JZ8FC005 系列 MCU

中文用户手册

Built - in 16 Bit PWM / 12Bit ADC / Touch Key / 1T 8051 64K Flash MCU

目 录

1 概述

JZ8FC005T 系列芯片是基于 1T 8051 内核的 8 位微控制器，通常情况下，运行速度比传统的 8051 芯片快 10 倍，性能更加优越。内置 64K Flash 程序存储器、4K 字节 SRAM 给用户开发带来了极大的方便。不仅保留了传统 8051 芯片的基本特性，还集成了 16 Bit PWM、12Bit ADC、UART、I²C、RGB_LED、Touch Key 控制器以及低电压检测(LVD)等功能模块。支持 IDLE、STOP 和低速运行三种省电模式以适应不同功耗要求的应用。强大的功能及优越的抗干扰性能使其可广泛应用于各种工业控制、家用照明、小家电、运动器材、大健康电子、电机控制等产品。

2 基本特性

内核

- CPU：1T 8051, 最高速度比传统 8051 快 10 倍
- 兼容 8051 指令集, 双 DPTR 工作模式

存储器

- Flash: 64K 字节，支持多次重复擦写
- Flash 可划分为程序空间和数据空间，数据空间可用于存储掉电需要保存数据，可省略 EEPROM
- RAM:256 字节内部 RAM，4K 字节外部 RAM

工作电压

- 工作电压：2.0 - 5.5V

时钟系统

- 内置低速 RC 振荡器：131KHz，精度为±2%（3.3V@25℃）
- 内置高速 RC 振荡器：24MHz，精度为±1%（3.3V@25℃）
- 外部高速振荡器：1 - 24MHz
- 外部时钟输入：1 - 24MHz
- 可编程 RC 振荡器 PFG：20 - 40MHz，内置自动跳频功能, 专门为雾化器功能而设计
- PFG 时钟频率校正模块：可实现以系统时钟为基准，对 PFG 时钟进行计数，以校正中心频率

TMC 功能

- 时钟源为内置低速 RC 振荡器，中断时间最小单位为 512 个低速 RC 振荡器时钟周期
- 可配置中断时间为 1-256 个最小单位时间

中断系统

- 15 个有效中断源
- 两级中断优先级，支持中断嵌套
- 10 个外部中断源，每个外部中断都可配置任意信号引脚作为中断输入脚

定时器

- 3 个 16 位通用定时器: 定时器 0，定时器 1，定时器 2

通用输入输出口（GPIO）

- 最多支持 18 个 GPIO 口，支持推挽、开漏、上拉、下拉、高阻模式 (型号不同，有不同）
- 8 个引脚(P0.0、P0.1、P1.0 ~ P1.5) 支持强灌电流，最高可达 95mA@5V,可用于高亮 LED 显示

模/数转换器（ADC）

- 支持 12 通道 12 位 ADC，内置运放和比较功能 (型号不同，有不同）
- 支持标准和快速两种模式，快速模式采样速度达 1MHz，可用于电机产品
- 支持 3 种基准电压源：VDD、内部基准、外部基准
- 选择内部电压为基准电压时可测量VDD 电压

运放（AMP）

- 运放A 为通用运放，运放A 内置校正机制，校正后全温条件下失调电压小于 0.5mV
- 运放 B 为无线充专用运放，用于 ASK 解码，放大倍数内部共有 4 档可选:33 倍、50 倍、75 倍、100 倍

无线充解码

- 内置两路 QI 硬件解码器
- 支持QI 无线充标准通信协议硬件解码，开发产品更便捷

PWM

- 支持 8 通道 PWM，在 16 位范围内可任意配置周期和占空比，其中 PWM[0~5] 每路都可独立设置任意引脚作为 PWM 输出引脚
- 支持互补模式和死区控制，可用于驱动直流无刷电机
- 支持可设置边沿对齐和中心对齐模式
- 支持可直接输出内部时钟功能
- 支持 PWM 中断
- 支持 800Kbps/S 扫描频率两路硬件级联模块，直接控制WS2812 或类似的驱动芯片，符合单色或七彩LED 灯带产品的需求。

触摸按键（Touch Key）

- 内置触摸感应控制器
- 最大支持 17 个触摸通道 (型号不同，有不同）
- 触摸可设置内部充电和内部基准，可有效抑制电源低频干扰
- 内置防水补偿机制
- 高抗干扰性，符合 EMC(CS)标准
- 支持触摸省电模式

低电压检测（LVD）

- 可配置宽电压检测范围：1.7V ~ 4.8V，每级 0.1V
- 可设置低电压复位或中断

复位模式

- 芯片支持多种复位源：硬复位，软复位，看门狗复位，低电压检测复位，上电/掉电复位

看门狗

- 27 位看门狗定时器，16 位调节精度，可配置看门狗复位或中断

通用串行接口（UART1/UART2）

- 支持 2 个 UART 接口
- 支持 1 字节接收缓存

SPI 接口

- 内置 1 个 4 线 SPI 接口，支持主从模式

I^2C 接口

- 内置 1 路 I^2C 接口，支持主从模式，支持标准/快速/高速模式

蜂鸣器

- 内置 1 路蜂鸣器驱动输出

乘除法器（MDU）

- 支持 1 个时钟周期 16 位 × 16 位乘法
- 支持 8 个时钟周期 32 位 ÷ 32 位除法
- 支持 1 个时钟周期 32 位数据左右移位操作

程序下载和仿真

- 支持ISP 和 IAP
- 支持在线仿真功能

低功耗

- STOP 模式，电流<7uA
- IDLE 模式，电流<15uA

封装类型：SOP20 / TSSOP20 / QFN20 /SOP16

3 芯片型号功能介绍

表 3-1　JZ8FC005T 系列具体型号功能特点

芯片型号	Flash 容量[BYTE]	外部 Ram[BYTE]	内部高速 RC 振荡器	内部低速 RC 振荡器	内部可编程 PFG 振荡器	GPIO 数量	通用 16 位定时器数量	UART 数量	I²C	SPI	16 bit PWM 通道数量	触摸按键数量	12 位 ADC 通道数量	级联 LED 驱动	ISP	片上仿真功能	工作电压	封装形式
JZ8FC005TS3	64K	4K	√	√	√	14	3	2	√	--	8	13	11	2	√	√	2.0-5.5	SOP16
JZ8FC005TT3	64K	4K	√	√	√	18	3	2	√	√	8	17	12	2	√	√	2.0-5.5	TSSOP20
JZ8FC005TS4	64K	4K	√	√	√	18	3	2	√	√	8	17	12	2	√	√	2.0-5.5	SOP20
JZ8FC005TN2	64K	4K	√	√	√	18	3	2	√	√	8	17	12	2	√	√	2.0-5.5	QFN20

4 系统框图

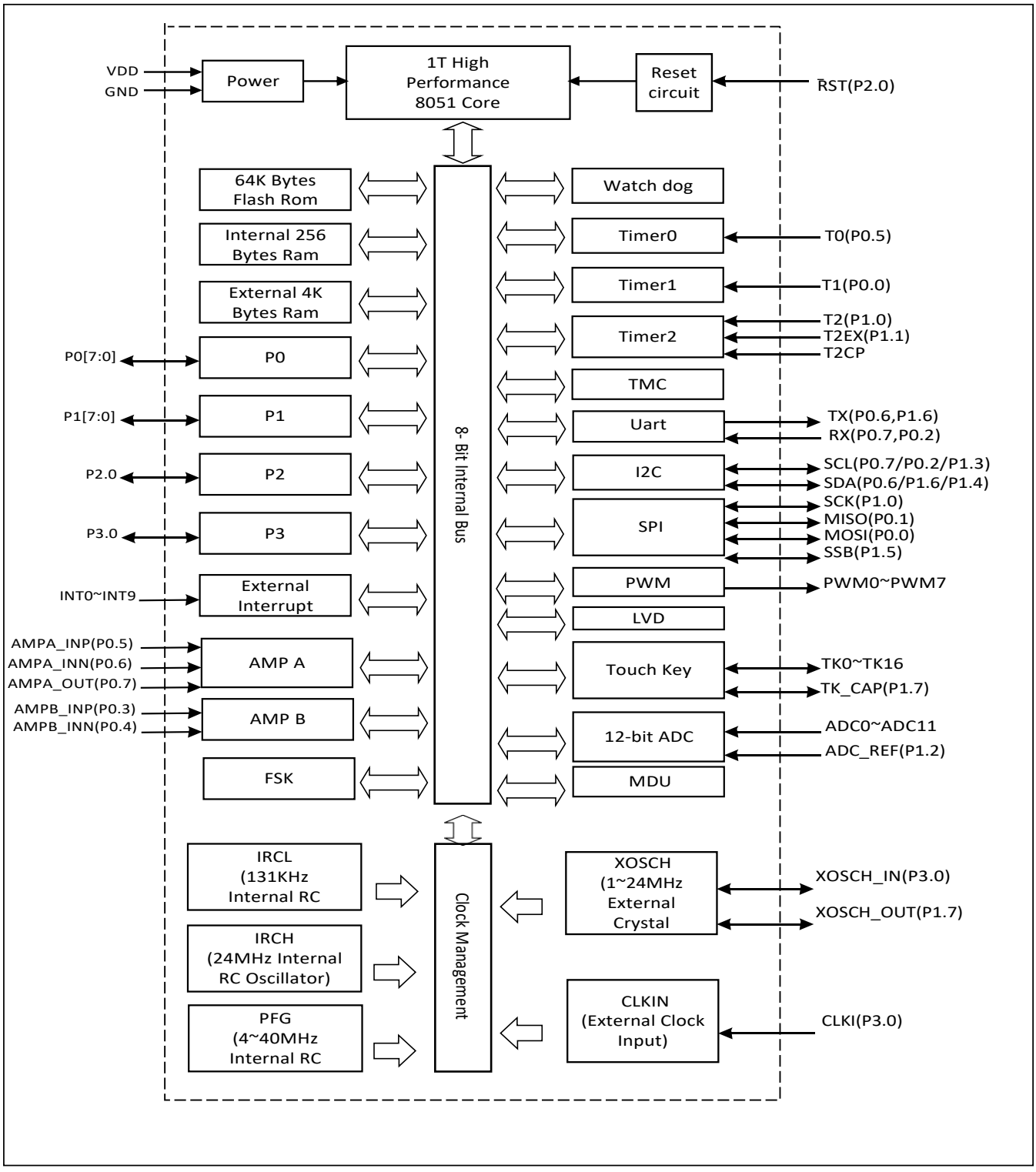


图 4-1-1 芯片框图

5 引脚封装及其描述

5.1 封装定义

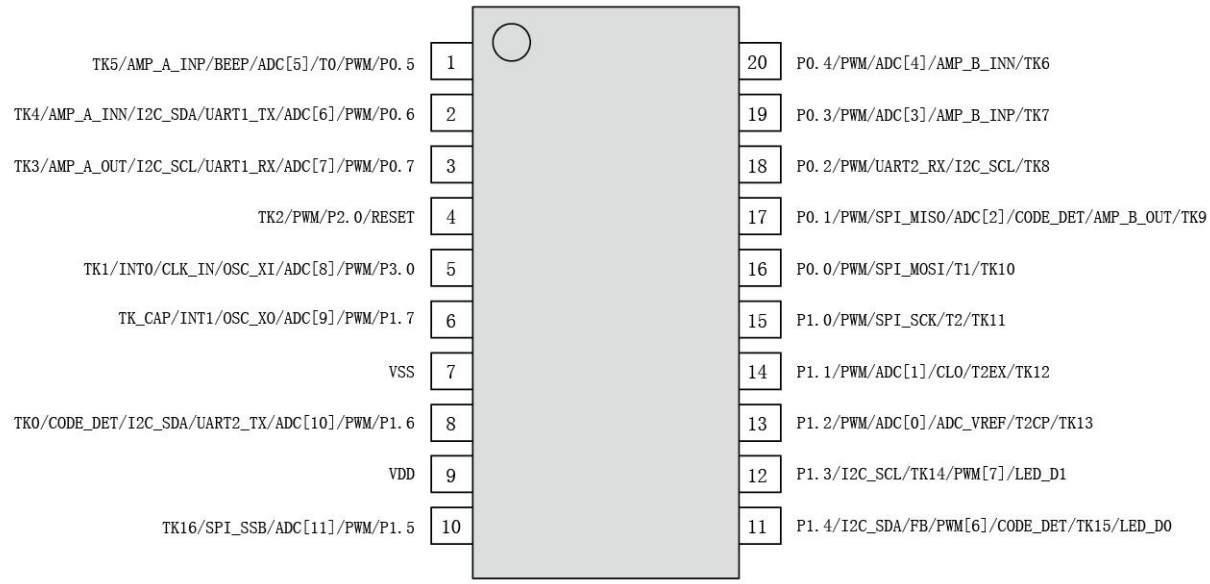


图 5-1-1 TSSOP20 管脚定义图

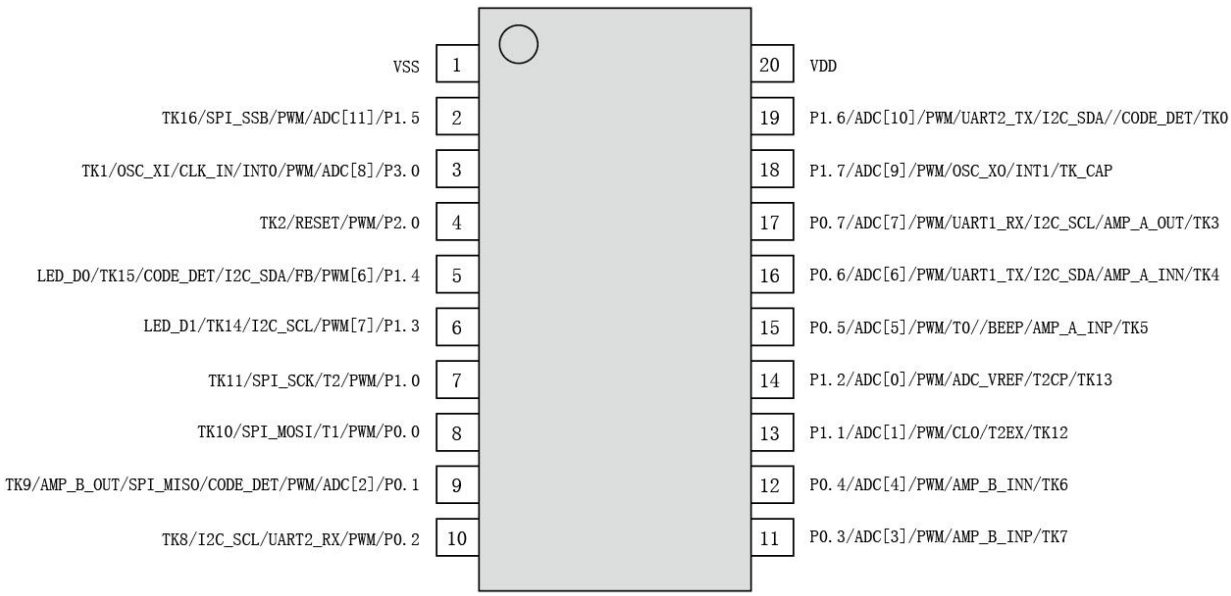


图 5-1-2 SOP20 管脚定义图

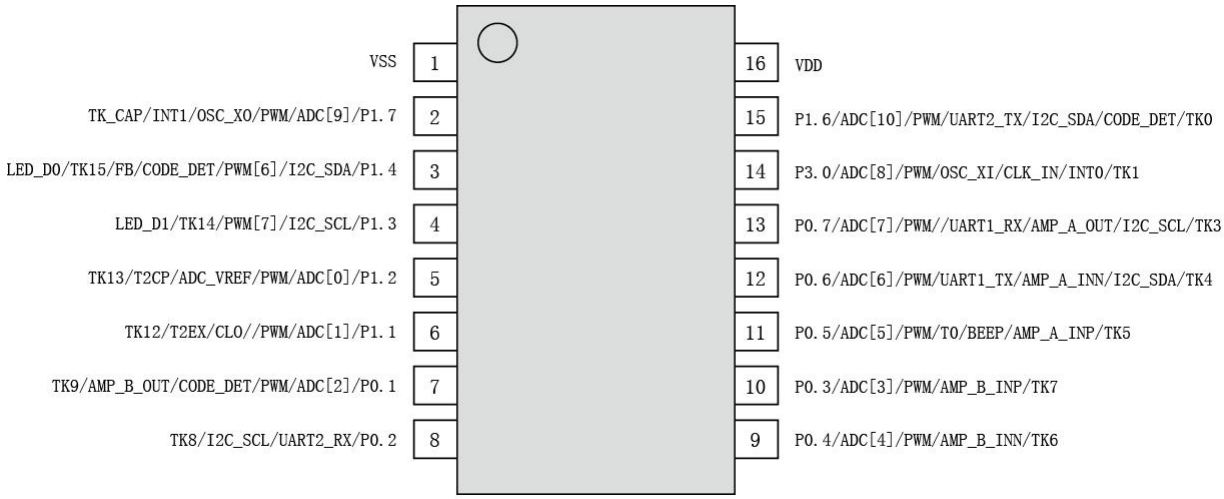


图 5-1-3 SOP16 管脚定义图

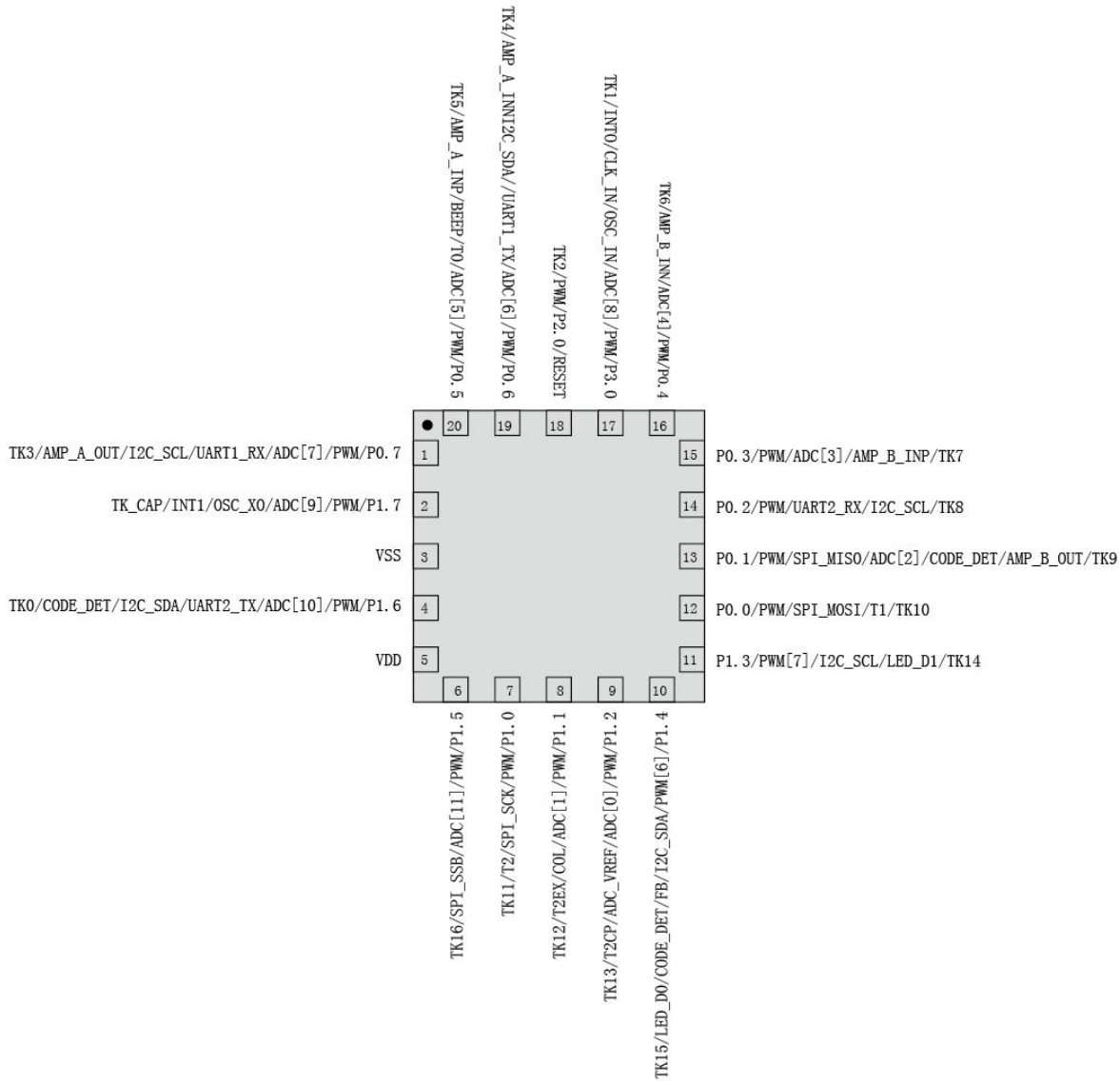


图 5-1-4 QFN20 管脚定义图

5.2 引脚描述

表 5-2-1 引脚描述

引脚序号				管脚名称	管脚功能	默认功能
TSSOP20	SOP20	QFN20	SOP16			
1	15	20	11	P0.5/PWM/T0/ADC[5]/BEEP/ AMP_A_INP/TK5	通用双向 I/O 口 PWM 信号输出 T0 输入 ADC 通道输入 BEEP 输出 AMP 通道输入 触摸按键模拟通道输入	通用双向 I/O 口
2	16	19	12	P0.6/PWM/UART1_TX/ADC[6]/ AMP_A_INN/I2C_SDA/TK4	通用双向 I/O 口 PWM 信号输出 UART1_TX 传输口 ADC 通道输入 AMP 通道输入 I_2C 数据传输口 触摸按键模拟通道输入	通用双向 I/O 口
3	17	1	13	P0.7/PWM/UART1_RX/ADC[7]/ AMP_A_OUT/I2C_SCL/TK3	通用双向 I/O 口 PWM 信号输出 UART1_RX 传输口 ADC 通道输入 AMP 通道输出 I_2C 时钟传输口 触摸按键模拟通道输入	通用双向 I/O 口
4	4	18		P2.0/PWM/RESET/TK2	通用双向 I/O 口 PWM 信号输出 硬件复位引脚 触摸按键模拟通道输入	硬件复位引脚
5	3	17	14	P3.0/PWM/OSC_XI/ADC[8]/CLK_IN/INT0/ TK1	通用双向 I/O 口 PWM 信号输出 OSC_XI 通道输入 ADC 通道输入 CLK_IN 通道输入 INT0 输入 触摸按键模拟通道输入	通用双向 I/O 口
6	18	2	2	P1.7/PWM/OSC_XO/ADC[9]/INT1/ TK_CAP	通用双向 I/O 口 PWM 信号输出 OSC_XO 通道输出 ADC 通道输入 INT1 输入 触摸外部电容输入口	通用双向 I/O 口
7	1	3	1	VSS	电源地引脚	电源地引脚
8	19	4	15	P1.6/PWM/UART2_TX/I2C_SDA/ ADC[10]/CODE_DET/TK0	通用双向 I/O 口 PWM 信号输出 UART2_TX 传输口 I2C 数据传输口 ADC 通道输入 CODE_DET 通道输入 触摸按键模拟通道输入	通用双向 I/O 口
9	20	5	16	VDD	芯片供电管脚	芯片供电管脚
10	2	6		P1.5/PWM/SPI_SSB/ADC[11]/TK16	通用双向 I/O 口 PWM 信号输出 ADC 通道输入 SPI_SSB 引脚 触摸按键模拟通道输入	通用双向 I/O 口

11	5	10	3	P1.4/I2C_SDA/FB/PWM[6]/CODE_DET/ TK15/LED_D0	通用双向 I/O 口 I2C 数据传输口 FB 输入 PWM 信号输出 CODE_DET 通道输入 触摸按键模拟通道输入 级联 LED 驱动输出	I2C 数据口
12	6	11	4	P1.3/I2C_SCL/TK14/PWM[7]/LED_D1	通用双向 I/O 口 I2C 时钟传输口 触摸按键模拟通道输入 PWM 信号输出 级联LED 驱动输出	I2C 时钟口
13	14	9	5	P1.2/PWM/ADC[0]/ADC_VREF/T2CP0/ TK13	通用双向 I/O 口 PWM 信号输出 ADC 通道输入 ADC_VREF 通道输入 T2CP0 引脚 触摸按键模拟通道输入	通用双向 IO 口
14	13	8	6	P1.1/PWM/ADC[1]/CLO/T2EX/TK12	通用双向 I/O 口 PWM 信号输出 ADC 通道输入 CLO 输出 T2EX 引脚 触摸按键模拟通道输入	通用双向 IO 口
15	7	7		P1.0/PWM/SPI_SCK/T2/TK11	通用双向 I/O 口 PWM 信号输出 SPI_SCK 引脚 T2 引脚 触摸按键模拟通道输入	通用双向 IO 口
16	8	12		P0.0/PWM/SPI_MOSI/T1 /TK10	通用双向 I/O 口 PWM 信号输出 SPI_MOSI 引脚 T1 引脚 触摸按键模拟通道输入	通用双向 IO 口
17	9	13	7	P0.1/PWM/SPI_MISO/ADC[2]/CODE_DET /AMP_B_OUT/TK9	通用双向 I/O 口 PWM 信号输出 SPI_MISO 引脚 ADC 通道输入 CODE_DET 通道输入 AMP_B_OUT 通道输出 触摸按键模拟通道输入	通用双向 IO 口
18	10	14	8	P0.2/PWM/UART2_RX/I2C_SCL/TK8	通用双向 I/O 口 PWM 信号输出 UART2_RX 引脚 I2C 时钟口 触摸按键模拟通道输入	通用双向 IO 口
19	11	15	10	P0.3/PWM/ADC[3]/AMP_B_INP/TK7	通用双向 I/O 口 PWM 信号输出 ADC 通道输入 AMP_B_INP 通道输入 触摸按键模拟通道输入	通用双向 IO 口
20	12	16	9	P0.4/PWM/ADC[4]/AMP_B_INN/TK6	通用双向 I/O 口 PWM 信号输出 ADC 通道输入 AMP_B_INN 通道输入 触摸按键模拟通道输入	通用双向 IO 口

备注：信号引脚复用功能设置方法详见表 15-2-5 和表 15-2-7

6 中央处理器（CPU）

6.1 CPU 简介

JZ8FC005T 系列芯片采用单周期 8051 CPU，与原来的 MCS-51 指令集完全兼容。CPU 采用流水线结构， 通常情况下，单周期 8051 CPU 的运行速度比标准 8051 处理器快 10 倍。

