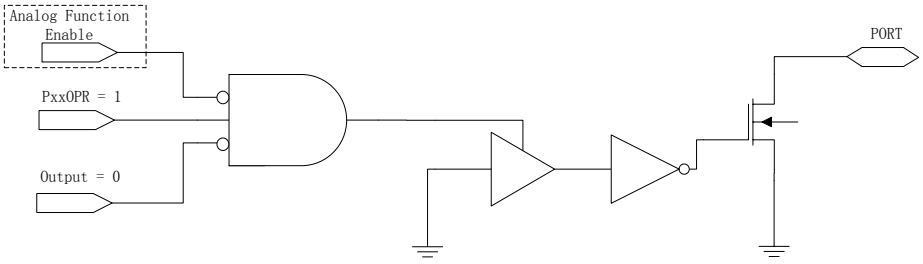


Figure 15-1-2 I/O Open-drain Mode Structure

The GPIO pull-down structure diagram is shown in Figure 15-1-3.

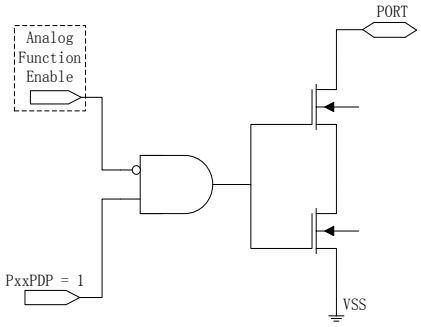


Figure 15-1-3 I/O Pull-down Mode Structure

The Figure 15-1-4 shows GPIO Pull-up Mode Structure。

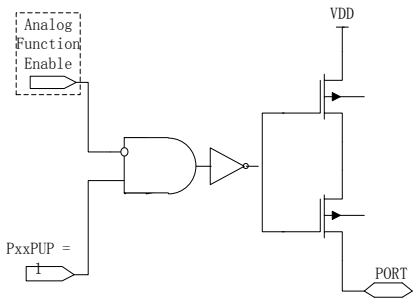


Figure 15-1-4 I/O Pull-up Mode Structure

15.2 Pin Register Description

Table 15-2-1 Register P0

80H	7	6	5	4	3	2	1	0
P0	P07	P06	P05	P04	P03	P02	P01	P00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0
Bit Number	Bit Symbol	Description						
7~0	P0x	Data register for pin P0x，valid when the pin function is set to GPIO 0:P0x is low level when the pin is set to input; when the pin set to output,P0x outputs low level signal 1:P0x is high level when the pin is set to input; when the pin set to output,P0x outputs high level signal						

Table 15-2-2 Register P1

90H	7	6	5	4	3	2	1	0
P1	P17	P16	P15	P14	P13	P12	P11	P10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0
Bit Number	Bit Symbol	Description						
7~0	P1x	Data register for pin P1x,valid when the pin function is set to GPIO 0: P1x is low level when the pin is set to input; when the pin set to output,P1x outputs low level signal 1: P1x is high level when the pin is set to input; when the pin set to output,P1x outputs high level signal						

Table 15-2-3 Register P2

A0H	7	6	5	4	3	2	1	0
P2	-	-	-	-	-	-	-	P20
R/W	-	-	-	-	-	-	-	R/W
Initial Value	-	-	-	-	-	-	-	0

Bit Number	Bit Symbol	Description
7~1	-	-
0	P20	Data register for pin P20，valid when the pin function is set to GPIO 0: P20 is low level when the pin is set to input; when the pin set to output,P20 outputs low level signal 1: P20 is high level when the pin is set to input; when the pin set to output,P20 outputs high level signal

Table 15-2-4 Register P3

B0H	7	6	5	4	3	2	1	0
P3	-	-	-	-	-	-	-	P30
R/W	-	-	-	-	-	-	-	R/W
Initial Value	-	-	-	-	-	-	-	0

Bit Number	Bit Symbol	Description
7~1	-	-
0	P30	Data register for pin P30, valid when the pin function is set to GPIO 0: P30 is low level when the pin is set to input; when the pin set to output,P3x outputs low level signal 1: P30 is high level when the pin is set to input; when the pin set to output,P3x outputs high level signal

Table 15-2-5 Pin Function Control Register

8000H	7	6	5	4	3	2	1	0
P00F	P00PUP	P00PDP	P00OPR	-	-	P00S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8001H	7	6	5	4	3	2	1	0
P01F	P01PUP	P01PDP	P01OPR	-	-	P01S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8002H	7	6	5	4	3	2	1	0
P02F	P02PUP	P02PDP	P02OPR	-	-	P02S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8003H	7	6	5	4	3	2	1	0
P03F	P03PUP	P03PDP	P03OPR	-	-	P03S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8004H	7	6	5	4	3	2	1	0
P04F	P04PUP	P04PDP	P04OPR	-	-	P04S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8005H	7	6	5	4	3	2	1	0
P05F	P05PUP	P05PDP	P05OPR	-	-	P05S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8006H	7	6	5	4	3	2	1	0
P06F	P06PUP	P06PDP	P06OPR	-	-	P06S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8007H	7	6	5	4	3	2	1	0
P07F	P07PUP	P07PDP	P07OPR	-	-	P07S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8008H	7	6	5	4	3	2	1	0

P10F	P10PUP	P10PDP	P10OPR	-	-	P10S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8009H	7	6	5	4	3	2	1	0
P11F	P11PUP	P11PDP	P11OPR	-	-	P11S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
800AH	7	6	5	4	3	2	1	0
P12F	P12PUP	P12PDP	P12OPR	-	-	P12S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
800BH	7	6	5	4	3	2	1	0
P13F	P13PUP	P13PDP	P13OPR	-	-	P13S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
800CH	7	6	5	4	3	2	1	0
P14F	P14PUP	P14PDP	P14OPR	-	-	P14S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
800DH	7	6	5	4	3	2	1	0
P15F	P15PUP	P15PDP	P15OPR	-	-	P15S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
800EH	7	6	5	4	3	2	1	0
P16F	P16PUP	P16PDP	P16OPR	-	-	P16S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
800FH	7	6	5	4	3	2	1	0
P17F	P17PUP	P17PDP	P17OPR	-	-	P17S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8010H	7	6	5	4	3	2	1	0
P20F	P20PUP	P20PDP	P20OPR	-	-	P20S		
R/W	R/W	R/W	R/W	-	-	R/W		

Initial Value	0	0	0	-	-	0	0	0
8018H	7	6	5	4	3	2	1	0
P30F	P30PUP	P30PDP	P30OPR	-	-	P30S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	1	0	0

Note:

PXnF represents P00F~P20F, where X=0,1,2,3 represents P0~P3, n=0,1,2,... ,7 (n=0 when X=2, 3).

Bit Number	Bit Symbol	Description
7	PnxPUP	Pull-up resistor enable control 0: disable pull-up resistor 1: enable pull-up resistor
6	PnxPDP	Pull-down resistor enable 0: disable pull-down resistor 1: enable pull-down resistor
5	PnxOPR	Open-drain enable control, only valid when the pin is set to be digital output 0: disable open-drain 1: enable open-drain

Table 15-2-6 Register PXnC

8120H	7	6	5	4	3	2	1	0
P00C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8121H	7	6	5	4	3	2	1	0
P01C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8122H	7	6	5	4	3	2	1	0
P02C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8123H	7	6	5	4	3	2	1	0
P03C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8124H	7	6	5	4	3	2	1	0
P04C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8125H	7	6	5	4	3	2	1	0

P05C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8126H	7	6	5	4	3	2	1	0
P06C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8127H	7	6	5	4	3	2	1	0
P07C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8128H	7	6	5	4	3	2	1	0
P10C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8129H	7	6	5	4	3	2	1	0
P11C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
812AH	7	6	5	4	3	2	1	0
P12C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
812BH	7	6	5	4	3	2	1	0
P13C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
812CH	7	6	5	4	3	2	1	0
P14C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
812DH	7	6	5	4	3	2	1	0
P15C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
812EH	7	6	5	4	3	2	1	0
P16C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
812FH	7	6	5	4	3	2	1	0
P17C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	

R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8130H	7	6	5	4	3	2	1	0
P20C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8138H	7	6	5	4	3	2	1	0
P30C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1

Note:

PXnC represents P00C~P20C, where X=0,1,2,3 represents P0~P3, n=0,1,2,... ,7 (when n=0， X=2, 3).

Bit Number	Bit Symbol	Description
7	-	-
6	SMIT_EN	Mode selection when I/O is an input function 0：Inverter mode 1：SMIT mode
5	PU_SEL	Pull-up resistor selection 0: weak pull-up (pull-up resistor is 45K) 1: Strong pull-up (pull-up resistor of 10K)
4	PD_SEL	Pull-down resistor selection 0: weak pull-down (pull-down resistor is 45K) 1: Strong pull-down (pull-down resistor of 15K)
3~2	DRV	Output strength selection , range: 0~3, the larger the value, the stronger the drive capability
1~0	SR	Output slope control , range: 0~3, the larger the value, the higher the IO flip slope (the faster the speed)

Table 15-2-7 Pin Multiplexing Function Mapping Table

Take value / Name	0	1	2	3	4	5	6	7
P00S	High resistance	Digital Input /T1/INT[n]/EPI[n]	Digital Output	High resistance	High resistance	PWM[n]	SPI_MOSI	T2CP[n]

P01S	High resistance	Digital Input /INT[n]/EPI[n]	Digital Output	ADC[2]	High resistance	PWM[n]	SPI_MISO	T2CP[n]
P02S	High resistance	Digital Input /INT[n]/EPI[n]	Digital Output	SWIM	UART2_RX	PWM[n]	I2C_SCL	T2CP[n]
P03S	High resistance	Digital Input /INT[n]/EPI[n]	Digital Output	ADC[3]	High resistance	PWM[n]	AMP_B_INP	T2CP[n]
P04S	High resistance	Digital Input /INT[n]/EPI[n]	Digital Output	ADC[4]	High resistance	PWM[n]	AMP_B_INN	T2CP[n]
P05S	High resistance	Digital Input /T0/INT[n]/EPI[n]	Digital Output	ADC[5]	BEEP	PWM[n]	AMP_A_INP	T2CP[n]
P06S	High resistance	Digital Input /INT[n]/EPI[n]	Digital Output	ADC[6]	UART1_TX	PWM[n]	AMP_A_INN	T2CP[n]
P07S	High resistance	Digital Input /INT[n]/EPI[n]	Digital Output	ADC[7]	UART1_RX	PWM[n]	AMP_A_OUT	T2CP[n]
P10S	High resistance	Digital Input /T2/INT[n]/EPI[n]	Digital Output	High resistance	High resistance	PWM[n]	SPI_SCK	T2CP[n]
P11S	High resistance	Digital Input /T2EX/INT[n]/EPI[n]	Digital Output	ADC[1]	High resistance	PWM[n]	High resistance	T2CP[n]
P12S	High resistance	Digital Input /INT[n]/EPI[n]	Digital Output	ADC[0]	ADC_VREF	PWM[n]	High resistance	T2CP[n] /T2CPO
P13S	High resistance	Digital Input /INT[n]/EPI[n]	Digital Output	High resistance	High resistance	PWM[n]	I2C_SCL	T2CP[n]
P14S	High resistance	Digital Input /INT[n]/EPI[n]	Digital Output	High resistance	FB	PWM[n]	I2C_SDA	T2CP[n]
P15S	High resistance	Digital Input /INT[n]/EPI[n]	Digital Output	ADC[11]	High resistance	PWM[n]	SPI_SSB	T2CP[n]
P16S	High resistance	Digital Input /INT[n]/EPI[n]	Digital Output	ADC[10]	UART2_TX	PWM[n]	I2C_SDA	T2CP[n]
P17S	High resistance	Digital Input /INT[n]/EPI[n]	Digital Output	ADC[9]	XOSCH_OUT	PWM[n]	High resistance	T2CP[n]
P20S	High resistance	Digital Input /INT[n]/EPI[n]	Digital Output	RESET	High resistance	PWM[n]	High resistance	T2CP[n]
P30S	High resistance	Digital Input /INT[n]/EPI[n]	Digital Output	ADC[8]	XOSCH_IN	PWM[n]	CLK_IN	T2CP[n]

15.3 Pin control Example

Set the Pin function

For instance, P20 is set to be push-pull output, the program is like：

```
---------------------------------------------------------------------------------------------
P00F = 2;
---------------------------------------------------------------------------------------------
```

P00 is set to be open-drain output, the program is like：

```
---------------------------------------------------------------------------------------------
P00F = (1<<5)|2;
---------------------------------------------------------------------------------------------
```

P00 is set to be open-drain output with pull-up enabled, the program is like：

```
---------------------------------------------------------------------------------------------
P00F = (1<<7) / (1<<5) / 2;
---------------------------------------------------------------------------------------------
```

P00 is set to be input with pull-up enabled, the program is like：

```
---------------------------------------------------------------------------------------------
P00F = (1<<7) / 1;
---------------------------------------------------------------------------------------------
```

16 Universal Asynchronous Receiver/Transmitter（UART1/UART2）

16.1 UART1 And UART2

16.1.1 Introduction

UART1 and UART2 are two full-duplex asynchronous serial data transceivers of identical design, and UARTx (x=1, 2, which stands for UART1 and UART2) also has a one-byte receive buffer. UARTx has two different operating modes, as shown in Table 16-1-1-1.

SMx	Mode	Description	Baud rate
0	A	9-bit asynchronous mode	CPUCLK/(32*(1024-SxREL))
1	B	8-bit asynchronous mode	CPUCLK/(32*(1024-SxREL))

Table 16-1-1-1 UARTx Working Mode

The UARTx is designed with a dedicated baud rate generator and the baud rate is configured through registers SxRELL, SxRELH.

Note: The baud rate of UARTx cannot be set by the SMOD bit of PCON register to multiply the frequency.

Figure 16-1-1-1 shows the schematic diagram of UARTx.

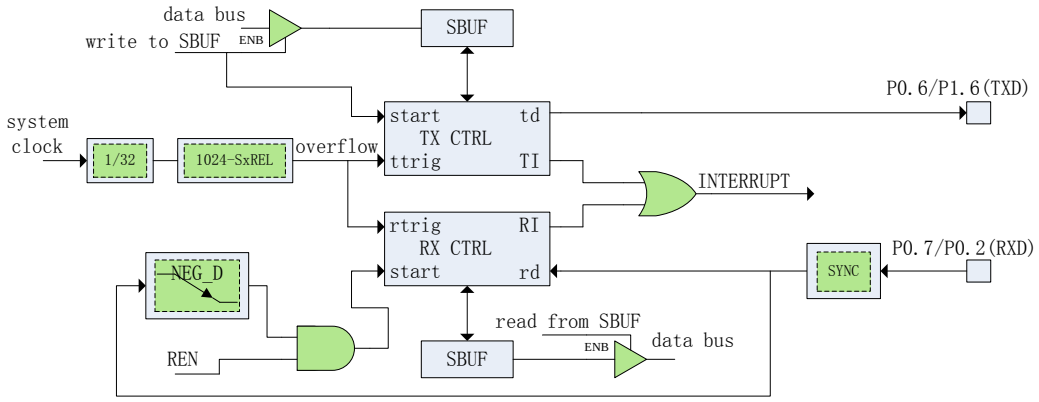


Figure 16-1-1-1 Schematic Diagram of UARTx Working Principle

- **ModeA**

In modeA, the UARTx can send and receive 9 bits of data simultaneously asynchronously. Writing data to register SxBUF initiates UARTx data transmission. The first bit transmitted is the start bit (for 0), followed by 9 bits of data (low first), the 9th bit of data is the TB8x bit of register SxCON, and the last bit transmitted is the stop bit (for 1). In the receive state, UARTx is synchronized by detecting the falling edge of pin RX. After the transmission process is completed, the low 8 bits of data are stored in register SxBUF and the 9th bit of data is stored in bit RB8x.

- **ModeB**

ModeB differs from ModeA in that ModeB is an 8-bit data transfer, and the stop bit holds a valid stop bit. Other functions are the same as ModeA.

- **UARTx Multi-Machine Communication**

There is a mechanism in UARTx mode A that is specifically applicable to multi-machine communication. When the SM2x position of register SxCON is 1, only the slave that receives the 9th bit data as 1 (RB8x=1) will generate the receive interrupt. Using this function, multi-machine communication can be performed; the slaves set all their SM2x bits to 1, and the host sets the 9th bit data to 1 when transmitting the slave's address, so that all the slaves will generate the receive interrupt; the slave's software uses their own address and the If they agree, the addressed slave sets SM2x=0, and then the host sets bit 9 to 0 when it continues to transmit the next data, because the other slaves still have SM2x set to 1, so that only the addressed slave generates a receive interrupt.

- **Fast Baud Rate Setting**

In the standard 51 MCU UART, the baud rate of UART is fixed as 32 divisions of the timer overflow rate. Since the CPU clock of JZ8FC003 series MCU is 16MHz (or 16MHz divisions), the configured baud rate has a large error compared with the standard baud rate, so the mechanism to correct the baud rate is designed in the JZ8FC003 series MCU. The baud rate of UART is not fixed to 32 divisions of the timer overflow rate, but can be set by register UDCKS. For example, when the baud rate of UART is fixed to 32 divisions of the timer overflow rate, Timer 2 is selected as the baud rate generator of UART, if the baud rate is configured to 115200, the calculation formula is: 16000000/32/115200=4.34, because the timer count can only take integer. So take 4 (that is, the timer overflows once every 4 system clock cycles), the error rate is about 8.5%, the error rate will be too large to cause abnormal communication. Because the system clock is fixed, to achieve a more accurate baud rate, only by modifying the frequency division factor to achieve. If you set the timer 5 clock cycles overflow, then: 16000000/115200/5 = 27.78. Take the number of frequency division as 28, then the baud rate is 114285, and 115200 compared to the error rate of about 0.8%, generally will not affect the UART communication. In addition, a smaller number of divisions can also achieve a higher baud rate configuration.

The default crossover factor of the chip is 32, which is the same as the standard 51. If you want to change the crossover coefficient, enable it by setting UDEx=1. The value of DNUM indicates a different crossover coefficient, see the description of register UDCKSx for details.

16.1.2 UARTx Register Description

Table 16-1-2-1 Register S1CON

9AH	7	6	5	4	3	2	1	0
S1CON	SM1	U1IE	SM21	REN1	TB81	RB81	TI1	RI1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0
Bit Number	Bit Symbol	Description						

7	SM1	UART1 mode selection , for more information please refer to Table 16-2-1-1
6	U1IE	UART1 interrupt enable , 1 enables
5	SM21	Multi-computer communication enable control, 1 enables
4	REN1	Serial receive enable control, 1 enables
3	TB81	The 9th data bit to transmit It will be transmitted as the 9th bit of the data in mode A and it is controlled by the software (For instance, parity check or multi-computer communication)
2	RB81	The 9th bit of the data received It will be used for UART1 to receive the 9th bit of the data in modeA. It is the stop bit in modeA;
1	TI1	Transmit interrupt flag, 1 indicates the interrupt, cleared by writing 0 to it
0	RI1	Receive interrupt flag, 1 indicates the interrupt, cleared by writing 0 to it

Table 16-1-2-2 Register S1BUF

9BH	7	6	5	4	3	2	1	0
S1BUF	S1BUF[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7~0	S1BUF	Receiver/Transmitter buffer Writing data to S1BUF will starts the data transmission Reading S1BUF will reads the data received

Table 16-1-2-3 Register S1RELL、S1RELH

9CH	7	6	5	4	3	2	1	0
S1RELL	S1RELL[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
9DH	7	6	5	4	3	2	1	0
S1RELH	-	-	-	-	-	-	S1REL[9:8]	
R/W	-	-	-	-	-	-	R/W	
Initial Value	-	-	-	-	-	-	0	0

Bit Number	Bit Symbol	Description
9~0	S1REL	Baud rate configuration register Baud rate is CPUCLK/(32 * (1024 - S1REL))

Table 16-1-2-4 Register S2CON

A1H	7	6	5	4	3	2	1	0
S2CON	SM2	U2IE	SM22	REN2	TB82	RB82	TI2	RI2
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7	SM2	UART2 mode selection , for more information please refer to Table 16-2-1-1
6	U2IE	Serial port 2 interrupt enable control, 1 enables
5	SM22	Multi-computer communication enable control, 1 enables
4	REN2	Serial receive enable control, 1 enables
3	TB82	The 9th data bit to transmit It will be transmitted as the 9th bit of the data in modeA and it is controlled by the software (For instance, parity check or multi-computer communication)
2	RB82	The 9_{th} bit of the data received It will be used for UART0 to receive the 9_{th} bit of the data in modeA. It is the stop bit in modeB;
1	TI2	Transmit interrupt flag, 1 indicates the interrupt, cleared by writing 0 to it
0	RI2	Receive interrupt flag, 1 indicates the interrupt, cleared by writing 0 to it

Table 16-1-2-5 Register S2BUF

A2H	7	6	5	4	3	2	1	0
S2BUF	S2BUF[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7~0	S2BUF	Receiver/Transmitter buffer Writing data to S2BUF will starts the data transmission Reading S2BUF will get the data already received

Table 16-1-2-6 Register S2REL

A3H	7	6	5	4	3	2	1	0
S2RELL	S2RELL[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
A4H	7	6	5	4	3	2	1	0
S2RELH	-	-	-	-	-	-	S2REL[9:8]	

R/W	-	-	-	-	-	-	R/W	
Initial Value	-	-	-	-	-	-	0	0

Bit Number	Bit Symbol	Description
9~0	S2REL	Baud rate configuration register Baud rate is CPUCLK/(32 * (1024 - S2REL))

Table 16-1-2-7 Register UDCKSx

8118H	7	6	5	4	3	2	1	0
UDCKS1	UDE1	-	-	DNUM1[4:0]				
R/W	R/W	-	-	R/W				
Initial Value	0	-	-	0	0	0	0	0
8119H	7	6	5	4	3	2	1	0
UDCKS2	UDE2	-	-	DNUM2[4:0]				
R/W	R/W	-	-	R/W				
Initial Value	0	-	-	0	0	0	0	0

Bit Number	Bit Symbol	Description
7	UDEx	Fast baud rate configuration enable control, 1 enables *Note:* *When UDE x=0, the UARTx baud rate is configured as original, UDEx=1, the UAR Tx baud rate is configured by DNUMx.*
6~5	-	-
4~0	DNUMx	Fast baud rate configuration register, valid only when UDEx=1 When sending, DNUMx >=0; when receiving, DNUMx>=6 BR = F_{sys}/(((DNUMx+1)*(1024-SxREL))

17 I²C Interface

17.1 Function Introduction

I²C modules enables the chip to communicate with peripheral I²C devices by serial transmission standard which complies with standard I²C specification. It can be set to either slave or master and configured to standard/fast/high speed mode.

17.2 I²C Main Features

- Simple but strong communication port, bi-directional bus with 2 wires
- Slave/Master mode configurable
- Able to operate in receiver/transmitter mode
- 7 bit slave address
- Supports multimaster's arbitration
- Broadcast function supported

17.3 I²C Function Description

I²C modules supports I²C standard bus specification. I²C bus includes 2 wires to transfer data among devices, one is SCL(Serial Clock) and the other is SDA(Serial Data), as Figure 17-3-1 shows. Since the it is open-drain port for I²C, there must be pull-up resistor on I²C bus. The pull-up resistor can be connected externally or enabled internally. Each device that connects to the bus has its own 7-bit address.