CPU 有以下特性：

1T 8051 CPU
兼容 8051 指令集，见指令集附录
双 DPTR，可用于数据快速搬移

6.2 寄存器描述

程序计数器 PC

程序计数器 PC 寄存器为 16 位，是专门用来控制指令执行顺序的寄存器，它没有寄存器地址。单片机上电或复位后，PC 值为 0，单片机从零地址开始执行程序。

累加器 ACC

累加器 ACC 是一个常用的专用寄存器，指令系统中采用 A 作为累加器的助记符，常用于存放算术或逻辑运算的操作数及运算结果。

通用寄存器 B

B 在乘除法运算中需要和ACC 配合使用。MUL AB 指令把ACC 和 B 中 8 位无符号数相乘，所得的 16 位乘积的低字节存放在A 中，高字节存放在 B 中。DIV AB 指令用 B 除以 A，整数商存放在 A 中，余数存放在 B 中。寄存器B 还可以用作通用暂存寄存器。

堆栈指针 SP

堆栈指针SP 是一个 8 位专用寄存器。它指示出堆栈顶部在内部 RAM 块中的位置。系统复位后，SP 初始化位 07H，使得堆栈事实上由 08H 单元开始，考虑 08H~1FH 单元分别 属于工作寄存器组 1~3，若在程序设计中用到这些区，则最好SP 改变为 80H 或更大的为宜。

数据指针 DPTR

数据指针DPTR0/DPTR1 是两个 16 位专用寄存器，它们的高位字节寄存器用DP0H/DP1H表示，低位字节寄存器用 DP0L/DP1L 表示，通过 DPS(PSW.1)可选择使用 DPTR0/DPTR1。每个 DPTR 既可以作为一个 16 位寄存器来处理，也可以作为 2 个独立的 8 位寄存器DP0H/DP1H 和 DP0L/DP1L 来处理。

状态寄存器 PSW

状态寄存器 PSW 是 CPU 的状态寄存器。在 CPU 做算术运算或者逻辑运算时，对应的 PSW 状态位会发生改变。

表 6-2-1 累加器 ACC

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

表 6-2-2 通用寄存器B

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

表 6-2-3 堆栈指针 SP

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

表 6-2-4 数据指针 DP0L

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

表 6-2-5 数据指针 DP0H

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

表 6-2-6 数据指针 DP1L

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

表 6-2-7 数据指针 DP1H

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

表 6-2-8 状态寄存器 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
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7	CY	进位标志位 0：算术或逻辑运算中，没有进位或借位发生 1：算术或逻辑运算中，有进位或借位发生
6	AC	辅助进位标志位 0：算术或逻辑运算中，没有辅助进位或借位发生 1：算术或逻辑运算中，有辅助进位或借位发生
5	F0	F0 标志位 用户自定义标志位
4~3	RS	R0~R7 寄存器页选择位 00：页 0（映射到 00H-07H) 01：页 1（映射到 08H-0FH) 10：页 2（映射到 10H-17H) 11：页 3（映射到 18H-1FH)
2	OV	溢出标志位 0：没有溢出发生 1：有溢出发生
1	DPS	DPTR 选择寄存器，0 为选择 DPTR0，1 为选择 DPTR1
0	P	奇偶校验位 0：累加器 A 值为 1 的位数为偶数 1：累加器 A 值为 1 的位数为奇数

表 6-2-9 寄存器 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
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~0	SPMAX	寄存器 SPMAX 用于记录 SP 的最大值，用户在应用程序中可查看此寄存器来判断堆栈有没有溢出风险

7 存储器系统

7.1 随机数据存储器（RAM）

JZ8FC005T 系列芯片提供了 256 字节内部 RAM 和 4K 字节外部 RAM，存储器地址分配如下：

- 低位 128 字节的内部 RAM（地址：00H ~ 7FH）可直接寻址或间接寻址。
- 高位 128 字节的内部 RAM（地址：80H ~ FFH）只能间接寻址。
- 外部 4K 字节外部 RAM（地址：0000H ~ 0FFFH）可通过MOVX 指令间接寻址。

图 7-1-1 RAM 组织结构图

7.2 特殊功能寄存器（SFR）

JZ8FC005T 系列芯片提供了兼容传统 8051 的 SFR 分布，SFR 和高 128 字节内部 RAM 共用地址 80H ~ FFH, 只能直接寻址，SFR 映射如表 7-2-1 所示。

表 7-2-1 特殊功能寄存器（SFR）映射表

	可位寻址	不可位寻址						
	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	TK0MSL	TK0MSH	TK1MSL	TK1MSH	TK2MSL	TK2MSH	INDEX
E8H	EXIE	TK3MSL	TK3MSH	TK4MSL	TK4MSH	TK5MSL	TK5MSH	IDCODE
E0H	ACC	LEDAT0	LEDAT1	LEDUTL	LEDUTH	LEFLG	MDUCON	MDUDAT
D8H	PWM0NUML	PWM0NUMH	PWMEN	PWMUPD	PWMCMAX	PWMCON	PWMCFG	PWMDIVL
D0H	PSW	PWMDIVH	PWMDUTL	PWMDUTH	PWMAIF	PWMBIF	PWMCIF	LVDCON
C8H	T2CON	T2MOD	T2CL	T2CH	TL2	TH2	-	-
C0H	TKCON	TKCFG	TKMTS	TKIF	TKCKS	TKPWC	LEWTML	LEWTMH
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	TKCHS0	TKCHS1	TKCHS2	TKCHS3	TKCHS4	TKCHS5	-
88H	TCON	TMOD	TL0	TL1	TH0	TH1	IT1CON	IT0CON
80H	P0	SP	DP0L	DP0H	DP1L	DP1H	PWCON	PCON

由于 SFR 地址空间有限，JZ8FC005T 系列芯片在外部RAM 地址空间增加了扩展特殊功能寄存器，扩展特殊功能寄存器映射如图表 7-2-2 所示。

表 7-2-2 扩展特殊功能寄存器映射表

	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	RCMSLL	RCMSLH	RCMSHL	RCMSHH	RCTAGL	TCTAGH	-
8018H	P30F	RCCON	HVTH	VCKDL	VCKDH	STEP	STEPNUM	-
8080H	CKCON	CKSEL	CKDIV	IHCFGL	IHCFGH	ILCFG	XHISEL	-
8088H	ADCALL	ADCALH	ADCPDLL	ADCPDLH	ADCPDHL	ADCPDHH	ADPWMTRIG	ADOPC
8090H	PFGCFGL	PFGCFGH	PFGTRIML	PFGTRIMH	ACPBC	-	-	-
8098H	PWMPS	PWMHS	PWMFBC	PWMFBS	PWMSBC	PWMBD	-	-
80A0H	T2CPPS	T2CPCHS	T2CPF	T2CPL	T2CPH	-	-	-
80A8H	TMCON	TMSNU	-	-	-	-	-	-
80B0H	SWICON	SWIDAT	SWISTA	SWIOVT	-	-	-	-
80B8H	AMPCON	AMPAOS	AMPOUT	-	-	-	-	-
80C0H	BZCON	BZDIVL	BZDIVH	BZDUTL	BZDUTH	-	-	-
80D0H	DCCR	DCDS	DCDB	DCBIT0HH	DCBIT0HL	DCBIT0LH	DCBIT0LL	DCBIT1HH
80D8H	DCBIT1HL	DCBIT1LH	DCBIT1LL	DCOVRH	DCOVRL	DCFCR	-	-
80E0H	DC1CR	DC1DS	DC1DB	DC1BIT0HH	DC1BIT0HL	DC1BIT0LH	DC1BIT0LL	DC1BIT1HH
80E8H	DC1BIT1HL	DC1BIT1LH	DC1BIT1LL	DC1OVRH	DC1OVRL	DC1FCR	-	-
80F8H	TSCMD	TSSTA	TSSWC	-	-	-	-	-
8100H	SPMAX	I2CIOS	-	-	-	-	-	-
8118H	UDCKS1	UDCKS2	-	-	-	-	FTCTL	TPCTL
8120H	P00C	P01C	P02C	P03C	P04C	P05C	P06C	P07C
8028H	P10C	P11C	P12C	P13C	P14C	P15C	P16C	P17C
8030H	P20C	-	-	-	-	-	-	-
8138H	P30C	-	-	-	-	-	-	-
FC00H	MECON	FSCMD	FSDAT	LOCK	PADRD	PTSL	PTSH	REPSET

7.3 Flash 存储器

7.3.1 功能简介

Flash 存储器包含 64K 字节 Flash 主数据区，Flash 存储器可重复擦写。Flash 存储器由一组特定的寄存器控制，用户可用这些寄存器进行读写擦、设置写保护等操作。

7.3.2 Flash 存储器组织结构

- Flash 由若干个扇区组成，扇区是进行擦除操作的最小单位，每个扇区大小为 512 字节
- Flash 可以按功能划分为程序区和数据区，划分单位为 512 字节，程序区用于存储用户的程序，数据区是用于存储一些掉电需要保存的数据。

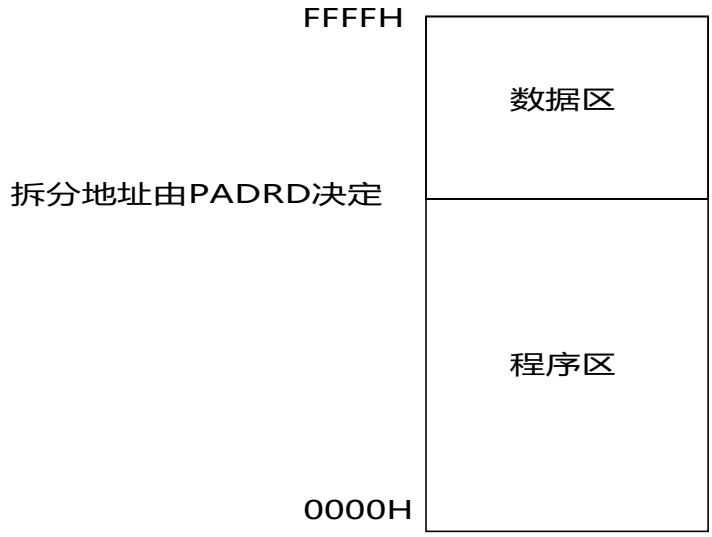


图 7-3-1 64K Flash 存储器结构

7.3.3 Flash 寄存器描述

表 7-3-3-1 寄存器 MECON

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

位编号	位符号	说明
7	-	-
6	DPSTB	IDLE/STOP 模式下 Flash 进入睡眠模式控制位 0：IDLE/STOP 模式下，Flash 处于正常工作模式 1：IDLE/STOP 模式下，Flash 进入睡眠模式 *备注：如果 DPSTB=1，当芯片进入 IDLE/STOP 模式，Flash 也同时进入睡眠模式，Flash 在睡眠模式的功耗为 50nA，当芯片退出 IDLE/STOP 模式，Flash 也同时退出睡眠模式。*
5~1	-	-
0	BOOT	设置软复位后程序启动空间选择位域 0：软复位后程序从 FLASH 启动运行 1：软复位后程序从 XRAM 启动运行

表 7-3-3-2 寄存器 FSCMD

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

位编号	位符号	说明
7~3	-	-
2~0	CMD	命令寄存器 000：无操作 100：Flash 整片擦除 001：读 Flash 数据区 010：写 Flash 数据区 011：擦除 Flash 数据区一个扇区 101：读 Flash 程序区 110：写 Flash 程序区 111：擦除 Flash 程序区一个扇区 *备注：* *1. 擦除命令执行后 CMD 自动清零。* *2. FLASH 写操作以 4 个字节为单位，每次操作写入 4 个字节数据。*

表 7-3-3-3 寄存器 FSDAT

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

位编号	位符号	说明
7~0	FSDAT	Flash 数据寄存器

表 7-3-3-4 寄存器 LOCK

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

位编号	位符号	说明
写操作		
7~0	LOCK	28H：对 Flash 可编程区解锁 29H：对 Flash 程序区解锁 2AH：对 Flash 数据区解锁 AAH：Flash 加锁，不能进行写擦操作
读操作		
7~4	-	
3	FLKF	可编程区解锁标志，1 表示已解锁
2	PLKF	程序区解锁标志，1 表示已解锁
1	DLKF	数据区解锁标志，1 表示已解锁
0	-	-

表 7-3-3-6 寄存器 PADRD

FC04H	7	6	5	4	3	2	1	0
PADRD	-	PADRD[7:0]						
R/W	R/W							
初始值	0	1	1	1	1	1	1	0

位编号	位符号	说明

7~0	PADRD	程序区和数据区划分配置寄存器 程序区和数据区以 512 字节为单位进行划分，当 PADRD>0 时， 程序区的地址空间为: 0 ~ (PADRD×512 - 1), 数据区的地址空间为: (PADRD×512) ~ FFFFH. *备注：* *1. 当 PADRD=0 时，整个 Flash 空间都是数据空间。* *2. PADRD 的最大值分别为 7FH，即至少要划分一个扇区作为数据空间。*

表 7-3-3-7 寄存器 PTS

FC05H	7	6	5	4	3	2	1	0
PTSL	PTS[7:0]							
R/W	R/W							
初始值	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				
初始值	-	-	-	0	0	0	0	0

位编号	位符号	说明
15~13	-	-
12~0	PTS	目标地址指针寄存器 *备注：* *如为写操作，PTS 必须设置为 4 的整数倍。*

7.3.4 Flash 控制例程

Flash 划分程序区和数据区

例如，64K 的Flash 空间划分最后 512 字节为数据空间，其余为程序空间，程序如下：

```
--------------------------------------------------------------------------------
PADRD = 0x7F; //程序区空间地址为：0~0xFDFF,数据区空间地址为：0xFE00~0xFFFF
--------------------------------------------------------------------------------
```

备注：以上设置数据区在 FLASH 中的物理地址是 0xFE00~0xFFFF，但是逻辑地址是 0x0000~0x01FF，读写数据区时应填写逻辑地址。

数据空间扇区擦除

例如，需要擦除数据空间扇区n，程序如下：

```
--------------------------------------------------------------------------------
FSCMD = 0;    //设置 CMD 为 0
LOCK = 0x2A; //数据空间解锁
PTSH = (unsigned char)((n*0x200)>>8); //设置扇区高位地址
PTSL = (unsigned char)(n*0x200); //设置扇区低位地址
FSCMD = 3;    //设置擦除命令
LOCK = 0xAA; //FLASH 加锁
```

备注：扇区序号 n=0、1、2……。

数据空间写入数据

例如，往数据空间地址为 0~255 写入数据 0xAA，程序如下：

```
unsigned char i;
FSCMD = 0; // 设 置 CMD 为 0
LOCK = 0x2A; //数据空间解锁
PTSH = (unsigned char)(0>>8); //设置数据首地址高 8 位
PTSL = (unsigned char)0; //设置数据首地址低 8 位
FSCMD = 2;  //设置写命令
for(i=0;i<256;i++)
{
    FSDAT = 0xAA;
}
FSCMD = 0;
LOCK = 0xAA; //FLASH 加锁
```

备注：

1．当连续写入数据时，只需设置首地址，每次写 FSDAT 后，数据指针寄存器 PTS 会自动累加。

2．读写数据区时，设置的地址是数据区的逻辑地址，而不是 FLASH 的物理地址，逻辑地址是从 0 开始的。

3．写入数据的时候，必须以字为单位，即写入的字节数要为 4 的倍数；

数据空间读出数据

例如，从数据空间地址为 0~255 读出数据到指针pBuf，程序如下：

```
unsigned char i, *pBuf;
FSCMD = 0;  //设置 CMD 为 0
LOCK = 0x2A; //数据空间解锁
PTSH = (unsigned char)(0>>8); //设置数据首地址高 8 位
PTSL = (unsigned char)0; //设置数据首地址低 8 位
FSCMD = 1;  //设置读命令
for(i=0;i<256;i++)
{
    *pBuf++ = FSDAT ; //连续写入数据
}
FSCMD = 0;
LOCK = 0xAA; //FLASH 加锁
```

备注：当连续读出数据时，只需设置首地址，每次读 FSDAT 后，数据指针寄存器 PTS 会自动累加。

7.4 外部 RAM 映射为程序空间

4K 字节的外部 RAM 可以映射为程序空间使用，映射地址为 0000H~0FFFH。用户可以下载程序到外部 RAM 空间，把 BOOT（详见寄存器 MECON）的值设置为 1，然后执行软复位，复位后程序从外部 RAM 空间开始执行（此时映射地址为 0000H~0FFFH）。映射程序区用来实现IAP 等功能特别方便。

8 中断系统

8.1 功能简介

JZ8FC005T 系列芯片有一个增强的中断控制系统，共有 15 个中断入口，每个中断入口有若干中断源，每个中断源有 2 级中断优先级。每个中断源都有独立的中断向量、优先级设置位、中断使能位、中断标志。CPU 在响应中断后，进入该中断对应的中断服务程序，接到 RETI 指令后将返回中断前状态。如果同时有多个有效中断产生中断请求，CPU 将根据设置的中断优先级依次响应；如果优先级相同，则根据它们的自然优先级（中断入口地址从低到高）依次响应。

8.2 中断逻辑

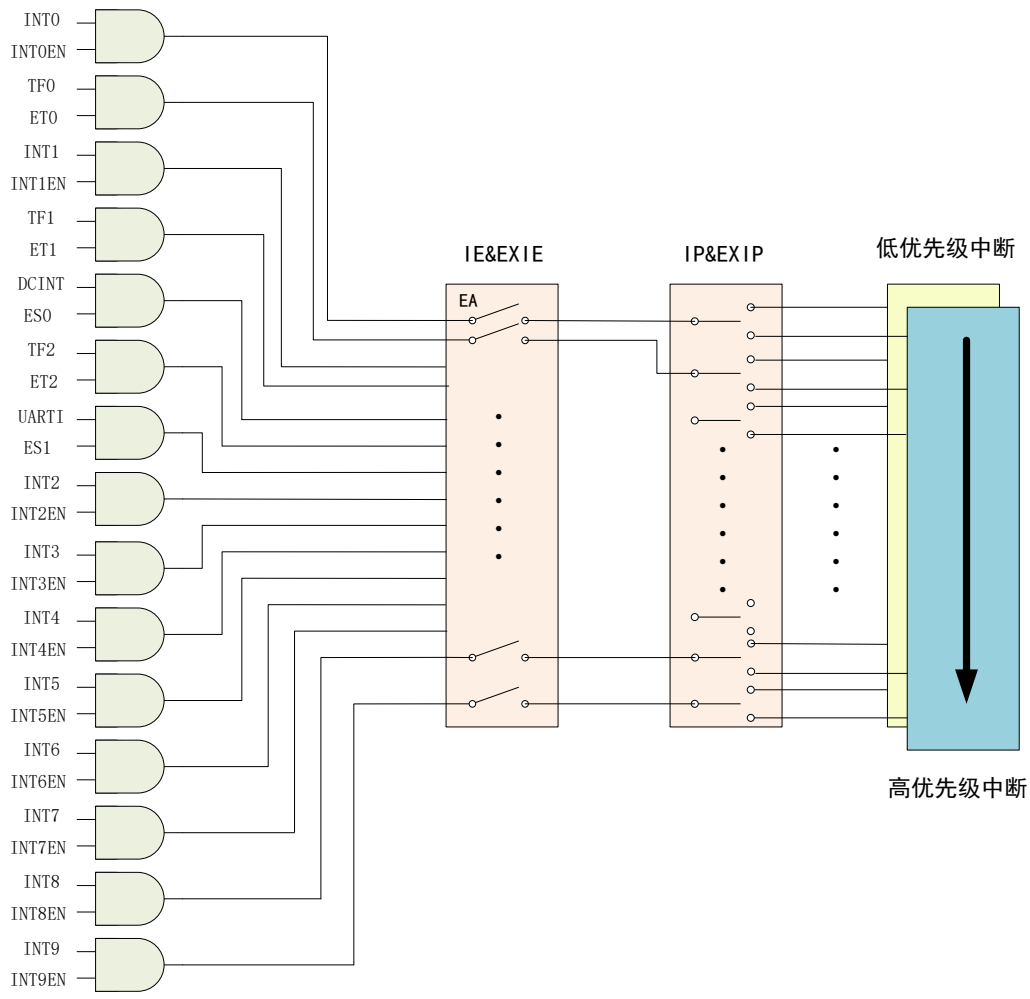


表 8-2-1 中断逻辑图

8.3 中断向量表

中断	中断源	向量	默认优先级
INT0	INT0	03H	0
TF0	定时器 0	0BH	1
INT1	INT1	13H	2
TF1	定时器 1	1BH	3
DCINT	无线充解码	23H	4
TF2	定时器 2	2BH	5
UART1	UART1	33H	6
INT2	ADC/外部中断 2	3BH	7
INT3	UART2/TK/外部中断 3	43H	8
INT4	LVD/外部中断 4	4BH	9
INT5	SPI/外部中断 5	53H	10
INT6	I2C/SWI/外部中断 6	5BH	11
INT7	WDT/外部中断 7	63H	12
INT8	TMC/外部中断 8	6BH	13
INT9	PWM/外部中断 9	73H	14

8.4 中断控制寄存器

表 8-4-1 寄存器 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
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7	EA	全局中断使能控制位 0：全局中断关闭 1：全局中断打开
6	ES1	UART1 中断使能控制位 0：UART1 中断关闭 1：UART1 中断打开
5	ET2	定时器 2 中断使能控制位 0：定时器 2 中断关闭 1：定时器 2 中断打开
4	ES0	无线解码中断使能控制位 0：无线解码中断关闭 1：无线解码中断打开
3	ET1	定时器 1 中断使能控制位 0：定时器 1 中断关闭 1：定时器 1 中断打开
2	EX1	外部中断 1 使能控制位 0：外部中断 1 中断关闭 1：外部中断 1 中断打开
1	ET0	定时器 0 中断使能控制位
0	EX0	外部中断 0 使能控制位 0：外部中断 0 中断关闭 1：外部中断 0 中断打开

表 8-4-2 寄存器 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
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7	INT9EN	中断 9 使能控制位（中断 9 用于 SAMPLE/PWM/外部中断 9） 0：关闭 1：打开
6	INT8EN	中断 8 使能控制位（中断 8 用于RTC/比较器计数器/外部中断 8） 0：关闭 1：打开
5	INT7EN	中断 7 使能控制位（中断 7 用于 WDT/MOTOR/外部中断 7） 0：关闭 1：打开
4	INT6EN	中断 6 使能控制位（中断 6 用于I2C/模拟比较器/外部中断 6） 0：关闭 1：打开
3	INT5EN	中断 5 使能控制位（中断 5 用于SPI/时钟监控/外部中断 5） 0：关闭 1：打开
2	INT4EN	中断 4 使能控制位（中断 4 用于LVD/外部中断 4） 0：关闭 1：打开
1	INT3EN	中断 3 使能控制位（中断 3 用于UART2/TK/外部中断 3） 0：关闭 1：打开
0	INT2EN	中断 2 使能控制位（中断 2 用于 ADC/外部中断 2） 0：关闭 1：打开

备注：EXIE 的使能控制位是对应中断向量的，各中断源的中断开关也要另外打开。例如：要开启外部中 2 的中断，除了设置 INT2EN 为 1，EPIE2（外部中断 2 使能位）也要设为 1。

表 8-4-3 寄存器 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
初始值	-	0	0	0	0	0	0	0

位编号	位符号	说明
7	-	-
6	PS1	UART1 优先级控制位 0：低优先级 1：高优先级
5	PT2	定时器 2 优先级控制位 0：低优先级 1：高优先级
4	PS0	DCINT 优先级控制位 0：低优先级 1：高优先级
3	PT1	定时器 1 优先级控制位

		0：低优先级 1：高优先级
2	PX1	外部中断 1 优先级控制位 0：低优先级 1：高优先级
1	PT0	定时器 0 优先级控制位 0：低优先级 1：高优先级
0	PX0	外部中断 0 优先级控制位 0：低优先级 1：高优先级

表 8-4-4 寄存器 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
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7	PX9	中断 INT9 优先级控制位 0：低优先级 1：高优先级
6	PX8	中断 INT8 优先级控制位 0：低优先级 1：高优先级
5	PX7	中断 INT7 优先级控制位 0：低优先级 1：高优先级
4	PX6	中断 INT6 优先级控制位 0：低优先级 1：高优先级
3	PX5	中断 INT5 优先级控制位 0：低优先级 1：高优先级
2	PX4	中断 INT4 优先级控制位 0：低优先级 1：高优先级
1	PX3	中断 INT3 优先级控制位 0：低优先级 1：高优先级
0	PX2	中断 INT2 优先级控制位 0：低优先级 1：高优先级

8.5 外部中断

8.5.1 外部中断介绍

INT0 和 INT1 在标准 8051 的基础上，增加了可选择任意输入口作为中断触发源的功能。系统还扩展了 8 个中断入口 INT2~INT9 作为外部中断，每个中断入口也可选择任意输入口作为中断触发源，扩展外部中断可单独设置上升沿或下降沿触发中断。每个外部中断都可以用于 STOP 模式唤醒。EPIF 为 INT2~INT9 外部中断状态寄存器。INT2~INT9 对应的各个配置寄存器 EPCON0~EPCON7 可通过配置索引寄存器 INDEX 为 0~7 来访问。

备注：INT0 和 INT1 可选择上升沿或下降沿触发，选择位分别为 IT0 和 IT1，详见寄存器 TCON 相关描述。

8.5.2 外部中断寄存器

表 8-5-2-1 寄存器 IT0CON

8FH	7	6	5	4	3	2	1	0
IT0CON	-	-	-	IT0PS[4:0]				
R/W	-	-	-	R/W				
初始值	-	-	-	0	0	0	0	0
位编号	位符号	说明						
7~5	-	-						
4~0	IT0PS[4:0]	INT0 中断管脚选择位 编号和管脚对应表参考**表 8-5-2-6**						

表 8-5-2-2 寄存器 IT1CON

8EH	7	6	5	4	3	2	1	0
IT1CON	-	-	-	IT1PS[4:0]				
R/W	-	-	-	R/W				
初始值	-	-	-	0	0	0	0	0
位编号	位符号	说明						
7~5	-	-						
4~0	IT1PS[4:0]	INT1 中断引脚选择位 编号和管脚对应表参考**表 8-5-2-6**						

表 8-5-2-3 寄存器 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
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						

7	EPIE9	外部中断 9 使能位
6	EPIE8	外部中断 8 使能位
5	EPIE7	外部中断 7 使能位
4	EPIE6	外部中断 6 使能位
3	EPIE5	外部中断 5 使能位
2	EPIE4	外部中断 4 使能位
1	EPIE3	外部中断 3 使能位
0	EPIE2	外部中断 2 使能位

表 8-5-2-4 寄存器 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
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7	EPIF9	外部中断 9 中断标志位，写 1 清零
6	EPIF8	外部中断 8 中断标志位，写 1 清零
5	EPIF7	外部中断 7 中断标志位，写 1 清零
4	EPIF6	外部中断 6 中断标志位，写 1 清零
3	EPIF5	外部中断 5 中断标志位，写 1 清零
2	EPIF4	外部中断 4 中断标志位，写 1 清零
1	EPIF3	外部中断 3 中断标志位，写 1 清零
0	EPIF2	外部中断 2 中断标志位，写 1 清零

表 8-5-2-5 寄存器 EPCON

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

备注：EPCON 为带索引寄存器，设置 INDEX=0~7 分别对应 EPCON0~EPCON7

位编号	位符号	说明
7	EPPL	外部中断触发沿选择位 0：上升沿 1：下降沿
6~5	-	-
4~0	EPPS[4:0]	中断引脚选择位域 编号和引脚对应表参考**表 8-5-2-6**

表 8-5-2-6 中断引脚编号索引表

引脚名称	编号	引脚名称	编号
P00	0	P20	16
P01	1	P30	其他
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		

8.5.3 外部中断控制例程

外部中断 0/1 控制例程

例如，使能外部中断 0，程序如下:

```
--------------------------------------------------------------------------------
void INT0_init(void)
{
    P10F = 1;           //P10 设置为输入功能
    IT0CON = 8;         //设置 P10 为 INT0 中断引脚
    EX0 = 1;            //INT0 中断使能
    IE0 = 1;            //外部中断 0 中断使能
    IT0 = 1;            //设置 INT0 为下降沿触发
    PX0 = 1;            //设置 INT0 中断为最高优先级
    EA = 1;             //总中断使能

}
void INT0_ISR (void) interrupt 0
{
    //外部中断 0 中断服务程序
}
--------------------------------------------------------------------------------
```

例如，使能外部中断 1，程序如下:

```
--------------------------------------------------------------------------------
void INT1_init(void)
{
    P10F = 1;           //P10 设置为输入功能
    IT1CON = 8;         //设置 P10 为 INT1 中断引脚
    EX1 = 1;            //INT1 中断使能
    IE1 = 1;            //外部中断 1 中断使能
    IT1 = 1;            //设置 INT1 为下降沿触发
    PX1 = 1;            //设置 INT1 中断为最高优先级
    EA = 1;             //总中断使能
}
void INT1_ISR (void) interrupt 2
{
    //外部中断 1 中断服务程序
}

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

外部中断 2~4 控制例程

以外部中断 2 为例，设置P10 为外部中断 2 中断输入引脚并开启外部中断 2，程序如下:

```
---------------------------------------------------------------------------------
void INT2_init(void)
{
    P10F = 1;                   //P10 设置为输入模式
    INDEX = 0;                  //EPCON 为带索引的寄存器，设置 INDEX=0 对应 INT2
    EPCON =  (1<<7) | 8;        //设置 P10 为 INT2 中断引脚，下降沿触发
    INT2EN = 1;                 //外部中断 2 中断使能
    EPIE |= 0x01;               //INT2 中断使能
    EA = 1;                     //总中断使能
}
void INT2_ISR (void) interrupt 7
{
    if(EPIF & 0x01)             //判断外部中断 2 中断标志
    {
        EPIF = 0x01;            //中断标志写 1 清 0
        //外部中断 2 中断服务程序
        ......
    }
}
---------------------------------------------------------------------------------
```

9 时钟系统

9.1 时钟系统介绍

JZ8FC005T 系列芯片共支持以下时钟源：

- 内置 24MHz RC 振荡器
- 内置 131KHz RC 振荡器
- 内置 20 - 40MHz 可编程 RC 振荡器
- 支持外部 1 - 24MHz 晶体振荡器
- 支持外部 1 - 24MHz 时钟输入

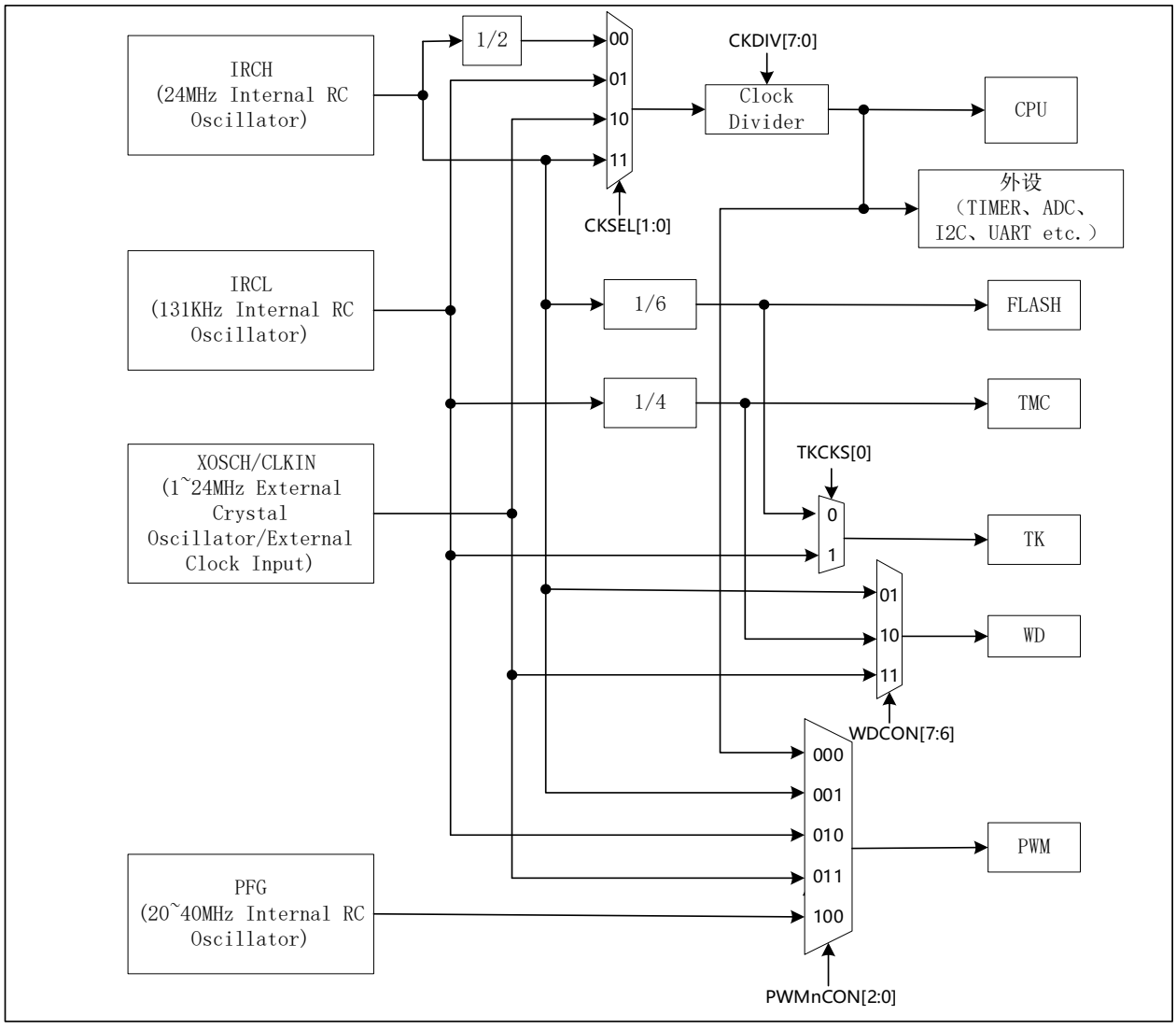


图 9-1-1 时钟结构图

用户可独立的管理各个时钟源，每个时钟源都可以单独打开或关闭，从而可以灵活控制功耗。

所有时钟源都可设置为系统时钟，也可分配到各种外设中，作为外设的时钟源，详细请参考外设部分介绍。

9.1.1 时钟专用名称定义

名称缩写	描述
IRCH	内置 24MHz RC 振荡器
IRCL	内置 131KHz RC 振荡器
PFG	内置 20~40MHz RC 振荡器
XOSCH	外部 1~24MHz 晶体振荡器
CLKIN	外部 1~24MHz 时钟输入

9.1.2 内置 24MHz RC 振荡器（IRCH）

IRCH 是芯片上电后默认的系统时钟，可通过寄存器 CKCON 的 IHCKE 位打开或关闭。芯片出厂后，IRCH 的频率校正为 24MHz@3.3V/25℃，时钟精度为±1%。

9.1.3 内置 131 KHz RC 振荡器（IRCL）

IRCL 可通过寄存器 CKCON 的 ILCKE 位打开或关闭。IRCL 设为系统时钟可实现系统低功耗。芯片出厂后，IRCL 的频率校正为 131KHz@3.3V/25℃，时钟精度为±2%。

9.1.4 内置 PFG 振荡器

PFG 振荡器由PFGCKE 使能，主要是作为PWM 的时钟源，专门为加湿器应用而设计。

■ **PFG 校准机制**

由于芯片出厂后 PFG 时钟频率存在偏差，并且具体应用中要求的 PFG 时钟频率也不相同，所以要求芯片具备校正 PFG 到目标频率的机制。校正模块以系统时钟为基准，可对 PFG 时钟进行计数或测量，应用软件再根据计数或测量值计算出当前 PFG 时钟频率，然后再进行频率调整，重复进行测量调整最终可使 PFG 时钟频率逼近目标频率。

校正模块有两种工作模式：计数模式和测量模式。

计数模式用于手动测量 PFG 时钟频率，设置 MODE（RCCON[7:6]）为 1 后启动 PFG 时钟计数，MODE 设置为 0 后计数停止，停止计数后计数值存入寄存器 RCMS（RCMSHH/RCMSHL/RCMSLH/RCMSLL）中。在应用中，用户可以在确定的时间段内启动 PFG 时钟计数，通过对计数值 RCMS 简单的计算，可以得到 PFG 的频率。

测量模式在若干个系统时钟周期内对 PFG 时钟进行计数，通过计数值来推算出 PFG 时钟频率。设置 MODE 为 2 开始测量，测量完成后计数值存入 RCMS 寄存器，MODE 自动清 0。为了提高测量精度，应尽量延长测量周期时间长度，可以通过配置寄存器 VCKD（VCKDH/VCKDL），设置一个测量周期为系统时钟周期的 VCKD 倍。这样，测量之后经过对计数值的简单计算，可以得到 PFG 的频率。计算公式如下：

PFG 的周期＝ (系统时钟周期x VCKD) ÷RCMS

■ **PFG 时钟跳频功能**

PFG 跳频功能专门为加湿器应用而设计，目的是为了降低加湿器产品对外射频干扰幅度。

PFG 跳频以校准后的时钟频率为中心点，以 STEP 设置的步进前后摆动，最大摆动幅度由 STEPNUM 设置，每次频率调整后，PFG 频率的设置值保存在 PFGRT，即跳频功能使能后，实际是 PFGRT 决定 PWM 的频率。跳频功能由TRIMEN（RCCON[3]）使能，每次跳频由 PWM 触发，PWM0NUM 可设置触发PFG 调整的周期数。

例如，校准后的 PFG 时钟频为 24MHz，24MHz 对应的 PFGCFG 值即为中心点，STEP 设置为 5，STEMNUM 设置为 3，PWM0NUM 设为 2，跳频功能使能后，每 3（PWM0NUM+1）个 PWM 周期，PFG 时钟频率调整一次，PFGRT 与 PFGCFG 的差值按以下顺序变化：5、10、15、10、5、0、-5、-10、-15、-10、-5、0、5、10、15……

注意，当需要改变中心点频率时，必须先把跳频功能关闭（即设置 TRIMEN=0），更新完中心点频率后再使能跳频功能。

9.1.5 外部高速晶体谐振器（XOSCH）和外部时钟输入（CLKIN）

XOSCH 和 CLKIN 共用连接引脚，所以在应用时只能二选一，由寄存器 CKCON 的 XHCS 位选择。XOSCH 可通过寄存器 CKCON 的 XHCKE 位打开或关闭，另外标志位 XHSTA 指示 XOSCH 时钟是否已稳定。

备注：硬件设计时晶振负载电容地尽量靠近芯片 GND 引脚。

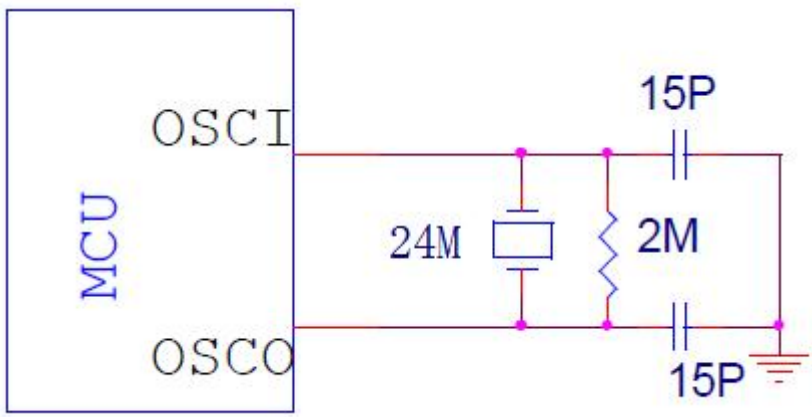


图 9-1-5-1 外部高速晶振典型电路图

9.2 时钟控制寄存器描述

表 9-2-1 寄存器 CKCON

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

位编号	位符号	说明
7	ILCKE	IRCL 使能控制位 1：打开 0：关闭 *备注：* *该位为 1 时，时钟模块打开，但是该位为 0 时，如果系统或者其他模块选择了该时钟源，该时钟仍然会被打开。*
6	IHCKE	IRCH 使能控制位 1：打开 0：关闭 *备注：* *该位为 1 时，时钟模块打开，但是该位为 0 时，如果系统或者其他模块选择了该时钟源，该时钟仍然会被打开。*
5	PFGCKE	PFG 使能控制位 1：打开 0：关闭
4	XHCS	XOSCH 时钟源选择位 0：XOSCH 时钟源为外挂高速晶振 1：XOSCH 时钟源为外部时钟输入 *备注：* *改写 XHCS 的值之前必须确保系统时钟没有选择外部时钟，否则改写动作将无效。*
3~2	-	-
1	XHCKE	XOSCH 使能控制位 1：打开 0：关闭 *备注：* *1 位有效时，时钟模块开启，但是该位为 0 时，如果系统或者其他模块选择了该时钟源，该时钟仍然会被打开。* *2 由于 XOSCH 为外部时钟，所以若想所以启动 XOSCH 还需要把对应的引脚功能配置为 XOSCH 的功能。*
0	XHSTA	XOSCH 时钟稳定信号标志，1 有效

表 9-2-2 寄存器 IHCFG

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

位编号	位符号	说明
15~9	-	-
8~0	IHCFG	IRCH 时钟频率设置寄存器

表 9-2-3 寄存器 ILCFG

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

位编号	位符号	说明
7~6	-	-
5~0	ILCFG	内部 131KHz 时钟配置寄存器

表 9-2-4 寄存器 RCCON

E9H	7	6	5	4	3	2	1	0
RCCON	MODE[1:0]		-	-	TRIMEN	-	-	-
R/W	R/W		-	-	R/W	-	-	-
初始值	0	0	-	-	0	-	-	-

位编号	位符号	说明
7~6	MODE	工作模式选择位 01：计数模式，设置 MODE 为 0 则退出计数模式 10：测量模式，完成后 MODE 自动清 0
-	-	-
-	-	-
3	TRIMEN	PFG 时钟跳频功能使能，1 有效
2-0	-	-

表 9-2-5 寄存器 VCKDL、VCKDH

EBH	7	6	5	4	3	2	1	0
VCKDL	VCKD[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
ECH	7	6	5	4	3	2	1	0
VCKDH	VCKD[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
15~0	VCKD	测量模式下，参考时钟分频倍数，为 VCKD 倍分频(VCKD>1)

表 9-2-6 寄存器 RCTAGL、RCTAGH

94H	7	6	5	4	3	2	1	0
RCTAGL	RCTAGL[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
95H	7	6	5	4	3	2	1	0

RCTAGH	RCTAGH[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
15~0	RCTAG	测量模式下，目标时钟分频倍数，为 RCTAG 倍分频(RCTAG>=1)						

表 9-2-7 寄存器 RCMSLL、RCMSLH、RCMSHL、RCMSHH

96H	7	6	5	4	3	2	1	0
RCMSLL	RCMS[7:0]							
R/W	R							
初始值	0	0	0	0	0	0	0	0
97H	7	6	5	4	3	2	1	0
RCMSLH	RCMS[15:8]							
R/W	R							
初始值	0	0	0	0	0	0	0	0
9EH	7	6	5	4	3	2	1	0
RCMSHL	RCMS[23:16]							
R/W	R							
初始值	0	0	0	0	0	0	0	0
9FH	7	6	5	4	3	2	1	0
RCMSHH	RCMS[31:24]							
R/W	R							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
31~0	RCMS	计数模式完成后，存放计数结果 测量模式完成后，存放测量结果						

9.3 系统时钟

系统时钟控制由寄存器 CKCON、CKSEL、CKDIV 完成。通过这些寄存器组，可以单独设置各时钟源的开关、系统时钟的切换和分频等操作。

9.3.1 系统时钟结构图

系统时钟结构图见图 9-3-1。

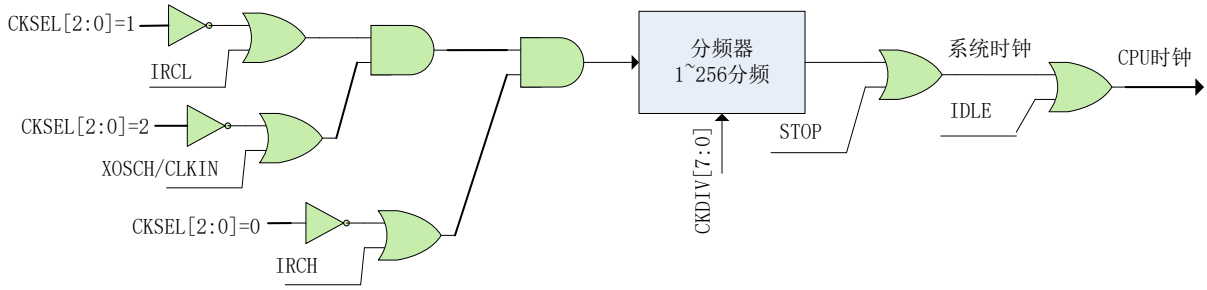


图 9-3-1 系统时钟结构图

9.3.2 系统时钟控制寄存器描述

表 9-3-2-1 寄存器 CKSEL

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

位编号	位符号	说明
7~3	-	-
2~0	CKSEL	系统时钟选择位 000：IRCH 二分频 001：IRCL 010：XOSCH/CLKIN 011：IRCH

表 9-3-2-2 寄存器 CKDIV

8082H	7	6	5	4	3	2	1	0
CKDIV	CKDIV[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
7~0	CKDIV	系统时钟分频： 00H：不分频 01H：2 分频 02H：3 分频 03H：4 分频 …… FFH：256 分频						

9.3.3 系统时钟控制方法及例程

设置系统时钟为 IRCH 的二分频

设置系统时钟为IRCH，程序如下：

```
--------------------------------------------------------------------------------
#define IHCKE                   (1<<6)
#define CKSEL_HALF_IRCH         0
void Sys_Clk_Set_Half_IRCH(void)
{
    CKCON |= IHCKE;                                 //IRCH 时钟使能
    CKSEL = (CKSEL&0xF8) | CKSEL_HALF_IRCH;         //系统时钟切换到 IRCH 的二分频
}
--------------------------------------------------------------------------------
```

设置系统时钟为 IRCH

设置系统时钟为IRCH，程序如下：

```
--------------------------------------------------------------------------------
#define IHCKE               (1<<6)
#define CKSEL_IRCH          3
void Sys_Clk_Set_IRCH(void)
{
    CKCON |= IHCKE;                                 //IRCH 时钟使能
    CKSEL = (CKSEL&0xF8) | CKSEL_IRCH;              //系统时钟切换到 IRCH
}
--------------------------------------------------------------------------------
```

设置系统时钟为 IRCL

设置系统时钟为IRCL，程序如下：

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

```
#define ILCKE               (1<<7)
#define CKSEL_IRCL          1
void Sys_Clk_Set_IRCL(void)
{
    CKCON |= ILCKE;                         //IRCL 时钟使能
    Delay_ms(1);                            //使能 IRCL 后延时 1ms，等待IRCL 稳定
    CKSEL = (CKSEL&0xF8) | CKSEL_IRCL;      //系统时钟切换到 IRCL
}
```

设置系统时钟为 XOSCH

设置系统时钟为 XOSCH，程序如下：

```
#define XHCS                (1<<4)
#define XHCKE               (1<<1)
#define XHSTA               (1<<0)
#define CKSEL_XOSCH         2
void Sys_Clk_Set_XOSCH(void)
{
    P17F = 4;                               //设置 P1.7 为外部高速晶振输出引脚
    P30F = 4;                               //设置 P3.0 为外部高速晶振输入引脚
    CKCON &= ~XHCS;
    CKCON |= XHCKE;                         //打开 XOSCH 时钟
    while(!(CKCON & XHSTA));                //等待时钟稳定
    CKSEL = (CKSEL&0xF8) | CKSEL_XOSCH; //设置系统时钟为 XOSCH
}
```

设置系统时钟为 CLKIN

设置系统时钟为 CLKIN，程序如下：