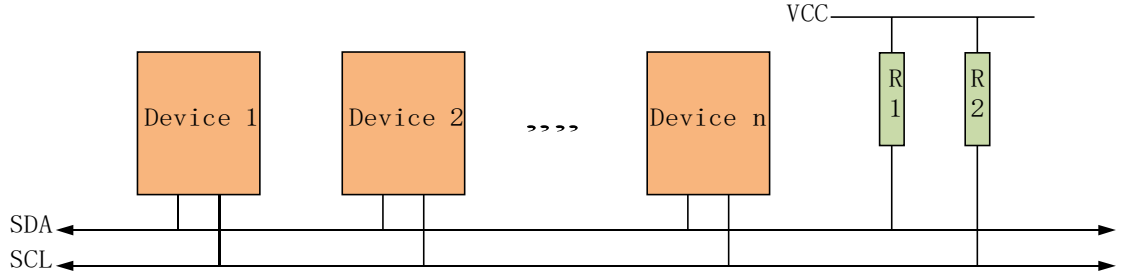


Figure 17-3-1 I2C Bus Interconnection Diagram

I²C module principle is as Figure 17-3-2 shows.

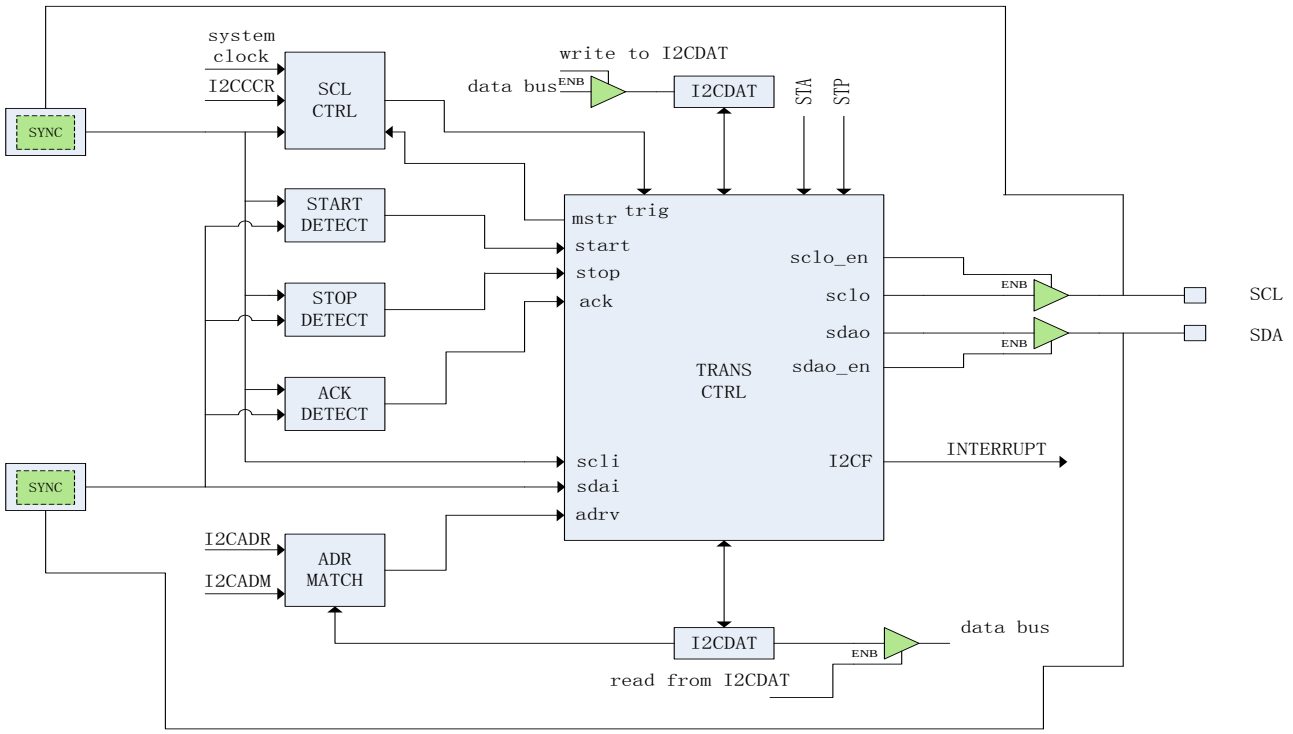


Figure 17-3-2 I²C Schematic Diagram of the Module Princip

- **I²C Mode Selection**

I2C can operates in the following 4 modes：slave transmit mode, slave receive mode, master transmit mode, master receive mode. I2C operates in slave mode by default. I2C changes to master mode after the START signal generated and returns slave mode when the arbitration fails or STOP signal is generated.

- **I2C Bus Data Transmission Pattern**

There are usually 4 stages for the standard I2C communication: START signal, slave address transfer, data transmission and STOP signal. The data transmitted on I2C bus is always 8 bits with the most significant bit sent first. There must be a ACK following every one byte data. However, there is no byte limits for the data transmission. The master sends STOP signal after the transmission is over and terminates the communication.

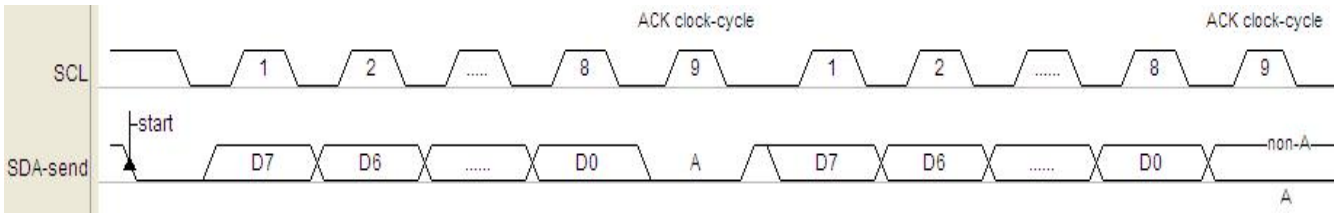


Figure 17-3-3 I2C Bus Data Transfer Format

- **Communication Process**

I2C enables the data transmission and generates the clock signal in Master mode. The serial data transmission always begins with START signal and ends with STOP signal. Both START and STOP signals are generated by the software in master mode. Setting STA=1 generates START signal and setting STP=1 generates STOP
Signal.
I2C can distinguish its address (7 bits) and the broadcast address in slave mode. Software can enable/disable its ability to recognize broadcast address by setting GCE.

Both address and data are transmitted in bytes. The address will be sent by the master after the START signal. The receiver must reply with a ACK signal in the 9th clock cycle after one byte information is transferred. The ACK can be set by AAK while it must be set before the one byte information transfer completes. When the one byte information is received, the ACK signal will be generated automatically.

Every time when one byte data is received/transmitted or arbitration fails (and etc.) there will be an interrupt flag I2CF. The status of the event will be indicated by register I2CSTA (for more information please refer to register I2CSTA). The software decides the next operation according to the status of the event when interrupt occurs. Clearing the interrupt flag I2CF will start the next operation. After the communication ends the host generates the STOP signal will also generate the interrupt flag I2CSTP at the slave side to indicate the completion of the communication process. When the interrupt flag I2CF is generated, if SHD=1, SCL will be pulled low by the slave before clearing I2CF, and the host will detect that SCL is released before the next operation; if SHD=0, the slave will not pull low SCL, which is designed to be compatible with the application that the host is software emulating I2C, at this time, the host's software must wait long enough for the slave to respond to each byte of data transfer is processed.

- **I2C Clock Settings**

When the I^{2}C interface is used as a slave, the clock of SCL is input by the host and is independent of the clock configuration of the slave. When acting as a slave, the sampling clock of I2C is set by SMPDIV (I2CCCR[7:5]), and the filtering function is automatically activated when SMPDIV is not 0. When acting as a master, the output clock frequency of SCL is determined by SMPDIV and I2CCKD(I2CCCR[4:0]) (please check the introduction of register section for details).

17.4 I^{2}C Communication Pin Mapping

There are different mappings for I^{2}C communication pins which could be selected by register I2CIOS. For more information please refer to register I2CIOS description.

17.5 Register Description

Table 17-5-1 Register I2CCON

B1H	7	6	5	4	3	2	1	0
I2CCON	I2CE	I2CIE	STA	STP	SHD	AAK	CBSE	STFE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7	I2CE	I^2C module enable control, 1 enables it
6	I2CIE	I^2C interrupt enable control, 1 enables it
5	STA	I^2C START signal transfer control,valid when it is 1, it will be cleared automatically when START signal detected
4	STP	I^2C STOP signal transfer control, valid when it is 1, it will be cleared automatically when STOP signal detected
3	SHD	When it is 1, if I2CF=1, I2CF will make SCL remain low after SCL becomes low
2	AAK	I^2C ACK signal transfer control, 1 enables it *Note :* *When I^2C is configured as slave, this bit must be set to 1 beforehand, otherwise even the address matches it will not reply ACK*
1	CBSE	CBUS compatible enable control When it is set to 1, the ACK will be ignored during the transmission to be compatible with CBUS bus. Since the address for CBUS bus is 7 bits, thus GCE must be set to 0.
0	STFE	When STFE=1, I2CF will be set to 1 if I^2C module detects the START signal

Table 17-5-2 Register I2CADR

B2H	7	6	5	4	3	2	1	0
I2CADR	GCE	I2CADRL[6:0]						
R/W	R/W	R/W						
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7	GCE	Broadcast address recognition(00H) enable control, 1 enables it
6~0	I2CADRL	I^2C slave address, only valid when it operates as slave *Note：* *(when AAK=1) when the address is 7 bits and the higher 7 bits of first received address matches I2CADR, reply with ACK and enters slave mode*

Table 17-5-3 Register I2CADM

B3H	7	6	5	4	3	2	1	0
I2CADM	SPFE	I2CADML[6:0]						
R/W	R/W	R/W						

Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7	SPFE	When SPFE=1, I2CF will be set to 1 if I^2C module detects the STOP signal
6~0	I2CADML	I^2C address mask by bit control, valid only when it operates as slave When I2CADM[n](n=0~6)=1, the corresponding address bit I2CADR[n] will not be compared (which means no matter what is received, it is seen as matched)

Table 17-5-4 Register I2CCCR

B4H	7	6	5	4	3	2	1	0
I2CCCR	SMPDIV[2:0]			I2CCKD[4:0]				
R/W	R/W							
Initial Value	0	0	1	0	0	0	0	0

Bit Number	Bit Symbol	Description
7~5	SMPDIV	I^2C sample clock setting, the I^2C sample clock is the smpdiv power division of 2 of the I2C operating clock, i.e. 000:F_{sample}=Fi2cclk 001:F_{sample}=Fi2cclk/2 010:F_{sample}=Fi2cclk/4 ... 111:F_{sample}=Fi2cclk/128
4~0	I2CCKD	I2C SCL output clock frequency setting, SCL output clock frequency is the (I2CCKD + 1) division of the sampling frequency, i.e. F_{scl}= F_{sample} /(I2CCKD +1)) Note: *1. When SMPDIV= 0, if the setting I2CCKD is less than 9, it will be calculated by 9 automatically.* *2. When SMPDIV>0, if the setting I2CCKD is less than 7, it will be calculated by 7 automatically.* *Note:* *1. When I2CCCR[7:5] = 0, if a value less than 9 is written to I2CCCR[4:0], the value of 9 will be calculated automatically.* *2. When I2CCCR[7:5] > 0, if a value less than 7 is written to I2CCCR[4:0], the value of 7 will be calculated automatically.*

Table 17-5-5 Register I2CDAT

B5H	7	6	5	4	3	2	1	0
I2CDAT	I2CDAT[7:0]							
R/W	R/W							

Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7~0	I2CDAT	Data buffer for receiving/transmission *Note：* *When I2CF is 1, it is recommended to make I2CF remain 1 when users overwrite/read I2CDAT. I2CF should be cleared after the process is over, and then the transmission continues so that there will be no transmission errors*

Table 17-5-6 Register I2CSTA

B6H	7	6	5	4	3	2	1	0
I2CSTA	I2CSTA[7:0]							
R/W	R							
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7~0	I2CSTA	I^2C status register 00H: (master/slave) bus error 08H: (master/slave)START signal detected (valid only when STFE=1) 18H: (master)address and write bit sent, ACK signal received 20H: (master)address and write bit sent, no ACK signal received 28H: (master)one byte data received/transmitted, ACK signal detected 30H: (master)one byte data received/transmitted, no ACK signal detected 38H: (master)arbitration lost(master will change to slave after arbitration lost) 40H: (master)address and read bit transmitted, ACK signal received 48H: (master)address and read bit transmitted, no ACK signal received 60H: (slave)address and write bit received, with ACK signal is sent 70H: (master/slave)broadcast address received with ACK signal is sent(master/slave will become slave) 80H: (slave)one byte data received/transmitted, ACK signal detected 88H: (slave)one byte data received/transmitted, no ACK signal detected A0H: (master/slave)STOP signal detected(valid only when SPFE=1) A8H: (slave)address and read bit received, with ACK signal is sent F8H: (master/slave) bus is idle

Table 17-5-7 Register I2CFLG

B7H	7	6	5	4	3	2	1	0
I2CFLG	-	-	-	-	-	-	-	I2CF
R/W	-	-	-	-	-	-	-	R

Initial Value	-	-	-	-	-	-	-	0

Bit Number	Bit Symbol	Description
7~1	-	-
0	I2CF	I^2C interrupt flag, 1 indicates the interrupt, cleared by writing 1 to it *Note：* *1. I2CF will be set to 1 every time after a one-byte data or the address transmission completes (with ACK/NAK received/sent)* *2. I2CF will be set to 1 when there is bus error* *3. If STFE=0, I2CF will not be set to 1 when START signal detected* *4. If SPFE=0, I2CF will not be set to 1 when STOP signal detected*

Table 17-5-8 Register I2CIOS

8101H	7	6	5	4	3	2	1	0
I2CIOS	I2CKS	-	-	-	-	-	-	I2CS
R/W	R/W	-	-	-	-	-	-	R/W
Initial Value	0	-	-	-	-	-	-	0

Bit Number	Bit Symbol	Description
7	I2CKS	I^2C operating clock selection 0: System clock 1: Internal high-speed clock
6~1	-	-
0	I2CS	I^2C pin selection 0: SCL on pin P1.3, SDA on pin P1.4 1: SCL at pin P0.2, SDA at pin P1.6

17.6 I^2C Control Example

I2C as master

For instance , the master sends 20 byte data to the slave cyclically, the program is like：

```
//I2CCON
#define I2CE(N)          (N<<7)
```

```
#define I2CIE(N)          (N<<6)
#define STA(N)            (N<<5)
#define STP(N)            (N<<4)
#define CKHD(N)           (N<<3)
#define AAK(N)            (N<<2)
#define CBSE(N)           (N<<1)
#define STFE(N)           (N<<0)
//I2CADR Definition
#define     GCE(N)        (N<<7) //N = 0~1
//I2CFLG 定义
#define I2CF              (1<<0)

#define I2C_ADDR          0xCA              // Define the I2C slave address
unsigned char xdata WriteBuffer[20]={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19};
void main(void)
{
    unsigned char i;
    EA = 1;                                 // Enable global interrupt
/************************************ Select I2C Port ************************************/
    I2CIOS = 0;                 // Set P14,P13 as I2C communication pins
    P14F = 6 | (1<<7);          // Set P14 as I2C SDA and enable pull-up function
    P13F = 6 | (1<<7);          // Set P13 as I2C SCL and enable pull-up function

//  I2CIOS = 1;                 // Set P16,P02 as I2C communication pins
//  P16F = 6 | (1<<7);          // Set P16 as I2C SDA and enable pull-up function
//  P02F = 6 | (1<<7);          // Set P02 as I2C SCL and enable pull-up function
// Choose one of the above two groups of ports
/*****************************************************************************************/
    I2CCON = I2CE(1) | I2CIE(0) | STA(0) | STP(0)| CKHD(1) | AAK(1)| CBSE(0) | STFE(1);
    I2CADR = GCE(0);
    I2CCCR = 0x4c;                          // Setting the I2C clock
    while(1)
    {
        I2CCON |= STA(1);               // I2C host sends START signal
        while(!(I2CFLG & I2CF));        // Waiting for the interrupt flag to be generated
        if(I2CSTA != 0x08)
        {
            I2CFLG |= I2CF;
            goto SEND_STOP;
        }
        I2CDAT = I2C_ADDR;              // Host sends slave address + write bit
        I2CFLG |= I2CF;                 // Clear the interrupt flag
        while(!(I2CFLG & I2CF));        // Waiting for the interrupt flag to be generated
        if(I2CSTA != 0x18)
```

```
            {
                I2CFLG |= I2CF;
                goto SEND_STOP;
            }

            I2CDAT = 0;                     // Host send data register address
            I2CFLG |= I2CF;                 // Clear the interrupt flag
            while(!(I2CFLG & I2CF));        // Waiting for the interrupt flag to be generated
            if(I2CSTA != 0x28)
            {
                I2CFLG |= I2CF;
                goto SEND_STOP;
            }
            for(i = 0; i < 20; i++)         // Host sends 20 data
            {
                I2CDAT =WriteBuffer[i];
                I2CFLG |= I2CF;             // Clear the interrupt flag
                while(!(I2CFLG & I2CF));    // Waiting for the interrupt flag to be generated
                if(I2CSTA != 0x28)
                {
                    I2CFLG |= I2CF;
                    goto SEND_STOP;
                }
            }
SEND_STOP:
            I2CCON |= STP(1);               // Send STOP signal
            I2CFLG |= I2CF;
            Delay_ms(100);
        }
    }
```

For instance, the master reads 20 byte data from the slave cyclically, the program is like：

```
#define I2C_ADDR        0xCA                // Define the I2C slave address
unsigned char xdata ReadBuffer[20];
void main(void)
{
    unsigned char i;
    EA = 1;                                 // Open global interrupt
/************************************ Select I2C Port ************************************/
    I2CIOS = 0;                 // Select P14,P13 as I2C communication pins
    P14F = 6 | (1<<7);          // Set P14 as I2C SDA and enable the upper
    P13F = 6 | (1<<7);          // Set P13 as I2C SCL and enable the upper
```

```
//      I2CIOS = 1;                     // Select P16,P02 as I2C communication pins
//      P16F = 6 | (1<<7);              // Set P16 as I2C SDA and enable the upper
//      P02F = 6 | (1<<7);              // Set P02 as I2C SCL and enable the upper
// Choose one of the above two groups of ports
/*****************************************************************************************/
      I2CCON = I2CE(1) | I2CIE(0) | STA(0) | STP(0)| CKHD(1) | AAK(1)| CBSE(0) | STFE(1);
      I2CADR = GCE(0);
      I2CCCR = 0x4c;                                  // Setting the I2C clock
      while(1)
      {
            I2CCON |= STA(1);                 // I2C host sends START signal
            while(!(I2CFLG & I2CF));          // Waiting for the interrupt flag to be generated
            if(I2CSTA != 0x08)
            {
                  I2CFLG |= I2CF;
                  goto SEND_STOP;
            }
            I2CDAT = I2C_ADDR;                // Host sends slave address + write bit
            I2CFLG |= I2CF;                   // Clear the interrupt flag

            while(!(I2CFLG & I2CF));          // Waiting for the interrupt flag to be generated
            if(I2CSTA != 0x18)
            {
                  I2CFLG |= I2CF;
                  goto SEND_STOP;
            }

            I2CDAT = 0;                       // Host send data register address
            I2CFLG |= I2CF;                   // Clear the interrupt flag
            while(!(I2CFLG & I2CF));          // Waiting for the interrupt flag to be generated
            if(I2CSTA != 0x28)
            {
                  I2CFLG |= I2CF;
                  goto SEND_STOP;
            }

            I2CCON |= STA(1);                 // I2C host sends START signal
            I2CFLG |= I2CF;                   // Clear the interrupt flag
            while(!(I2CFLG & I2CF));          // Waiting for the interrupt flag to be generated
            if(I2CSTA != 0x08)
            {
                  I2CFLG |= I2CF;
                  goto SEND_STOP;
```

```
        }

        I2CDAT = I2C_ADDR+1;                // Host sends slave address + read bit
        I2CFLG |= I2CF;                     // Clear the interrupt flag
        while(!(I2CFLG & I2CF));            // Waiting for the interrupt flag to be generated
        if(I2CSTA != 0x40)
        {
            I2CFLG |= I2CF;
            goto SEND_STOP;
        }
        I2CCON |= AAK(1);                   // Set answer bit

        for(i = 0; i < 20; i++)
        {
            I2CFLG |= I2CF;                 // Clear the interrupt flag
            while(!(I2CFLG & I2CF));        // Waiting for the interrupt flag to be generated
            if(I2CSTA != 0x28 && I2CSTA != 0x30)
            {
                I2CFLG |= I2CF;
                goto SEND_STOP;
            }
            ReadBuffer[i] = I2CDAT;         // Read data to data register
            if(i < 19)
            {
                I2CCON |= AAK(1);           // If it is not the last byte, preset ACK status
            }
            else
            {
                I2CCON &= ~AAK(1);          // If it is the last byte, no ACK is sent
            }
        }
SEND_STOP:
        I2CCON |= STP(1);                   // Send STOP signal
        I2CFLG |= I2CF;
        Delay_ms(100);
    }
}
```

I2C As Slave Routines

As a slave, it supports the host to write or read data with the following procedure.

```
#define I2C_ADDR        0xCA            // Define the I2C slave address
unsigned char I2CDataIndex;
```

```
    unsigned char regAddr;
    bit iicReadMode;
    unsigned char xdata Buffer[20]={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19};// Set the initial value of data register to 0~19
    void INT6_ISR(void) interrupt 11
    {
        unsigned char Sta_Temp;
        if(I2CFLG & I2CF)                       //IIC    interrupt
        {
            Sta_Temp = I2CSTA;
            if(Sta_Temp == 0x60)                // Receive slave address + write bit
            {
                I2CDataIndex = 0xFF;            // Set to 0xFF means that the first byte received later is the address
                iicReadMode = 0;                // Set to slave receive status
                I2CCON |= AAK(1);
            }
            else if(Sta_Temp == 0x80)           // Send or receive one byte of data, an answer signal is detected
            {
                if(iicReadMode)                 // Send one byte of data
                {
                    I2CDataIndex++;
                    I2CDAT = Buffer[I2CDataIndex + regAddr];    // Load the data into the transmit register and wait for the host to read it
                }
                else                                            // One byte of data received
                {
                    if(I2CDataIndex == 0xFF)                    // Address
                    {
                        regAddr = I2CDAT;                       // The first byte received is considered to be the address
                        I2CDataIndex = 0;                       // Set the index value to 0
                        I2CCON |= AAK(1);
                    }
                    else                                        // Data
                    {
                        Buffer[I2CDataIndex + regAddr] = I2CDAT; // Received data is loaded into the data register
                        I2CDataIndex++;                         // Index value accumulation
                        I2CCON |= AAK(1);
                    }
                }
            }
            else if(Sta_Temp==0xA8)                             // Receive slave address + read bit, send ACK signal
            {
                I2CDAT = Buffer[I2CDataIndex + regAddr];        // Load the data into the transmit register and wait for the host to read it
```

```
                iicReadMode = 1;                    // Set to slave transmit state
            }
            else if(Sta_Temp == 0x88)               // Send or receive one byte of data, an answer signal is detected
            {
            }
            I2CFLG |= I2CF;                         // Clear the interrupt flag
        }
    }


    void main(void)
    {
        EA = 1;                                     // Open global interrupt
    /************************************ Select I2C Port ************************************/
        I2CIOS = 0;                 // Select P14,P13 as I2C communication pins
        P14F = 6 | (1<<7);          // Set P14 as I2C SDA and enable pull-up function
        P13F = 6 | (1<<7);          // Set P13 as I2C SCL and enable pull-up function

    //  I2CIOS = 1;                 // Select P16,P02 as I2C communication pins
    //  P16F = 6 | (1<<7);          // Set P16 as I2C SDA and enable pull-up function
    //  P02F = 6 | (1<<7);          // Set P02 as I2C SCL and enable pull-up function
    // Choose one of the above two groups of ports
    /*****************************************************************************************/
        I2CCON = I2CE(1) | I2CIE(1) | STA(0) | STP(0)| CKHD(1) | AAK(1)| CBSE(0) | STFE(0);
        I2CADR = GCE(0)|(I2C_ADDR>>1);          // Set I2C slave address
        I2CCCR = 0x20;                          // Setting the I2C clock sample clock
        INT6EN = 1;                             // I2C interrupt on
        while(1)
        {
        }
    }
```

---

18 PWM

18.1 PWM Function Introduction

The JZ8FC003 series chips have up to 6 channels of PWM outputs, and the PWM period and duty cycle can be configured in any 16-bit range. Each PWM channel supports edge-aligned mode and center-symmetric mode. In addition, PWM also supports deadband control and complementary output, when set to complementary mode, the six PWM channels form three pairs of complementary channels. PWM supports software and hardware brake function, and can set PWM pause output. This feature of PWM is specially designed for brushless DC motor drive.