```
#define XHCS                (1<<4)
#define CKSEL_XCLK          2
void Sys_Clk_Set_XCLK_IN(void)
{
    P30F = 6;                               //设置 P3.0 为外部高速时钟输入引脚
    CKCON |= XHCS;
    CKSEL = (CKSEL&0xF8) | CKSEL_XCLK;  //设置系统时钟为 CLKIN
}
```

---

10 供电和复位系统

10.1 供电系统

在 JZ8FC005T 系列芯片 VDD 和 VSS 引脚间接入 2.0V - 5.5V 的电源，用于供电。模拟系统由 VDD 以及 LDO 供电，数字系统只需LDO 供电。

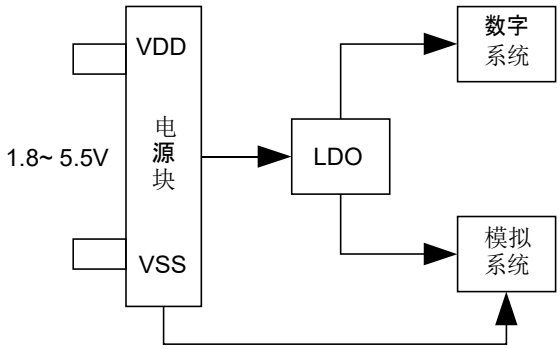


图 10-1-1 供电系统示意图

图 10-1-2 为芯片供电典型电路图。

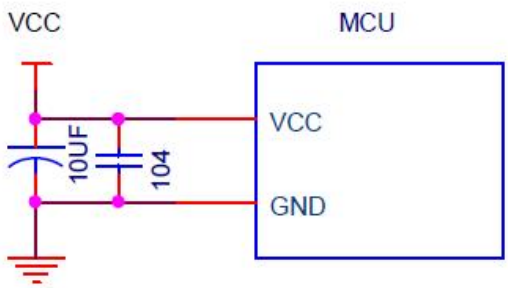


图 10-1-2 芯片供电典型电路图

重要提醒：*1. 以上电路中，滤波电容 10uF 和 104 为芯片供电电路标配，不可省略，此电容须靠近芯片电源引脚摆放，否则有可能会导致芯片工作异常。*

2.以上电路及元件参数仅供参考，根据外围工作环境及不同电压供电参数可能需要修改。

10.1.1 LDO 功能简介

JZ8FC005T 系列芯片有一个内置的低压差线性稳压器（LDO）。LDO 模块为芯片提供核心电压。LDO 的

输出电压通过 VLEVEL 位（PWCON[2:0]）设置， VLEVEL 默认值为 3，对应的输出电压为 1.58V。当 VDD/VSS 小于 VLEVEL 位设定的输出电压时，LDO 直接输出 VDD；当 VDD/VSS 大于设定电压时，LDO 输出设定的电压。LDO 设置高电压有助于时钟模块快速启动，而设置为较低电压时有助于降低芯片功耗。LDO 有两种不同的工作模式：高功率模式和低功率模式，通过 VHL 位（PWCON[3]）设置。两种模式下，LDO 的负载能力不相同。在高功率模式，LDO 能提供的电流更大，但自身功耗也更大，低功率模式则反之。当系统处于正常工作状态时，LDO 一般都设置为高功率模式，而低功率模式一般应用于省电模式，例如 STOP、IDLE、低速运行模式等。

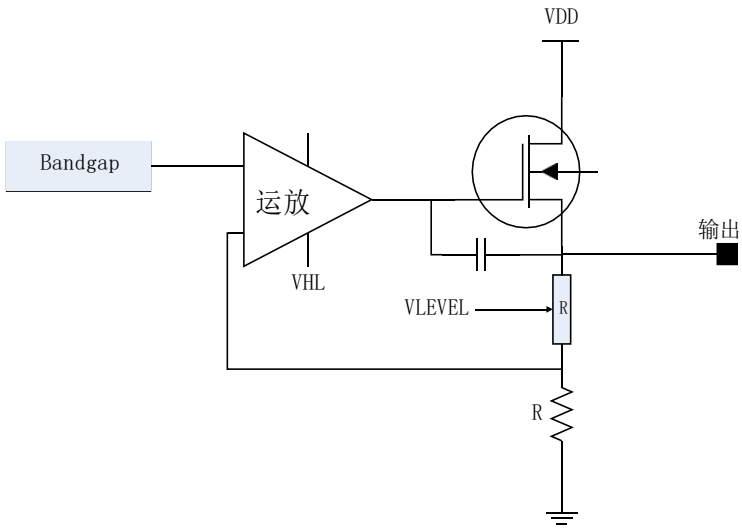


图 10-1-3 LDO 模块示意图

10.1.2 LDO 控制寄存器

表 10-1-2-1 寄存器 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		
初始值	0	1	1	1	1	1	0	1

位编号	位符号	说明
7~4	FLEVEL	内部基准电压（Bandgap）输出调整位域 0000：0.825V 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

3	VHL	LDO 工作模式控制位 0：低功率模式 1：高功率模式
2~0	VLEVEL	LDO 输出电压设置位域 000：1.31V 001：1.37V 010：1.43V 011：1.49V 100：1.55V 101：1.61V 110：1.67V 111：1.73V

10.2 复位系统

JZ8FC005T 系列芯片有多个内部和外部复位源，如图 10-2-1 所示。

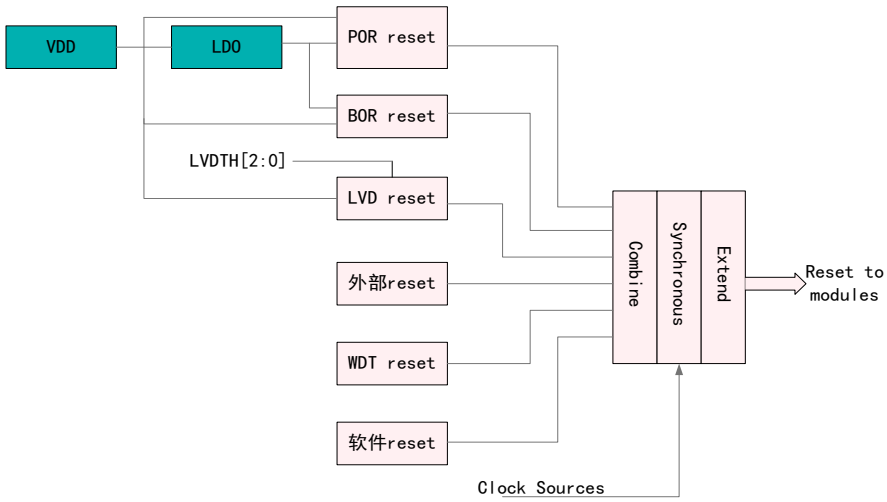
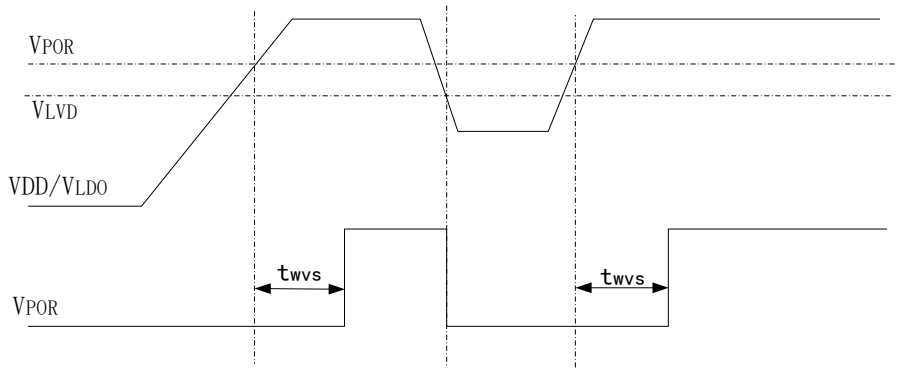


图 10-2-1 复位系统结构图

- **上电复位（POR）**

系统上电呈现逐渐上升的曲线形式，需要一定时间才能达到正常的工作电压。上电复位是基于电源电压 VDD 和内部LDO 的输出电压，当电压低于检测阈值时，上电复位信号有效。
上电复位电路能够保证芯片在上电过程中处于复位状态，芯片上电后能够从一个已知的稳定的状态开始运行。上电复位信号也会被芯片内部的计数器展宽，以保证上电后内部的各种模拟模块能够进入稳定的工作状态。



twvs：等待电压稳定时间

图 10-2-2 上电复位电路示例及上电过程

- **掉电复位（BOR）**

利用掉电复位，可以为芯片提供电源跌落(例如受到干扰或者负载变化)的预警信号。一旦发现电源电压 VDD 或内部 LDO 的输出电压下降到某一个阈值时，就使芯片及时复位以免系统工作状态不正常或者程序执行错误。

- **低电压复位**

低电压检测（LVD）可以在多种工作模式下持续监控电源电压 VDD。当 VDD 低于 LVD 设定的域值电压超

过 20us 就可以产生复位信号（前提是 LVD 设置为复位模式）。

- **外部复位**

通过拉低复位引脚(RESET)，可以从外部源复位器件。在正常工作情况下，RESET 可以复位整个芯片，在 STOP 状态，硬复位会唤醒芯片后再复位。一般情况下，RESET 被内部上拉拉高，不会影响内部的复位电路。

- **看门狗复位**

看门狗定时器负责监控处理器执行指令的情况，通过合适的配置，如果看门狗定时器在特定时间段内未被刷新，则可以产生复位信号。上电复位后，看门狗定时器是关闭的，用户需要时，再配置开启。

- **软复位**

芯片可以在程序控制下执行软复位。通过对 PCON 寄存器中的 SWRST 位写 1，CPU 可以发出复位指令。

上电掉电复位及外部硬复位将复位所有的电路，LVD 和 WDT 的复位不能复位其本身电路，但可以复位其他电路（例如：WDT 复位产生后，WDT 模块电路没有复位，WDT 寄存器还保持复位之前的状态，但 WDT 之外的电路已经复位了）。LVD/WDT 和软复位都不能复位存储控制电路。软复位后，程序将从 BOOT 配置指向的位置开始运行。所有复位产生之后，PC 都将指向地址 0。

11 功耗管理

JZ8FC005T 系列芯片有三种不同的低功耗模式: IDLE 模式、STOP 模式、低速运行模式。IDLE 模式时系统功耗小于 15uA，STOP 模式时系统功耗小于 7uA，低速运行时功耗小于 90uA。

11.1 IDLE 模式

在 IDLE 模式下，CPU 将停止工作。进入 IDLE 模式前，除了主时钟，其他的时钟源根据需要都可选择关闭，以便节省功耗。同样地，进入 IDLE 模式前，可根据需要设定芯片某些外设的开关。打开的外设在 IDLE 状态下仍然可以正常工作。

设置进入 IDLE 模式前，需要先查看一下寄存器 IDLST（IDLSTH 和 IDLSTL），如果所有位都为 0，则设置进入 IDLE 模式后，CPU 将正常进入 IDLE 模式。如果 IDLST 的位不全为 0，即使有设置进入 IDLE 模式的操作，CPU 也不会进入 IDLE 模式，而是继续停留在正常工作模式。此时用户需先把 IDLST 对应位的中断处理完成，再重新设置进入IDLE 模式的动作。

所有复位事件和任何中断事件都将唤醒芯片。中断唤醒 CPU 后，芯片首先将恢复时钟，然后响应该中断，进入该中断的服务程序。退出中断服务程序后，芯片将执行置位 IDLE 指令后面的指令。退出 IDLE 模式时，IDLE 位将自动清零。

需要注意的是，在置位IDLE 的指令后面需要紧接两条nop 指令，防止程序出错。

11.2 STOP 模式

STOP 模式是比 IDLE 更深层次的低功耗模式。STOP 模式可以停止所有时钟（包括主时钟）和时钟产生电路。如果 WDT 和 RTC 处于打开状态，则它们使用的时钟模块将处于工作状态，可以有选择地关闭 WDT 和 RTC 以节省功耗。

类似于 IDLE 模式，进入 STOP 模式前，需要先查看 STPST（STPSTH 和 STPSTL）寄存器，若有置 1 的位存在，需要先行处理，以确保能顺利进入STOP 模式。

STOP 模式可以通过外部中断、LVD 中断或复位、硬复位、RTC 中断、WDT 中断或复位、时钟监控中断、触摸中断来唤醒。如果是中断唤醒，那么唤醒 MCU 后，芯片首先将恢复时钟，然后响应该中断，进入该中断的服务程序。退出中断服务程序后，芯片将执行置位 STOP 指令后面的指令。退出 STOP 模式时，STOP 位将自动清零。

为了更好的唤醒芯片，推荐在进入 STOP 模式前切换系统时钟到内部时钟，因为唤醒时，外部时钟需要更多时间去等待稳定。

在进入 STOP 模式时，最后一个时钟沿将关闭系统时钟，然后芯片完全进入 STOP 模式。需要注意的是，在置位STOP 的指令后面需要紧接三条nop 指令，防止程序出错。

备注：*进入 STOP/IDLE 模式时，设置 LDO 为低功率模式可有效降低待机功耗，但是退出 STOP/IDLE 模式时，一定要把 LDO 设置回高功率模式，否则会导致芯片工作异常。*

11.3 低速运行模式

由于芯片的功耗与运行速度直接相关，所以把主时钟切换到低速时钟运行也可以显著降低功耗。系统设为IRCL（频率为 131KHz）时的电流小于 90uA。

11.4 低功耗相关寄存器描述

表 11-4-1 寄存器 PCON

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

位编号	位符号	说明
7~6	-	-
5	SWRST	软复位控制位，1 有效 设置 SWRST=1 产生软复位，复位产生后自动清 0。
4~3	-	-
2	TSMODE	在线仿真模式标志位，为 1 表示芯片正工作于在线仿真模式
1	STOP	STOP 模式控制位，1 有效 当设置 STOP=1 且 STPST 为 0 时，芯片进入 STOP 模式，退出 STOP 模式后自动清 0
0	IDLE	IDLE 模式控制位，1 有效 当设置 IDLE=1 且IDLST 为 0 时，芯片进入 IDLE 模式，退出 IDLE 模式后自动清 0

表 11-4-2 寄存器 IDLST

FCh	7	6	5	4	3	2	1	0
IDLSTL	-	IDLSTL[6:0]						
R/W	-	R						
初始值	-	0	0	0	0	0	0	0

位编号	位符号	说明
7	ADCINT/EPIF[0]	IDLE 模式时，ADC/外部中断 2 的中断状态
6	UART1INT	IDLE 模式时， UART1 的中断状态
5	T2INT	IDLE 模式时，定时器 2 的中断状态
4	DCINT	IDLE 模式时，无线码的中断状态
3	TF1	IDLE 模式时，定时器 1 的中断状态
2	PIF[1]	IDLE 模式时，外部中断 1 的中断状态
1	TF0	IDLE 模式时，定时器 0 的中断状态
0	PIF[0]	IDLE 模式时，外部中断 0 的中断状态

FDH	7	6	5	4	3	2	1	0
IDLSTH	-	IDLSTH[6:0]						
R/W	-	R						
初始值	-	0	0	0	0	0	0	0

位编号	位符号	说明
7	-	-
6	PWM/EPIF[7]	IDLE 模式时，PWM/外部中断 9 的中断状态

5	TMCINT/EPIF[6]	IDLE 模式时，TMC/外部中断 8 的中断状态
4	WDTINT/EPIF[5]	IDLE 模式时，WDT/外部中断 7 的中断状态
3	I2CINT/SWIINT/EPIF[4]	IDLE 模式时，I2C/SWI/外部中断 6 的中断状态
2	SPIINT/EPIF[3]	IDLE 模式时，SPI/外部中断 5 的中断状态
1	LVDINT/EPIF[2]	IDLE 模式时，UART2/外部中断 4 的中断状态
0	UART2INT/EPIF[1]	IDLE 模式时，ADC/外部中断 3 的中断状态

表 11-4-2 寄存器 STPST

FEH	7	6	5	4	3	2	1	0
STPST	STPSTL							
R/W	R							
初始值	0	0	0	0	0	0	0	0
位编号	位符号			说明				
7	EPWKF[5]			STOP 模式时，外部中断 7 中断状态				
6	EPWKF[4]			STOP 模式时，外部中断 6 中断状态				
5	EPWKF[3]			STOP 模式时，外部中断 5 中断状态				
4	EPWKF[2]			STOP 模式时，外部中断 4 的中断状态				
3	EPWKF[1]			STOP 模式时，外部中断 3 的中断状态				
2	EPWKF[0]			STOP 模式时，外部中断 2 的中断状态				
1	PWKF[1]			STOP 模式时，外部中断 1 的中断状态				
0	PWKF[0]			STOP 模式时，外部中断 0 的中断状态				
FFH	7	6	5	4	3	2	1	0
STPST	STPSTH							
R/W	R							
初始值	0	0	0	0	0	0	0	0
位编号	位符号			说明				
7	TMWKF			STOP 模式时， TMC 的中断状态				
6	WDTWKF			STOP 模式时，WDT 的中断状态				
5	I2CWKF/SWIWKF			STOP 模式时， I^2C/SWI 的中断状态				
4	-			-				
3	LVDWKF			STOP 模式时， LVD 的中断状态				
2	-			-				
1	EPWKF[7]			STOP 模式时，外部中断 9 中断状态				
0	EPWKF[6]			STOP 模式时，外部中断 8 中断状态				

11.5 低功耗模式控制例程

STOP 模式例程

STOP 模式程序如下：

```
-------------------------------------------------------------------------------------------
void Stop(void)
{
    bit IE_EA;
    I2CCON = 0;                 //关闭 I2C，否则无法关闭主时钟
    SWICON |= 0x01;             //关闭单线通信功能，否则无法关闭主时钟
    CKCON = 0;                  //所有时钟关闭
    PWCON &= ~0x08;             //设置 LDO 进入低功率模式
    MECON |= (1<<6);            //设置 FLASH 进入深度睡眠状态
    while(STPSTH|STPSTL);       //如果有中断未响应,等待中断被响应
    IE_EA = EA;                 //保存全局中断使能位状态
    EA = 0;
    PCON |= 0x02;               //进入 STOP 模式
    _nop_();
    _nop_();
    _nop_();
    EA = IE_EA;                 //恢复原全局中断开关状态
    PWCON |= 0x08;              //退出 STOP 模式后，把 LDO 设回高功率模式
}
-------------------------------------------------------------------------------------------
```

IDLE 模式例程

IDLE 模式程序如下：

```
-------------------------------------------------------------------------------------------
#define IHCKE           (1<<6)
#define ILCKE           (1<<7)

#define CKSEL_IRCH    0
#define CKSEL_IRCL    1

void Idle(void)
{
    CKCON |= ILCKE;             //IRCL 时钟使能
    Delay_ms(1);                //使能 IRCL 后延时 1ms，等待 IRCL 稳定
    CKSEL = (CKSEL&0xF8) | CKSEL_IRCL;   //系统时钟切换到 IRCL
    I2CCON = 0;                 //关闭 I2C，否则无法关闭主时钟
    SWICON |= 0x01;             //关闭单线通信功能，否则无法关闭主时钟
    CKCON = 0;                  //所有时钟关闭
    PWCON &= ~0x08;             //设置 LDO 进入低功率模式
```

```
    MECON |= (1<<6);            //设置 FLASH 进入深度睡眠状态
    while(IDLSTH|IDLSTL);       //如果有中断未响应,等待中断被响应
    PCON |= 0x01;               //进入 IDLE 模式
    _nop_();
    _nop_();
    _nop_();
    PWCON |= 0x08;              //退出 IDLE 模式后，把 LDO 设回高功率模式
}
```

备注：由于进入 IDLE 后，主时钟仍是打开的，如果进入 IDLE 前主时钟是高速时钟，进入 IDLE 模式后功耗仍会很大，所以进入 IDLE 之前需要把主时钟切换到低速时钟。

低速运行模式例程

低速运行模式程序如下：