18.2 PWM Function Description

Each PWM channel has a dedicated 16-bit counter, the PWM period is set through register PWMDIV, and register PWMDUT corresponds to the duty cycle of PWM. PWM is enabled through register PWMEN, and each bit of register PWMEN corresponds to one channel of PWM. PWM module has a PWM data update register PWMUPD, when rewriting The PWM module has a PWM data update register PWMUPD, when rewriting registers PWMDIV, PWMDUT and PWMCKD, register PWMUPD must set the corresponding bit to update the data, when the data is refreshed PWMUPD corresponding bit is automatically cleared to 0. PWM can set PWM pin output inversion through the PWMTOG bit. PWM has multiple clock sources to choose from, the clock source is set in two PWM units, respectively : PWM0 and PWM1, PWM2 and PWM3, PWM4 and PWM5, that is, the clock source of each group of PWM is set jointly, and the clock source is selected through PWM0, PWM2, PWM4 corresponding to the control register PWMCON of PWMMCKS. In addition, the clock division frequency of each PWM can be set independently by PWMCKD. Each PWM can choose to select any pin as PWM output pin through register PWMPS, see the description in the register section for details.

- **Edge-aligned mode and center-aligned mode**

The PWM edge-aligned mode and center-aligned mode are selected by the PWMMS bit. when PWM is enabled, the PWM counter starts counting from 0, and the PWM pin outputs high (PWMTOG=0) when the count value is less than PWMDUT, and low (PWMTOG=0) when the count value is greater than or equal to PWMDUT. In edge-aligned mode, when the count value is equal to PWMDIV, one PWM cycle is completed and the PWM counter resets to 0 and starts counting in the next cycle. In center-aligned mode, when the count value reaches the PWMDIV value, the count direction is reversed and the decreasing count starts. During this phase, again, the PWM pin outputs high (PWMTOG=0) when the count value is less than the PWMDUT, and low (PWMTOG=0) when the count value is greater than or equal to the PWMDUT; when the count decreases to 0, one When the count decreases to 0, one PWM cycle is completed and the count starts again with the next cycle of accumulative counting.

When edge-aligned mode and center-aligned mode single PWM output waveform are shown in Figure 18-2-1 and Figure 18-2-2 (Note: All PWM waveform below satisfy the condition PWMDIV>PWMDUT>0). As shown in the figure, setting the same PWMDIV and PWMDUT values, one PWM cycle in center-aligned mode is twice as long as that in edge-aligned mode.

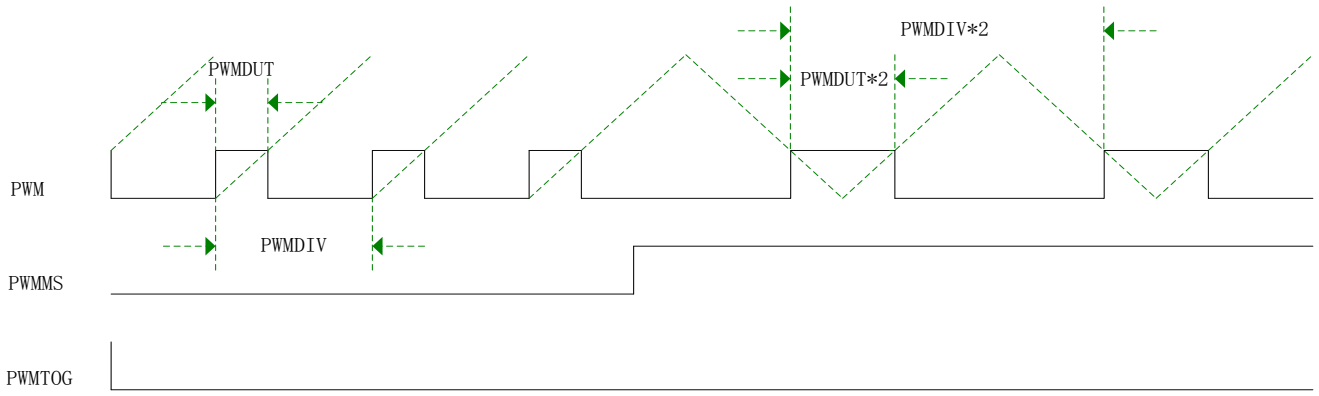


Figure 18-2-1 PWM Output Waveform When PWMTOG = 0

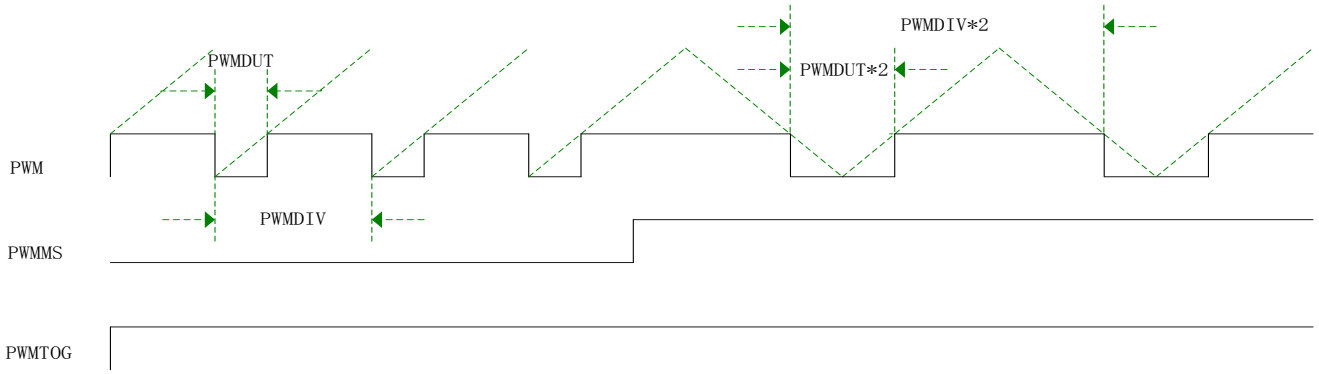


Figure 18-2-2 PWM Output Waveform When PWMTOG = 1

When PWMDIV=0, PWM pin output the PWM clock directly. If PWMCKD=0, PWM pin's output is the selected clock source. When PWMDIV is not 0 but PWMDUT=0, PWM pin outputs low level signal (PWMTOG＝0)；while PWMDUT>=PWMDIV>0, PWM pin outputs high level signal(PWMTOG＝0)

- **Complementary model**

In the complementary mode, 6 PWMs can form 3 pairs of complementary channels: PWM0 and PWM1, PWM2 and PWM3, PWM4 and PWM5. The complementary mode of PWM is set by the PWMMOD bit of PWMCON, the control register of PWM1, PWM3 and PWM5. In the complementary mode, PWM alignment, period, duty cycle, and prescaler clock are set by the registers corresponding to PWM0, PWM2, and PWM4, and only PWMTOG is still controlled independently by the corresponding registers of each channel. the PWM complementary mode schematic is shown in Figure 18-2-3

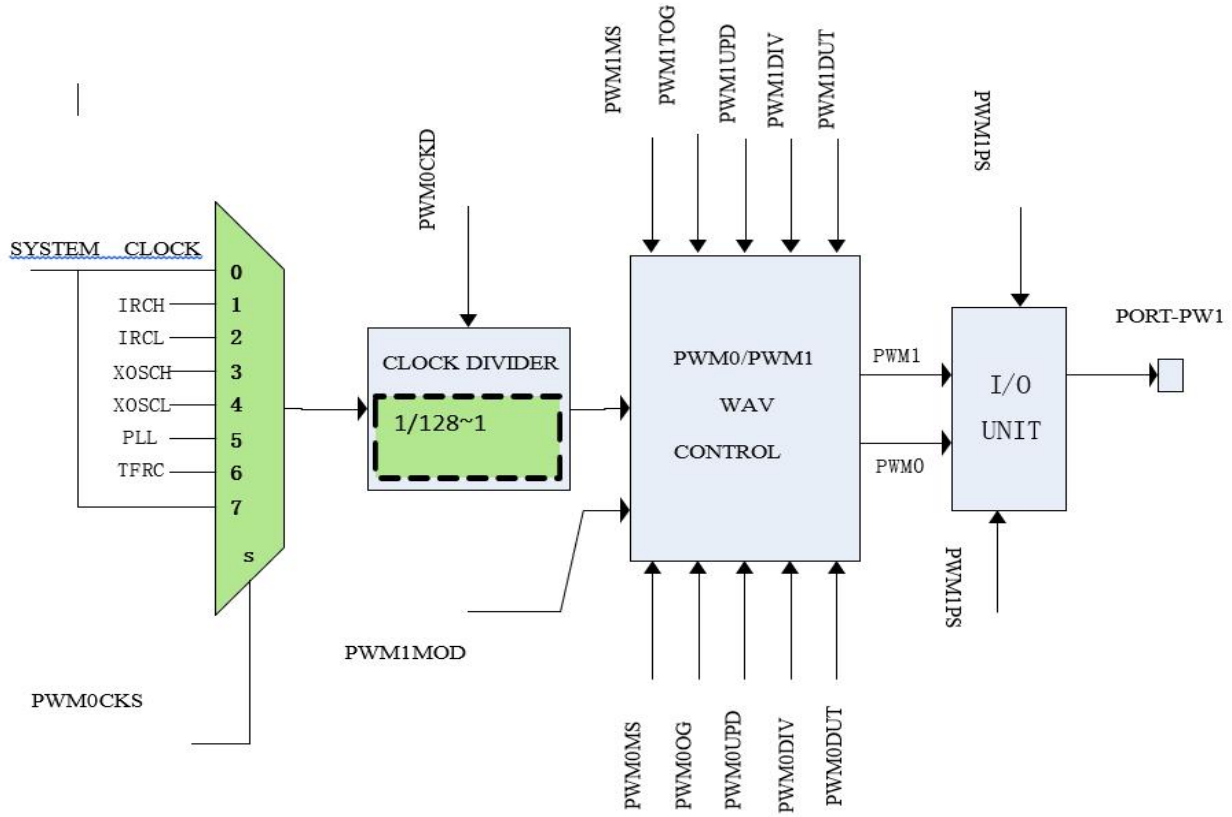


Figure 18-2-3 Schematic diagram of PWM0 and PWM1

The waveform of each PWM output group are complementary in phase, as shown in Figure 18-2-4 and Figure 18-2-5 (taking PWM0 and PWM1 as examples).

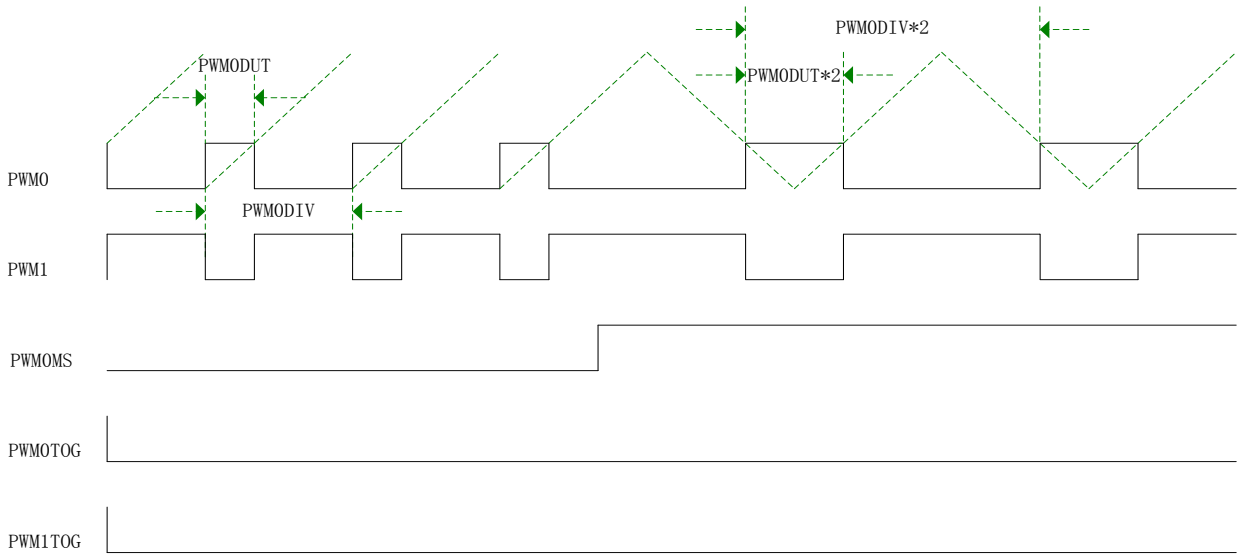


Figure 18-2-4 PWM0 and PWM1 output complementary waveform when PWMTOG = 0

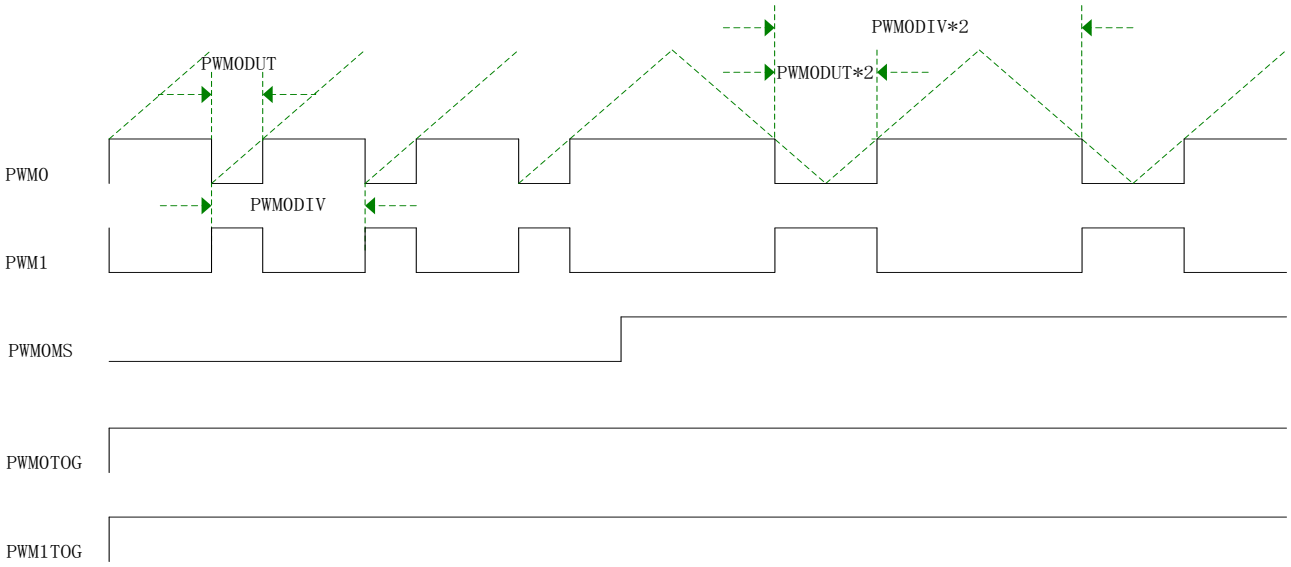


Figure 18-2-5 PWM0 and PWM1 Output Complementary Waveforms When PWMTOG = 1

- **Deadband control**

In bridge driver circuit, deadband control is a must to avoid the two half bridges are conducted at the same time.The deadbandcan be controlled by PWM1, PWM3,PWM5 and PWM7's corresponding register PWMDIV and PWMDUT. PWMDIV sets the deadband on the left and PWMDUT sets the deadband on the right. The deadband must meet the requirements below (take PWM0 and PWM1, for example)：

In edge fixed mode, PWMDIV1<PWMDUT0 and PWMDUT1<(PWMDIV0 – PWMDUT0);

In center fixed mode, PWMDIV1<(PWMDIV0 – PWMDUT0) x 2 or PWMDUT1<(PWMDIV0 – PWMDUT0) x 2.

Figure 18-2-6 shows the output waveform for deadband control (taking PWM0,PWM1 for example).

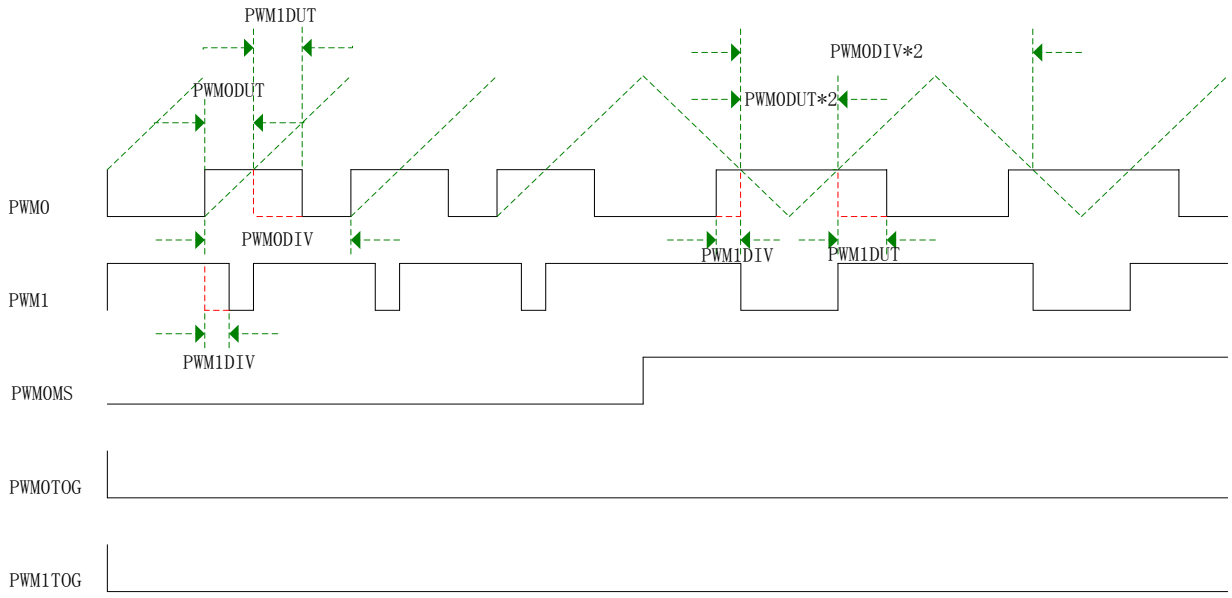


Figure 18-2-6 PWM0 and PWM1 Deadband Control Waveform When PWMTOG＝0

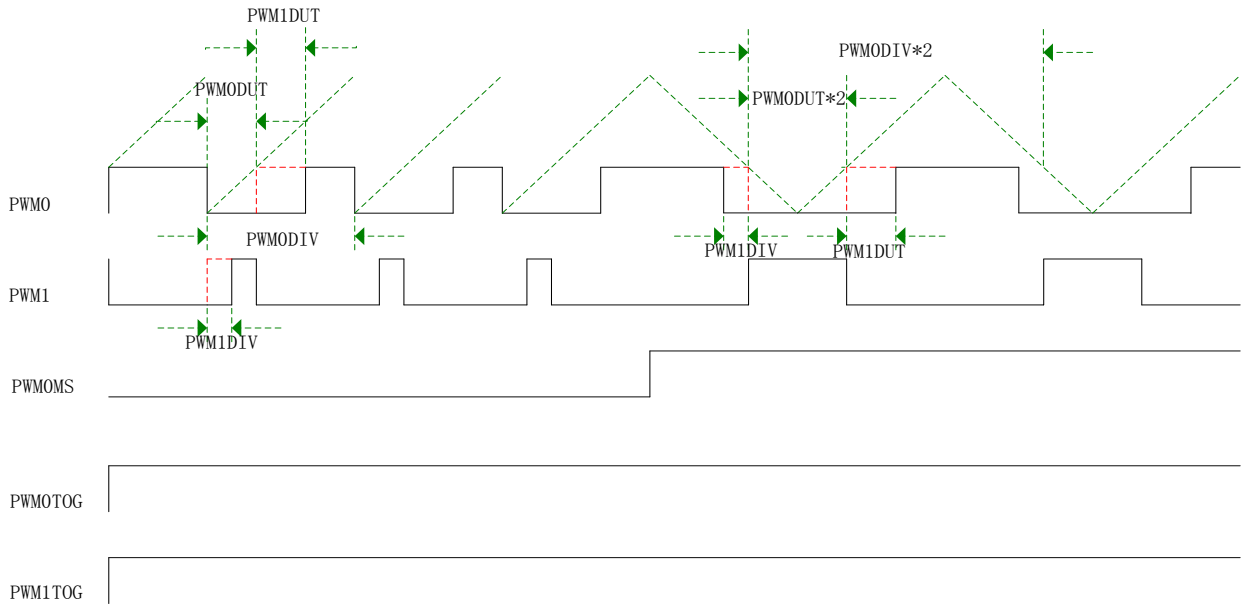


Figure 18-2-7 PWM0 and PWM1 Deadband Control Waveform When PWMTOG＝1

- **PWM Interrupt**

PWM interrupt in enabled by PWMTIE, PWMZIE, PWMPIE and PWMNIE in register PWMCON. PWMTIE enables the interrupt when PWM counter's count reaches the peak (which equals PWMDIV). PWMZIE enables the interrupt when PWM counter's count reaches the bottom(which equals 0). PWMNIE enables the interrupt when the output pin falling edge comes. PWMPIE enables the interrupt when the output pin rising edge comes. PWMTIE and PWMZIE's corresponding interrupts do not exist in edge fixed mode. Register PWMAIF, PWMBIF, PWMCIF and PWMDIF are the interrupt status registers for the 6 channel's interrupts. PWMxTIF, PWMxZIF, PWMxNIF, PWMxPIF correspond to PWMTIE, PWMZIE, PWMNIE, PWMPIE respectively.