```
#define ILCKE  (1<<7)
#define CKSEL_IRCL 1
void LowSpeedMode(void)
{
    CKCON |= ILCKE;              //打开 IRCL
    CKSEL = (CKSEL&0xF8) | CKSEL_IRCL;   //系统时钟设置为 IRCL
    I2CCON = 0;                 //关闭 I2C 模块，I2C 默认是使能的，如果 I2C 不关闭将无法关闭 IRCH 时钟
    SWICON |= 0x01;              //关闭单线通信功能，否则无法关闭主时钟
    CKCON = 0;                  //关闭所有时钟
    PWCON &= ~0x08;              //设置 LDO 进入低功率模式
}
```

***备注**：退出低速运行模式后，必须把 LDO 设置回高功率模式，参考 STOP/IDLE 例程。*

---

12 通用定时器（定时器 0,定时器 1,定时器 2）

12.1 定时器 0

12.1.1 定时器 0 介绍

定时器或计数器功能通过 CT0 位（TMOD[2]）来选择，CT0=0 选择为定时器，CT0=1 选择为计数器。作为定时器时，时钟是系统时钟的 12 分频。作为计数器时，时钟是 T0 的输入时钟。由于检测 T0 输入边沿变化需要 2 个时钟周期，所以作为计数器时最大的输入波特率是内部系统时钟频率的 1/2。T0 输入信号在占空比上没有限制，然而为了完全识别 0 或 1 的状态，信号至少需要保持 1 个内部系统时钟周期时间。定时器 0 有 4 个工作模式，通过T0M0、T0M1 位(TMOD[1:0])来选择。

- **模式 0**

在此模式下，定时器 0 作为 13 位定时器/计数器，TH0 存放 13 位定时器/计数器的高 8 位，TL0[4:0]存放低 5 位，而 TL0[7:5]是无效的，在读取时应被忽略。当定时器 0 溢出，中断标志位 TF0（TCON[5]）会被置 1。中断被响应后，TF0 位会自动清 0。当GATE0（TCON[3]）=0 时，定时器/计数器由TR0（TCON[4]）位使能计数，当GATE0=1 时，定时器/计数器由引脚INT0 控制使能，INT0 为高电平时计数，INT0 为低电平则停止计数。

- **模式 1**

此模式下，定时器 0 作为 16 位定时器/计数器，除此之外，功能与模式 0 完全相同。

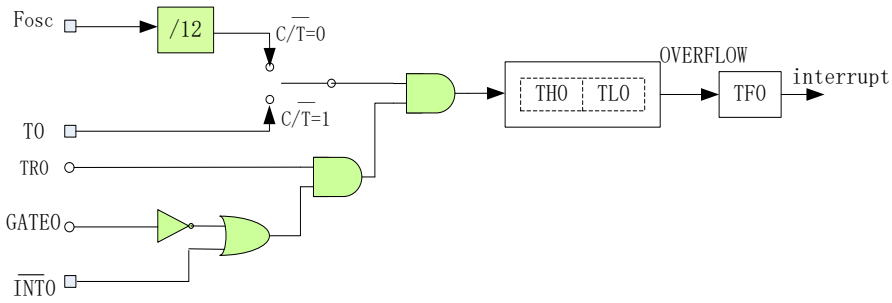


图 12-1-1-1 定时器 0 的模式 0 与 1

- **模式 2**

在此模式中，定时器 0 作为 8 位自动重载定时器/计数器，只有 TL0 自动累加。当 TL0 计数溢出时，不但产生中断标志TF0，而且从TH0 中自动装载计数初始值到TL0。其他设置方法和模式 0、1 相同。

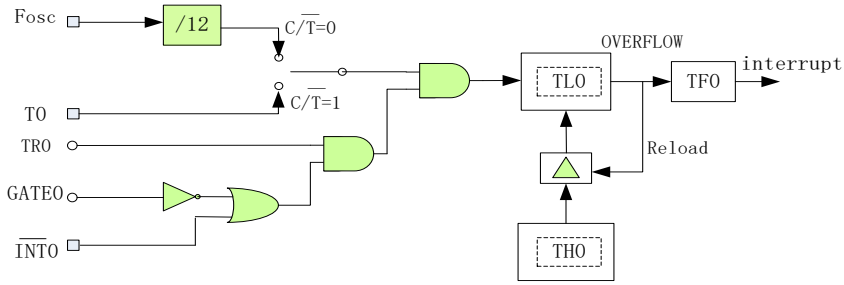


图 12-1-1-2 定时器 0 的模式 2

- **模式 3**

在此模式中，TL0 和 TH0 作为两个独立的 8 位定时器/计数器。TL0 可以作为定时器或计数器，而 TH0 只能作为定时器。其中 TL0 占用定时器 0 的控制位 CT0、GATE0、TR0、TF0、INT0，而 TH0 只能占用定时器 1 的控制位 TR1、TF1。其他控制方法和模式 0、1 相同。当定时器 0 工作于模式 3 时，定时器 1 和 TH0 共用控制位 TR1，但定时器 1 由于 TF1 已被 TH0 占用，所以只能工作于不需要产生中断的场合，例如作为 UART 的波特率产生器。

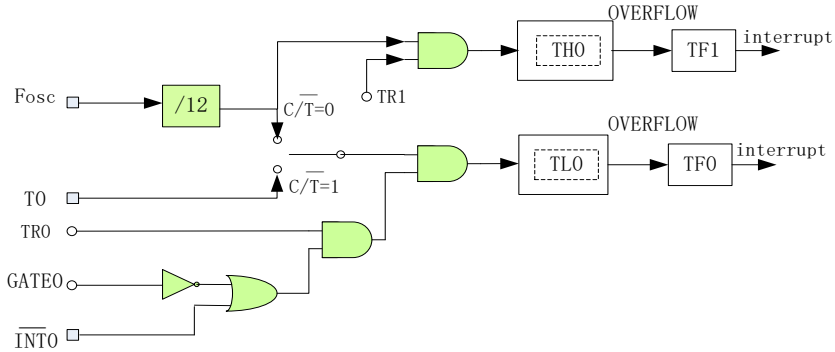


图 12-1-1-3 定时器 0 的模式 3

12.1.2 定时器 0 寄存器描述

表 12-1-2-1 寄存器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
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7	TF1	定时器 0 模式 3 的 TH0 溢出/定时器 1 溢出标志位，中断被响应后自动清 0.
6	TR1	定时器 1 运行控制位，1 有效
5	TF0	定时器 0 溢出标志位，中断被响应后自动清 0.
4	TR0	定时器 0 运行控制位，1 有效
3	IE1	外部中断 1 使能位，1 有效
2	IT1	外部中断 1 触发类型控制位 0：外部中断 1 在输入管脚上升沿时触发 1：外部中断 1 在输入管脚下降沿时触发
1	IE0	外部中断 0 使能位，1 有效
0	IT0	外部中断 0 触发类型控制位 0：外部中断 0 在输入管脚上升沿时触发 1：外部中断 0 在输入管脚下降沿时触发

表 12-1-2-2 寄存器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
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7	GATE1	定时器 1 门控控制位，1 有效。有效时定时器 1 由INT1 控制开关
6	CT1	定时器 1 计数器/定时器选择位 0：定时器，时钟为系统时钟 12 分频 1：计数器，时钟为 T1 输入时钟
5	T1M1	[T1M1,T1M0]为定时器 1 模式选择位 00：模式 0，TL1 和TH1 组成 13 位定时器/计数器 01：模式 1，TL1 和TH1 组成 16 位定时器/计数器 10：模式 2，TL1 作为 8 位定时器/计数器，TH1 作为自动重载寄存器 11：模式 3，此模式会锁住 TH1/TL1，等效于 TR1=0
4	T1M0	
3	GATE0	定时器 0 门控控制位，1 有效。有效时定时器 0 由INT0 控制开关
2	CT0	定时器 0 计数器/定时器选择位 0：定时器，时钟为系统时钟 12 分频 1：计数器，时钟为 T0 输入时钟
1	T0M1	[T0M1,T0M0]为定时器 0 模式选择位 00：模式 0，TL0 和TH0 组成 13 位定时器/计数器 01：模式 1，TL0 和TH0 组成 16 位定时器/计数器 10：模式 2，TL0 作为 8 位定时器/计数器，TH0 作为自动重载寄存器 11：模式 3，TL0 和TH0 作为两个完全独立的 8 位定时器/计数器
0	T0M0	

表 12-1-2-3 寄存器TL0

8AH	7	6	5	4	3	2	1	0
TL0	TL0							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~0	TL0	定时器 0 模式 0/1 计数值的低字节，模式 2/3 计数值

表 12-1-2-4 寄存器TH0

8CH	7	6	5	4	3	2	1	0
TH0	TH0							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~0	TH0	定时器 0 模式 0/1 计数值的高字节，模式 2 重载值，模式 3 计数值

12.2 定时器 1

12.2.1 定时器 1 介绍

定时器或计数器功能通过 CT1 位（TMOD[6]）来选择，CT1=0 选择为定时器，CT1=1 选择为计数器。作为定时器时，时钟是系统时钟的 12 分频。作为计数器时，时钟是 T1 的输入时钟。由于检测 T1 输入边沿变化需要 2 个时钟周期，所以作为计数器时最大的输入波特率是内部系统时钟频率的 1/2。T1 输入信号在占空比上没有限制，然而为了完全识别 0 或 1 的状态，信号至少需要保持 1 个内部系统时钟周期时间。定时器 1 有 4 个工作模式，通过T1M0、T1M1 位(TMOD[5:4])来选择。

- **模式 0**

  在此模式下，定时器 1 作为 13 位定时器/计数器，TH1 存放 13 位定时器/计数器的高 8 位，TL1[4:0]存放低 5 位，而TL1[7:5]是无效的，在读取时应被忽略。当定时器 1 溢出，中断标志位 TF1（TCON[7]）会被置 1。 中断被响应后，TF1 位会自动清 0。当 GATE1（TCON[7]）=0 时，定时器/计数器由 TR1（TCON[6]）位使能计数，当 GATE1=1 时，定时器/计数器由引脚 INT1 控制使能，INT1 为高电平时计数，INT1 为低电平则停止计数。

- **模式 1**

  在此模式下，定时器 1 作为 16 位定时器/计数器，TH1 存放 16 位定时器/计数器的高 8 位，TL1 存放低 8 位。当定时器 1 溢出，中断标志位 TF1（TCON[7]）会被置 1。中断被响应后，TF1 位会自动清 0。当 GATE1（TCON[7]）=0 时，定时器/计数器由 TR1（TCON[6]）位使能计数，当 GATE1=1 时，定时器/计数器由引脚INT1 控制使能，INT1 为高电平时计数，INT1 为低电平则停止计数。

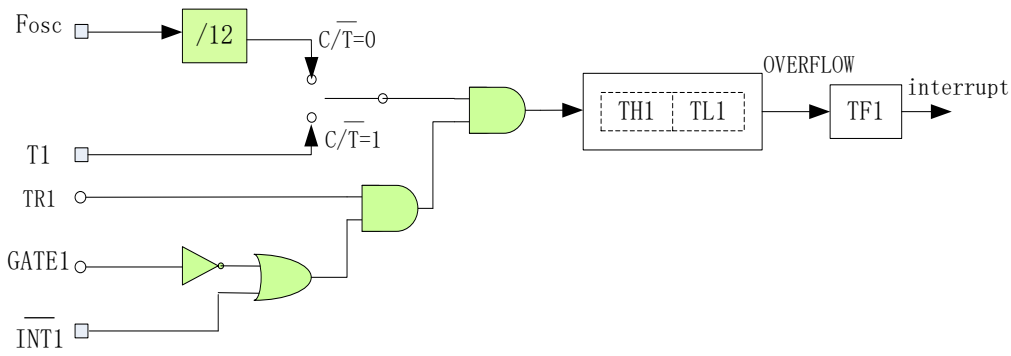


图 12-2-1 定时器 1 的模式 0 和 1

- **模式 2**

  在此模式中，定时器 1 作为 8 位自动重载定时器/计数器，只有 TL1 自动累加。当 TL1 计数溢出时，不但产生中断标志TF1，而且从TH1 中自动装载计数初始值到TL1。其他设置方法和模式 0、1 相同。

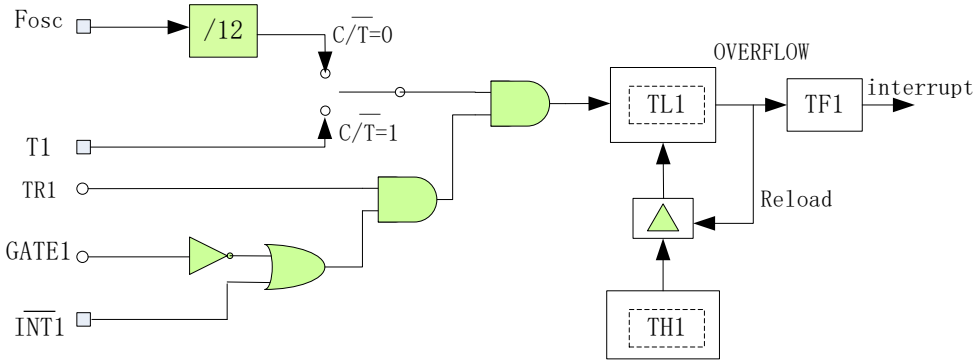


图 12-2-2 定时器 1 的模式 2

- **模式 3**
  此模式下，TH1、TL1 会被锁住，等效于TR1=0。

12.2.2 定时器 1 寄存器描述

寄存器TCON 和 TMOD 见表 12-1-2-1 和表 12-1-2-2。

表 12-2-2-1 寄存器TL1

8BH	7	6	5	4	3	2	1	0
TL1	TL1							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
7~0	TL1	定时器 1 模式 0/1 计数值的低字节，模式 2/3 计数值						

表 12-2-2-2 寄存器TH1

8DH	7	6	5	4	3	2	1	0
TH1	TH1							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
7~0	TH1	定时器 1 模式 0/1 计数值的高字节，模式 2 重载值，模式 3 计数值						

12.3 定时器 2

12.3.1 功能简介

定时器 2 是一个 16 位（TH2、TL2）的定时器/计数器。T2P0、T2P1 位可选择不同的控制方式或时钟源。当 T2P=0、3 时，选择系统时钟作为定时器 2 时钟（注意：和定时器 0、1 不同的是，时钟没有经过 12 分频）；

当 T2P=0 时，定时器 2 由 TR2 位使能；当 T2P=2 时，由 T2 电平门控，T2 为高时，计数使能，T2 为低时，计数停止。当 T2P=1、2 时，选择 T2 的输入信号作为计数时钟，当 T2P=1 时，检测 T2 的下降沿计数，当 T2P=2 时，检测T2 的上升沿。

定时器 2 可通过 T2M0、T2M1 位设置不同的工作模式。当 T2M=0 时，定时器 2 工作于定时器/计数器模式，TH2、TL2 作为 16 位计数器自动累加；在此模式下，通过设置 T2R0、T2R1 位可选择两种不同的重载模式或关闭重载功能，在重载模式下，T2CH、T2CL 存放重载值，当 T2R=2 时，定时器 2 溢出会从 T2CH、T2CL 装载计数初值到 TH2、TL2，而当 T2R=3 时，在引脚 T2EX 下降沿进行重载。当重载事件发生后，重载中断标志 RF2 置 1，如果定时器 2 中断使能会触发重载中断，RF2 通过写 1 清 0。

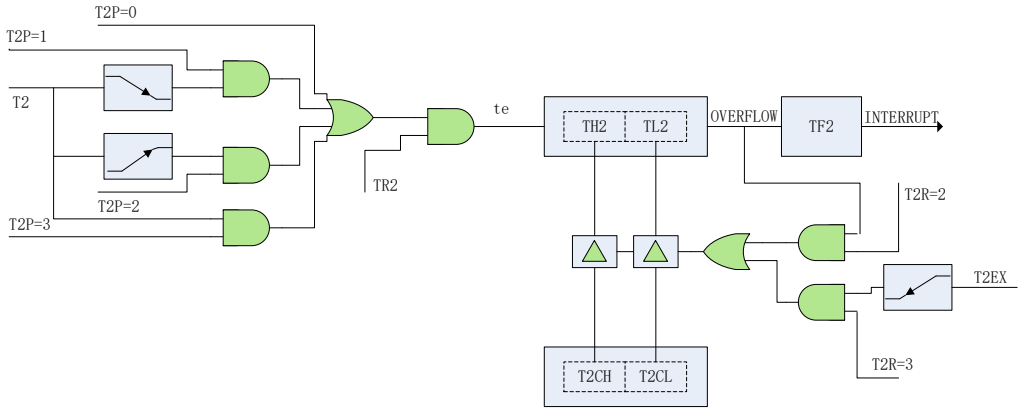


图 12-3-1-1 定时器 2 的重载模式

当 T2M=1 时，定时器 2 工作于比较模式，当计数值 TH2、TL2 大于 T2CH、T2CL 时，引脚 T2CP 输出高，否则 T2CP 输出低。

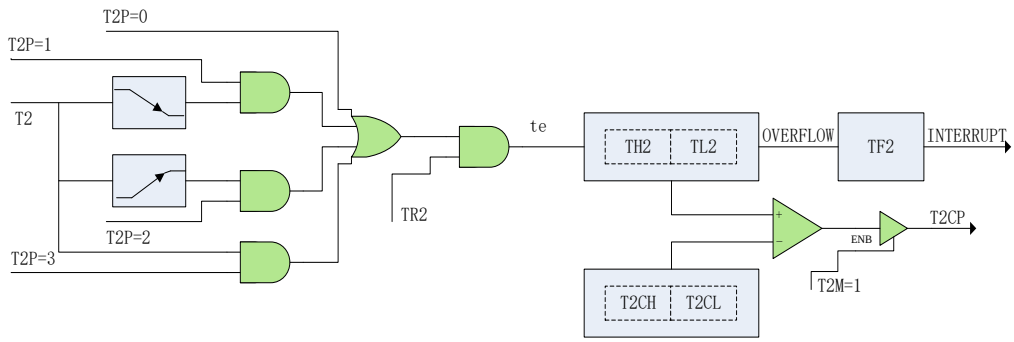


图 12-3-1-2 定时器 2 的比较模式

当 T2M=2 或 3 时，定时器 2 工作于抓取模式。当T2M=2 时，当引脚 T2CP 触发沿发生时，定时器 2 的计数值 TH2、TL2 被锁存到 T2CH、T2CL，触发沿可通过 CCFG 位设置，当抓取事件产生后，抓取中断标志 CF2 置 1，如果定时器 2 中断使能会触发抓取中断，CF2 通过写 1 清 0。当 T2M=3 时，写寄存器 T2CL 将产生锁存的触发事件，而写T2CL 的值不保存，在此模式下，抓取事件不会置位 CF2。

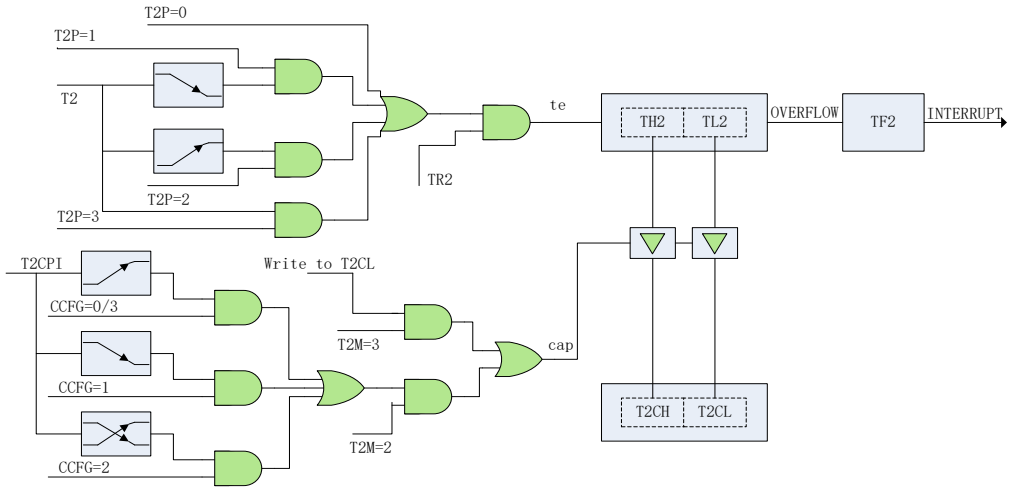


图 12-3-1-3 定时器 2 的抓取模式

12.3.2 定时器 2 寄存器描述

表 12-3-2-1 寄存器 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
初始值	-	0	0	0	0	0	0	0

位编号	位符号	说明
7	-	-
6	TR2	定时器 2 运行控制位，1 有效
5	T2R1	[T2R1,T2R0]是定时器 2 重载模式选择位 10：模式 0 11：模式 1 其他：重载功能关闭
4	T2R0	
3	T2IE	定时器 2 中断使能位，1 有效
2	-	-
1	T2P1	[T2P1,T2P0]是定时器 2 引脚 T2 功能选择位 00：定时器 2 使用内部系统时钟计数，没有使用 T2 01：定时器 2 检测 T2 下降沿计数 10：定时器 2 检测 T2 上升沿计数 11：定时器 2 使用内部系统时钟计数，通过 T2 门控
0	T2P0	

表 12-3-2-2 寄存器 T2MOD

C9H	7	6	5	4	3	2	1	0
T2MOD	TF2	CF2	RF2	CCFG1	CCFG0	-	T2M1	T2M0
R/W	-	-	-	R/W	R/W	-	R/W	R/W
初始值	-	-	-	0	0	-	0	0

位编号	位符号	说明
7	TF2	Timer2 计数器溢出中断标志，写 1 清 0

6	CF2	抓取中断标志，写 1 清 0
5	RF2	自动重载中断标志，写 1 清 0
4	CCFG1	[CCFG1,CCFG0]抓取模式触发沿选择位，在 T2M＝3 或 T2M＝4 时有效 01：下降沿 10：上升或下降沿 其它值：上升沿
3	CCFG0	
2	-	-
1	T2M1	工作模式选择位 00：定时器/计数器模式 01：比较模式 10：抓取模式 0 11：抓取模式 1
0	T2M0	

表 12-3-2-3 寄存器T2CL

CAH	7	6	5	4	3	2	1	0
T2CL	T2CL							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
7~0	T2CL	在重载模式，T2CL 是重载值的低字节 在比较模式，T2CL 是比较值的低字节 在抓取模式，T2CL 保存捕获值的低字节						

表 12-3-2-4 寄存器T2CH

CBH	7	6	5	4	3	2	1	0
T2CH	T2CH							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
7~0	T2CH	在重载模式，T2CH 是重载值的高字节 在比较模式，T2CH 是比较值的高字节 在捕获模式，T2CH 保存捕获值的高字节						

表 12-3-2-5 寄存器TL2

CCH	7	6	5	4	3	2	1	0
TL2	TL2							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
7~0	TL2	定时器 2 计数值的低字节						

表 12-3-2-6 寄存器TH2

CDH	7	6	5	4	3	2	1	0
TH2	TH2							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~0	TH2	定时器 2 计数值的高字节

表 12-3-2-7 寄存器 T2CPPS

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

备注：

1. T2CS=0 时，T2CPS 为普通寄存器，仅作为捕获通道 0 的引脚选择控制寄存器。

2. T2CS=1 时，T2CPPS 为索引寄存器，由 INDEX0/1/2 分别指向 T2CPPS0/1/2，控制捕获通道 0/1/2。

位编号	位符号	说明
7~5	-	-
4~0	T2CPPS	Timer2 捕获引脚选择位域 00000：选择 P0.0 00001：选择 P0.1 00010：选择 P0.2 00011：选择 P0.3 00100：选择 P0.4 00101：选择 P0.5 00110：选择 P0.6 00111：选择 P0.7 01000：选择 P1.0 01001：选择 P1.1 01010：选择 P1.2 01011：选择 P1.3 01100：选择 P1.4 01101：选择 P1.5 01110：选择 P1.6 01111：选择 P1.7 10000：选择 P2.0 其他：选择 P3.0

表 12-3-2-8 寄存器 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
初始值	0	-	-	-	-	0	0	0

位编号	位符号	说明
7	T2CS	捕获通道 0 寄存器组选择寄存器 0：捕获通道 0 选择旧寄存器组，标志位为 T2MOD.CF，捕获值为 T2C 1：捕获通道 0 选择新寄存器组，标志位为 T2CPF.CF20，捕获值为 T2CP 备注： 1. 无论 T2CS 为何值，受影响的寄存器的操作仅限于捕获功能的操作，其他功能的操作仍然不变。 2. 当 T2CS=0 时，T2CPCH0E/T2CPCH1E/T2CPCH2E 都无法控制对应的捕获通道的使能和关闭。
6~3	-	-
2	T2CPCH2E	捕获通道使能控制位，1 表示捕获通道 2 使能
1	T2CPCH1E	捕获通道使能控制位，1 表示捕获通道 1 使能
0	T2CPCH0E	捕获通道使能控制位，1 表示捕获通道 0 使能

表 12-3-2-9 寄存器 T2CPF

80A2H	7	6	5	4	3	**2**	**1**	**0**
T2CPF	-	-	-	-	-	CF22	CF21	CF20
R/W	-	-	-	-	-	R/W	R/W	R/W
初始值	-	-	-	-	-	0	0	0

备注：

1. 寄存器 CF20/CF21/CF22 仅在 T2CS=1 时有效，默认无效，详细参考 T2CPCHS 描述。

2. 当寄存器 CF20/CF21/CF22 无效时，读出来的值始终为 0。

位编号	位符号	说明
7~3	-	-
2	CF22	捕获通道 T2CP2 的抓取中断标志，写 1 清 0
1	CF21	捕获通道 T2CP1 的抓取中断标志，写 1 清 0
0	CF20	捕获通道 T2CP0 的抓取中断标志，写 1 清 0

表 12-3-2-10 寄存器 T2CP

80A3H	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
T2CL	T2CP[7:0]							
R/W	R/W							
初始值	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							
初始值	0	0	0	0	0	0	0	0

备注：

1. T2CP 为索引寄存器，由 INDEX0/1/2 分别指向寄存器 T2CP0/1/2。

2. 当 T2CS=0 时，写 T2CP 无效，读 T2CP 将始终为某一固定值 (0 或其他值，取决于清除 T2CS 时，它的状态)。

3. 详细对比参考寄存器 T2C 的描述。

位编号	位符号	说明
15~0	T2CP	捕获值存储寄存器

13 看门狗定时器（WDT）

13.1 看门狗定时器(WDT)功能简介

看门狗定时器是一个可选时钟源的 27 位减法计数器，时钟为 24MHz 下计数时间范围为 0.128ms – 4.096s，有 16 位调节精度。看门狗主要用于监控系统，避免 CPU 因为外界干扰出现死机。如果软件不能在溢出前刷新看门狗定时器，看门狗将产生内部复位或者中断。写 A5H 到寄存器WDFLG 将刷新看门狗，读 WDFLG 可得到看门狗状态。在STOP 模式下，如果看门狗处于使能状态，则看门狗所选的时钟源正常工作，此时如果看门狗设为中断，看门狗中断可唤醒 CPU。

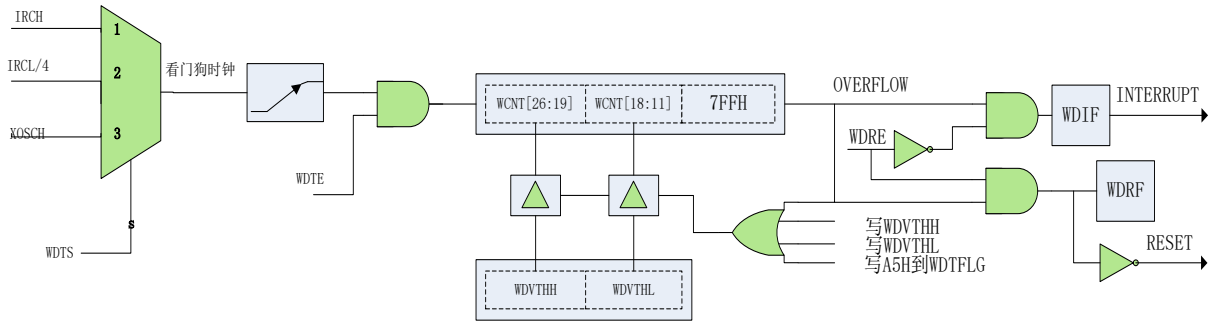


图 13-1-1 看门狗模块结构图

13.2 看门狗定时器(WDT)寄存器描述

表 13-2-1 寄存器 WDCON

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

位编号	位符号	说明
7~6	WDTS	WDT 时钟选择位 01：选择 IRCH 10：选择 IRCL 四分频 11：选择 XOSCH
5~1	-	
0	WDRE	WDT 功能选择位 0：WDT 溢出后产生中断 1：WDT 溢出后产生复位

表 13-2-2 寄存器 WDFLG

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

位编号	位符号	说明
7~2	-	-
1	WDIF	WDT 中断标志，写 A5H 时将清除该标志
0	WDRF	WDT 复位标志，写 A5H 时将清除该标志

表 13-2-3 寄存器 WDVTHL、WDVTHH

ACH	7	6	5	4	3	2	1	0
WDVTHL	WDVTH[7:0]							
R/W	R/W							
初始值	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							
初始值	-	0	0	0	0	0	0	0

位编号	位符号	说明
15~0	WDVTH	WDT 阈值设置寄存器，计算公式如下： WDT 触发时间＝ (WDVTH * 800H+7FFH) * clock cycle

13.3 看门狗定时器控制例程

看门狗中断模式例程

例如，看门狗时钟设置为 IRCH，IRCH 的频率为 24MHz，看门狗设置为中断模式，溢出时间为 1 秒，程序如下：

```
---------------------------------------------------------------------------------
#define WDTS_IRCH           (1<<5)
#define WDTS_IRCL           (2<<5)

#define WDRE_reset          (1<<0)
#define WDRE_int            (0<<0)
void WDT_init(void)
{
    WDCON = WDTS_IRCH | WDRE_int; //设置看门狗时钟源为 IRCH，模式为中断模式
    WDVTHH = 0x16;                //看门狗中断阈值高八位设置 当前值为 1s
    WDVTHL = 0xE2;                //看门狗中断阈值低八位设置
    INT7EN = 1;                   //开启看门狗中断
    WDFLG = 0xA5;                 //喂狗
    EA = 1;                       //开启总中断
}
void WDT_ISR (void) interrupt 12
{
    if(WDFLG & 0x02)
    {
        //看门狗中断服务程序
        WDFLG = 0xA5;//刷新看门狗
    }
}
---------------------------------------------------------------------------------
```

看门狗复位模式例程

例如，看门狗时钟设置为 IRCH，IRCH 的频率为 24MHz，看门狗设置为复位模式，溢出时间为 1 秒，程序如下：

```
---------------------------------------------------------------------------------
#define WDTS_IRCH           (1<<5)
#define WDTS_IRCL           (2<<5)

#define WDRE_reset          (1<<0)
#define WDRE_int            (0<<0)

void WDT_init(void)
{
    WDCON = WDTS_IRCH | WDRE_reset;       //设置看门时钟为 IRCH, 看门狗复位模式
```

```
    WDVTHH = 0x16;                  //看门狗中断阈值高八位设置 当前值为 1s
    WDVTHL = 0xE2;                  //看门狗中断阈值低八位设置
    WDFLG = 0xA5;                    //刷新看门狗
}
-------------------------------------------------------------------------------------
```

14 TMC 定时器

14.1 TMC 功能简介

TMC 定时器的时钟源为 IRCL，中断的最小单位为 512 个 IRCL 时钟周期，可配置中断时间为 1~256 个最小单位时间。在STOP/IDLE 模式下，TMC 中断可唤醒 CPU。

14.2 TMC 寄存器描述

表 14-2-1 寄存器TMCON

80A8H	7	6	5	4	3	2	1	0
TMCON	TME	-	-	-	-	-	-	TMF
R/W	R/W	-	-	-	-	-	-	R
初始值	0	-	-	-	-	-	-	0
位编号	位符号	说明						
7	RTCE	TME 模块使能，1 有效						
6~1	-	-						
0	TMF	TMC 中断标志，1 有效，写 1 清 0						

表 14-2-2 寄存器TMSNU

80A9H	7	6	5	4	3	2	1	0
TMSNU	TMSNU[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	-
位编号	位符号	说明						
7~0	TMSNU	TMC 中断时间配置寄存器，TMC 的中断时间为（TMSNU+1）×512×Tircl *注：Tircl 为 IRCL 时钟一个周期时间。*						

14.3 TMC 控制例程

设置 TMC 为最小单位时间中断（即 512 个IRCL 时钟周期），程序如下：

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

#define IHCKE           (1<<6)
#define ILCKE           (1<<7)

void TMC_ISR  (void)  interrupt 13
{
    if(TMCON & TMF)             //判断 TMC 中断标志
    {
        TMCON |= TMF;           //清除 TMC 中断标志

    }
}

void TMC_init(void)
{
    CKCON |= ILCKE;    //使能 IRCL 时钟
    TMCON = TME(1); //TMC 使能
    TMSNU = 0;              //设置中断时间，中断时间 = TMSNU * 512 * Tircl
    INT8EN =1;
    EA = 1;         //开启总中断
}
---------------------------------------------------------------------------------------
```

15 通用输入输出口（GPIO）及复用定义

15.1 功能简介

JZ8FC005T 系列芯片最大封装有 18 个 I/O 引脚，每个引脚都是复用功能引脚，不仅能独立编程为输入/输出口，而且还能设置为其他功能引脚。每个引脚都分配了一个功能设置寄存器 PnxF（分别对应引脚 Pnx，其中 n=0、1、2、3, 代表 P0、P1、P2、P3，x=0~7, 代表 Pn.0~Pn.7），用户可通过寄存器 PnxF 配置引脚的主功能和其他选项。详见寄存器部分介绍。

GPIO 的主要特性如下：

- 可配置为高阻模式
- I/O 结构可独立设置上拉下拉电阻
- 输出模式可选开漏输出或推挽输出
- 数据输出锁存支持读-修改-写
- 支持 2.0~5.5V 宽电压范围
- 8 个 I/O(P0.0、P0.1、P1.0 ~ P1.5)设计了强灌电流模式(详细参数请查看电气特性章节)

GPIO 推挽模式结构图如图 15-1-1 所示。

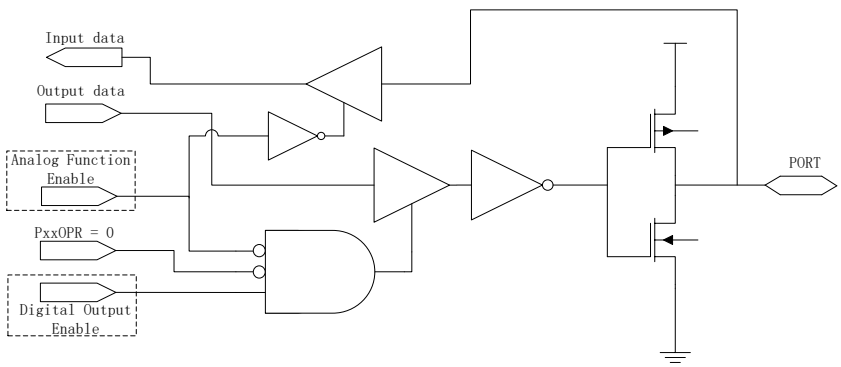


图 15-1-1 I/O 推挽模式结构示意图

GPIO 开漏模式结构图如图 15-1-2 所示。

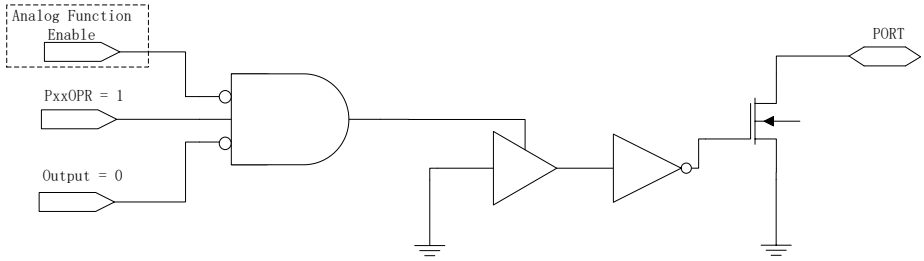


图 15-1-2 I/O 开漏模式结构示意图

GPIO 下拉结构图如图 15-1-3 所示。

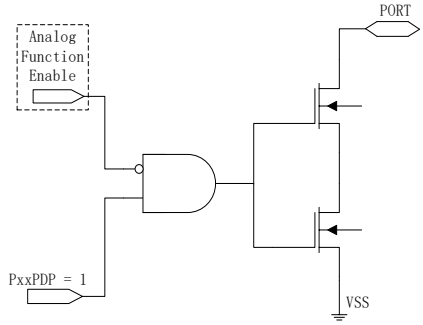


图 15-1-3 I/O 下拉模式结构示意图

GPIO 上拉结构图如图 15-1-4 所示。

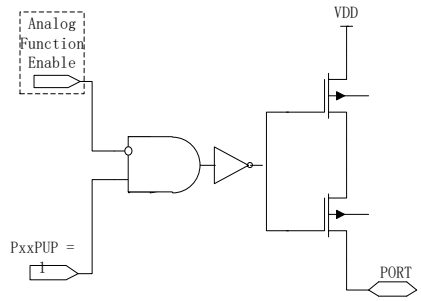


图 15-1-4 I/O 上拉模式结构示意图

15.2 引脚寄存器描述

表 15-2-1 寄存器 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
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
7~0	P0x	引脚 P0x 的数据寄存器，管脚功能设置为 GPIO 时有效 0：设为输入时 P0x 电平为低，设为输出时 P0x 输出低电平 1：设为输入时 P0x 电平为高，设为输出时 P0x 输出高电平						

表 15-2-2 寄存器 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
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~0	P1x	引脚 P1x 的数据寄存器，管脚功能设置为 GPIO 时有效 0：设为输入时 P1x 电平为低，设为输出时 P1x 输出低电平 1：设为输入时 P1x 电平为高，设为输出时 P1x 输出高电平

表 15-2-3 寄存器 P2

A0H	7	6	5	4	3	2	1	0
P2	-	-	-	-	-	-	-	P20
R/W	-	-	-	-	-	-	-	R/W
初始值	-	-	-	-	-	-	-	0

位编号	位符号	说明
7~1	-	-
0	P20	引脚 P20 的数据寄存器，管脚功能设置为 GPIO 时有效 0：设为输入时 P20 电平为低，设为输出时 P20 输出低电平 1：设为输入时 P20 电平为高，设为输出时 P20 输出高电平

表 15-2-4 寄存器 P3

B0H	7	6	5	4	3	2	1	0
P3	-	-	-	-	-	-	-	P30
R/W	-	-	-	-	-	-	-	R/W
初始值	-	-	-	-	-	-	-	0

位编号	位符号	说明
7~1	-	-
0	P30	引脚 P30 的数据寄存器，管脚功能设置为 GPIO 时有效 0：设为输入时 P30 电平为低，设为输出时 P30 输出低电平 1：设为输入时 P30 电平为高，设为输出时 P30 输出高电平

表 15-2-5 引脚功能控制寄存器

8000H	7	6	5	4	3	2	1	0
P00F	P00PUP	P00PDP	P00OPR	-	P00S			
R/W	R/W	R/W	R/W	-	R/W			
初始值	0	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			
初始值	0	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			
初始值	0	0	0	-	0	1	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			
初始值	0	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			
初始值	0	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			
初始值	0	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			
初始值	0	0	0	-	1	0	0	1
8007H	7	6	5	4	3	2	1	0
P07F	P07PUP	P07PDP	P07OPR	-	P07S			
R/W	R/W	R/W	R/W	-	R/W			
初始值	0	0	0	-	1	0	0	1
8008H	7	6	5	4	3	2	1	0
P10F	P10PUP	P10PDP	P10OPR	-	P10S			
R/W	R/W	R/W	R/W	-	R/W			
初始值	0	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			
初始值	0	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			
初始值	0	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			
初始值	0	0	0	-	0	1	1	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			
初始值	0	0	0	-	0	1	1	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			
初始值	0	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			
初始值	0	0	0	-	0	1	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			
初始值	0	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			
初始值	0	0	0	-	0	0	1	1
8018H	7	6	5	4	3	2	1	0
P30F	P30PUP	P30PDP	P30OPR	-	P30S			
R/W	R/W	R/W	R/W	-	R/W			
初始值	0	0	0	-	0	0	0	0

备注：

PXnF 代表 P00F~P20F，其中 X=0,1,2,3 代表 P0~P3，n=0,1,2,...,7(当 X=2，3 时 n=0)。

位编号	位符号	说明
7	PnxPUP	上拉电阻使能控制位 0: 上拉电阻关闭 1: 上拉电阻打开
6	PnxPDP	下拉电阻使能控制位 0: 下拉电阻关闭 1: 下拉电阻打开
5	PnxOPR	开漏使能控制位，引脚设为数字输出时才有效 0: 开漏关闭 1: 开漏打开

表 15-2-6 寄存器 PXnC

8120H	7	6	5	4	3	2	1	0
P00C	SINK_EN	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	R/W	R/W	R/W	R/W	R/W		R/W	
初始值	0	1	1	1	1	1	1	1
8121H	7	6	5	4	3	2	1	0
P01C	SINK_EN	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	R/W	R/W	R/W	R/W	R/W		R/W	
初始值	0	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	
初始值	-	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	
初始值	-	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	
初始值	-	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	
初始值	-	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	
初始值	-	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	
初始值	-	1	1	1	1	1	1	1
8128H	7	6	5	4	3	2	1	0
P10C	SINK_EN	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	R/W	R/W	R/W	R/W	R/W		R/W	
初始值	0	1	1	1	1	1	1	1
8129H	7	6	5	4	3	2	1	0
P11C	SINK_EN	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	R/W	R/W	R/W	R/W	R/W		R/W	
初始值	0	1	1	1	1	1	1	1
812AH	7	6	5	4	3	2	1	0
P12C	SINK_EN	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	R/W	R/W	R/W	R/W	R/W		R/W	
初始值	0	1	1	1	1	1	1	1
812BH	7	6	5	4	3	2	1	0
P13C	SINK_EN	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	R/W	R/W	R/W	R/W	R/W		R/W	
初始值	0	1	1	1	1	1	1	1
812CH	7	6	5	4	3	2	1	0
P14C	SINK_EN	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	R/W	R/W	R/W	R/W	R/W		R/W	
初始值	0	1	1	1	1	1	1	1
812DH	7	6	5	4	3	2	1	0
P15C	SINK_EN	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	R/W	R/W	R/W	R/W	R/W		R/W	
初始值	0	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	
初始值	-	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	
初始值	-	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	
初始值	-	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	
初始值	-	1	1	1	1	1	1	1

备注：

PXnC 代表 P00C~P20C，其中 X=0,1,2,3 代表 P0~P3，n=0,1,2,...,7(当 X=2，3 时 n=0)。

位编号	位符号	说明
7	-	-
6	SMIT_EN	为 1 输入的 SMIT 使能，为 0 输入是反相器使能
5	PU_SEL	为 1 时上拉电阻 10K，为 0 时上拉电阻 45K
4	PD_SEL	为 1 时下拉电阻 15K，为 0 时下拉电阻 45K
3~2	DRV	输出强度选择 00：10mA 01：20mA 10：40mA 11：70mA
1~0	SR	输出斜率控制 00：最慢斜率控制 01： 10： 11：最快斜率控制

表 15-2-7 引脚复用功能映射表

取值 名称	0	1	2	3	4	5	6	7	8	9
P00S	高阻	数字输入 /T1/INT[n] /EPI[n]	数字输出	高阻	高阻	PWM[n]	SPI_MOSI	T2CP[n]	TK[10]	高阻
P01S	高阻	数字输入 /INT[n]/EPI[n]	数字输出	ADC[2]	高阻	PWM[n]	SPI_MISO	T2CP[n]	TK[9]	高阻
P02S	高阻	数字输入 /INT[n]/EPI[n]	数字输出	SWIM	UART2_RX	PWM[n]	I2C1_SCL	T2CP[n]	TK[8]	高阻
P03S	高阻	数字输入 /INT[n]/EPI[n]	数字输出	ADC[3]	高阻	PWM[n]	AMP_B_INP	T2CP[n]	TK[7]	高阻
P04S	高阻	数字输入 /INT[n]/EPI[n]	数字输出	ADC[4]	STADC0	PWM[n]	AMP_B_INN	T2CP[n]	TK[6]	高阻
P05S	高阻	数字输入 /T0/INT[n]/EPI [n]	数字输出	ADC[5]	BEEP	PWM[n]	AMP_A_INP	T2CP[n]	TK[5]	高阻
P06S	高阻	数字输入 /INT[n]/EPI[n]	数字输出	ADC[6]	UART1_TX	PWM[n]	AMP_A_INN	T2CP[n]	TK[4]	I2C0_SCL

P07S	高阻	数字输入 /INT[n]/EPI[n]	数字输出	ADC[7]	UART1_RX	PWM[n]	AMP_A_OUT	T2CP[n]	TK[3]	I2C0_SDA
P10S	高阻	数字输入 /T2/INT[n]/EPI [n]	数字输出	高阻	高阻	PWM[n]	SPI_SCK	T2CP[n]	TK[11]	高阻
P11S	高阻	数字输入 /T2EX/INT[n]/EPI[n]	数字输出	ADC[1]	高阻	PWM[n]	CLO	T2CP[n]	TK[12]	高阻
P12S	高阻	数字输入 /INT[n]/EPI[n]	数字输出	ADC[0]	ADC_VREF	PWM[n]	高阻	T2CP[n] /T2CPO	TK[13]	高阻
P13S	高阻	数字输入 /INT[n]/EPI[n]	数字输出	高阻	STADC1	PWM[n]	I2C2_SCL	T2CP[n]	TK[14]	LED1
P14S	高阻	数字输入 /INT[n]/EPI[n]	数字输出	高阻	FB	PWM[n]	I2C2_SDA	T2CP[n]	TK[15]	LED0
P15S	高阻	数字输入 /INT[n]/EPI[n]	数字输出	ADC[11]	高阻	PWM[n]	SPI_SSB	T2CP[n]	TK[16]	高阻
P16S	高阻	数字输入 /INT[n]/EPI[n]	数字输出	ADC[10]	UART2_TX	PWM[n]	I2C1_SDA	T2CP[n]	TK[0]	高阻
P17S	高阻	数字输入 /INT1/INT[n]/EPI[n]	数字输出	ADC[9]	32M_O	PWM[n]	高阻	T2CP[n]	TKCAPS	高阻
P20S	高阻	数字输入 /INT[n]/EPI[n]	数字输出	RESET	高阻	PWM[n]	高阻	T2CP[n]	TK[2]	高阻
P30S	高阻	数字输入 /INT0/INT[n] /EPI[n]	数字输出	ADC[8]	32M_I	PWM[n]	CLK_IN	T2CP[n]	TK[1]	高阻

15.3 引脚控制例程

引脚功能设置

例如，P00 设置为推挽输出，程序如下：

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

P00 设置为开漏输出，程序如下：

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

P00 设置为开漏输出，并且打开上拉，程序如下：

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

P00 设置为输入功能，并且打开上拉，程序如下：

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

16 通用串行接口（UART1/UART2）

16.1 UART1 和 UART2

16.1.1 介绍

UART1 和 UART2 是设计完全相同的两个全双工异步串行数据收发器，UARTx（x=1、2,代指 UART1、UART2）也有一字节的接收缓存。UARTx 有两种不同的工作模式，如表 16-1-1-1 所示。

SMx	模式	描述	波特率
0	A	9 位异步模式	CPUCLK/(32*(1024-SxREL))
1	B	8 位异步模式	CPUCLK/(32*(1024-SxREL))

表 16-1-1-1 UARTx 工作模式

UARTx 设计了专门的波特率发生器，波特率通过寄存器 SxRELL、SxRELH 来配置。

备注：UARTx 的波特率不能通过 PCON 寄存器的 SMOD 位来设置倍频。

图 16-1-1-1 是 UARTx 的原理示意图。

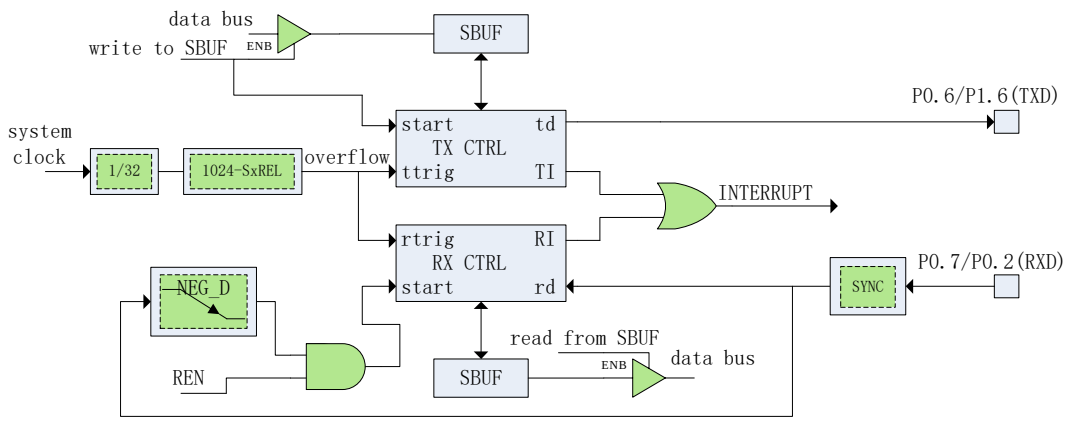


图 16-1-1-1 UARTx 工作原理示意图

- **模式 A**

在模式 A，UARTx 可异步同时收发 9 位数据。写入数据到寄存器 SxBUF 会启动 UARTx 数据发送。第一个传送的位是开始位（为 0），然后是 9 位数据（低位先发），第 9 位数据是寄存器 SxCON 的 TB8x 位，最后传送的是停止位（为 1）。在接收状态，UARTx 通过检测引脚 RX 的下降沿来同步。传送过程完成后，低 8 位数据存放在寄存器 SxBUF，第 9 位数据存放在 RB8x 位。

- **模式 B**

模式 B 和模式 A 不同的是，模式 B 是 8 位数据传输，停止位存放的是有效停止位。其他功能和模式 A 一致。

- **UARTx 多机通信**

在 UARTx 模式 A 中有一个专门适用于多机通信的机制。当寄存器 SxCON 的 SM2x 位置 1，只有接收到第 9 位数据为 1（RB8x=1）的从机才会产生接收中断，利用这个功能可进行多机通信，从机将它们的 SM2x 位都置为 1，主机传送从机的地址时将第 9 位数据设为 1，这样所有的从机都会产生接收中断；从机的软件用它们自己的地址和接收的地址进行比较，如果一致，被寻址的从机设置 SM2x=0，然后主机继续传送后面的数据时设置第 9 位为 0，因为其他的从机SM2x 仍然设为 1，这样就只有被寻址的从机才会产生接收中断。

16.1.2 UARTx 寄存器描述

表 16-1-2-1 寄存器 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
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7	SM1	UART1 模式选择位，详见表 16-1-1-1
6	U1IE	UART1 中断使能位，1 有效
5	SM21	多机通信使能位，1 有效
4	REN1	串行接收使能位，1 有效
3	TB81	发送数据的第 9 位 在模式 A，这个位用于 UART1 传送数据，对应传送数据的第 9 位 （例如奇偶校验或多主机通信），由软件控制
2	RB81	接收数据的第 9 位 在模式 A，这个位用于 UART1 接收数据，对应接收数据的第 9 位； 在模式 B，这个位是接收到的停止位
1	TI1	传送中断标志位，1 有效，写 1 清 0
0	RI1	接收中断标志位，1 有效，写 1 清 0

表 16-1-2-2 寄存器 S1BUF

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

位编号	位符号	说明
7~0	S1BUF	UART1 收发缓冲器 写 S1BUF 将开始发送所写的数据 读 S1BUF 将得到已经接收的数据

表 16-1-2-3 寄存器 S1RELL、S1RELH

9CH	7	6	5	4	3	2	1	0
S1RELL	S1RELL[7:0]							
R/W	R/W							

初始值	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	
初始值	-	-	-	-	-	-	0	0

位编号	位符号	说明
9~0	S1REL	波特率配置寄存器 波特率为 CPUCLK/(32 * (1024 - S1REL))

表 16-1-2-4 寄存器 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
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7	SM2	UART2 模式选择位，详见表 16-1-1-1
6	U2IE	串口 2 中断使能位，1 有效
5	SM22	多机通信使能位，1 有效
4	REN2	串行接收使能位，1 有效
3	TB82	发送数据的第 9 位 在模式 A，这个位用于串口 1 传送数据，对应传送数据的第 9 位（例如奇偶校验或多主机通信），由软件控制
2	RB82	接收数据的第 9 位 在模式 A，这个位用于 UART2 接收数据，对应接收数据的第 9 位 在模式 B，这个位是接收到的停止位
1	TI2	传送中断标志位，1 有效，写 1 清 0
0	RI2	接收中断标志，1 有效，写 1 清 0

表 16-1-2-5 寄存器 S2BUF

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

位编号	位符号	说明
7~0	S2BUF	收发缓冲器 写 S2BUF 将开始发送所写的数据 读 S2BUF 将得到已经接收的数据

表 16-1-2-6 寄存器 S2REL

A3H	7	6	5	4	3	2	1	0
S2RELL	S2RELL[7:0]							
R/W	R/W							
初始值	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	
初始值	-	-	-	-	-	-	0	0

位编号	位符号	说明
9~0	S2REL	波特率配置寄存器 波特率为 CPUCLK/(32 * (1024 - S2REL))

表 16-1-2-7 寄存器 UDCKSx

8118H	7	6	5	4	3	2	1	0
UDCKS1	UDE1	-	-	DNUM1[4:0]				
R/W	R/W	-	-	R/W				
初始值	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				
初始值	0	-	-	0	0	0	0	0

位编号	位符号	说明
7	UDEx	快速波特率配置使能控制位，1 有效 ***备注：*** *UDEx=0 时，UARTx 波特率按照原来的配置，UDEx=1，UARTx 波特率由 DNUMx 来配置。*
6~5	-	-
4~0	DNUMx	快速波特率配置寄存器，仅在 UDEx=1 时有效 发送时，须满足 DNUMx>=0；接收时，DNUMx>=6 $BRx = F_{sys}*(1/((DNUMx+1)*(1024-SxRELL)))$

17 I²C 接口

17.1 功能简介

I²C 模块支持芯片与外围 I²C 器件以标准 I²C 协议进行串行数据传输，可设置为主机或从机模式，通过合理配置可使I²C 支持标准/快速/高速模式。

17.2 I²C 主要特点

- 简单且强大而灵活的通讯接口，双向两线总线
- 可设置为主机或从机模式
- 可以工作于发送器模式或接收器模式
- 7 位从机地址
- 支持多主机仲裁
- 支持广播功能

17.3 I²C 功能描述

I²C 模块支持 I²C 标准总线协议。I²C 总线用 2 根线在设备间传输数据，分别为 SCL（串行时钟线）和 SDA（串行数据线），如图 17-3-1 所示。由于 I²C 端口是开漏结构，所以 I²C 总线上必须有上拉电阻，上拉电阻可以外接也可以在芯片内部打开。每个连接在总线上的设备都有一个唯一的 7 位地址。

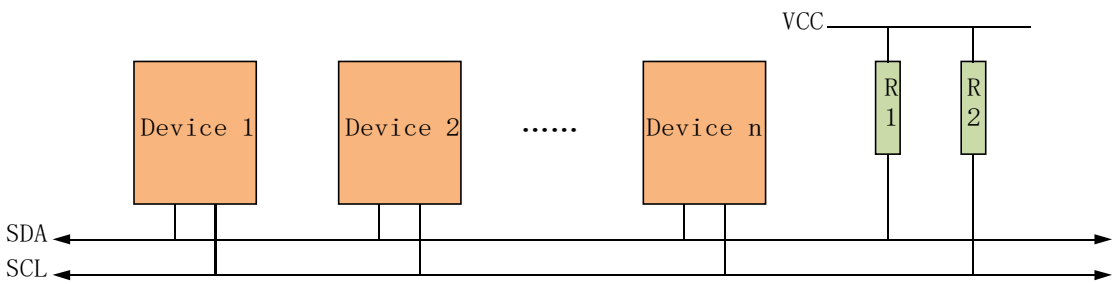


图 17-3-1 I2C 总线互连图

I²C 模块原理示意图如图图 17-3-2 所示。

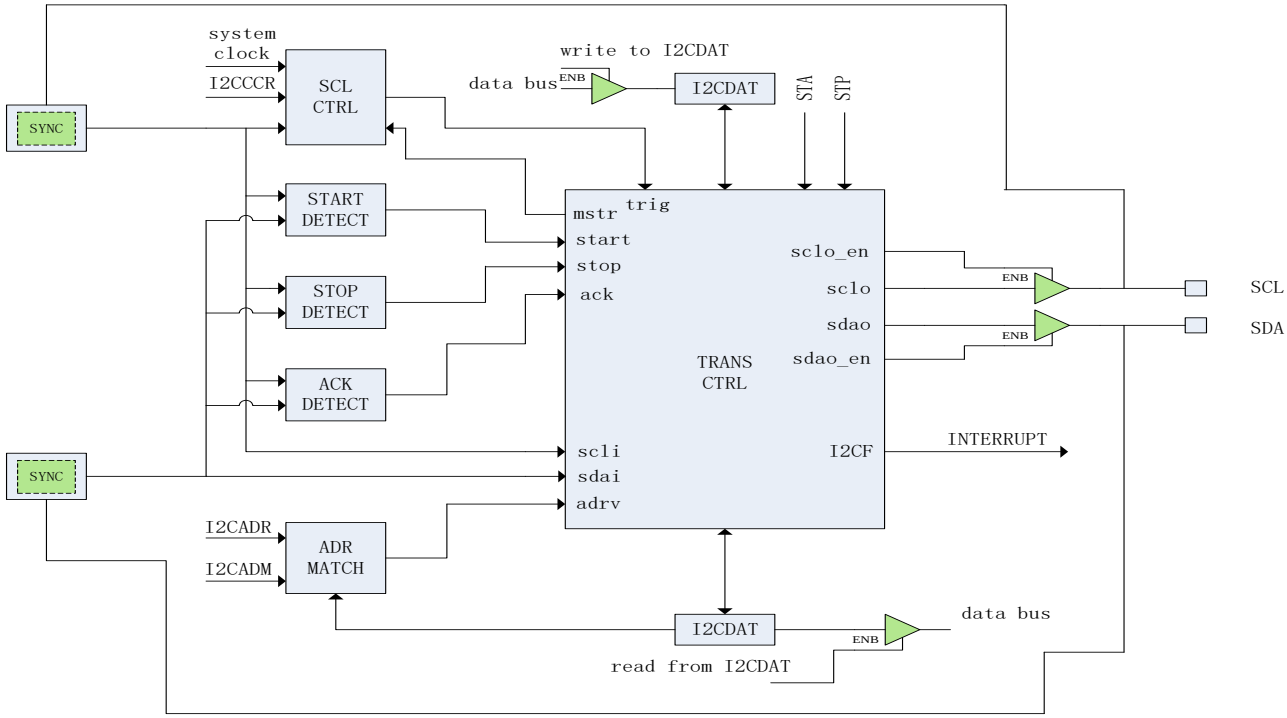


图 17-3-2 I²C 模块原理示意图

- **I²C 模式选择**

I²C 可以在以下 4 种模式中的一种运行：从机发送模式、从机接收模式、主机发送模式、主机接收模式。默认情况下，I²C 处于从机模式。I²C 在产生开始信号后自动从从机模式切换到主机模式，当仲裁失败或产生 STOP 信号后又自动切回从机模式。

- **I²C 总线数据传输格式**

一般情况下，标准的 I²C 通信由四部分组成：开始信号、从机地址传输、数据传输和结束信号。I²C 总线上传送的数据均为 8 位，高位先发，每发送一个字节后都必须跟随一个应答位，每次通信的数据字节数没有限制；在全部数据传送结束后，由主机发送停止信号，结束通信。

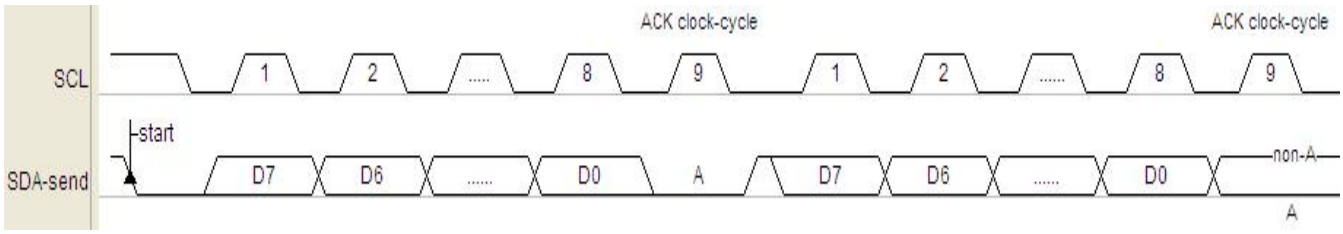


图 17-3-3 I2C 总线数据传输格式

- **通信过程**

在主机模式下，I²C 接口启动数据传输并产生时钟信号。串行数据传输总是以 START 信号开始，以 STOP 信号结束。START 信号和 STOP 信号都是在主机模式下通过软件控制产生的，START 信号通过设置 STA=1 产生，而STOP 信号通过设置 STP=1 产生。

在从机模式下，I²C 接口能识别自身地址（7 位地址）和广播地址。软件能通过 GCE 位使能或禁止广播地址的识别。

地址和数据以字节为单位进行传输，地址会跟在 START 信号之后由主机发送。在一个字节传输的 8 个时

钟后的第 9 个时钟周期内，接收器必须回送一个应答位给发送器。应答位通过 AAK 位设置，设置应答位必须在一个字节传输完之前设置，接收器完成一个字节接收时，应答信号自动产生。数据传输过程中，数据发送/接收完一字节、仲裁失败等事件都会产生中断标志 I2CF，而事件的状态则由寄存器 I2CSTA 指示（详细请参考寄存器 I2CSTA 介绍），软件应在产生中断标志后根据事件的状态设置数据传输的下一步操作，清除中断标志 I2CF 将启动下一步操作。通信结束后主机产生 STOP 信号也会在从机端产生中断标志 I2CSTP，指示通信过程的完成。当中断标志 I2CF 产生时，如果 SHD=1，在没有清除 I2CF 之前，SCL 会被从机拉低，主机检测到 SCL 被释放后才会进行下一步操作；如果 SHD=0，从机不会拉低 SCL，这样设计是为了兼容主机是软件模拟 I2C 的应用，此时，主机的软件必须等待足够长的时间让从机响应每字节数据传输的处理。

当 I²C 接口作为从机时，SCL 的时钟由主机输入，和从机的时钟配置无关。作为从机时，需要保证 SCL 为低电平的宽度最少为 6.5 个系统时钟，而高电平最少为 2.5 个系统时钟。所以，外部主机发送的 SCL 频率最高为系统时钟频率的 1/9。

17.4 I²C 通信引脚的映射

为了方便硬件设计，I²C 通信引脚可以有不同的映射，由寄存器 I2CIOS 设置不同的值来选择。详细见寄存器I2CIOS 的描述。

17.5 寄存器描述

表 17-5-1 寄存器 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
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7	I2CE	I²C 模块使能位，1 有效
6	I2CIE	I²C 中断使能位，1 有效
5	STA	I²C 发送 START 信号控制位，1 有效，检测到 START 信号后将自动清 0
4	STP	I²C 发送 STOP 信号控制位，1 有效，检测到 STOP 信号后将自动清 0
3	SHD	为 1 时，如果I2CF 为 1，那么当 SCL 变低之后，I2CF 将会使 SCL 保持在低的状态
2	AAK	I²C 发送 ACK 信号控制位，1 有效 *备注：* *当 I²C 接口配置为从机模式时，这一位须预先置 1，否则即使地址匹配也不会回复 ACK，从而无法被寻址。*
1	CBSE	CBUS 兼容使能位 当这一位设置为 1 时，将会使传输忽略 ACK 位的状态判断，以兼容 CBUS 总线。
0	STFE	为 1 时，I²C 模块检测到 START 信号时将置位I2CF