In addition, an interrupt may be triggered only when the interrupt event has occurred for several times. The number of times can be set by register PWMCMX. For instance, if PWMCMX=3 and PWMPIE=1, then there will be a rising edge interrupt when PWM pin rising edge has come 4 times.

- **PWM Brake Function**

Each PWM can set the brake function independently, and the brake function includes software brake and hardware brake. The software brake is generated by software write register PWMSBC, PWMSBE0~PWMSBE5 correspond to PWM0~5 respectively, when PWMSBE0~PWMSBE5 write 1, the brake event is generated, the output level of PWM pin after brake is set by register PWMBD, PWMBD0~5 correspond to PWM0~5 respectively, when PWMBDx (x=0~5) is 0, the corresponding PWM pin after brake is output low, and vice versa. When PWMBDx(x=0~5) is 0, the corresponding PWM pin outputs low level after braking, and vice versa outputs high level. The hardware brake is triggered by the FB pin, enabled by setting PWMFB=1. When PWMFB=0, the brake is triggered by the FB pin high, and when PWMFB=1 is set, the brake is triggered by the FB pin low. The hardware brake interrupt is enabled by setting PWMFBIE=1, and PWMFBIF is the hardware brake interrupt flag. Each PWM can be selected by PWMFBSx (x=0~5 corresponding to

PWM0~5 respectively) whether the brake is triggered by FB level or not, and the state after the brake is the same as the software brake, which is also set by PWMBD.

- **PWM pause function**

The PWM pause function is set in groups of two: PWM0/1, PWM2/3, PWM4/5. The PWMHS register is the PWM pause function control register, PWMHS0 controls PWM0/1, PWMHS2 controls PWM2/3, and PWMHS4 controls PWM4/5. When PWMHSx (x=0/2/4) is set to 1, the corresponding two PWM outputs remain in their original state, and when PWMHSx is 0, the corresponding PWM returns to normal output.

- **PWM interrupt triggered ADC function**

The ADC conversion function can be enabled by register ADPWMTRIG, ADTRIG0~5 corresponds to PWM0~5. When ADTRIGx(x=0~5)=1, the corresponding PWM interrupt is generated, the ADC conversion is started. Note: ADC initialization and PWM interrupt initialization need to be set separately, ADTRIGx is only used as an enable switch, and PWM interrupt triggering ADC is equivalent to making the AST bit (ADCON[7]) set to 1.

18.3 PWM Register Description

Table 18-3-1 Register PWMEN

DAH	7	6	5	4	3	2	1	0
PWMEN	-	-	PWMEN[5:0]					
R/W	-	-	R/W					
Initial Value	-	-	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7~6	-	-
7~0	PWMEN	5~0 bit correspond to PWM channel 7~0's enable control, 1 enables it

Table 18-3-2 Register PWMUPD

DBH	7	6	5	4	3	2	1	0
PWMUPD			PWMUPD[5:0]					
R/W			R/W					
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7~0	PWMUPD	5~0 bit correspond to PWM channel 7~0's data refresh enable control, 1 enables it *Note:* *To refresh the channel's data(PWMDIV/PWMDUT/PWMCKD), set the corresponding position to 1 and the data will be refreshed when the PWM counter overflows. The corresponding bit will be cleared after the data refresh.*

Table 18-3-3 Register PWMCMX

DCH	7	6	5	4	3	2	1	0
PWMCMX	PWMCMX[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Note: PWMMX is a register with index, set INDEX=0~5 corresponding to PWMMX0~PWMCMX5 respectively								
Bit Number	Bit Symbol	Description						
7~0	PWMCMX	The interval number setting register of each PWM channel interrupt valid. The number of intervals=PWMCMX+1. For example, if INDEX=0 and PWMMX=7, then all interrupts of PWM0 will generate 8 interrupt events before the interrupt flag is set.						

Table 18-3-4 Register PWMCON

DDH	7	6	5	4	3	2	1	0
PWMCON①	PWMTIE	PWMZIE	PWMPIE	PWMNIE	PWMMS	PWMCKS[2:0]		
R/W	R/W	R/W	R/W	R/W	R/W	R/W		
Initial Value	0	0	0	0	0	0	0	0
PWMCON②	PWMTIE	PWMZIE	PWMPIE	PWMNIE	PWMMS	-	-	PWMMOD
R/W	R/W	R/W	R/W	R/W	R/W	-	-	R/W
Initial Value	0	0	0	0	0	-	-	0

Bit Number	Bit Symbol	Description
Note: *1. PWMCON① is the channel control for PWM0/PWM2/PWM4* *PWMCON② is the channel control for PWM1/PWM3/PWM5* *2. PWMCON is also register with index, INDEX=0~5 correspond to PWMCON0~PWMCON5 respectively*		
7	PWMTIE	PWM counter peak interrupt enable control, 1 enables it
6	PWMZIE	PWM counter bottom interrupt enable control, 1 enables it
5	PWMPIE	PWM rising edge interrupt enable control, 1 enables it
4	PWMNIE	PWM falling edge interrupt enable control, 1 enables it
3	PWMMS	PWM mode selection 0: edge fixed 1: center fixed
2~0	PWMCKS	PWM working clock selection 001：IRCH 010：IRCL 011：XOSCH/CLK_IN Others: System clock

		Note： *PWM0/PWM1 are set by PWMCKS0；* *PWM2/PWM3 are set by PWMCKS2；* *PWM4/PWM5 are set by PWMCKS4；*
0	PWMMOD	Complementary mode enable control, 1 enables it *Note：* *When PWMMOD1=1, PWM0 and PWM1 enter the complementary mode* *When PWMMOD3=1, PWM2 and PWM3 enter the complementary mode* *When PWMMOD5=1, PWM4 and PWM5 enter the complementary mode*

Table 18-3-5 Register PWMCFG

DEH	7	6	5	4	3	2	1	0
PWMCFG	PWMTOG	PWMCKD[6:0]						
R/W	R/W	R/W						
Initial Value	0	0	0	0	0	0	0	0
Note：PWMCFG is register with index, INDEX=0~5 corresponds to PWMCFG0~PWMCFG5								
Bit Number	Bit Symbol	Description						
7	PWMTOG	PWM Output Inverse Enable Register ,1 enables it						
6~0	PWMCKD	PWM Operating Clock Prescaler Configuration Register 0000000: no division 0000001: frequency divided by 2 0000010: frequency divided by 3 ..... 1111110: frequency divided by 127 1111111: frequency divided by 128						

Table 18-3-6 Registers PWMDIVL、PWMDIVH

DFH	7	6	5	4	3	2	1	0
PWMDIVL	PWMDIV[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
D1H	7	6	5	4	3	2	1	0
PWMDIVH	PWMDIV[15:8]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Note：PWMDIV is register with index, INDEX=0~5 corresponds to PWMDIV0~PWMDIV5								
Bit Number	Bit Symbol	Description						
15~0	PWMDIV	PWM cycle configuration PWMDIV1/PWMDIV3/PWMDIV5/PWMDIV7 are for different use in						

		complementary mode, please refer to register PWMDUT description

Table 18-3-7 Registers PWMDUTL, PWMDUTH

D2H	7	6	5	4	3	2	1	0
PWMDUTL	PWMDUT[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
D3H	7	6	5	4	3	2	1	0
PWMDUTH	PWMDUT[15:8]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Note： PWMDUT is register with index, INDEX=0~5 corresponds to PWMDUT0~PWMDUT5

Bit Number	Bit Symbol	Description
15~0	PWMDUT	PWM duty cycle setting In complementary mode, , PWMDUT1/PWMDUT3/PWMDUT5/PWMDUT7 are for different use as the table below：

PWMDIV1	Controls width of the deadband on the left for PWM0/PWM1
PWMDUT1	Controls width of the deadband on the right for PWM0/PWM1
PWMDIV3	Controls width of the deadband on the left for PWM2/PWM3
PWMDUT3	Controls width of the deadband on the right for PWM2/PWM3
PWMDIV5	Controls width of the deadband on the left for PWM4/PWM5
PWMDUT5	Controls width of the deadband on the right for PWM4/PWM5 right of PWM4/PWM5

Table 18-3-8 Register PWMAIF

D4H	7	6	5	4	3	2	1	0
PWMAIF	PWM1TIF	PWM1ZIF	PWM1PIF	PWM1NIF	PWM0TIF	PWM0ZIF	PWM0PIF	PWM0NIF
R/W	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7	PWM1TIF	PWM1 counter peak interrupt flag, cleared when 1 is written to it
6	PWM1ZIF	PWM1 counter bottom interrupt flag, cleared when 1 is written to it
5	PWM1PIF	PWM1 rising edge interrupt flag, cleared when 1 is written to it
4	PWM1NIF	PWM1 falling edge interrupt flag, cleared when 1 is written to it

3	PWM0TIF	PWM0 counter peak interrupt flag, cleared when 1 is written to it
2	PWM0ZIF	PWM0 counter bottom interrupt flag, cleared when 1 is written to it
1	PWM0PIF	PWM0 rising edge interrupt flag, cleared when 1 is written to it
0	PWM0NIF	PWM0 falling edge interrupt flag, cleared when 1 is written to it

Table 18-3-9 Register PWMBIF

D5H	7	6	5	4	3	2	1	0
PWMBIF	PWM3TIF	PWM3ZIF	PWM3PIF	PWM3NIF	PWM2TIF	PWM2ZIF	PWM2PIF	PWM2NIF
R/W	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7	PWM3TIF	PWM3 counter peak interrupt flag, cleared when 1 is written to it
6	PWM3ZIF	PWM3 counter bottom interrupt flag, cleared when 1 is written to it
5	PWM3PIF	PWM3 rising edge interrupt flag, cleared when 1 is written to it
4	PWM3NIF	PWM3 falling edge interrupt flag, cleared when 1 is written to it
3	PWM2TIF	PWM2 counter peak interrupt flag, cleared when 1 is written to it
2	PWM2ZIF	PWM2 counter bottom interrupt flag, cleared when 1 is written to it
1	PWM2PIF	PWM2 rising edge interrupt flag, cleared when 1 is written to it
0	PWM2NIF	PWM2 falling edge interrupt flag, cleared when 1 is written to it

Table 18-3-10 Register PWMCIF

D6H	7	6	5	4	3	2	1	0
PWMCIF	PWM5TIF	PWM5ZIF	PWM5PIF	PWM5NIF	PWM4TIF	PWM4ZIF	PWM4PIF	PWM4NIF
R/W	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7	PWM5TIF	PWM5 counter peak interrupt flag, cleared when 1 is written to it
6	PWM5ZIF	PWM5 counter bottom interrupt flag, cleared when 1 is written to it
5	PWM5PIF	PWM5 rising edge interrupt flag, cleared when 1 is written to it
4	PWM5NIF	PWM5 falling edge interrupt flag, cleared when 1 is written to it
3	PWM4TIF	PWM4 counter peak interrupt flag, cleared when 1 is written to it
2	PWM4ZIF	PWM4 counter bottom interrupt flag, cleared when 1 is written to it
1	PWM4PIF	PWM4 rising edge interrupt flag, cleared when 1 is written to it
0	PWM4NIF	PWM4 falling edge interrupt flag, cleared when 1 is written to it

Table 18-3-11 Register PWMPS

8098H	7	6	5	4	3	2	1	0

PWMPS	-	-	-	PWMPS[4:0]				
R/W	-	-	-	R/W				
Initial Value	-	-	-	?	?	?	?	?

Note：

1. PWMPS is the index register, which is pointed by INDEX0~5 to PWMPS0~5 respectively to control PWM0~5 channels.

2. PWMPS0[4:0] initial value of 01001, select P1.2.

PWMPS1[4:0] initial value of 01001, select P1.1.

PWMPS2[4:0] initial value of 01000, select P1.0.

PWMPS3[4:0] initial value of 00000, select P0.0.

PWMPS4[4:0] initial value of 00001, select P0.1.

The initial value of PWMPS5[4:0] is 00011, and P0.3 is selected.

3. When PWM0~5 are turned on at the same time, PWMPS0~5 must select different pins respectively, otherwise it will lead to errors; If more than two PWM are enabled and a pin is selected at the same time, the selected pin will output 0.

Bit Number	Bit Symbol	Description
7~5	-	-
4~0	PWMPS	PWM Pin Select Bit Field 00000: Select P0.0 00001: Select P0.1 00010: Select P0.2 00011: Select P0.3 00100: Select P0.4 00101: Select P0.5 00110: Select P0.6 00111: Select P0.7 01000: Select P1.0 01001: Select P1.1 01010: Select P1.2 01011: Select P1.3 01100: Select P1.4 01101: Select P1.5 01110: Select P1.6 01111: Select P1.7 10000: Select P2.0 Others: Select P3.0

Table 18-3-12 Register PWMHS

8099H	7	6	5	4	3	2	1	0

PWMHS	-	-	-	PWMHS4	-	PWMHS2	-	PWMHS0
R/W	-	-	-	R/W	-	R/W	-	R/W
Initial Value	-	-	-	0	-	0	-	0

Bit Number	Bit Symbol	Description
7~5	-	-
4	PWMHS4	PWM4/PWM5 hold enable control, 1 enables it
3	-	-
2	PWMHS2	PWM2/PWM3 hold enable control , 1 enables it
1	-	-
0	PWMHS0	PWM0/PWM1 hold enable control , 1 enables it

Table 18-3-13 Register PWMFBC

809AH	7	6	5	4	3	2	1	0
PWMFBC	PWMFBIF	-	PWMFBIE	-	-	-	PWMFBL	PWMFBE
R/W	R	-	R/W	-	-	-	R/W	R/W
Initial Value	0	-	0	-	-	-	0	0

Bit Number	Bit Symbol	Description
7	PWMFBIF	PWM Fault Pin brake interrupt flag,1 enables it , write 1 to clear 0
6	-	-
5	PWMFBIE	PWM Fault Pin Brake Interrupt enable control,1 enables it
4~2	-	-
5	PWMFBL	PWM Fault Pin Brake Level Selection 0: Select high level brake 1: Select low level brake
0	PWMFBE	PWM Fault Pin Brake Enable control,1 enables it

Table 18-3-14 Register PWMFBS

809BH	7	6	5	4	3	2	1	0
PWMFBS	-	-	PWMFBS5	PWMFBS4	PWMFBS3	PWMFBS2	PWMFBS1	PWMFBS0
R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	-	-	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7~6	-	-
5	PWMFBS5	PWM5 Fault Pin brake selector Bit 0: PWM5 is not affected by the Fault Pin signal 1:When the Fault Pin signal is valid, PWM5 will be in the braking state
4	PWMFBS4	PWM4 Fault Pin brake selector Bit 0: PWM4 is not affected by the Fault Pin signal 1: When the Fault Pin signal is valid, PWM4 will be in the braking state

3	PWMFBS3	PWM3 Fault Pin brake selector Bit 0: PWM3 is not affected by the Fault Pin signal 1: When the Fault Pin signal is valid, PWM3 will be in the braking state
2	PWMFBS2	PWM2 Fault Pin brake selector Bit 0: PWM2 is not affected by the Fault Pin signal 1: PWM2 will be in braking state when Fault Pin signal is valid
1	PWMFBS1	PWM1 Fault Pin Brake Selector Bit 0: PWM1 is not affected by the Fault Pin signal 1: PWM1 will be in the braking state when the Fault Pin signal is valid
0	PWMFBS0	PWM0 Fault Pin Brake Select Bit 0: PWM0 is not affected by the Fault Pin signal 1: When the Fault Pin signal is valid, PWM0 will be in the braking state

Table 18-3-15 Register PWMSBC

809CH	7	6	5	4	3	2	1	0
PWMSBC	-	-	PWMSBE5	PWMSBE4	PWMSBE3	PWMSBE2	PWMSBE1	PWMSBE0
R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	-	-	0	0	0	0	0	0

位编号	位符号	说明
7~6	-	-
5	PWMSBE5	PWM5 software brake enable , 1 brake
4	PWMSBE4	PWM4 software brake enable , 1 brake
3	PWMSBE3	PWM3 software brake enable , 1 brake
2	PWMSBE2	PWM2software brake enable, 1 brake
1	PWMSBE1	PWM1 software brake enable, 1 brake
0	PWMSBE0	PWM0 software brake enable, 1 brake

Table 18-3-16 Register PWMBD

809DH	7	6	5	4	3	2	1	0
PWMBD	-	-	PWMBD5	PWMBD4	PWMBD3	PWMBD2	PWMBD1	PWMBD0
R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	-	-	1	0	1	0	1	0

Note:

Software and hardware brakes, the levels are controlled by this register.

Bit Number	Bit Symbol	Description
7~6	-	-
5	PWMBD5	PWM5 Brake level selection 0: Hold low when brakes are applied 1:Hold high when brakes are applied
4	PWMBD4	PWM4 Brake level selection 0: Hold low when brakes are applied

		1: Hold high when brakes are applied
3	PWMBD3	PWM3 Brake level selection 0: Hold low when brakes are applied 1: Hold high when brakes are applied
2	PWMBD2	PWM2 Brake level selection 0: Hold low when brakes are applied 1: Hold high when brakes are applied
1	PWMBD1	PWM1 Brake level selection 0: Hold low when brakes are applied 1: Hold high when brakes are applied
0	PWMFBD0	PWM0 Brake level selection 0: Hold low when brakes are applied 1: Hold high when brakes are applied

Table 18-3-17 Register ADPWMTRIG

808eH	7	6	5	4	3	2	1	0
PWMHS	-	-	ADTRIG5	ADTRIG4	ADTRIG3	ADTRIG2	ADTRIG1	ADTRIG0
R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	-	-	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7~6	-	-
5	ADTRIG5	PWM5's interrupt triggers ADC enable, 1 enables it
4	ADTRIG4	PWM4's interrupt triggers ADC enable,1 enables it
3	ADTRIG3	PWM3's interrupt triggers ADC enable, 1 enables it
2	ADTRIG2	PWM2's interrupt triggers ADC enable, 1 enables it
1	ADTRIG1	PWM1's interrupt triggers ADC enable,1 enables it
0	ADTRIG0	PWM0's interrupt triggers ADC enable, 1 enables it

18.4 PWM Control Example

Single channel PWM output

For instance , PWM0 outputs 30KHz clock with duty cycle 50%, and generate PWM rising edge interrupt:

```
---------------------------------------------------------------------------------------------
//PWMCON
#define TIE(N)          (N<<7)
#define ZIE(N)          (N<<6)
#define PIE(N)          (N<<5)
#define NIE(N)          (N<<4)
#define MS(N)           (N<<3)
#define MOD(N)          (N<<0)
//PWMCFG
#define TOG(N)          (N<<7)
#define PWM_CH0         0
#define IHCKE           (1<<6)
#define CKS_IH          (1<<0)

#define PWMDIV_V                (32000000/30000)        // When the PWM clock is other clock frequency, you need to modify the parameters accordingly
#define PWMDUT_V                (PWMDIV_V/2)            // Duty cycle of 50%
void PWM_init(void)
{
    CKCON |= IHCKE;                                     // Enable the IRCH clock
    INDEX = PWM_CH0;                                    // Set the INDEX value corresponding to PWM0
    PWMCON = TIE(0) | ZIE(0) | PIE(0) | NIE(0) | MS(0) | CKS_IH ;  // Set PWM clock source to IRCH
    PWMCFG = TOG(0) | 0;
    PWMPS = 0;                                          // Set P00 as PWM0 output pin
    P00F    = 5;
    PWMDIVH     =     (unsigned char)(PWMDIV_V>>8);
    PWMDIVL     =     (unsigned char)(PWMDIV_V);
    PWMDUTH     =     (unsigned char)(PWMDUT_V>>8);
    PWMDUTL     =      (unsigned char)(PWMDUT_V);
    PWMUPD = (1<<PWM_CH0);                              // PWMDIV, PWMDUT update enable
    while(PWMUPD);                                      // Waiting for the update to complete
    PWMEN = (1<<PWM_CH0);                               // PWM0 enable

    INDEX = PWM_CH0;                                    // Set the INDEX value corresponding to PWM0
    PWMCON |=TIE(0) | ZIE(0) | PIE(1) | NIE(0);         // Enabling rising edge interrupts
    PWMCMAX = 0;                                        // Set each rising edge to trigger
    INT9EN = 1;
}
void INT9_ISR(void) interrupt 14    using 1
```

```
{
    if(PWMAIF & PIF0)           // PWM rising edge interrupt
    {
        PWMAIF = PIF0;          // Clear the interrupt flag
    }
}
```

PWM output clock

For instance, PWM0 outputs IRCH and the program is as follows.

```
void PWM_init(void)
{
    CKCON |= IHCKE;                     // Enable the IRCH clock
INDEX = PWM_CH0;                                    //Set the INDEX value corresponding to PWM0
    PWMCON = TIE(0) | ZIE(0) | PIE(0) | NIE(0) | MS(0) | CKS_IH ; // Set the INDEX value corresponding to PWM0
    PWMCFG = TOG(0) | 0;
    PWMPS = 0;                          // Set P00 as PWM0 output pin
    P00F    = 5;
    PWMDIVH     = 0;                    // PWMDUT are set to 0 to output the clock directly
    PWMDIVL     = 0;
    PWMDUTH     = 0;
    PWMDUTL     = 0;
    PWMUPD = (1<<PWM_CH0);              // PWMDIV, PWMDUT update enable
    while(PWMUPD);                      // Waiting for the update to complete
    PWMEN = (1<<PWM_CH0);               // PWM0 enable
}
```

---

19 Analog/Digital Converter (ADC)

19.1 Function Introduction

Analog/digital converter is a 12-bit successive approximation(SAR) ADC, with at most 12 input channels. The clock source for ADC is the system clock with frequency division configurable. There are ADC multiple reference voltages for ADC. When internal voltage is selected as the reference voltage, it can be used to test the power supply voltage for the chip and there will be correction to ensure the chip's consistency as well. There is also a compare mode for it with threshold configurable. Once it goes beyond the threshold, a corresponding interrupt occurs. the ADC module has built-in op-amp, and the op-amp gain can be set (can only be reduced). Alternatively, the ADC can be used in combination with op-amp A. The ADC can directly detect the output of op-amp A.

19.2 Main Features

- 12 bit resolution
- 12 input channels at most
- Supports ADC interrupt
- ADC clock frequency division configurable
- Alternate reference voltage：internal reference voltage, VDD, external reference voltage
- Automatic data correction supported when internal reference voltage is selected
- Support configurable comparator mode
- Support the detection signal through the op-amp reduction and then conversion, reduction multiplier can be selected
- ADC can directly detect op-amp A output
- Input voltage range：VSS<=VIN<=VDD。

19.3 Block Diagram

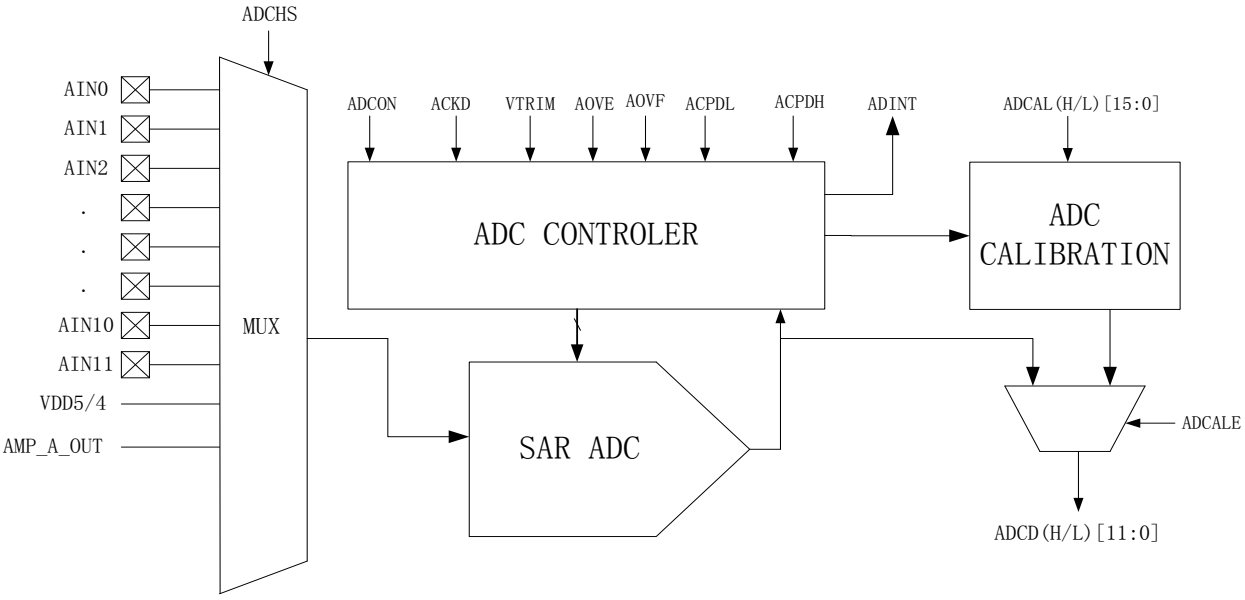


Figure 19-3-1 ADC Architecture

19.4 Function Introduction

ADC can be enabled by AST. When AST=1, the input voltage selected by ADCHS will be analog/digital converted. The clock for ADC is the system clock with frequency division set by ACKD beforehand. When ADC clock is constant, the time for single conversion is set by HTME. The conversion time is (13+2^HTME) ADC clock cycle periods. 12-bit A/D will be stored in register ADCDH and ADCDL after the conversion. AST will be cleared automatically 2.5 clock cycles later. The interrupt flag ADCIF will be set to 1 at the same time. If ADC interrupt is enabled then, ADC interrupt occurs. The shortest ADC conversion time is 0.5us. Figure 19-4-1 is the sequence diagram for ADC conversion.

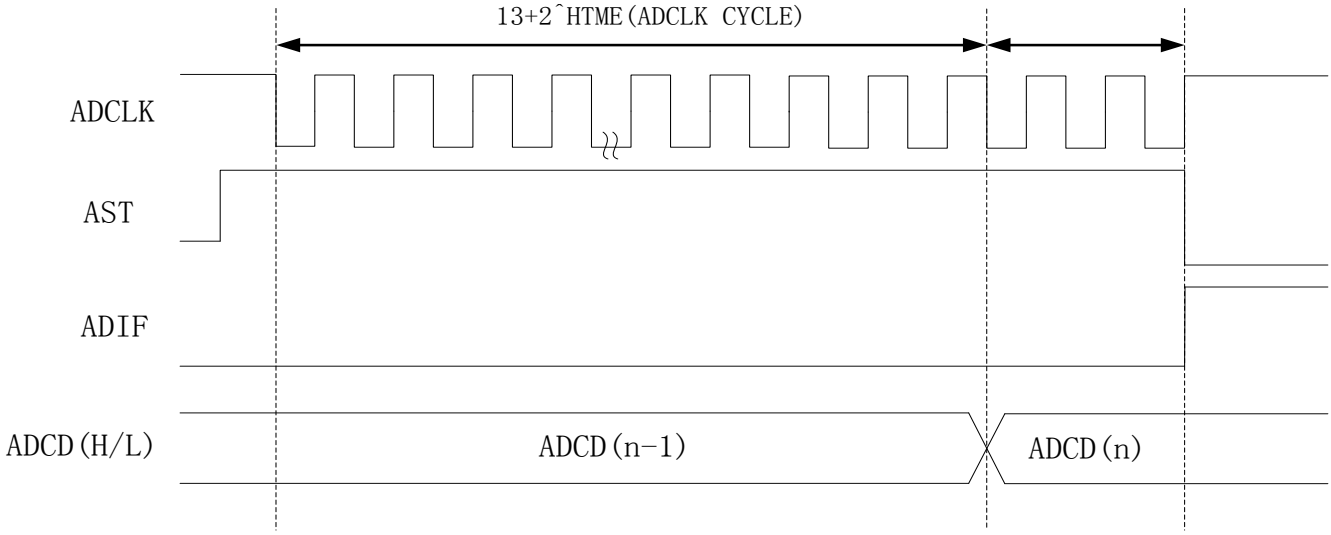


Figure 19-4-1 ADC Sequence Diagram

- **Compare Mode**

This mode is enabled by AOVE. When AOVE=1, ADC conversion result ADCD will be compared to threshold ADCPDL and ADCPDH after the conversion. When ADCD exceeds the threshold range, the compare interrupt flag will be set to 1. If the ADC interrupt is enabled then, the interrupt occurs.

- **ADC Data Calibration**

When internal voltage(1.5V) is selected as the reference voltage, due to the discreteness of the chips, the internal voltage in each chip can not be exactly the same which induces different ADC conversion results consequentially. Thus, it is necessary to correct the AD value after the conversion. The internal voltage will be tested and a correction value will be obtained when chips leave factory. When the chip's powered on, the correction value will be loaded into register ADCALL and ADCALH. The accurate AD value will be obtained by calculation according to the correction value. The final accurate result for AD will be stored in register ADCD. The function can be enabled by ADCALE. Users only need to set ADCALE=1 and the correction will be done automatically.

- **ADC built-in op-amp**

The ADC has a built-in op-amp and the gain can be set via register ADOPC. Note that the built-in op-amp can only reduce the signal, not amplify it.

19.5 Register Description

Table 19-5-1 Register ADCON

B9H	7	6	5	4	3	2	1	0
ADCON	AST	ADIE	ADCIF	HTME			VSEL[1:0]	
R/W	R/W	R/W	R/W	R/W			R/W	
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7	AST	ADC conversion enable control, the conversion starts when 1 is written to it, the hardware will clear it automatically after the conversion
6	ADIE	ADC interrupt enable control, 1 enables it
5	ADCIF	ADC interrupt flag, write 1 to clear 0
4~2	HTME	The number of sampling periods is 2 power HTME
1~0	VSEL	ADC reference voltage selection 00: internal 1.5V(INNER_VREF) as reference voltage 01: external VDD 10: external VREF 11: internal 1.5V(INNER_VREF) as reference voltage *Note: When the reference voltage is selected as external VREF, the voltage of VREF must be greater than 1.1V.*

Table 19-5-2 Register ADCFGL

BAH	7	6	5	4	3	2	1	0
ADCFGL	ACKD			ADCALE	ADCHS			
R/W	R/W			R/W	R/W			
Initial Value	0	0	0	1	0	0	0	0

Bit Number	Bit Symbol	Description
7~5	ACKD	ADC clock frequency division setting 000: no division 001: frequency divided by 2 010: frequency divided by 4 … 111: frequency divided by 14
4	ADCALE	ADC calibration enable control, 1 enables it Valid only when the internal 1.5V is selected as the reference voltage. When ADCALE=1, ADC conversion result will be calibrated according to
3~0	ADCHS	ADC channel enable selection 0000： disable the channels 0001： enable channel AD_CH[0](P12) 0010： enable channel AD_CH[1](P11) 0011： enable channel AD_CH[2](P01) 0100： enable channel AD_CH[3](P03) 0101： enable channel AD_CH[4](P04) 0110： enable channel AD_CH[5](P05) 0111： enable channel AD_CH[6](P06) 1000： enable channel AD_CH[7](P07) 1001： enable channel AD_CH[7](P30) 1010： enable channel AD_CH[7](P17) 1011： enable channel AD_CH[7](P16) 1100： enable channel AD_CH[7](P15) 1101： Detect 1/4 enable of VDD 1110： Detect AMP_A_OUT Others： disable the channels

Table 19-5-3 Register ADCFGH

BBH	7	6	5	4	3	2	1	0
ADCFGH	AOVF	AOVE	VTRIM					
R/W	R/W	R/W	R/W					
Initial value	0	0	1	0	0	0	1	1

Bit number	Bit Symbol	instruction
7	AOVF	overflow flag in compare mode

6	AOVE	Compare mode enable control, 1 enables it
5~0	VTRIM	Internal 1.5V reference voltage correction register, with accuracy ±1mV

Table 19-5-4 Register ADCAL

8088H	7	6	5	4	3	2	1	0
ADCALL	ADCAL[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
8089H	7	6	5	4	3	2	1	0
ADCALH	ADCAL[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	instruction
15~0	ADCAL	ADC calibration register, valid only when ADCALE=1 and the internal 1.5V is selected as reference voltage. When it is valid, the ADC output is： ADCDL=(ADC conversion result*ADCAL)/32768 Note: ADCAL is automatically loaded by the system after power-on, and users are not allowed to modify it.

Table 19-5-5 Register ADCPDL

808AH	7	6	5	4	3	2	1	0
ADCPDLL	ADCPDL[3:0]				-	-	-	-
R/W	R/W				-	-	-	-
Initial value	0	0	0	0	0	0	0	0
808BH	7	6	5	4	3	2	1	0
ADCPDLH	ADCPDL[11:4]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	instruction
15~0	ADCPDL	Threshold lower limit setting register in compare mode

Table 19-5-6 Register ADCPDH

808CH	7	6	5	4	3	2	1	0

ADCPDHL	ADCPDH[3:0]				-	-	-	-
R/W	R/W				-	-	-	-
Initial value	0	0	0	0	0	0	0	0
808DH	7	6	5	4	3	2	1	0
ADCPDHH	ADCPDH[11:4]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	instruction
15~0	ADCPDH	Threshold upper limit setting register in compare mode

Table 19-5-7 Register ADCD

BCH	7	6	5	4	3	2	1	0
ADCDL	ADCDL[3:0]				-			
R/W	R/W				-			
Initial value	0	0	0	0	-	-	-	-
BDH	7	6	5	4	3	2	1	0
ADCDH	ADCDH[11:4]							
R/W	R/W							

Initial value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
11~0	ADCD	ADC conversion result

Table 19-5-8 Register ADOPC

808FH	7	6	5	4	3	2	1	0
ADOPC	-	-	-	-	-	-	GAIN[1:0]	
R/W	-	-	-	-	-	-	R/W	
Initial Value	-	-	-	-	-	-	0	0

Bit Number	Bit Symbol	Description
7~2	-	-
1~0	GAIN	Op-amp scaling multiplier selection 00：No scaling 01：4 times smaller 10：3 times smaller 11：2 times smaller

19.6 ADC Control Example

For instance, external VDD is selected as the ADC reference voltage, channel 0 selected, ADC interrupt enabled, the program is like：

```
---------------------------------------------------------------------------------------------
//ADCON Definition
#define AST(N)          (N<<7)          // ADC start, AST=0 end
#define ADIE(N)         (N<<6)          // Interrupt Enable
#define ADIF            (1<<5)          // Interruption flags
#define HTME(N)         (N<<2)          //N=0-7         // Sampling time setting, clock period to the power of 2 to the HTME
#define VSEL(N)         (N)             //N=0-3         // Selecting the reference voltage 0 - internal 1-VDD 2-external
//ADCFGL Definition
#define ACKD(N)         (N<<5)          //N=0-7         // ADC clock division Frequency division multiplier = (ACKD+1)
#define ADCALE(N)       (N<<4)          // ADC correction, internal reference voltage selected to be valid
#define ADCHS(N)        (N)             //N=0-15        // ADC channel selection, 1-13 corresponds to channels 0-12
void ADC_init(void)
{
    P11F = 3; // Set P11 to ADC pin function
    ADCON = AST(0) | ADIE(1) | HTME(7) | VSEL(1); // Set ADC reference voltage to VDD
    ADCFGL = ACKD(7) | ADCALE(1) | ADCHS(2);    // Select ADC1 channel
    ADCON |= AST(1);                    // enable AD conversion
    INT2EN = 1;                         // enable INT2 interrupt
}

void ADC_ISR (void) interrupt 7
{
    unsigned int    AD_Value;
    if(ADCON & ADIF)
    {
        ADCON |= ADIF;                      // Clear interrupt flag
        AD_Value = ADCDH*256 + ADCDL;       // Read AD value
        AD_Value >>= 4;
        ADCON |= AST(1);                    // enable the next AD conversion
    }
}
---------------------------------------------------------------------------------------------
```

20 Operational Amplifier (AMP)

20.1 Function Introduction

JZ8FC003 series chips have two built-in operational amplifiers, among which, operational amplifier A is a general-purpose operational amplifier, and operational amplifier B is a dedicated communication decoding operational amplifier for wireless charging.

Op amp A has a built-in correction mechanism. After factory correction, the offset voltage is less than 0.5mV under full temperature conditions, and the correction parameters are automatically loaded by the system when the system is powered on.

The output terminal of the operational amplifier B is connected to the wireless charging and decoding module inside the chip as the input signal of the decoding module.

20.2 Register Description

Table 20-2-1 Register DCDS

80D1H	7	6	5	4	3	2	1	0
DCDS	AMP_A_EN	-	AMP_B_EN	AMP_B_SM_EN	AMP_B_OUT_EN	-	-	-
R/W	R/W	-	R/W	R/W	R/W	-	-	-
Initial Value	0	-	0	1	0	-	-	-

Bit Number	Bit Symbol	Description
7	AMP_A_EN	Enable signal of AMP_A, 1 enable, 0 disable
6	-	-
5	AMP_B_EN	Enable signal of AMP_B, 1 enable, 0 disable
4	AMP_B_SM_EN	The output of AMP_B is Schmitt control, 1 is Schmitt output, 0 is inverter output
3	AMP_B_T_EN	AMP_B output enable, when it is 1, AMP_B output is connected to P01 pin, when it is 0, no output
2~0	-	-

Table 20-2-2 Register AMPOS

80B9H	7	6	5	4	3	2	1	0
AMPOS			AOTDIR	AOTDAT				
R/W			R/W	R/W	R/W	R/W	R/W	R/W
Initial Value			-	-	-	-	-	-

Note: This register value is automatically loaded by the system at power-on, and the user is not allowed to modify it.

Bit Number	Bit Symbol	Description
7~6		
5	AOTDIR	Adjust the offset voltage in the positive direction when it is 1, and adjust it in the negative direction when it is 0
4~0	AOTDAT	Offset voltage adjustment range=0.4* AOTDAT mV

21 Buzzer (BUZZER)

21.1 Function Description

JZ8FC003 series chips have a built-in BUZZER output.

The clock source of BUZZER is fixed as the system clock. When BZE=1, BUZZER is enabled, the period of BUZZER is set by BZDIV, and the duty cycle is set by BZDUT. After BUZZER is enabled, the BZH bit can set the BUZZER output fixed level, and the level value is determined by the BZD bit.

21.2 Register Description

Table 21-2-1 Register BZCON

80C0H	7	6	5	4	3	2	1	0
BZCON	-	-	-	-	-	BZD	BZH	BZE
R/W	-	-	-	-	-	R/W	R/W	R/W
Initial Value	-	-	-	-	-	0	0	0

Bit Number	Bit Symbol	Description
7~3	-	-
2	BZD	BUZZER output level value, valid only when BZH=1
1	BZH	BUZZER output fixed level enable, 1 enable, BUZZER output value of BZD
0	BZE	BUZZER enable bit, 1 enable *Note：* *For BUZZER to be enabled, the corresponding pin must also be selected as the BUZZER function.*

Table 21-2-2 Register BZDIV

80C1H	7	6	5	4	3	2	1	0
BZDIVL	BZDIV[7:0]							
R/W	R/W							
Initial Value	0	1	1	1	1	1	1	1
80C2H	7	6	5	4	3	2	1	0
BZDIVH	BZDIV[15:8]							
R/W	R/W							
Initial Value	0	0	1	1	1	1	1	0

Bit Number	Bit Symbol	Description

15~0	BZDIV	BUZZER Frequency Configuration Register $F_{buzzer} = F_{sys}/(BZDIV+1)$ *Note：* *F_{buzzer} is the output frequency of BUZZER and F_{sys} is the frequency of the system clock.*

Table 21-2-3 Register BZDUT

80C3H	7	6	5	4	3	2	1	0
BZDUTL	BZDUT[7:0]							
R/W	R/W							
Initial Value	0	0	1	1	1	1	1	1
80C4H	7	6	5	4	3	2	1	0
BZDUTH	BZDUT[15:8]							
R/W	R/W							
Initial Value	0	0	0	1	1	1	1	1

Bit Number	Bit Symbol	Description
15~0	BZDUT	BUZZER Duty Cycle Configuration Register High level time = T_{sys} *(BZDUT+1) *Note：* *1. T_{sys} is the period of the system clock.。* *2. The BUZERR output is meaningful when the value of BZDUT is less than BZDIV.。*

22 Low Voltage Detection (LVD)

22.1 Function Introduction

Low voltage detection (LVD) is used to monitor the chip's own power supply VDD, with four voltage levels available: 2.0V, 2.7V, 3.7V, 4.4V. When VDD is lower than the voltage set, either interrupt or reset occurs.

Note: **Due the manufacturing process, the LVD trigger voltage may be slightly different**

Figure 22-1-1 shows the architecture of LVD.

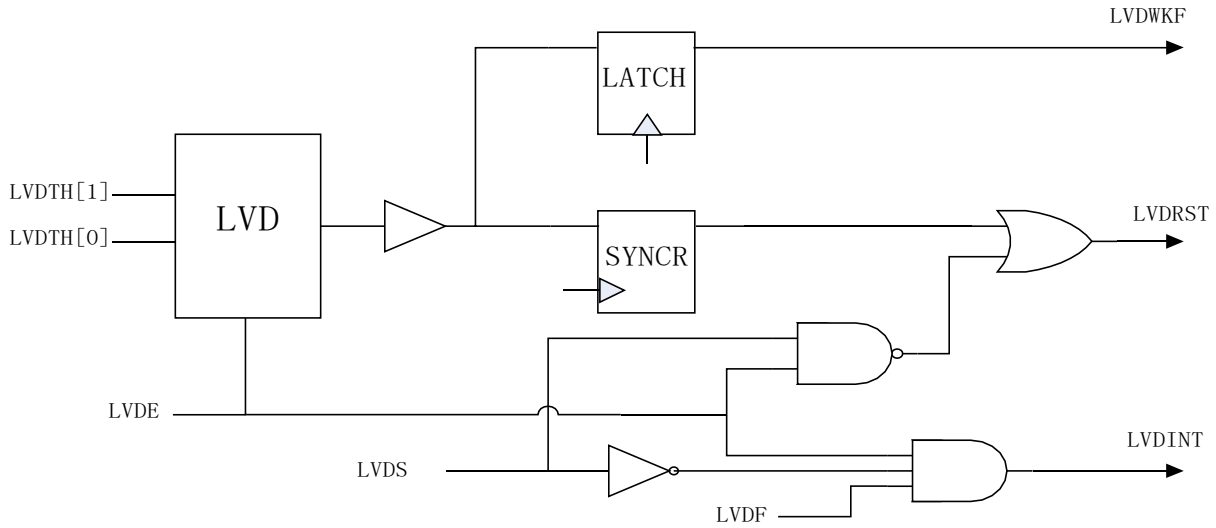


Figure 22-1-1 LVD Schematic

22.2 Function Description

LVD function is enabled by LVDE and the trigger voltage is set by LVDTH. When VDD is lower than the trigger voltage, the LVDF will be set to1. If LVDS=0 then, there will be an interrupt; if LVDS=1, it will generate a reset signal. However, LVD reset signal will not reset itself, which means register LVDCON remains its status. As a result, if VDD is still lower than the trigger voltage set by users after the reset, it will be reset forever. Similarly, the interrupt will occur repeatedly if VDD is still lower than the trigger voltage set by users after the interrupt.

22.3 Register Description

Table 22-3-1 Register LVDCON

EFH	7	6	5	4	3	2	1	0
LVDCON	LVDE	LVDS	LVDF	-	-	-	LVDTH[1:0]	
R/W	R/W	R/W	R/W	-	-	-	R/W	R/W
Initial Value	0	0	0	-	-	-	0	0

Bit Number	Bit Symbol	Description
7	LVDE	LVD enable control, 1 enables it
6	LVDS	LVD function selection 0: Interrupt 1: Reset
5	LVDF	LVD flag ,cleared when 1 is written to it
4~2	-	-
1~0	LVDTH	LVD trigger level selection 00: 2.0V 01: 2.7V 10: 3.7V 11: 4.4V

22.4 LVD Control Example

LVD interrupt example

For instance , set LVD to interrupt mode with trigger voltage 3.7V, the program is like：

```
---------------------------------------------------------------------------------------------
#define LVDE(N)          (N<<7)    //N=0~1
#define LVDS_int         (0<<6)
#define LVDF             (1<<5)
#define LVDTH_3p7V    2
void LVD_init(void)
{
      LVDCON = LVDE(1) | LVDS_int |    LVDTH_3p7V;// enables LVD and set it to interrupt mode, set the trigger
voltage to 3.7V
      INT4EN = 1;                    // enables INT4 interrupt
      EA = 1;                        // Enable total interruption
}
void INT4_ISR (void) interrupt 9     // LVD interrupt service program
{
      if(LVDCON & LVDF)
      {
            LVDCON |= LVDF;          // Clear LVD interrupt flag

      }
}
---------------------------------------------------------------------------------------------
```

LVD reset example

For instance , set LVD to reset mode with trigger voltage 3.7V, the program is like：.

```
---------------------------------------------------------------------------------------------
#define LVDE(N)          (N<<7)    //N=0~1
#define LVDS_reset       (1<<6)
#define LVDTH_3p7V    2
void LVD_init(void)
{
      LVDCON = LVDE(1) | LVDS_reset| LVDTH_3p7V;// /enables LVD and set it to reset mode, set the trigger voltage
to 3.7V
}
---------------------------------------------------------------------------------------------
```

23 Wireless Charger Decoding

23.1 Function Introduction

This module is a wireless charging communication decoding module, which is combined with op amp B in wireless charging applications to decode the output signal of op amp B, and has a built-in automatic verification mechanism and filtering mechanism.