表 17-5-2 寄存器 I2CADR

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

位编号	位符号	说明

7	GCE	识别广播地址（00H）使能位，1 有效
6~0	I2CADRL	I²C 从机地址，作为从机时有效 *备注：* *（在 AAK 为 1 的前提下）7 位地址模式时，接收的第一个地址字节高 7 位和 I2CADR 匹配，则回复 ACK，进入从机模式。*

表 17-5-3 寄存器 I2CADM

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

位编号	位符号	说明
7	SPFE	为 1 时，I²C 模块检测到 STOP 信号时将置位 I2CF
6~0	I2CADML	I²C 从地址按位屏蔽寄存器，为从机时有效 当 I2CADM[n](n=0~6)=1 时，对应的地址位 I2CADR[n]将不比对（即认为无论收到 1 还是 0 都算匹配）。

表 17-5-4 寄存器 I2CCCR

B4H	7	6	5	4	3	2	1	0
I2CCCR	I2CCCR[7:0]							
R/W	R/W							
初始值	0	0	1	0	0	0	0	0

位编号	位符号	说明
7~0	I2CCCR	I²C 时钟配置寄存器 采样频率为 I²C 工作时钟的 $2^{I2CCR[7:5]}$分频，当 I2CCR[7:5]等于 000：$F_{sample}=F_{i2cclk}$ 001：$F_{sample}=F_{i2cclk}/2$ 010：$F_{sample}=F_{i2cclk}/4$ … 111：$F_{sample}=F_{i2cclk}/128$ 输出频率为采样频率的(I2CCCR[4:0]+1)分频， $F_{scl}=F_{i2cclk}/(2^{I2CCCR[7:5]}*(I2CCCR[4:0]+1))$ 例如 I2CCCR[4:0]=9 时，当 I2CCR[7:5]等于 000：$F_{scl}=F_{i2cclk}/(1*10)$ 001：$F_{scl}=F_{i2cclk}/(2*10)$ 010：$F_{scl}=F_{i2cclk}/(4*10)$ … 111：$F_{scl}=F_{i2cclk}/(128*10)$ ***备注：*** *1. 当I2CCCR[7:5]=0 时，如果对 I2CCCR[4:0]写小于 9 的值，将自动按 9 的值计算。* *2. 当I2CCCR[7:5]>0 时，如果对 I2CCCR[4:0]写小于 7 的值，将自动按 7 的值计算。*

表 17-5-5 寄存器 I2CDAT

B5H	7	6	5	4	3	2	1	0

I2CDAT	I2CDAT[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~0	I2CDAT	发送和接收数据缓存 *备注：* *当I2CF 为1 时，建议改写/读取 I2CDAT 时，让 I2CF 保持在 1，等处理完成之后再清除 I2CF，以继续传输，这样可以避免总线发生不必要的错误。*

表 17-5-6 寄存器 I2CSTA

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

位编号	位符号	说明
7~0	I2CSTA	I²C 状态寄存器 00H:（主/从）总线错误 08H:（主/从）检测到 START 信号（只在 STFE=1 时才有效） 18H:（主）已发送地址+写位，已接收到应答信号 20H:（主）已发送地址+写位，无接收到应答信号 28H:（主）已发送/接收一字节数据，已检测到应答信号 30H:（主）已发送/接收一字节数据，无检测到应答信号 38H:（主）失去仲裁（主机失去仲裁后会变为从机） 40H:（主）已发送地址+读位，已接收到应答信号 48H:（主）已发送地址+读位，无接收到应答信号 60H:（从）已接收地址+写位，已发送出应答信号 70H:（主/从）已接收广播地址，已发送出应答信号（主机或从机都会变为从机） 80H:（从）已发送/接收一字节数据，已检测到应答信号 88H:（从）已发送/接收一字节数据，无检测到应答信号 A0H:（主/从）检测到 STOP 信号（只在 SPFE=1 时才有效） A8H:（从）已接收地址+读位，已发送出应答信号 F8H:（主/从）总线空闲

表 17-5-7 寄存器 I2CFLG

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

位编号	位符号	说明
7~1	-	-
0	I2CF	I²C 中断标志，1 有效，写 1 清 0 *备注：* *1. 每字节地址或数据传输完成后（收到/发送完 ACK/NAK），将置位 I2CF。* *2. 总线出错时，将置位 I2CF。* *3. 当STFE=0 时，检测到 START 信号，I2CF 不会置 1。* *4. 当SPFE=0 时，检测到 STOP 信号，I2CF 不会置 1。*

表 17-5-8 寄存器I2CIOS

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

位编号	位符号	说明
7	I2CKS	I2C 工作时钟选择位 0：系统时钟 1：内部高速时钟
6~2	-	-
1~0	I2CS	I2C 引脚选择位 0：SCL 在引脚 P0.7，SDA 在引脚 P0.6 1：SCL 在引脚 P0.2，SDA 在引脚 P1.6 2: SCL 在引脚 P1.3，SDA 在引脚 P1.4

17.6 I²C 控制例程

I²C 作为主机例程

例如，主机循环向从机写入 20 字节数据，程序如下：

```
//I2CCON 定义
#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 定义
#define   GCE           (1<<7)
//I2CFLG 定义
#define I2CF            (1<<0)
#define I2CSTP          (1<<1)

#define I2C_ADDR        0xCA            //定义 I2C 从机地址
unsigned char xdata WriteBuffer[20]={1,2,3,4,5,6,7,8,9,10,11,12,12,14,15,16,17,18,19,20};
unsigned char xdata ReadBuffer[20];
bit printf_flag = 0;

void main(void)
{
    unsigned char i;
    CKSEL = 3;
    EA = 1;                                 //开全局中断
```

```
/**********选择 I2C 端口**************************************************************/
    //0：SCL 在引脚 P0.7，SDA 在引脚 P0.6
    //1：SCL 在引脚 P0.2，SDA 在引脚 P1.6
    //2：SCL 在引脚 P1.3，SDA 在引脚 P1.4

    //    I2CIOS = 0;
    //    P06F = P06_I2C_SDA_SETTING | PU_EN;
    //    P07F = P07_I2C_SCL_SETTING | PU_EN;

    //    I2CIOS = 1;
    //    P16F = P16_I2C_SDA_SETTING | PU_EN;
    //    P02F = P02_I2C_SCL_SETTING | PU_EN;

    I2CIOS = 2;
    P14F = P14_I2C_SDA_SETTING | PU_EN;
    P13F = P13_I2C_SCL_SETTING | PU_EN;

/*******************************************************************************************/

    I2CCON = I2CE(1) | I2CIE(0) | STA(0) | STP(0)| CKHD(1) | AAK(1)| CBSE(0) | STFE(1);
    I2CADR = GCE(0);
    I2CCCR = 0x4C;                                  //设置 I2C 时钟
    while(1)
    {
        I2CCON |= STA(1);                           //I2C 主机发送 START 信号
        while(!(I2CFLG & I2CF));    //等待中断标志产生

        if(I2CSTA != 0x08)
        {
            I2CFLG  |= I2CF;
            goto SEND_STOP;
        }
        I2CDAT = I2C_ADDR;                          //主机发送从机地址+写位
        I2CFLG  |= I2CF;                            //清除中断标志

        while(!(I2CFLG & I2CF));    //等待中断标志产生
        if(I2CSTA != 0x18)
        {
            I2CFLG  |= I2CF;
            goto SEND_STOP;
        }

        I2CDAT = 0;                                 //主机发送数据寄存器地址
        I2CFLG  |= I2CF;                            //清除中断标志
```

```
            while(!(I2CFLG & I2CF));      //等待中断标志产生
            if(I2CSTA != 0x28)
            {
                I2CFLG |= I2CF;
                goto SEND_STOP;
            }

            for(i = 0; i < 20; i++)             //主机发送 20 数据
            {
                I2CDAT =      WriteBuffer[i];
                I2CFLG |= I2CF;                       //清除中断标志
                while(!(I2CFLG & I2CF));//等待中断标志产生
                if(I2CSTA != 0x28)
                {
                    I2CFLG |= I2CF;
                    goto SEND_STOP;
                }
            }
SEND_STOP:
            I2CCON |= STP(1);                   //发送 STOP 信号
            I2CFLG |= I2CF;
            Delay_ms(10);
        }
}
```

例如，主机循环从从机读取 20 字节数据，程序如下：

```
#define I2C_ADDR        0xCA            //定义 I2C 从机地址
#define DATA_LEN        20
unsigned char xdata ReadBuffer[DATA_LEN];
void main(void)
{
    unsigned char i;
    CKSEL = 3;
    EA = 1;                                 //开全局中断

/**********选择 I2C 端口**********************************************************/
    //0：SCL 在引脚 P0.7，SDA 在引脚 P0.6
    //1：SCL 在引脚 P0.2，SDA 在引脚 P1.6
    //2：SCL 在引脚 P1.3，SDA 在引脚 P1.4

    //  I2CIOS = 0;
    //  P06F = P06_I2C_SDA_SETTING |PU_EN;
    //  P07F = P07_I2C_SCL_SETTING |PU_EN;
```

```
//    I2CIOS = 1;
//    P16F = P16_I2C_SDA_SETTING | PU_EN;
//    P02F = P02_I2C_SCL_SETTING | PU_EN;

    I2CIOS = 2;
    P14F = P14_I2C_SDA_SETTING | PU_EN;
    P13F = P13_I2C_SCL_SETTING | PU_EN;

/*****************************************************************************************/

    I2CCON = I2CE(1) | I2CIE(0) | STA(0) | STP(0)| CKHD(1) | AAK(1)| CBSE(0) | STFE(1);
    I2CADR = GCE(0);
    I2CCCR = 0x4C;                                  //设置 I2C 时钟
    while(1)
    {
        I2CCON |= STA(1);             //I2C 主机发送 START 信号
        while(!(I2CFLG & I2CF));      //等待中断标志产生

        if(I2CSTA != 0x08)
        {
            I2CFLG |= I2CF;
            goto READ_MASS_STOP;
        }
        I2CDAT = I2C_ADDR;            //主机发送从机地址+写位
        I2CFLG |= I2CF;               //清除中断标志

        while(!(I2CFLG & I2CF));      //等待中断标志产生
        if(I2CSTA != 0x18)
        {
            I2CFLG |= I2CF;
            goto READ_MASS_STOP;
        }

        I2CDAT = 0;                   //主机发送数据寄存器地址
        I2CFLG |= I2CF;               //清除中断标志
        while(!(I2CFLG & I2CF));      //等待中断标志产生
        if(I2CSTA != 0x28)
        {
            I2CFLG |= I2CF;
            goto READ_MASS_STOP;
        }

        I2CCON |= STA(1);             //I2C 主机发送 START 信号
        I2CFLG  |= I2CF;              //清除中断标志
```

```
    while(!(I2CFLG & I2CF));            //等待中断标志产生
    if(I2CSTA != 0x08)
    {
        I2CFLG |= I2CF;
        goto READ_MASS_STOP;
    }

    I2CDAT = I2C_ADDR+1;                //主机发送从机地址+读位
    I2CFLG |= I2CF;                     //清除中断标志
    while(!(I2CFLG & I2CF));            //等待中断标志产生
    if(I2CSTA != 0x40)
    {
        I2CFLG |= I2CF;
        goto READ_MASS_STOP;
    }
#if (DATA_LEN == 1)    //读取单字节
    I2CCON &= ~AAK(1);
    I2CFLG |= I2CF;                     //清除中断标志
    while(!(I2CFLG & I2CF));            //等待中断标志产生
    ReadBuffer[0] = I2CDAT;             //读取数据到数据寄存器
#else                   //连续读取多字节  I2CCON |=
    AAK(1); //设置应答位 for(i = 0; i < DATA_LEN;
    i++)
    {
        I2CFLG |= I2CF;                 //清除中断标志
        while(!(I2CFLG & I2CF));        //等待中断标志产生
        if((I2CSTA != 0x28)&&(I2CSTA != 0x30))
        {
            I2CFLG |= I2CF;
            goto READ_MASS_STOP;
        }
        ReadBuffer[i] = I2CDAT;         //读取数据到数据寄存器
        if(i == (DATA_LEN - 2))
        {
            I2CCON &= ~AAK(1);  //读最后一字节，不发送 ACK
        }
        else
        {
            I2CCON |= AAK(1);       //如果不是最后一字节，预设 ACK 状态
        }
    }
#endif
READ_MASS_STOP:
    I2CCON |= STP(1);                   //发送 STOP 信号
    I2CFLG |= I2CF;
```

```
        Delay_ms(100);
    }
}
```

I2C 作为从机例程

作为从机，支持主机写入或读取数据，程序如下：

```
#define I2C_ADDR        0xCA            //定义 I2C 从机地址
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};//设置数据寄存器初值为 0~19
void I2C_init(void)
{
    /**********选择 I2C 端口**************************************************************/
    //0：SCL 在引脚 P0.7，SDA 在引脚 P0.6
    //1：SCL 在引脚 P0.2，SDA 在引脚 P1.6
    //2：SCL 在引脚 P1.3，SDA 在引脚 P1.4

    //    I2CIOS = 0;
    //    P06F = P06_I2C_SDA_SETTING | PU_EN;
    //    P07F = P07_I2C_SCL_SETTING | PU_EN;

    //    I2CIOS = 1;
    //    P16F = P16_I2C_SDA_SETTING | PU_EN;
    //    P02F = P02_I2C_SCL_SETTING | PU_EN;

    I2CIOS = 2;
    P14F = P14_I2C_SDA_SETTING | PU_EN;
    P13F = P13_I2C_SCL_SETTING | PU_EN;
    /*****************************************************************************************/

    I2CCON = I2CE(1) | I2CIE(1) | STA(0) | STP(0)| CKHD(1) | AAK(1)| CBSE(0) | STFE(0);
    I2CADR = GCE(0)|(I2C_ADDR>>1);          //设置 I2C 从机地址
    I2CCCR = 0x20;                                          //设置 I2C 时钟采样时钟
    INT6EN = 1;                                             //I2C 中断开启
}

void INT6_ISR(void) interrupt 11
{
    unsigned char Sta_Temp;

    if(I2CFLG & I2CF)                       //IIC  interrupt
    {
```

```
        Sta_Temp = I2CSTA;
        if(Sta_Temp == 0x60)                    //接收到从机地址+写位
        {
            I2CDataIndex = 0xFF;                //设置为 0xFF 表示后面接收到的第一个字节为地址
            iicReadMode = 0;                    //设置为从机接收状态
            I2CCON |= AAK(1);
        }
        else if(Sta_Temp == 0x80)               //发送或接收一字节数据，已检测到应答信号
        {
            if(iicReadMode)                     //发送一字节数据
            {
                I2CDataIndex++;
                I2CDAT = Buffer[I2CDataIndex + regAddr];    //把数据装载到发送寄存器，等待主机读取
            }
            else                                //接收到一字节数据
            {
                if(I2CDataIndex == 0xFF)        //地址
                {
                    regAddr = I2CDAT;           //接收到的第一个字节认为是地址
                    I2CDataIndex = 0;           //设置索引值为 0
                    I2CCON |= AAK(1);
                }
                else                            //数据
                {
                    Buffer[I2CDataIndex + regAddr] = I2CDAT;    //接收到的数据装载到数据寄存器
                    I2CDataIndex++;             //索引值累加
                    I2CCON |= AAK(1);
                }
            }
        }
        else if(Sta_Temp==0xA8)                 //接收到从机地址+读位，发送 ACK 信号
        {
            I2CDAT = Buffer[I2CDataIndex + regAddr]; //把数据装载到发送寄存器，等待主机读取
            iicReadMode = 1;                    //设置为从机发送状态
        }
        else if(Sta_Temp == 0x88)               //发送或接收一字节数据，已检测到应答信号
        {
        }
        I2CFLG |= I2CF;                         //清除中断标志
    }
}

void main(void)
{
    CKSEL = 3;
```

```
    I2C_init();
    EA = 1; //开启全局中断
    while(1)
    {
    }
}
```

---

18 PWM

18.1 PWM 功能简介

JZ8FC005T 系列芯片最多有 8 通道PWM 输出，PWM 周期和占空比可在 16 位范围内任意配置。其中PWM[0~5]每路都可独立设置任意引脚作为PWM 输出引脚，PWM[0~5]每路都支持边沿对齐模式和中心对称模式。此外，PWM[0~5]还支持死区控制及互补输出，设置为互补模式时，6 路 PWM 组成 3 对互补通道。PWM 支持软件及硬件刹车功能，并可设置 PWM 暂停输出，PWM 的这种特性是专门针对无刷直流电机驱动而设计的。PWM[6~7]这两路PWM 为固定引脚，PWM6(LED_D0)/PWM7(LED_D1)支持级联LED 驱动，扫描频率大于 400Hz/S，数据发送速度 800Kbps/S，可直接控制 WS2812 或类似的驱动芯片，符合单色或七彩 LED 灯带产品的需求。

18.2 PWM(0~5)功能描述

每路 PWM 通道都有一个专门的 16 位计数器，PWM 的周期通过寄存器 PWMDIV 来设置，而寄存器 PWMDUT 则对应 PWM 的占空比。PWM 通过寄存器 PWMEN 使能，寄存器 PWMEN 的每一位对应 PWM 的一个通道。PWM 模块有一个 PWM 数据更新寄存器 PWMUPD，当改写寄存器 PWMDIV、PWMDUT 和 PWMCKD 时，寄存器 PWMUPD 必须置位相应位才会更新数据，当数据刷新之后 PWMUPD 相应位自动清 0。PWM 可通过 PWMTOG 位设置 PWM 引脚输出反相。PWM 有多种时钟源可以选择，时钟源是以两路 PWM 为单位进行设置的，分别是：PWM0 和 PWM1、PWM2 和 PWM3、PWM4 和 PWM5，也就是说，每组 PWM 的时钟源是共同设置的，时钟源通过 PWM0、PWM2、PWM4 对应的控制寄存器 PWMCON 的 PWMCKS 来选择。另外，每路 PWM 的时钟分频可通过 PWMCKD 独立设置。每路 PWM 都可选择选择任意的引脚作为 PWM 输出引脚，通过寄存器PWMPS 选择，详见寄存器部分描述。

- **边沿对齐模式和中心对齐模式**

PWM 的边沿对齐模式和中心对齐模式通过 PWMMS 位来选择。PWM 使能后，PWM 计数器从 0 开始累加计数，当计数值小于 PWMDUT 时，PWM 引脚输出高电平（PWMTOG=0 ），当计数值大于或等于 PWMDUT 时，PWM 引脚输出低电平（PWMTOG=0）。在边沿对齐模式下，当计数值与 PWMDIV 相等时，一个 PWM 周期完成，PWM 计数器重新置 0 并开始下一周期计数。在中心对齐模式下，当计数值达到 PWMDIV 值时，计数方向反转，开始递减计数，在此阶段内，同样是计数值小于 PWMDUT 时，PWM 引脚输出高电平（PWMTOG=0），计数值大于或等于 PWMDUT 时，PWM 引脚输出低电平（PWMTOG=0）；当计数递减到 0 时，一个 PWM 周期完成，计数重新开始下一个周期累加计数。

当边沿对齐模式和中心对齐模式单路 PWM 输出波形见图 18-2-1 和图 18-2-2（注：以下所有 PWM 波形满足条件 PWMDIV>PWMDUT>0）。如图所示，设置相同的 PWMDIV 和 PWMDUT 值，中心对齐模式一个 PWM 周期是边沿对齐模式的两倍。

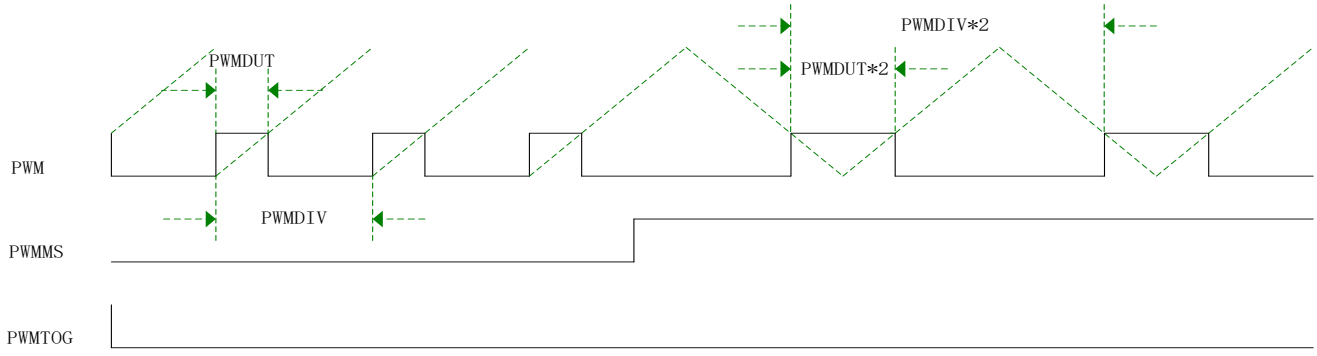


图 18-2-1 PWMTOG＝0 时 PWM 输出波形

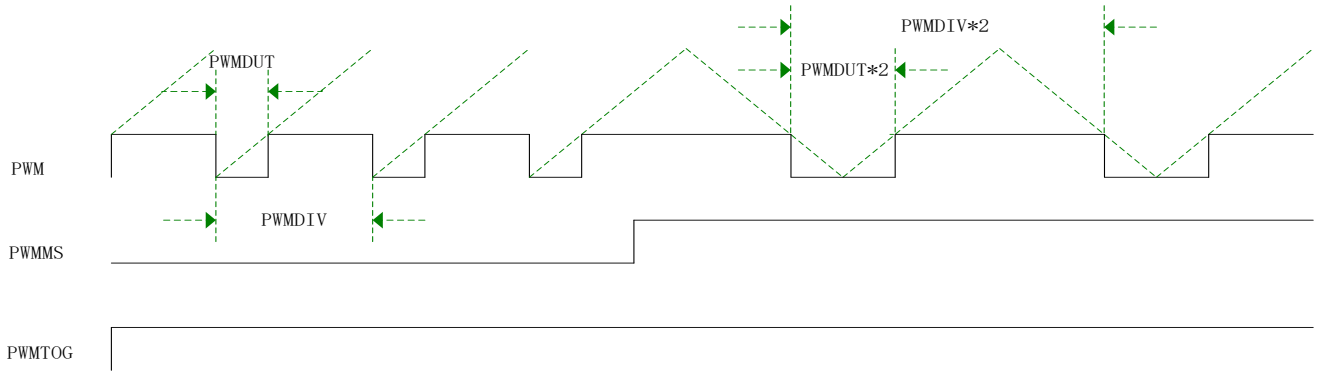


图 18-2-2 PWMTOG＝1 时 PWM 输出波形

值得注意的是，当 PWMDIV=0 时，PWM 引脚直接输出 PWM 时钟，如果 PWMCKD=0，PWM 引脚输出的是所选的时钟源的时钟信号；当 PWMDIV 不为 0，而 PWMDUT=0 时，PWM 引脚输出低电平（PWMTOG＝0）；当PWMDUT>=PWMDIV>0 时，PWM 引脚输出高电平（PWMTOG＝0）。

- **互补模式**

在互补模式下，6 路 PWM 可组成 3 对互补通道：PWM0 和 PWM1、PWM2 和 PWM3、PWM4 和 PWM5。PWM 的互补模式是通过 PWM1、PWM3、PWM5 的控制寄存器 PWMCON 的 PWMMOD 位设置的。在互补模式，PWM 对齐方式、周期、占空比、预分频时钟都是由 PWM0、PWM2、PWM4 对应的寄存器设置，只有 PWMTOG 仍是由各通道对应寄存器独立控制。PWM 互补模式原理图如图 18-2-3 所示。