23.2 Register Description

Table 23-2-1 Register DCCR

80D0H	7	6	5	4	3	2	1	0
DCDR	EN	WBIE	ERRIE	SSEL	P2SF	WBF	ERRF	
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7	EN	FSK decode enable bit 0: FSK decoding disable 1: FSK decoding enable
6	WBIE	One-byte FSK decoding completion interrupt enable bit0： One-byte data FSK decoding completion interrupt disable 1: One-byte data FSK decoding completion interrupt enable and receive start bit interrupt enable from Preamble
5	ERRIE	Decode error interrupt enable bit 0: Decoding error interrupt disable 1: Decode error interrupt enable
4	SSEL	FSK signal source selection ： 0: FSK signal from internal op-amp 1: FSK signal from the pin (pin selection is set in the DCFCR register)
3	P2SF	The start bit (indicating the start of data reception) flag is received from the Preamble bit Manual write 1 clear 0 (Note: The corresponding interrupt enable bit is merged with WBIE)
2	WBF	One-byte data decoding completion flag bit Manual write 1 clear 0
1	ERRF	Decoding error flags ： Manual write 1 clear 0
0	OVESEL	Timeout generates error settings ： 0: Timeout does not generate an error 1: Timeout will generate an error

		(Note: In any case, whenever there is a timeout, the hardware will automatically reset internally and wait for a new data transfer)

Table 23-2-2 Register DCDS

80D1H	7	6	5	4	3	2	1	0
DCDS						BITERRF	STERRF	PARITY
R	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7~3	AMP_A_EN	The op amp control bit, see the description in the op amp chapter for details.
2	ERRSF2	Error Cause Flag. The flag is automatically updated by the hardware when an error occurs and is automatically cleared to 0 at the start of a new transmission Error reason decoding： 000: No error 001: Timeout error (any situation where a timeout occurs will be indicated) 010: Signal pulse width out of threshold range error 011: Parity checksum error 100: START error (START received 1) 101: STOP error (STOP received 0)
1	ERRSF1	
0	ERRSF0	

Table 23-2-3 Register DCDB

80D2H	7	6	5	4	3	2	1	0
DCDB	DCDB							
R	R							
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7~0	DCDB	One-byte data received

Table 23-2-4 Register DCB0HH

80D3H	7	6	5	4	3	2	1	0
DCB0HH	DCB0HH							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7~0	DCB0HH	High 8 bits of the high threshold of BIT0

		Note: Setting of pulse width : Suppose the pulse width of bit0 is X BIT0L * T（perclk）< X ± 2^(DCFCR[4:0]+2) * T(perclk)< BIT0H * T（perclk）

Table 23-2-5 Register DCB0HL

80D4H	7	6	5	4	3	2	1	0
DCB0HL	DCB0HL							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit Number	Bit Symbol	Description						
7~0	DCB0HL	Low 8 bits of the high threshold of BIT0						

Table 23-2-6 Register DCB0LH

80D5H	7	6	5	4	3	2	1	0
DCB0LH	DCB0LH							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit Number	Bit Symbol	Description						
7~0	DCB0LH	High 8 bits of the low threshold of BIT0						

Table 23-2-7 Register DCB0LL

80D6H	7	6	5	4	3	2	1	0
DCB0LL	DCB0LL							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit Number	Bit Symbol	Description						
7~0	DCB0LL	Low 8 bits of the high threshold of BIT0						

Table 23-2-8 Register DCB1HH

80D7H	7	6	5	4	3	2	1	0
DCB1HH	DCB1HH							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7~0	DCB1HH	High 8 bits of the high threshold of BIT1 Note: Setting of BIT1 pulse width : Suppose the pulse width of bit1 is X BIT1L * $T_{(perclk)}$ < X ± 2^(DCFCR[4:0]+2) * $T_{(perclk)}$< BIT1H * $T_{(perclk)}$

Table 23-2-9 Register DCB1HL

80D8H	7	6	5	4	3	2	1	0
DCB1HL	DCB1HL							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit Number	Bit Symbol	Description						
7~0	DCB1HL	Low 8 bits of the high threshold of BIT1						

Table 23-2-10 Register DCB1LH

80D9H	7	6	5	4	3	2	1	0
DCB1LH	DCB1LH							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit Number	Bit Symbol	Description						
7~0	DCB1LH	High 8 bits of the low threshold of BIT1						

Table 23-2-11 Register DCB1LL

80DAH	7	6	5	4	3	2	1	0
DCB1LL	DCB1LL							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit Number								
Bit Number	Bit Symbol	Description						
7~0	DCB1LL	Low 8 bits of the high threshold of BIT1						

Table 23-2-12 Register DCOVRH

80DBH	7	6	5	4	3	2	1	0
DCOVRH	DCOVRH							

R	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit Number	Bit Symbol	Description						
7~0	DCOVRH	High 8 bits of the timeout threshold register						

Table 23-2-13 Register DCOVRL

80DCH	7	6	5	4	3	2	1	0
DCOVRL	DCOVRL							
R	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit Number	Bit Symbol	Description						
7~0	DCOVRL	Low 8 bits of the timeout threshold register						

Table 23-2-14 Register DCFCR

80DDH	7	6	5	4	3	2	1	0
DCFCR	DCFCR							
R	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7	SPIN_SEL1	FSK signal source pin setting. 00: FSK signal from P01 01: FSK signal from P14 10: FSK signal from P16 11: No signal
6	SPIN_SEL0	
5		
4		
3	FCR3	Interval setting of sampling points for FSK signals (for filtering)： 0000: sampling point interval of 2 perclk cycles 0001: sampling point interval of 4 perclk cycles 0010: Sampling point interval of 8 perclk cycles 0010: Sampling point interval of 16 perclk cycles 0010: Sampling point interval of 32 perclk cycles 0010: Sampling point interval of 64 perclk cycles 0010: Sampling point interval of 128 perclk cycles 0010: Sampling point interval of 256 perclk cycles 0010: Sampling point interval of 512 perclk cycles 0010: Sampling point interval of 1024 perclk cycles
2	FCR2	
1	FCR1	
0	FCR0	

24 Multiplier/Divider Unit(MDU)

24.1 Function Introduction

There are four types operation for MDU: 32 bit dividing 32 bit, 16 bit multiplying 16 bit, left shift, right shift. It takes one clock cycle periods to complete multiplication and shift operation and 8 clock cycle periods for division.

24.2 Architecture

Figure 24-2-1 shows the principle of multiplication circuit

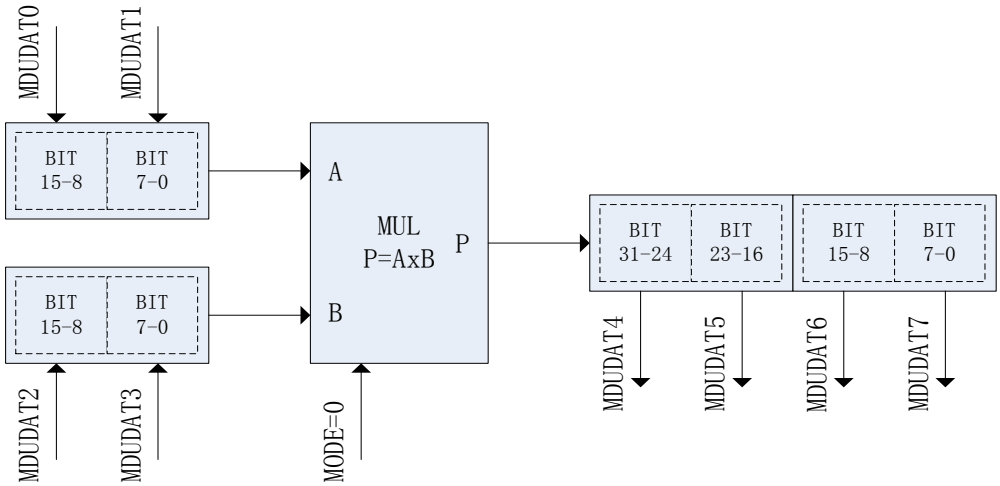


Figure 24-2-1 Multiplication Circuit Schematic

Figure 24-2-2 shows the principle of division circuit

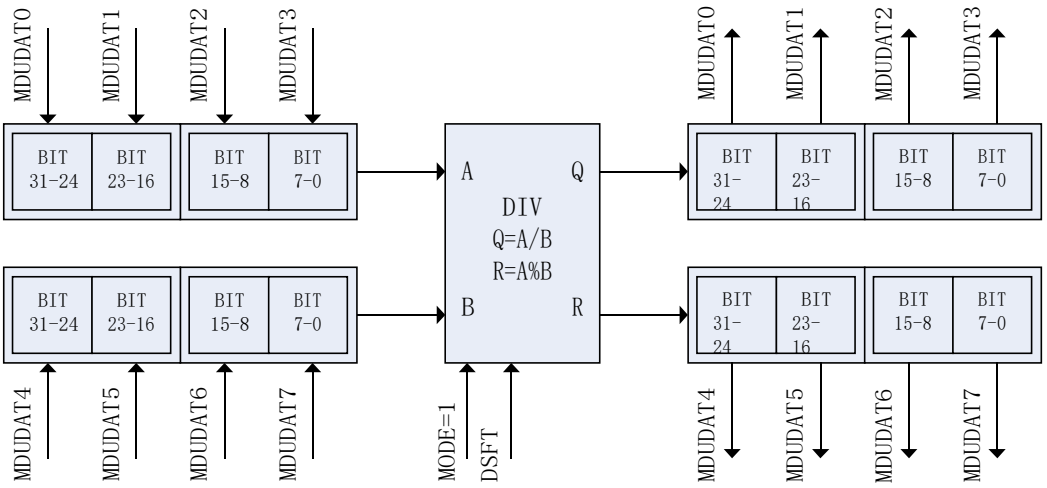


Figure 24-2-2 Division Circuit Schematic

Figure 24-2-3 shows the principle of shift circuit

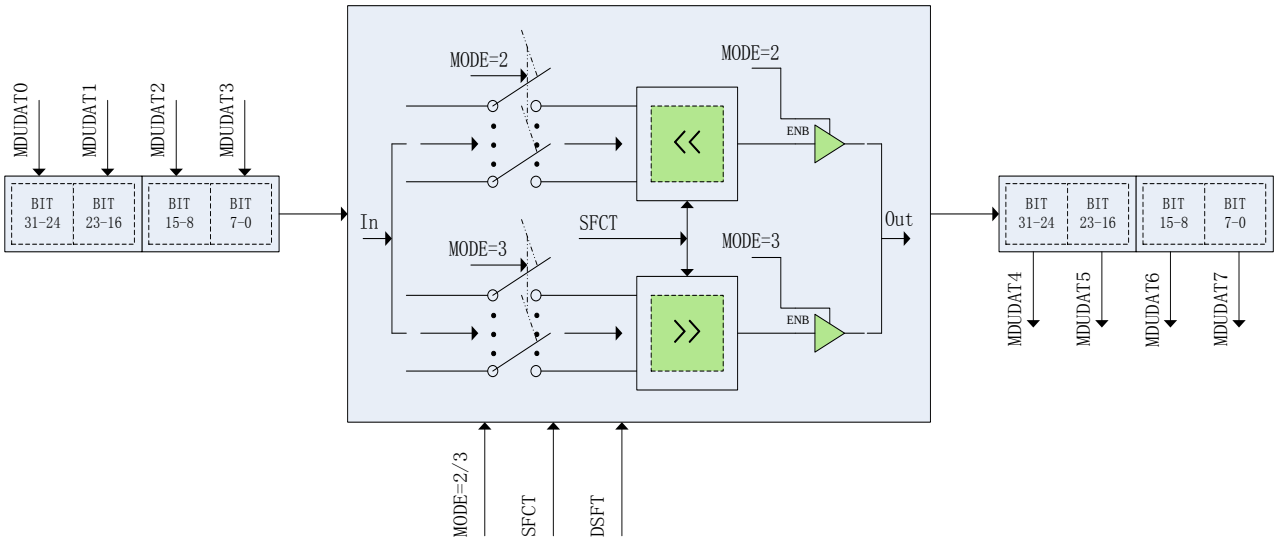


Figure 24-2-3 Shift Circuit Schematic

24.3 Function Description

24.3.1 Multiplier

When MODE=0, MDU is set as a 16bit×16bit multiplier. The multiplicand will be written to register MDUDAT0 and MDUDAT1, with multiplier written to register MDUDAT2 and MDUDAT3. It takes one clock cycle to complete the operation. The product will be obtained instantly after the multiplicand and multiplier are written to the registers. The product will be stored in register MDUDAT4, MDUDAT5, MDUDAT6 and MDUDAT7.

24.3.2 Divider

When MODE=1, MDU is set as a 32 bit ÷32 bit divider. The numerator will be written to register MDUDAT0, MDUDAT1, MDUDAT2 and MDUDAT3, with denominator written to register MDUDAT4, MDUDAT5, MDUDAT6 and MDUDAT7. DSFT must be set to 1 start the operation after the numerator and denominator are written to the registers. DSFT will be cleared automatically after the operation. The quotient will be stored in register MDUDAT0, MDUDAT1, MDUDAT2 and MDUDAT3, with remainder stored in register MDUDAT4, MDUDAT5, MDUDAT6 and MDUDAT7. Since it takes 8 clock cycles to do the division, hence users have to wait for 8 clock cycles or until DSFT is cleared and then read the result.

24.3.3 Shifter

When MODE=2 or MODE=3, MDU is set as a shifter(left shift when MODE=2, right shift when MODE=3). The number of bits to shift is set by SFCT, with the 32-bit data to be shifted written to register MDUDAT0, MDUDAT1, MDUDAT2 and MDUDAT3. Setting DSFT=1 will start the operation. Due to it takes one clock cycle only, the result can be read immediately after DSFT=1. The result will be stored in register MDUDAT4, MDUDAT5, MDUDAT6 and MDUDAT7.

24.4 Register Description

Table 24-4-1 Register MDUCON

E6H	7	6	5	4	3	2	1	0
MDUCON	MODE[1:0]		DSFT	SFCT[4:0]				
R/W	R/W		R/W	R/W				
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7~6	MODE	Operation mode selection 00: Multiplication 01: Division 10: Left shift 11: Right shift
5	DSFT	Enable control for division and shift operation. 1 starts the operation. Invalid for multiplication.
4~0	SFCT	The number of times to shift = (SFCT + 1). Invalid for multiplication and division.

Table 24-4-2 Register MDUDAT

E7H	7	6	5	4	3	2	1	0

MDUDAT	MDUDAT[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Note: MDUDAT is with index register, set INDEX=0~7 corresponding to MDUDAT0~MDUDAT7 respectively

Bit Number	Bit Symbol	Description
7~0	MDUDAT	MDU data storage. No order requirement for reading/writing MDUDAT0th～MDUDAT7th For multiplication, MDUDAT0th： 15th～8th bit of the multiplicand MDUDAT1： 7th～0th bit of the multiplicand MDUDAT2： 15th～8th bit of the multiplier MDUDAT3： 7th～0th bit of the multiplier For product, MDUDAT4： 31th～24th bit of the product MDUDAT5： 23th～16th bit of the product MDUDAT6： 15th～ 8th bit of the product MDUDAT7th： 7th～ 0th bit of the product For division, MDUDAT0th： 31th～24th bit of the numerator； MDUDAT1： 23th～16th bit of the numerator； MDUDAT2： 15th～8th bit of the numerator； MDUDAT3： 7th～0th bit of the numerator. MDUDAT4： 31th～24th bit of the denominator； MDUDAT5： 23th～16th bit of the denominator； MDUDAT6： 15th～ 8th bit of the denominator； MDUDAT7th： 7th～ 0th bit of the denominator 。 Division result： MDUDAT0th： 31th～24th bit of the quotient； MDUDAT1： 23th～16th bit of the quotient； MDUDAT2： 15th～8th bit of the quotient； MDUDAT3： 7th～0th bit of the quotient. MDUDAT4： 31th～24th bit of the remainder； MDUDAT5： 23th～16th bit of the remainder； MDUDAT6： 15th～ 8th bit of the remainder； MDUDAT7th： 7th～0th bit of the remainder. For shift , MDUDAT0th： 15th～8th bit of the source data； MDUDAT1： 7th～0th bit of the source data； MDUDAT2： 15th～8th bit of the source data； MDUDAT3： 7th～0th bit of the source data 。 Shift result ： MDUDAT4： 31th～24th bit of the destination data； MDUDAT5： 23th～16th bit of the destination data； MDUDAT6： 15th～ 8th bit of the destination data；

		MDUDAT7th： 7th～ 0th bit of the destination data 。 Note： For division operation, if the denominator is 0th, then the module will not do the operation. When DSFT is 1, overwriting the numerator or denominator will not influence the result for division.

24.5 MDU Control Example

Define the union below first:

```
---------------------------------------------------------------------------------------------
typedef union
{
      unsigned long int     dwVal;
      unsigned int          wVal[2];
      unsigned char         bVal[4];
}
  DWORD_UNION;

typedef union
{
      unsigned int     wVal;
      unsigned char   bVal[2];
}
  WORD_UNION;
---------------------------------------------------------------------------------------------
```

◆ **Example for Multiplication**

For instance , 65535 is multiplied by 1000, the program is like：

```
---------------------------------------------------------------------------------------------
#define MOD_MULT            (0<<6)
void Mult(void)
{
      WORD_UNION Faciend;             // the multiplicand
      WORD_UNION Multiplier;          // the multiplier
      DWORD_UNION Product;            // product
      Faciend.wVal = 65535;
      Multiplier.wVal = 1000;

      MDUCON = MOD_MULT;// Set MDU to multiplication mode

      INDEX = 0;
      MDUDAT = Faciend.bVal[0]; // write the higher 8 bits of the multiplicand
      INDEX = 1;
      MDUDAT = Faciend.bVal[1]; // write the lower of the multiplicand
      INDEX = 2;
      MDUDAT = Multiplier.bVal[0];//write the higher 8 bits of the multiplier
      INDEX = 3;
      MDUDAT = Multiplier.bVal[1];//write the higher 8 bits of the multiplier
```

```
    INDEX = 4;
    Product.bVal[0] = MDUDAT; //read the 24th~31th bit of the product
    INDEX = 5;
    Product.bVal[1] = MDUDAT; //read the 16th~23th bit of the product
    INDEX = 6;
    Product.bVal[2] = MDUDAT; //read the 8th~15th bit of the product
    INDEX = 7;
    Product.bVal[3] = MDUDAT; // read the 0th~7th bit of the product
}
```

◆ **Example for Division**

For instance , 0xFFFFFFFF is divided by 0x10000000, the program is like：

```
 ---------------------------------------------
#define MOD_DIV             (1<<6)

#define DSFT                (1<<5)
void Divid(void)
{
    DWORD_UNION Dividend;       // numerator
    DWORD_UNION Divisor;        //denominator
    DWORD_UNION Quotient;       //quotient
    DWORD_UNION Remainder;      // Remainder

    Dividend.dwVal = 0xffffffff;
    Divisor.dwVal =     0x10000000;

    MDUCON = MOD_DIV; //set MDU as the division mode

    INDEX = 0;
    MDUDAT = Dividend.bVal[0];      // write 24th~31th bit of the numerator
    INDEX = 1;
    MDUDAT = Dividend.bVal[1];      // write 16th~23th bit of the numerator
    INDEX = 2;
    MDUDAT = Dividend.bVal[2];      // write 8th~15th bit of the numerator
    INDEX = 3;
    MDUDAT = Dividend.bVal[3];      // write 0th~7th bit of the numerator

    INDEX = 4;
    MDUDAT = Divisor.bVal[0];       // write 24th~31th bit of the denominator
    INDEX = 5;
    MDUDAT = Divisor.bVal[1];       // write 16th~23th bit of the denominator
    INDEX = 6;
```

```
    MDUDAT = Divisor.bVal[2]; // write 8th~15th bit of the denominator
    INDEX = 7;
    MDUDAT = Divisor.bVal[3];  // write 0th~7th bit of the denominator

    MDUCON      |=   DSFT;      // enables the division operation
    while(MDUCON & DSFT);     // wait for the result

    INDEX = 0;
    Quotient.bVal[0] = MDUDAT;// read the 24th~31th bit of quotient
    INDEX = 1;
    Quotient.bVal[1] = MDUDAT;// read the 16th~23th bit of quotient
    INDEX = 2;
    Quotient.bVal[2] = MDUDAT;// read the 8th~15th bit of quotient
    INDEX = 3;
    Quotient.bVal[3] = MDUDAT;// read the 0th~7th bit of quotient

    INDEX = 4;
    Remainder.bVal[0]= MDUDAT;      // read the 24th~31th bit of remainder

    INDEX = 5;
    Remainder.bVal[1]= MDUDAT;      // read the 16th~23th bit of remainder

    INDEX = 6;
    Remainder.bVal[2]= MDUDAT;      // read the 8th~15th bit of remainder

    INDEX = 7;
    Remainder.bVal[3]= MDUDAT;      // read the 0th~7th bit of remainder

}
-----------------------------------------------------------------------------------------
```

◆ **Example for Shift Operation**

For instance , 0x88880001 left(or right) shift 8 bits, the program is like：

```
-----------------------------------------------------------------------------------------
#define MOD_SHIFT_LEFT          (2<<6)
#define MOD_SHIFT_RIGHT         (3<<6)
#define DSFT                    (1<<5)
void Shift(void)
{
    DWORD_UNION SourceData;       // Source Data
    DWORD_UNION DestinationData; // destatioinn Data

    MDUCON = MOD_SHIFT_LEFT|7;      //set the left shift and number of times
    //MDUCON = MOD_SHIFT_RIGHT|7;  //set the right shift and number of times
```

```
    SourceData.dwVal = 0x88880001;

    INDEX = 0;
    MDUDAT = SourceData.bVal[0]; //write 24th~31th bit of the source data
    INDEX = 1;
    MDUDAT = SourceData.bVal[1]; // write 16th~23th bit of the source data
    INDEX = 2;
    MDUDAT = SourceData.bVal[2]; // write 8th~15th bit of the source data
    INDEX = 3;
    MDUDAT = SourceData.bVal[3]; // write 0th~7th bit of the source data

    MDUCON |= DSFT;        // enables shift operation

    INDEX = 4;
    DestinationData.bVal[0] = MDUDAT;     // read 24th~31th bit of the destination data
    INDEX = 5;
    DestinationData.bVal[1] = MDUDAT;     // read 16th~23th bit of the destination data
    INDEX = 6;
    DestinationData.bVal[2] = MDUDAT;     // read 8th~15th bit of the destination data
    INDEX = 7;
    DestinationData.bVal[3] = MDUDAT;     // read 0th~7th bit of the destination data
}
-------------------------------------------------------------------------------------------
```

25 SPI

25.1 Function Introduction

SPI enables the chip to support half/full duplex synchronous data transmission with other devices. The peripheral devices could be other MCU, ADC, sensor or FLASH and etc. SPI nus can be 3 or 4 wires and the features are as follows.

- Both Master and Slave mode supported
- The transmission can either starts from the least or most significant bit
- 4 programmable bit rates
- Programmable polarity and phase
- Send end interrupt flag
- Write conflict flag for protection
- Support error interrupt in master mode

Figure 25-1-1 and Figure 25-1-2 are the principle schematics for SPI Master Mode and Slave Mode respectively.

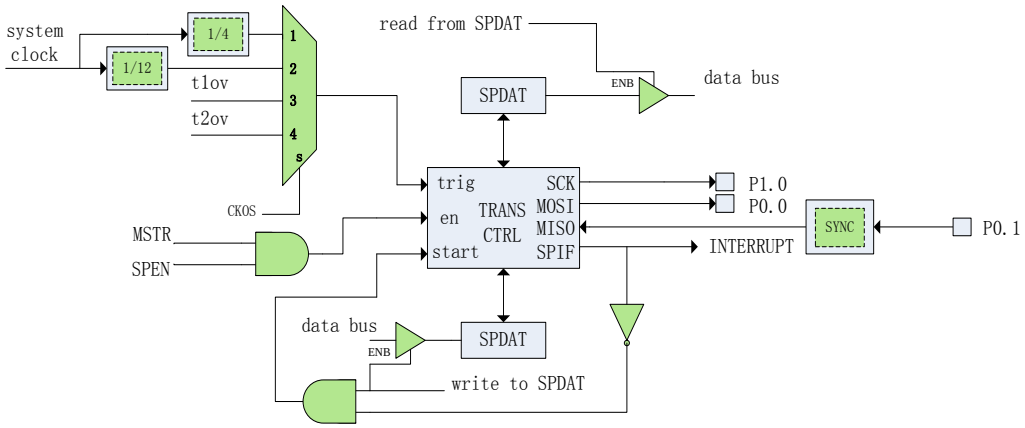


Figure 25-1-1 SPI Master Mode Schematic

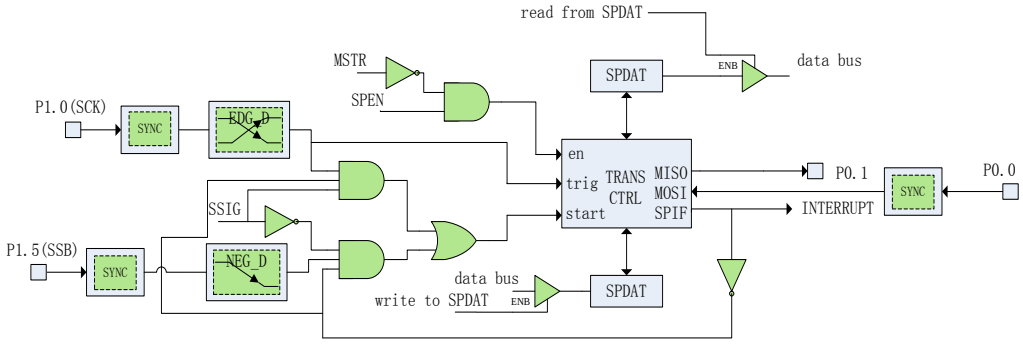


Figure 25-1-2 SPI Slave Mode Schematic

The operating modes for SPI is shown by the following table.

Table 25-1-1 SPI Operation Mode

Name	Description
Host Mode	All the transmission behavior can will be originated by the master, including SCK and SSB signal generation. When MSTR(SPCON[4]) is 1, SPI operates in Master mode. Users must select another pin as the chip select pin and connects slave's. SSB level will be pulled down before the data transmission and pulled up after the transmission. Writing register SPDAT will starts the data transmission in Master mode Data is shifted out from MOSI on the valid edge of the clock.
Slave Mode	When MSTR is set to 0, SPI operates in Slave mode. When SSIG(SPCON[5])=1, SSB is in valid and there are only 3 wires for SPI communication and the default chip select is valid; when SSIG=0, SSB is valid and low level SSB implies that the slave is selected.

Table 25-1-2 SPI Port/Pin Description

Name	Description
MOSI	Master output, slave input This pin is the master's data output port when SPI operates as master; it is the slave's data input port when SPI operates as slave
MISO	Master input, slave output This pin is the master's data input port when SPI operates as master; it is the slave's data output port when SPI operates as slave
SCK	Serial clock This pin is the serial clock output port when SPI operates as master; it is the serial clock input port when SPI operates as slave
SSB	Slave selection This pin is the master's slave select output port when SPI operates as master; it is the slave select input port when SPI operates as slave

Table 25-1-3 SPI Phase and Polarity

Name	Description
CPHA	Phase control 0： SCK take data samples when even edges(1,3,5,...,15)come 1： SCK take data samples when odd edges(2,4,6,...,16)com
CPOL	Polarity control 0： SCK level is low when it is idle 1： SCK level is high when it is idle

Referring to Table25-1-3, the real waveform during the transmission is as Figure25-1-3 and 25-1-4 shows.

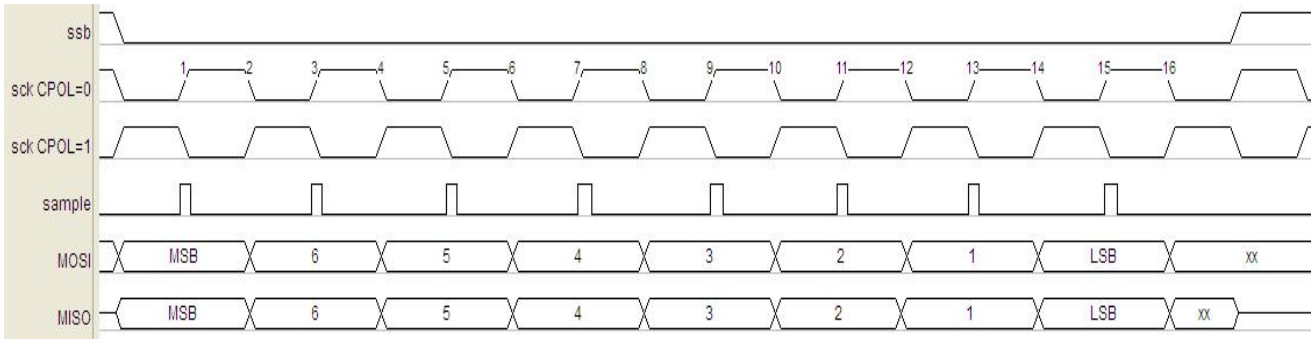


Figure 25-1-3 Sequence Diagram when CPHA=0

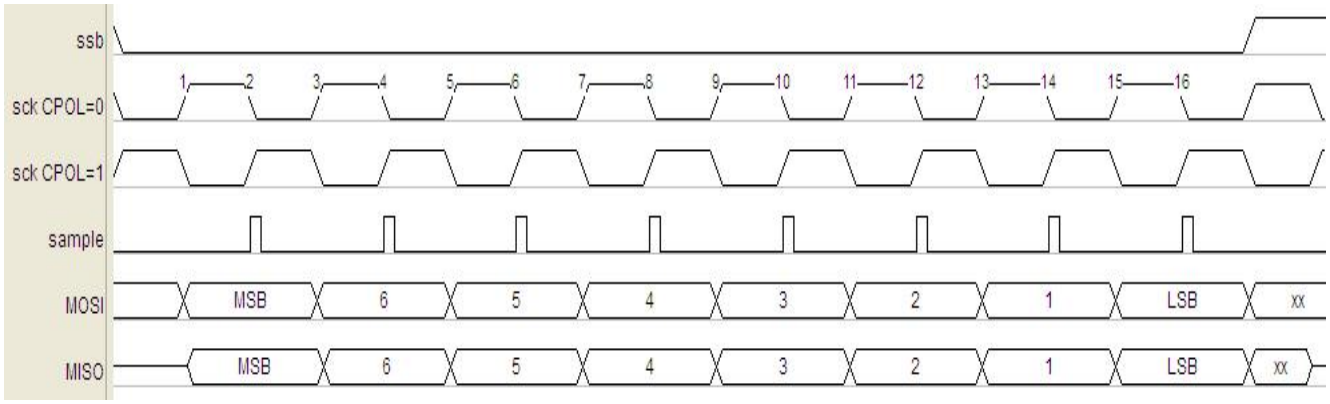


Figure 25-1-4 Sequence Diagram when CPHA=1

25.2 Register Description

Table 25-2-1 Register SPCON

A5H	7	6	5	4	3	2	1	0
SPCON	SPEN	LSBF	SSIG	MSTR	CPOL	CPHA	CKOS[1:0]	
R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7	SPEN	SPI module enable bit, 1 enables it
6	LSBF	Low or high priority transmit/receive selection bit 0: High position first 1: Low position first
5	SSIG	SSB pin disable control, the default value is 0 which indicates SSB signal is valid
4	MSTR	Host/Slave selection 0： Slave 1： Master
3	CPOL	Clock polarity selection 0： the clock signal is low by default 1： the clock signal is high by default
2	CPHA	Clock phase selection 0： take samples when the clock leaves idle state 1： take samples when the clock returns to idle state
1~0	CKOS	SPI output clock selection 00： 1/8 system clock 01： 1/24 system clock 10： Timer1 overflow flag used, transfer data every 2 overflows 11： Timer2 overflow flag used, transfer data every 2 overflows

Table 25-2-2 Register SPDAT

A6H	7	6	5	4	3	2	1	0
SPDAT	RBUF[7:0]							
R/W	R							
Initial Value	0	0	0	0	0	0	0	0
SPDAT	TBUF[7:0]							
R/W	W							
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description

7~0	SPDAT	Writing SPDAT will write to TBUF while reading SPDAT will read from RBUF

Table 25-2-3 Register SPSTA

A7H	7	6	5	4	3	2	1	0
SPSTA	SPIE	-	-	-	-	WCOL	MODF	SPIF
R/W	R/W	-	-	-	-	R/W	R/W	R/W
Initial Value	0	-	-	-	-	0	0	0

Bit Number	Bit Symbol	Description
7	SPIE	SPI interrupt enable control, 1 enables it
6~3	-	-
2	WCOL	Write conflict flag. During the data transmission, if software writes SPDAT while the data can not be written in, then there will be a write conflict flag. 1 indicates the conflict and it will be cleared by writing 1 to it. No interrupts when it is valid.
1	MODF	Fail mode flag with 1valid, it indicates that SSB logical level is not correct. It will be cleared by writing 1 to it. There will be interrupt when it is 1
0	SPIF	Data transmission complete flag, 1 indicates the data transmission is over and it will be cleared by writing 1 to it. There will be interrupt when it is 1 .

25.3 SPI Control Example

SPI master example

As the Master, SPI sends 10 byte data to the Slave, the program is like：

```
-------------------------------------------------------------------------------------------------------------------------- ----------
#define SPEN(N)         (N<<7)
#define LSBF(N)         (N<<6)
#define SSIG(N)         (N<<5)
#define MSTR(N)         (N<<4)
#define CPOL(N)         (N<<3)
#define CPHA(N)         (N<<2)
#define CKOS(N)         (N)     // 3:set system clock as the overflow of Timer2
                                //2:set system clock as the overflow of Timer1
                                //1:set system clock as its 1/8
                                //0:set system clock as its 1/4

// define SPSTA
#define SPIE            (1<<7)  // SPI interrupt enable
#define WCOL            (1<<2)  // Write conflict flag
#define MODF            (1<<1)  // Fail Mode
#define SPIF            (1<<0)  // Transmission complete
unsigned char txIndex;
unsigned char txBuf[10]={0,1,2,3,4,5,6,7,8,9};
unsigned char spi_int_flag;
void SPI_init(void)
{
    P10F = 6; // Set P10 to SPI SCK
    P01F = 6; // Set P01 to SPI MISO
    P00F = 6; // Set P00 to SPI MOSI
    P15F = 2; // Set P15 to output function as SPI chip select pin
    P15 = 1;   // Set the chip select pin to high
    SPCON = SPEN(1) | LSBF(0) | SSIG(1) | MSTR(1) | CPOL(0) | CPHA(0) |CKOS(1);// High first, SSB active,  host,
1/24 system clock
    SPSTA |= SPIE;
    INT5EN = 1;
    spi_int_flag=0;
}
void INT5_ISR (void) interrupt 10
{
    if(SPSTA & SPIF)                    // SPI Interrupt
    {
        SPSTA |= SPIF;                  // Clear SPI interrupt flag
        spi_int_flag = 1;
```

```
        }
        if(SPSTA&MODF)
        {
            SPSTA|=MODF;
        }
    }
    void main(void)
    {
        SPI_init();
        EA = 1;
        txIndex = 0;
        while(1)
        {
            P15=0;
            SPDAT=txBuf[txIndex];
            while(!spi_int_flag);
            spi_int_flag=0;
            P15=1;
            Delay_ms(10);
            txIndex++;
            if(txIndex==10)
                txIndex=0;
        }
    }
```

SPI slave example

As the slave, SPI receives data from the master, the program is like：

```
unsigned char rxIndex;
unsigned char rxBuf[10];
void SPI_init(void)
{
    P10F = 6; // Set P10 to SPI SCK
    P01F = 6; // Set P01 to SPI MISO
    P00F = 6; // Set P00 to SPI MOSI
    P15F = 6; // Set P15 to output function as SPI chip select pin
    SPCON = SPEN(1) | LSBF(0) | SSIG(0) | MSTR(0) | CPOL(0) | CPHA(0) |CKOS(0);// High first, SSB active, slave,
1/8 system clock
    SPSTA |= SPIE;
    INT5EN = 1;
}
void INT5_ISR (void) interrupt 10
{
```

```
    if(SPSTA & SPIF)                    // SPI Interrupt
    {
        SPSTA |= SPIF;                  // Clear SPI interrupt flag
        rxBuf[rxIndex++] = SPDAT;       // Receive one byte of data
        if(rxIndex==10)
            rxIndex=0;
    }
    if(SPSTA&MODF)
    {
        SPSTA|=MODF;
    }
}
void main(void)
{
    SPI_init();
    EA = 1;
    rxIndex= 0;
    while(1)
    {
    }
}
```

---

26 SWIM

26.1 Introduction

The SWIM interface is a single-wire communication interface, designed for single-wire emulation.

26.2 Register Description

Table 2-10-1 Register SWICON

80B0H	7	6	5	4	3	2	1	0
SWICON	SWIF	SWEF	SWIE	-	-	-	SWH	SWD
R/W	R	R	R/W	-	-	-	R/W	R/W
Initial Value	0	0	0	-	-	-	0	0

Bit Number	Bit Symbol	Description
7	SWIF	SWIM interrupt flag, 1 enable Writing 1 to SWIF will clear the SWIF When the module is in transmit phase, writing SWIDAT will clear the SWIF Notes： *1. An interrupt will be generated when each byte transfer is completed.* *2. Timeout counter overflow will generate an interrupt.* *3. When the SWI signal is at 0 and SWIF is 1, if SWH is 1, it will keep SWI at 0 until SWIF or SWH is cleared to 0 to pull up the SWIsignal.*
6	SWEF	SWIM checksum error flag, 1 means checksum error, no interrupt will be requested, write 1 to clear 0
5	SWIE	SWIM interrupt enable flag, 1 valid
4~2	-	-
1	SWH	SWIF keep SWI signal low enable 0: When SWIF is 1, after the SWI signal becomes low, SWI will become high as long as the line is not pulled low 1: When SWIF is 1, after the SWI signal becomes low, the SWI will remain low regardless of whether the line is pulled low or not
0	SWD	SWIM de-energized 0: SWIM module enable 1: SWIM module disabled

Table 2-10-2 Register SWIDAT

80B1H	7	6	5	4	3	2	1	0
SWIDAT	SWIDAT[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
7~0	SWIDAT	SWIM data register Writing a value to this register during the idle phase will start sending COMMAND The BYTE NUMBERS/ADDRESS/DATA will start to be sent when writing to this register during the send phase. Note ： *1. When the module is in the receive phase, SWIF is 1 to indicate that the received data is valid.* *2. When the module is in the receive phase, the write value to this register will be ignored.*

Table 2-10-3 Register SWISTA

80B2H	7	6	5	4	3	2	1	0
SWISTA	SWISTA[7:0]							
R/W	R							
Initial Value	0	1	1	1	1	1	1	0

Bit Number	Bit Symbol	Description
7~0	SWISTA	SWIM status code 00000010： Host sent command 00000011： Host sent byte number 00000100：Host sent address bit[15] to bit[8] 00000101：Host sent address bit[7] to bit[0] 00000110：Host TX data 00000111：Host RX data 01000010：Device received command 01000011： Device received byte number 01000100：Device received address bit[15] to bit[8] 01000101：Device received address bit[7] to bit[0] 01000110：Device RX data 01000111：Device TX data 01111110：Host/Device time overflow error 01111111：Host/Device idle other： reserved

Table 2-10-4 Register SWIOVT

80B3H	7	6	5	4	3	2	1	0
SWIOVT	-	-	-	-	SWIOVT[3:0]			
R/W	-	-	-	-	R/W			
Initial Value	-	-	-	-	1	1	1	1

Bit Number	Bit Symbol	Description
7~0	SWIOVT	SWIM overflow time control register (in 1-bit data length as the basic unit) Overflow time is (SWIOVT + 1) x 128 bits of data Note： *After entering debug mode, the host can rewrite this register with the corresponding write command.*

27 Program Download and Simulation

27.1 Program Download

The JZ8FC003 series mainly uses the ISP method to download programs, and there are two kinds of ports to choose from：

Two-wire communication port：

To download via I2C port, 4 wires need to be connected: VDD, I2C_SCL(P13), I2C_SDA(P14), GND.

Single wire communication port：

To download via SWIM port, you need to connect 3 wires: VDD, SWIM(P02), GND.

For more details on the program download procedure, please refer to the "JZ8FC003 Series Chip Development Tools User's Guide".

27.2 Online Simulation

The JZ8FC003 series chips support in-circuit emulation, similar to program download, and also have two types of ports to choose from：

Two-wire communication port：

For in-circuit emulation via I2C port, 3 wires need to be connected: I2C_SCL(P13), I2C_SDA(P14), GND.

Single wire communication port：

For online emulation via SWIM port, 3 wires need to be connected: RESET, SWIM(P02), GND. It should be noted that before entering emulation,The program inside the chip cannot set the RESET pin as GPIO function, otherwise it cannot enter the emulation mode.

When TSME=0 (PCON[3]), the chip is forbidden to enter the emulation mode. When the chip enters emulation mode, the TSMODE bit (PCON[2]) is set to 1. The application can judge the status of this bit to decide whether to switch to low-speed clock or enter power-saving mode.

For more details about the simulation function please refer to the documents related to emulator

28 Electrical Characteristics

28.1 Limit Parameter

Parameters	Minimum value	Maximum value	Unit
DC voltage for power supply	-0.3	6	V
Input voltage for I/O pin	-0.3	VDD+0.3	V
Working temperature	-40	85	℃
Storage temperature	-55	125	℃
CPU working frequency	-	16	MHz

Note：When the parameters exceed the limits above, the working status of the chip is unpredictable which may lead to severe damage to the chip. Working in such environment for a long time will influence the reliability of the chip.

28.2 DC Electrical Specification

Parameter	Symbol	Operating Voltage	Minimum	Typical	Maximum	Unit	Conditions
Working current	Iop1	VDD=1.8V		1.96		mA	The system clock is IRCH (16MHz), with other clocks disabled. No load for all the output pins, with LDO set to high power consumption mode.. No floating for digital input pins. All the peripherals are disabled, with CPU executing instruction NOP.
		VDD=3.3V		2.26			
		VDD=5V		2.27			
	Iop2	VDD=1.8V		23.2		uA	The system clock is IRCL(131KHZ), with other clocks disabled. No load for all the output pins, with LDO set to low power consumption mode, output voltage is 1.61V. No floating for digital input pins. All the peripherals are disabled, with CPU executing instruction NOP.
		VDD=3.3V		24			
		VDD=5V		24.4			
Current for STOP mode	I_{stp}	VDD=1.8V		5.1		uA	All the clocks disabled. No load for all the output pins, with LDO set to low power consumption mode.No floating for digital input pins. All the peripherals
		VDD=3.3V		5.3			
		VDD=5V		5.5			

							are disabled. Flash enters sleep mode and CPU enters STOP mode.
Current for IDLE mode	I_{idl1}	VDD=1.8V		1.00		mA	The system clock is IRCH (16MHz), with other clocks disabled. No load for all the output pins, with LDO set to low power consumption mode. No floating for digital input pins. All the peripherals are disabled. Flash enters sleep mode and CPU enters IDLE mode.
		VDD=3.3V		1.11			
		VDD=5V		1.12			
	I_{idl2}	VDD=1.8V		14.1		uA	The system clock is set to IRCL (131KHz), other clocks are off, all output pins are unloaded, all digital input pins are not floating, all peripherals are off, the LDO is set to low power mode, and the CPU enters IDLE mode.
		VDD=3.3V		14.5			
		VDD=5V		14.8			
IO port input high voltage (Smit mode on)	V_{hi1}	VDD=1.8V	0.75	-	1.8	V	-
		VDD=3.3V	1.20		3.3		
		VDD=5V	1.50		5		
IO port input high voltage (Smit mode off)	V_{hi2}	VDD=1.8V		0.5*VDD	VDD	V	-
		VDD=3.3V					
		VDD=5V					
IO port input low voltage (Smit mode on)	V_{lo1}	VDD=1.8V	0	-	0.62	V	-
		VDD=3.3V	0	-	0.85		
		VDD=5V	0	-	1.20		
IO port input low voltage (smit mode off)	V_{lo2}	VDD=1.8V	0	0.5*VDD		V	-
		VDD=3.3V					
		VDD=5V					
IO port push current	I_{pu}	VDD=3.3V	-	6	-	mA	IO set to push-pull output mode, drive capability set to maximum, Vol = VDD - 0.3V
		VDD=5V	-	8	-		
IO port sink current	I_{ol}	VDD=3.3V	-	12	-	mA	IO set to push-pull output mode, drive capability set to maximum, Vol = GND + 0.3V
		VDD=5V	-	17	-		
IO port strong pull-down resistor	Rd1	VDD=1.8~5.5V		15		KΩ	-

IO port weak pull-down resistor	Rd2	VDD=1.8~5.5V	-	45	-	KΩ	-
IO port strong pull-up resistor	Ru1	VDD=1.8~5.5V	-	10	-	KΩ	-
IO port weak pull-up resistor	Ru2	VDD=1.8~5.5V		45		KΩ	

Note: The above parameters are typical chip test results randomly selected for reference.

28.3 AC Electrical Specification

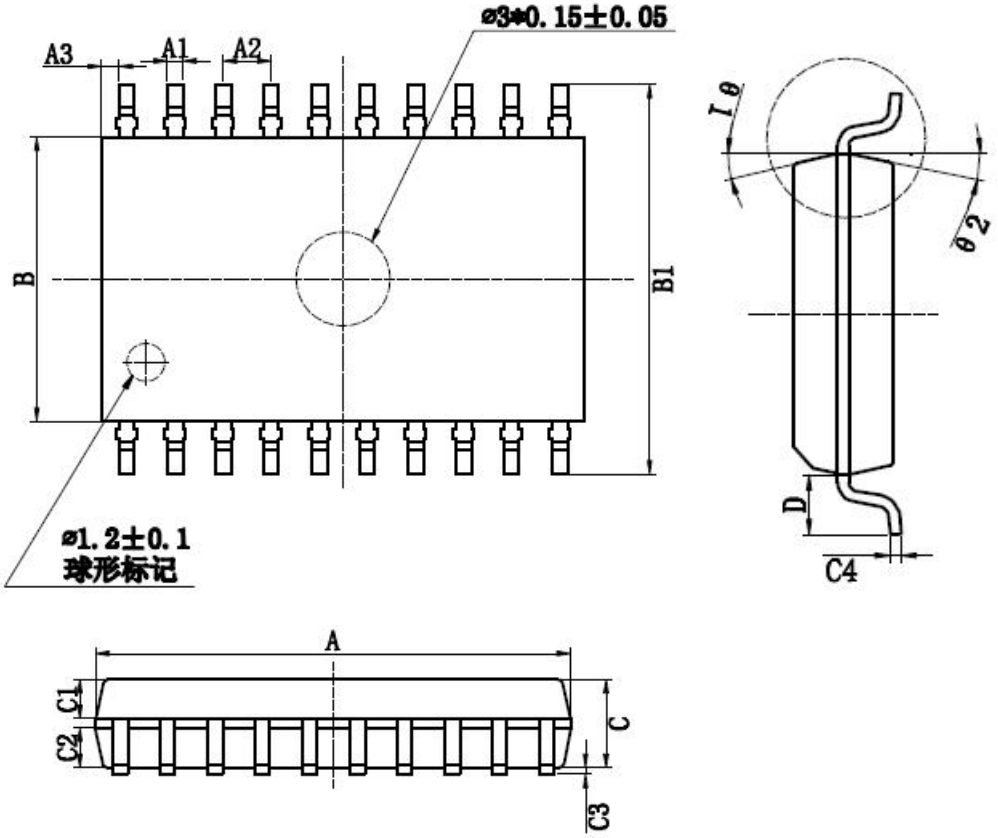
AC Electrical Specification(VDD=1.8-5.5V, TA=25℃, unless there are other explanations)

Chip Parameters	Symbol	Minimum value	Typical values	Maximum value	Unit	Conditions
Time to start oscillation for IRCL	Trc1	-	50	-	us	IRCL frequency 131Khz
Time to start oscillation for IRCH	Trc2	-	10	-	us	IRCH frequency 32MHz
Time of the reset pulse	Trst	-	0.5	-	us	

Note: VDD=3.3V,TA=25℃,internal high speed clock factory frequency is 32MHz,accuracy is ±1%.

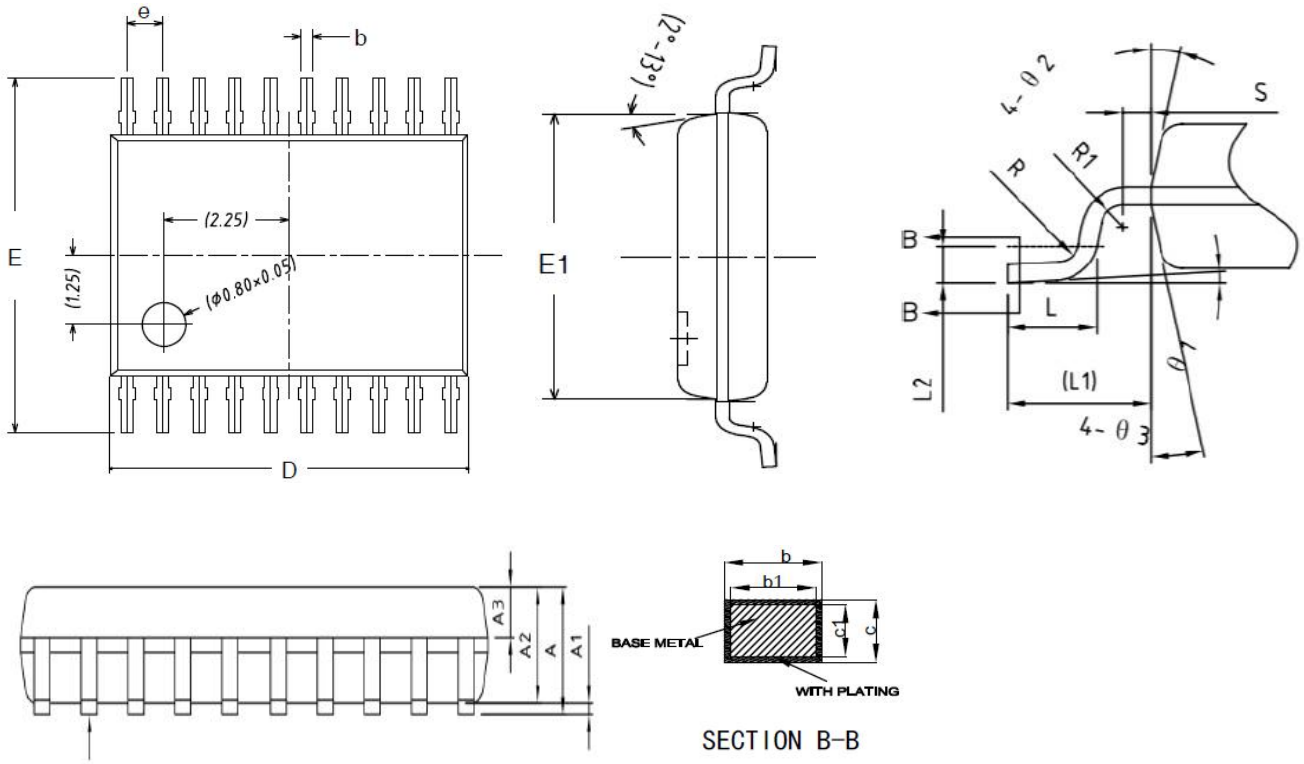
29 Package Type

Package form： SOP20



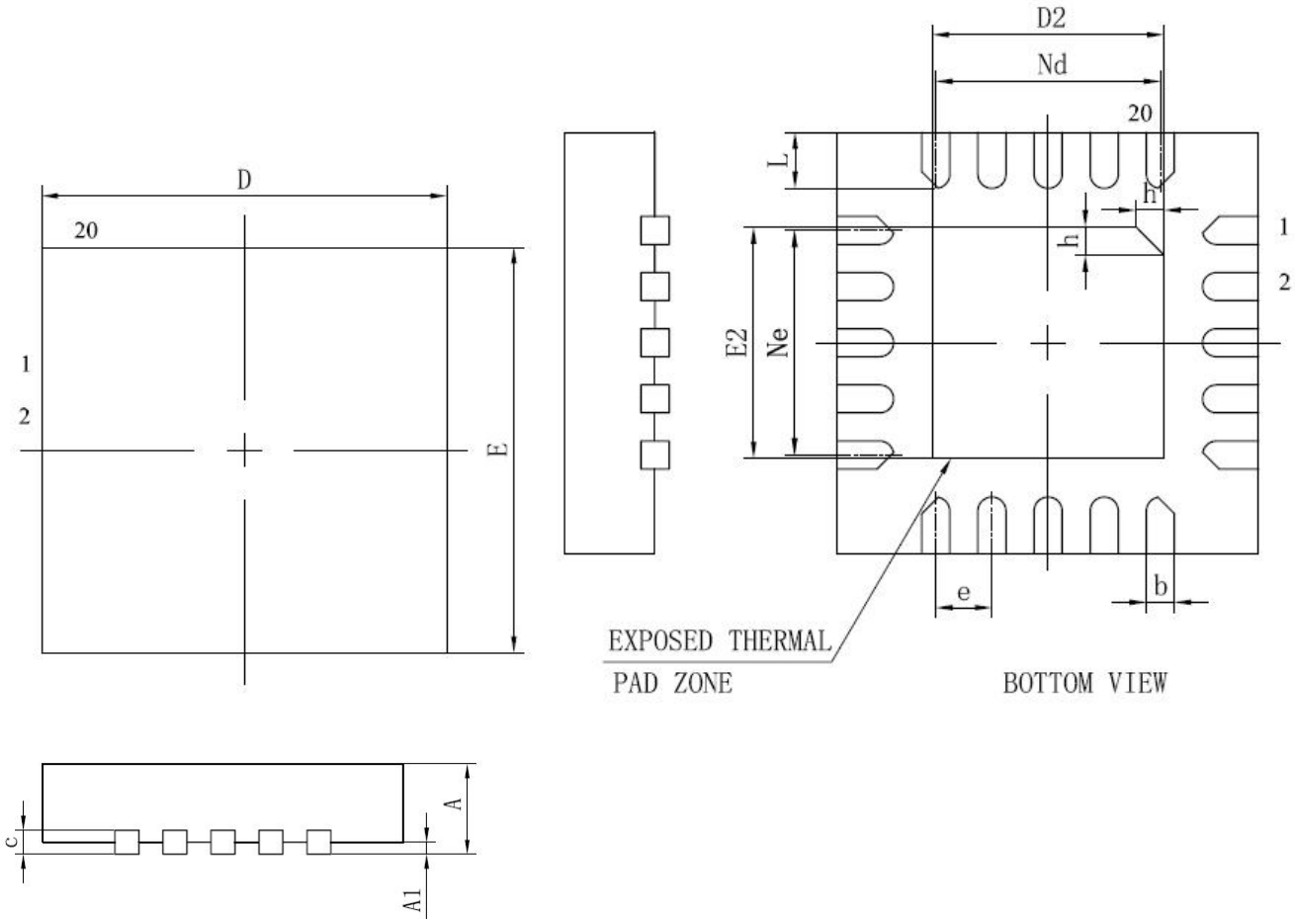
Sequence number	Minimum (mm)	Standard (mm)	Maximum (mm)
A	12.65	12.70	12.80
A1	0.381	0.40	0.431
A2	1.24	1.27	1.30
A3	0.45	0.455	0.46
B	7.40	7.50	7.60
B1	10.206	10.30	10.406
C	2.18	2.23	2.28
C1	0.938	1.0	1.038
C2	0.938	1.0	1.038
C3	0.145	0.175	0.205
D	1.353	1.40	1.453
C4	0.246	0.25	0.262

Package form： TSSOP20



Sequence number	Minimum (mm)	Standard (mm)	Maximum (mm)
A	1.0	---	1.1
A1	0.05	---	0.15
A2	---	---	0.95
A3	0.39	---	0.40
b	0.20	0.22	0.24
c	0.10	---	0.19
c1	0.10		0.15
D	6.40	6.45	6.50
E	6.25	6.40	6.55
E1		4.35	4.40
L	0.50	0.60	0.70
e	0.55	0.65	0.75
L2	0.25BSC		
R	0.09		
L1	1.0REF		

Package form：QFN20(3X3MM)



Serial number	Minimum value (mm)	Standard value (mm)	Maximum value (mm)
A	0.70	0.75	0.80
A1	---	0.02	0.05
b	0.15	0.20	0.25
c	0.18	0.20	0.25
D	2.90	3.00	3.10
D2	1.55	1.65	1.75
e	0.40BSC		
Ne	1.60BSC		
Nd	1.60BSC		
E	2.90	3.00	3.10
E2	1.55	1.65	1.75
L	0.35	0.40	0.45
h	0.20	0.25	0.30

30 Appendix

Appendix 1 Instruction Set Quick Reference Table

Instruction	Description	Description	Cycles
DATA TRANSFER			
MOV A,Rn	Move register to A	(A) ← (Rn)	1
MOV A,direct	Move direct byte to A	(A) ← (direct)	1
MOV A,@Ri	Move indirect RAM to A	(A) ← ((Ri))	1
MOV A,#data8	Move 8-bit immediate data to A	(A) ← #data	1
MOV Rn,A	Move A to register	(Rn) ← (A)	1
MOV Rn,direct	Move direct byte to register	(Rn) ← (direct)	2
MOV Rn,#data8	Move 8-bit immediate data to register	(Rn) ← #data	1
MOV direct,A	Move A to direct byte	(direct) ← (A)	1
MOV direct,Rn	Move register to direct byte	(direct) ← (Rn)	2
MOV direct,direct	Move direct byte to direct byte	(direct) ← (direct)	2
MOV direct,@Ri	Move indirect RAM to direct byte	(direct) ← ((Ri))	2
MOV direct,#data8	Move 8-bit immediate data to direct byte	(direct) ← #data	2
MOV @Ri,A	Move A to indirect RAM	((Ri)) ← (A)	1
MOV @Ri,direct	Move direct byte to indirect RAM	((Ri)) ← (direct)	2
MOV @Ri,#data8	Move 8-bit immediate data to indirect RAM	((Ri)) ← #data	1
MOV DPTR,#data16	Load Data Pointer with 16-bit constant	(DPTR) ← #data116	2
MOV A,@A+DPTR	Move Code byte relative to DPTR to A	(A) ← ((A)) + (DPTR)	2
MOV A,@A+PC	Move Code byte relative to FFe to A	(PC) ← (PC) + 1 (A) ← ((A) + (PC))	2
MOVX A,@Ri	Move External RAM (8-bit addr) to A	(A) ← ((Ri))	2
MOVX A,@DPTR	Move External RAM (16-bit addr) to A	(A) ← ((DPTR))	2
MOVX @Ri,A	Move A to External RAM (8-bit addr)	((Ri)) ← (A)	2
MOVX @DPTR,A	Move A to External RAM (16-bit addr)	(DPTR) ← (A)	2
PUSH direct	Push direct byte onto stack	(SP) ← (SP) + 1 ((SP)) ← (direct)	2
POP DIRECT	Pop direct byte from stack	(direct) ← ((SP)) (SP) ← (SP) - 1	2
XCH A,Rn	Exchange register with A	(A) ↔ (Rn)	1
XCH A,direct	Exchange direct byte with A	(A) ↔ (direct)	1

XCH A,@Ri	Exchange indirect RAM with A	(A) ↔ ((Ri))	1
XCHD A,@Ri	Exchange low-order Digit indirect RAM with A	(A.3,...,A.0) ↔ ((Ri).3,...,(Ri).0)	1
SWAP A	Swap nibbles within A	(A.3,...,A.0) ↔ (A.7,...,A.4)	1
ARITHMETIC OPERATIONS			
ADD A, Rn	Add register to A	(A) ← (A) + (Rn)	1
ADD A, direct	Add direct byte to A	(A) ← (A) + (direct)	1
ADD A, @Ri	Add indirect RAM to A	(A) ← (A) + ((Ri))	1
ADD A, #data8	Add 8-bit immediate data to A	(A) ← (A) + #data	1
ADDC A, Rn	Add register to A with Carry	(A) ← (A) + (C) + (Rn)	1
ADDC A, direct	Add direct byte to A with Carry	(A) ← (A) + (C) + (direct)	1
ADDC A, @Ri	Add indirect RAM to A with Carry	(A) ← (A) + (C) + ((Ri))	1
ADDC A, #data8	Add 8-bit immediate data to A with Carry	(A) ← (A) + (C) + #data	1
SUBB A, Rn	Subtract register from A with Borrow	(A) ← (A) - (C) - (Rn)	1
SUBB A, direct	Subtract direct byte from A with Borrow	(A) ← (A) - (C) - (direct)	1
SUBB A, @Ri	Subtract indirect RAM from A with Borrow	(A) ← (A) - (C) - ((Ri))	1
SUBB A, #data8	Subtract immediate data from A with Borrow	(A) ← (A) - (C) - #data	1
INC A	Increment A	(A) ← (A) + 1	1
INC Rn	Increment register	(Rn) ← (Rn) + 1	1
INC direct	Increment direct byte	(direct) ← (direct) + 1	1
INC @Ri	Increment indirect RAM	((Ri)) ← ((Ri)) + 1	1
INC DPTR	Increment Data Pointer	(DPTR) ← (DPTR) + 1	2
DEC A	Decrement A	(A) ← (A) – 1	1
DEC Rn	Decrement register	(Rn) ← (Rn) - 1	1
DEC direct	Decrement direct byte	(direct) ← (direct) - 1	1
DEC @Ri	Decrement indirect RAM	((Ri)) ← ((Ri)) - 1	1
MUL AB	Multiply A & B (A × B => BA)	temp16 ← (A) X (B) (A)←(temp.7,temp.6,...,temp.0)	4

		(B)←(temp.15,temp.14,...,temp.8)	
DIV AB	Divide A by B(A/B => A +B)	QUO ← (A) / (B) .....REM (A) ← QUO (B) ← REM	4

DA A	Decimal Adjust A	IF (A.3,...,A.0) > 9 || AC = 1
		THEN
		temp16 ← (A) + 0x06
		(A) ← (temp.7,...,temp.0)
		
		IF (temp16) > 0xFF
		THEN
		CY ← 1
		
		IF (A.7,...,A.4) > 9 || CY = 1
		THEN
		temp16 ← (A) + 0x60
		(A) ← (temp.7,...,temp.0)
		
		IF (temp16) > 0xFF
		THEN
		CY ← 1
			1

LOGICAL OPERATIONS			
ANL A, Rn	AND register to A	(A) ← (A) & (Rn)	1
ANL A, direct	AND direct byte to A	(A) ← (A) & (direct)	1
ANL A, @Ri	AND indirect RAM to A	(A) ← (A) & ((Ri))	1
ANL A, #data8	AND 8-bit immediate data to A	(A) ← (A) & #data	1
ANL direct, A	AND A to direct byte	(direct) ← (direct) & (A)	1
ANL direct, #data8	AND 8-bit immediate data to direct byte	(direct) ← (direct) & #data	2
ORL A, Rn	OR register to A	(A) ← (A) | (Rn)	1

ORL A, direct	OR direct byte to A	(A) ← (A) | (direct)	1
ORL A, @Ri	OR indirect RAM to A	(A) ← (A) | ((Ri))	1
ORL A, #data8	OR 8-bit immediate data to A	(A) ← (A) | #data	1
ORL direct, A	OR A to direct byte	(direct) ← (direct) | (A)	1
ORL direct, #data8	OR 8-bit immediate data to direct byte	(direct) ← (direct) | #data	2
XRL A, Rn	Exclusive-OR register to A	(A) ← (A) ^ (Rn)	1
XRL A, direct	Exclusive-OR direct byte to A	(A) ← (A) ^ (direct)	1
XRL A, @Ri	Exclusive-OR indirect RAM to A	(A) ← (A) ^ ((Ri))	1
XRL A, #data8	Exclusive-OR 8-bit immediate data to A	(A) ← (A) ^ #data	1
XRL direct, A	Exclusive-OR A to direct byte	(direct) ← (direct) ^ (A)	1
XRL direct, #data8	Exclusive-OR 8-bit immediate data to direct byte	(direct) ← (direct) ^ #data	2
CLR A	Clear A	(A) ← 0	1
CPL A	Complement A	(A) ← /(A)	1
RL A	Rotate A Left	(A) ← (A.6,A.5,...,A.0,A.7)	1
RLC A	Rotate A Left through Carry	C ← A.7 (A) ← (A.6,A.5,...,A.0,C)	1
RR A	Rotate A Right	(A) ← (A.0,A.7,...,A.2,A.1)	1
RRC A	Rotate A Right through Carry	C ← A.0 (A) ← (C,A.7,...,A.2,A.1)	1
PROGRAM AND MACHINE CONTROL			
ACALL addr11	Absolute subroutine call	(PC) ← (PC) + 2 (SP) ← (SP) + 1 ((SP)) ← (PC7-0) (SP) ← (SP) + 1 ((SP)) ← (PC15-8) (PC10-0) ← page address	2
LACLL addr16	Long subroutine call	(PC) ← (PC) + 3 (SP) ← (SP) + 1 ((SP)) ← (PC7-0)	2

		((SP)) ← (PC15-8) (PC) ←addr15-0	
RET	Return from subroutine	(PC15-8) ← ((SP)) (SP) ← (SP) - 1 (PC7-0) ← ((SP)) (SP) ← (SP) - 1	2
RETI	Return from interrupt	(PC15-8) ← ((SP)) (SP) ← (SP) - 1 (PC7-0) ← ((SP)) (SP) ← (SP) - 1	2
AJMP addr11	Absolute Jump	(PC) ← (PC) + 2 (PC10-0) ← page address	2
LJMP addr16	Long Jump	(PC) ← (PC) + 3 (SP) ← (SP) + 1 ((SP)) ← (PC7-0) (SP) ← (SP) + 1 ((SP)) ← (PC15-8) (PC10-0) ←addr15-0	2
SJMP rel	Short Jump (relative addr)	(PC) ← (PC) + 2 (PC) ← (PC) + rel	2
JMP @A+DPTR	Jump indirect relative to DPTR	(PC) ← (A) + (DPTR)	2
JZ rel	Jump if A is Zero	(PC) ← (PC) + 2 IF (A) = 0 THEN (PC) ← (PC) + rel	2
JNZ rel	Jump if A is Not Zero	(PC) ← (PC) + 2 IF (A) <> 0 THEN (PC) ← (PC) + rel	2

CJNE A, direct, rel	Compare direct to A & Jump if Not Equal	(PC) ← (PC) + 3
		IF (A) <> (direct)
		THEN
		(PC) ← (PC) +
		relative offset
		IF (A) < (direct)
		THEN
		(C) ← 1
			2

		ELSE (C) ← 0	

CJNE A, #data8, rel	Compare 8-bit immediate to A & Jump if Not Equal	(PC) ← (PC) + 3
		IF (A) <> data
		THEN
		(PC) ← (PC) + relative offset
		IF (A) < data
		THEN
		(C) ← 1
		ELSE
		(C) ← 0
			2

CJNE Rn, #data8, rel	Compare 8-bit immediate to reg. & Jump if Not Equal	(PC) ← (PC) + 3
		IF (Rn) <> data
		THEN
		(PC) ← (PC) + relative offset
		IF (Rn) < data
		THEN
		(C) ← 1
		ELSE
		(C) ← 0
			2

CJNE @Ri, #data8, rel	Compare 8-bit immediate to ind. & Jump if Not Equal	(PC) ← (PC) + 3
		IF ((Ri)) <> data
		THEN
		(PC) ← (PC) + relative offset
		IF ((Ri)) < data
		THEN
		(C) ← 1
		ELSE
		(C) ← 0
			2

DJNZ Rn, rel	Decrement register & Jump if Not Zero	(PC) ← (PC) + 2 (Rn) ← (Rn) - 1 IF (Rn) <> 0 THEN (PC) ← (PC) + rel	2
DJNZ direct, rel	Decrement direct byte & Jump if Not Zero	(PC) ← (PC) + 2 (direct) ← (direct) - 1 IF (direct) <> 0 THEN (PC) ← (PC) + rel	2
NOP	No operation	(PC) ← (PC) + 1	1

BOOLEAN VARIABLE MANIPULATION			
CLR C	Clear Carry flag	(C) ← 0	1
CLR bit	Clear direct bit	(bit) ← 0	1
SETB C	Set Carry flag	(C) ← 1	1
SETB bit	Set direct bit	(bit) ← 1	1
CPL C	Complement Carry flag	(C) ← /(C)	1
CPL bit	Complement direct bit	(bit) ← /(bit)	1
ANL C, bit	AND direct bit to Carry flag	(C) ← (C) & (bit)	2
ANL C, /bit	AND complement of direct bit to Carry flag	(C) ← (C) & /(bit)	2
ORL C, bit	OR direct bit to Carry flag	(C) ← (C) | (bit)	2
ORL C, /bit	OR complement of direct bit to Carry flag	(C) ← (C) | /(bit)	2
MOV C, bit	Move direct bit to Carry flag	(C) ← (bit)	1
MOV bit, C	Move Carry flag to direct bit	(bit) ← (C)	2
JC rel	Jump if Carry flag is set	(PC) ← (PC) + 2 IF (C) = 1 THEN (PC) ← (PC) + rel	2
JNC rel	Jump if No Carry flag	(PC) ← (PC) + 2 IF (C) = 0 THEN (PC) ← (PC) + rel	2
JB bit, rel	Jump if direct Bit is set	(PC) ← (PC) + 3 IF (bit) = 1 THEN (PC) ← (PC) + rel	2
JNB bit, rel	Jump if direct Bit is Not set	(PC) ← (PC) + 3 IF (bit) = 0 THEN (PC) ← (PC) + rel	2
JBC bit, rel	Jump if direct Bit is set & Clear bit	(PC) ← (PC) + 3 IF (bit) = 1 THEN (bit) ← 0 (PC) ← (PC) + rel	2
Pseudo-instruction			
ORG	Set program start address		
END	Mark the end of source code		
EQU	Define constants		
SET	Define integer numbers		
DATA	Assign a value to the data address		
BYTE	Assigning values to byte type symbols		
WROD	Assigning values to word type symbols		
BIT	Name the address of the bit		
ALTNAME	Replace reserved words with custom names		
DB	Load a contiguous block of memory with byte-type data		
DW	Load a contiguous block of memory with word data		
DS	Set aside a contiguous storage area or load specified bytes		
INCLUDE	Insert a source file into the program		

TITLE	Add a header row to the list file
NOLIST	No list file is generated during assembly
NOCODE	When the condition is compiled, the list is not generated if the condition is false