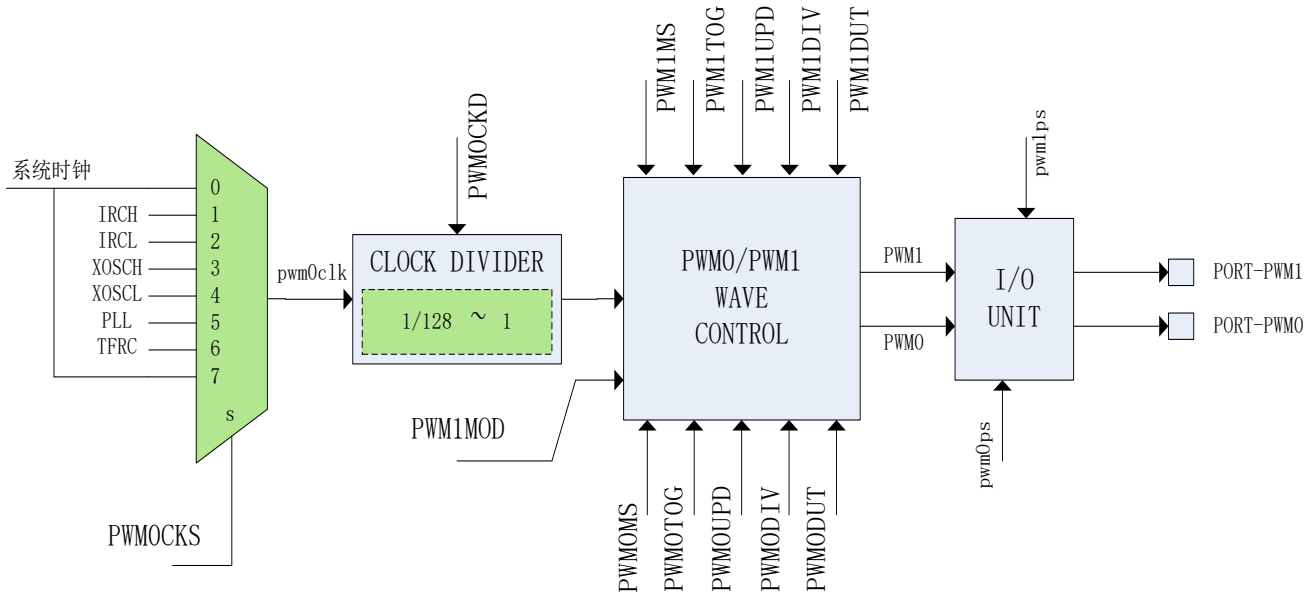


图 18-2-3 PWM0、PWM1 原理示意图

每组 PWM 输出的波形在相位上互补，如图 18-2-4 和图 18-2-5 所示（以 PWM0、PWM1 为例）。

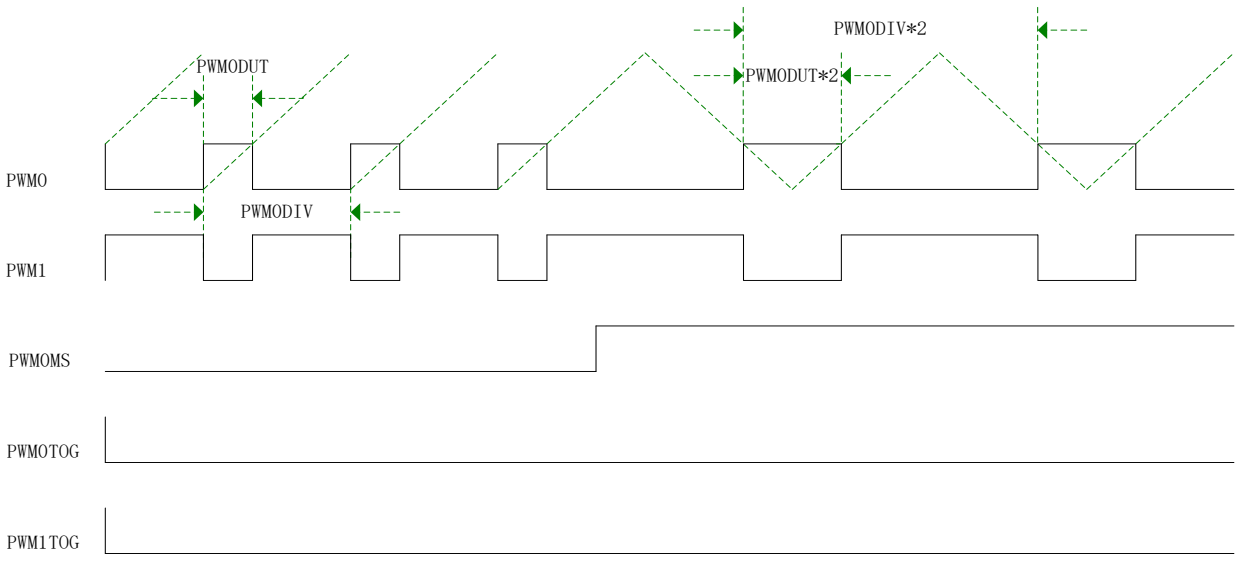


图 18-2-4 PWMTOG＝0 时 PWM0、PWM1 输出互补波形

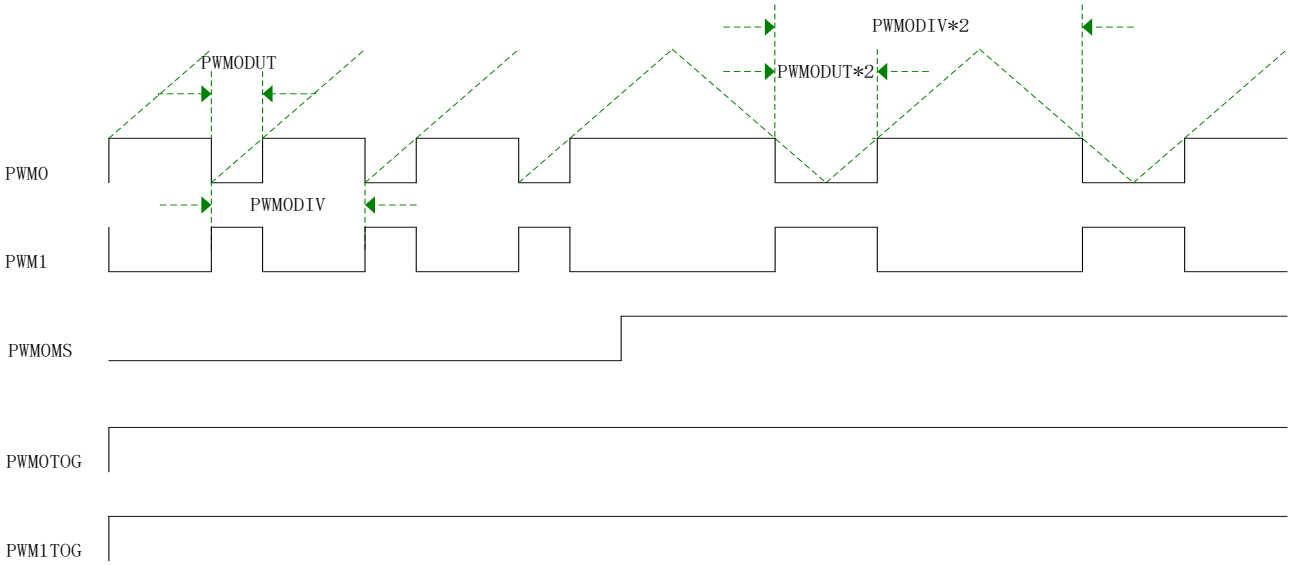


图 18-2-5 PWMTOG＝1 时 PWM0、PWM1 输出互补波形

- **死区控制**

在桥形驱动电路中，为防止上下半桥同时导通，需要在 PWM 互补信号中插入死区控制。死区时间由 PWM1、PWM3、PWM5 对应的寄存器 PWMDIV 和 PWMDUT 设置，PWMDIV 设置的是左边的死区时间，而 PWMDUT 设置的是右边的死区时间。设置死区时间需要满足以下条件（以PWM0、PWM1 为例）：

在边沿对齐模式，PWMDIV1<PWMDUT0 且 PWMDUT1<(PWMDIV0 – PWMDUT0);

在中心对齐模式，PWMDIV1<(PWMDIV0 – PWMDUT0) x 2 或 PWMDUT1<(PWMDIV0 – PWMDUT0) x 2。

死区控制输出波形如图 18-2-6 所示（以PWM0、PWM1 为例）。

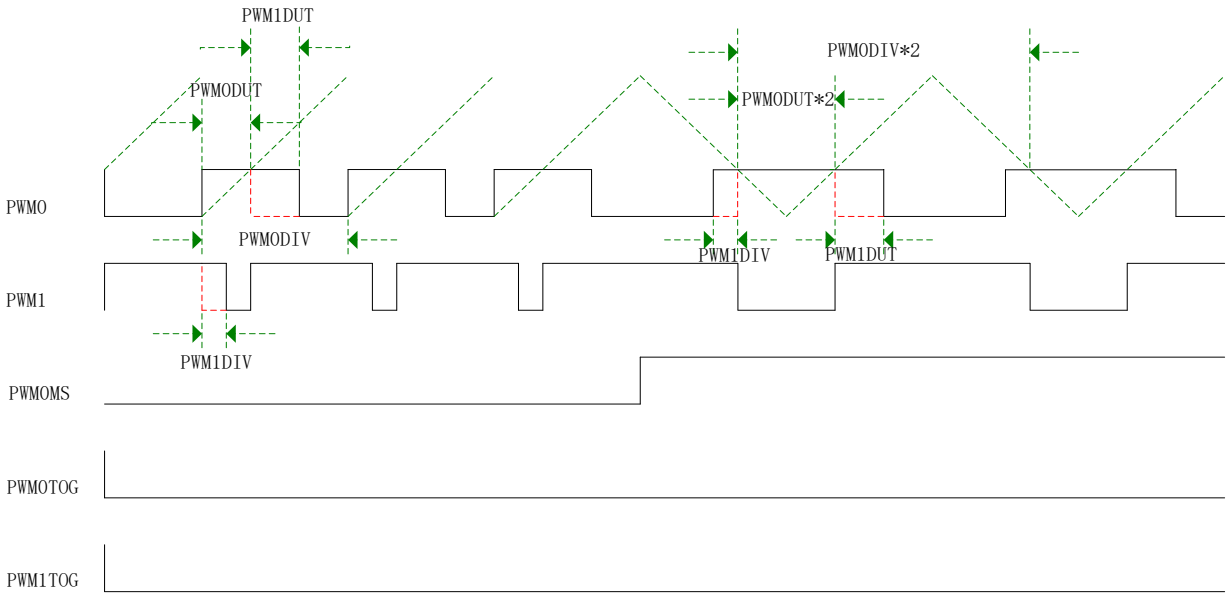


图 18-2-6 PWMTOG＝0 时 PWM0、PWM1 死区控制波形

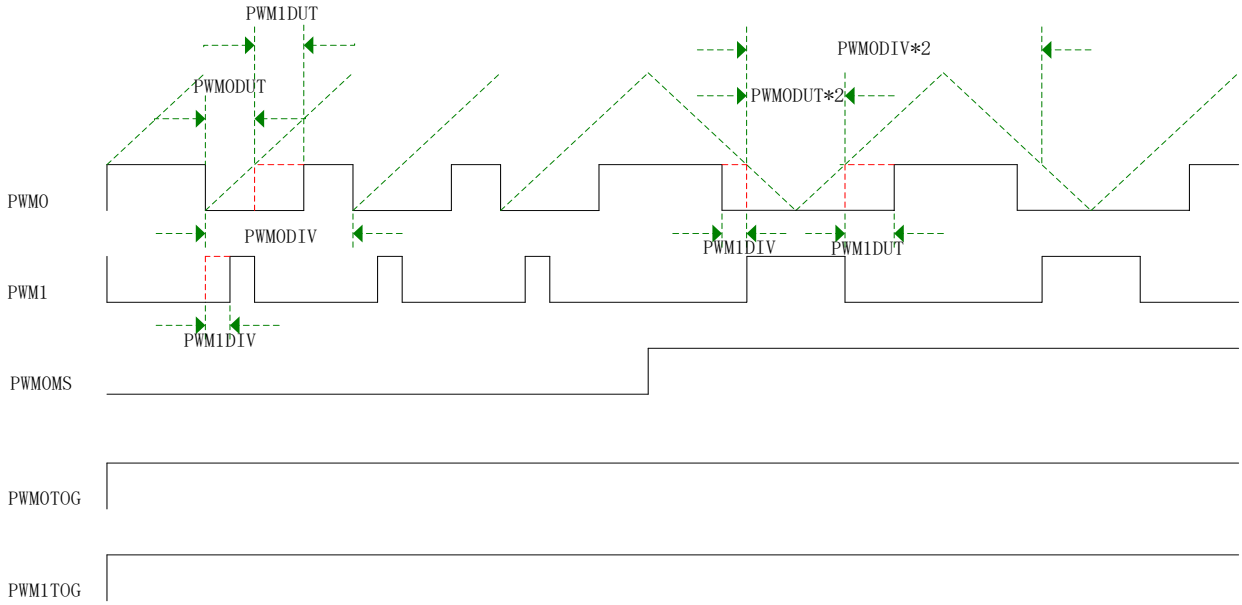


图 18-1-6 PWMTOG＝1 时 PWM0、PWM1 死区控制波形

- **PWM 中断**

PWM 中断通过寄存器 PWMCON 的 PWMTIE、PWMZIE、PWMPIE、PWMNIE 位使能，PWMTIE 位对应的是 PWM 计数器计数到顶点（即等于 PWMDIV)产生的中断，PWMZIE 位对应的是 PWM 计数器计数到最低点（即等于 0）时产生的中断，PWMNIE 对应的是输出引脚下降沿产生的中断，PWMPIE 对应的是输出引脚上升沿产生的中断。其中，在边沿对齐模式，没有 PWMTIE 位和 PWMZIE 位对应的中断。寄存器 PWMAIF、PWMBIF、PWMCIF、PWMDIF 是 6 个通道的中断状态寄存器，其中，PWMxTIF、PWMxZIF、PWMxNIF、PWMxPIF 分别对应使能位PWMTIE、PWMZIE、PWMNIE、PWMPIE。

另外，PWM 可通过寄存器 PWMCMX 设置多少次中断事件发生才产生一次中断，例如设置 PWMCMX=3、PWMPIE=1，那么 PWM 引脚 4 次上升沿才会产生一次上升沿中断。

- **PWM 刹车功能**

每路 PWM 都可独立设置刹车功能，刹车功能包括软件刹车和硬件刹车。软件刹车通过软件写寄存器 PWMSBC 产生，PWMSBE0~PWMSBE5 分别对应 PWM0~5，当 PWMSBE0~PWMSBE5 写 1 时，产生刹车事件， 刹车后 PWM 引脚的输出电平由寄存器 PWMBD 设置， PWMBD0~5 分别对应 PWM0~5，当 PWMBDx(x=0~5)为 0，刹车后相应的 PWM 引脚输出低电平，反之输出高电平。硬件刹车由 FB 引脚触发，通过设置 PWMFBE=1 使能，当 PWMFBL=0，FB 引脚高电平触发刹车，当设置 PWMFBL=1，FB 引脚低电平触发刹车。硬件刹车中断通过设置 PWMFBIE=1 使能，PWMFBIF 为硬件刹车中断标志。每路 PWM 可通过 PWMFBSx(x=0~5 分别对应 PWM0~5)选择是否由 FB 电平触发刹车，刹车后的状态和软件刹车相同，也是由 PWMBD 设置。

- **PWM 暂停功能**

PWM 暂停功能以两路为一组进行设置，分别为：PWM0/1、PWM2/3、、PWM4/5. 寄存器 PWMHS 为 PWM 暂停功能控制寄存器，PWMHS0 控制 PWM0/1，PWMHS2 控制 PWM2/3，PWMHS4 控制 PWM4/5。当设置 PWMHSx（x=0/2/4）为 1 时，对应的两路 PWM 输出保持在原来的状态，当 PWMHSx 为 0 时，对应 PWM 恢复为正常输出。

- **PWM 中断触发 ADC 功能**

可通过寄存器 ADPWMTRIG 使能 PWM 中断触发 ADC 转换功能，ADTRIG0~5 对应 PWM0~5。当

ADTRIGx(x=0~5)=1，对应的 PWM 中断产生时，ADC 转换启动。注意：ADC 的初始化和 PWM 中断初始化需要另外设置，ADTRIGx 只是作为使能开关，PWM 中断触发ADC 相当于使 AST 位(ADCON[7])置 1.

- **跳频功能**

■ PWM0 可以通过设置 PWM0NUM 定时调节 PFG 时钟的频率，每次调整 PFG 频率的步进由时钟控制器中的 STEP 决定，向上/向下调整的最高步进数由 STEPNUM 决定。（详细请参看“PFG 时钟跳频功能”介绍）

18.3 PWM(6~7)功能描述

每路 PWM 通道都有一个专门的 16 位计数器，PWM 的周期通过寄存器 PWMDIV 来设置，而寄存器 PWMDUT 则对应 PWM 的占空比。PWM 通过寄存器 PWMEN 使能，寄存器 PWMEN 的每一位对应 PWM 的一个通道。PWM 可通过 PWMTOG 位设置 PWM 引脚输出反相。PWM 有多种时钟源可以选择，每路时钟源都是单独进行设置的，对应的控制寄存器为 PWMCON 的 PWMCKS。另外，每路 PWM 的时钟分频可通过 PWMCKD 独立设置。

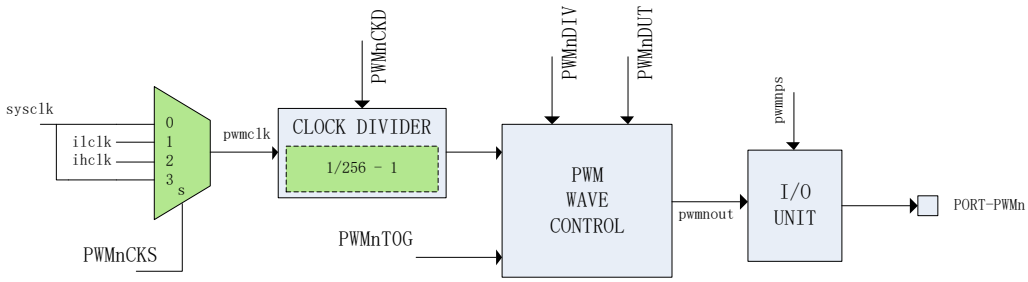


图 18-3-1 PWM 原理示意图

- **PWM 输出波形**

PWM 使能后，PWM 计数器开始累加计数，当计数值不大于 PWMDUT 时，PWM 引脚输出高电平（PWMTOG=0 ），当计数值大于 PWMDUT 时， PWM 引脚输出低电平（PWMTOG=0 ）。当计数值与 PWMDIV 相等时，一个PWM 周期完成，PWM 计数器重置并开始下一周期计数，此时将产生PWM 中断。

当 PWM 波形满足条件PWMDIV>PWMDUT>0 时，PWM 波形如图所示。

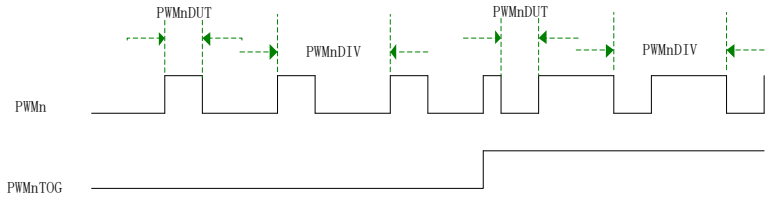


图 18-3-2 PWM 输出波形

值得注意的是，当 PWMDIV=0 时，PWM 引脚直接输出 PWM 时钟，如果 PWMCKD=0，PWM 引脚输出的是所选的时钟源的时钟信号；如果 PWMCKD != 0，PWM 引脚输出的是所选的时钟源的 1/（PWMCKD+1）频率的时钟信号；当 PWMDIV 不为 0，而 PWMDUT=0 时，PWM 引脚输出低/高电平（PWMTOG＝0/1）；当 PWMDUT>=PWMDIV>0 时，PWM 引脚输出高/低电平（PWMTOG＝0/1）。

- **单线级联LED 驱动**

PWM6/PWM7 通道支持单线级联LED 驱动，级联LED 的典型驱动时序图如图 18-3-4 所示。

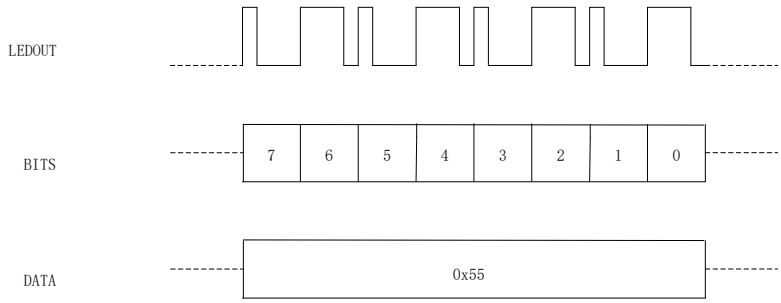


图 18-3-4 级联 LED 时序图

位码示意图如图 18-3-5 所示。

图 18-3-5 位码示意图

在级联 LED 时序图中，位码 0 的高电平时间宽度由 PWMDUT 配置，位码 1 的高电平时间宽度由 LEDUTH 配置，而位周期时间由 PWMDIV 配置。当 PWMMOD 不为 0 时，级联 LED 驱动使能，LEDAT0/LEDAT1 分别为 LEDn（n=0/1）的数据寄存器，当 LEFn(n=0/1)为 0 时，可以向 LEDAT0/LEDAT1 写入 LED 数据。写入 LEDAT0/LEDAT1 即启动 LED 驱动数据发送，当 LEDn(n=0/1)发送器正处于发送状态时，LEBSYn(n=0/1)置 1，当发送器处于空闲状态时，LEBSYn 变为 0。LED 发送器有一字节的发送缓存，当数据寄存器和缓存寄存器都有数据时，LEFn(n=0)位置 1，当缓存寄存器的数据发送完后，会自动从数据寄存器中加载，同时 LEFn(n=0)位置 0，LEFn(n=0)=0 表示可以重新向 LEDAT0/LEDAT1 装载数据。当 PWMMOD 不为 0 时，PWMMOD 也同时表示发送完PWMMOD 个字节后插入等待时间，等待时间由LEWTM 来配置。

当 PWMPOL=1 时， LEDAT0/LEDAT1 的数据反相，即：例如写入 01010101B，实际发送出来的是 10101010B。

18.4 PWM 寄存器描述

表 18-4-1 寄存器 PWMEN

DAH	7	6	5	4	3	2	1	0
PWMEN	PWMEN[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
7~0	PWMEN	7~0 位分别对应 PWM 通道 7~0 的使能控制位，1 有效						

表 18-4-2 寄存器 PWMUPD

DBH	7	6	5	4	3	2	1	0
PWMUPD			PWMUPD[5:0]					
R/W			R/W					
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
7~0	PWMUPD	5~0 位对应 PWM 通道 5~0 的数据更新使能控制位，1 有效 *备注：* *配置某个通道的数据（PWMDIV/PWMDUT/PWMCKD）之后，把 PWMUPD 对应通道的位置 1，这样才会使数据在 PWM 计数器溢出之后更新进去，而对应位在数据更新完成之后将会自动清 0。*						

表 18-4-3 寄存器 PWMCMAX

DCH	7	6	5	4	3	2	1	0
PWMCMAX	PWMCMAX[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
备注：PWMCMX 是带索引的寄存器，设置 INDEX=0~5 分别对应 PWMCMAX0~PWMCMAX5								
位编号	位符号	说明						
7~0	PWMCMAX	PWM 各通道中断有效的间隔次数设置寄存器。 间隔次数=PWMCMAX+1，例如设置 INDEX=0、PWMCMAX=7，那么 PWM0 所有中断都会产生 8 次中断事件时才会置位中断标志。						

表 18-4-4 寄存器 PWM0NUM

D8H	7	6	5	4	3	2	1	0
PWM0NUML	PWM0NUM [7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
D9H	7	6	5	4	3	2	1	0
PWM0NUMH	PWM0NUM[15:8]							
R/W	R/W							
初始值	0	0	0	1	0	0	0	0
位编号	位符号	说明						
15~0	PWM0NUM	PWM0 频率发生变化的 PWM 周期数（PWM0NUM+1 个 PWM 周期频率变化一次），PWM0NUM 最小值为 1，设置为 0 时 PFG 时钟频率为基准频率，不会进行频率调整。						

表 18-4-5 寄存器 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		
初始值	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
初始值	0	0	0	0	0	-	-	0
PWMCON③	-	PWMPOL	PWMMOD[5:3]			PWMCKS[2:0]		
R/W	-	R/W	R/W			R/W		
初始值	-	0	0	0	0	0	0	0

位编号	位符号	说明
备注： *1 PWMCON①是 PWM0/PWM2/PWM4 通道的控制寄存器；* *PWMCON②是 PWM1/PWM3/PWM5 通道的控制寄存器。* *PWMCON③是 PWM6/PWM7 通道的控制寄存器。* *2 PWMCON 是带索引的寄存器，设置 INDEX=0~7 分别对应 PWMCON0~7*		
7	PWMTIE	PWM 计数器顶点中断使能控制位，1 有效
6	PWMZIE	PWM 计数器最低点中断使能控制位，1 有效
5	PWMPIE	PWM 上升沿中断使能控制位，1 有效
4	PWMNIE	PWM 下降沿中断使能控制位，1 有效
3	PWMMS	PWM 模式选择位 0：边沿对齐模式 1：中心对齐模式
2~0	PWMCKS	PWM 工作时钟选择位 001：IRCH 010：IRCL 011：XOSCH/CLK_IN 100：PFG 其他：系统时钟 *备注：* *PWM0/PWM1，都由 PWMCKS0 来配置；* *PWM2/PWM3，都由 PWMCKS2 来配置；* *PWM4/PWM5，都由 PWMCKS4 来配置。*
0	PWMMOD	互补模式使能寄存器，1 有效 *备注：* *设置 PWMMOD1=1，PWM0、PWM1 进入互补模式；* *设置 PWMMOD3=1，PWM2、PWM3 进入互补模式；* *设置 PWMMOD5=1，PWM4、PWM5 进入互补模式。*
位编号	位符号	说明
备注：PWM6/PWM7 可驱动级联 LED，所以增加与 LED 驱动有关的控制位。		
7	-	-
6	PWMPOL	PWM 作为 LED 驱动时，发送数据取反使能控制位，1 有效 ***备注：*** *1. 当 PWMMOD！=0 时，对应的 PWMPOL 的值才有意义；* *2. 当 PWMPOL=1 时，如果对应的 LEDAT=01010101B，那么实际上发送的将会是 10101010B。*
5~3	PWMMOD	PWM 作为 LED 驱动时，连续发送字节数配置寄存器，0 表示 PWM 不作为 LED 驱动使用，1~7 表示 PWM 每发送 1~7 字节数据就暂停 1 次 ***备注：*** *详细使用参考 LEWTM*
2~0	PWMCKS	PWM 工作时钟选择位 001：IRCH 010：IRCL 011：XOSCH/CLK_IN

		100：PFG 其他：系统时钟

表 18-4-6 寄存器 PWMCFG

DEH	7	6	5	4	3	2	1	0
PWMCFG	PWMTOG	PWMCKD[6:0]						
R/W	R/W	R/W						
初始值	0	0	0	0	0	0	0	0
备注：PWMCFG 是带索引的寄存器，设置 INDEX=0~7 分别对应 PWMCFG0~PWMCFG7								
位编号	位符号	说明						
7	PWMTOG	PWM 输出取反使能寄存器，1 有效						
6~0	PWMCKD	PWM 工作时钟预分频配置寄存器 0000000：不分频 0000001：2 分频 0000010：3 分频 …… 1111110：127 分频 1111111：128 分频						

表 18-4-7 寄存器 PWMDIVL、PWMDIVH

DFH	7	6	5	4	3	2	1	0
PWMDIVL	PWMDIV[7:0]							
R/W	R/W							
初始值	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							
初始值	0	0	0	0	0	0	0	0
备注：PWMDIV 是带索引的寄存器，设置 INDEX=0~7 分别对应 PWMDIV0~PWMDIV7								
位编号	位符号	说明						
15~0	PWMDIV	PWM 周期配置寄存器 在互补模式下，PWMDIV1/PWMDIV3/PWMDIV5/PWMDIV7 有不同的含义，参考寄存器 PWMDUT 相关描述						

表 18-4-8 寄存器 PWMDUTL、PWMDUTH

D2H	7	6	5	4	3	2	1	0
PWMDUTL	PWMDUT[7:0]							
R/W	R/W							
初始值	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							
初始值	0	0	0	0	0	0	0	0
备注：PWMDUT 是带索引的寄存器，设置 INDEX=0~7 分别对应 PWMDUT0~PWMDUT7								
位编号	位符号	说明						

<table>
<tr><td>15~0</td><td>PWMDUT</td><td>PWM 占空比配置寄存器
在互补模式下，PWMDUT1/PWMDUT3/PWMDUT5 有不同的含义，如下表：

<table><tr><td>PWMDIV1</td><td>控制 PWM0/PWM1 的左边的死区的宽度</td></tr><tr><td>PWMDUT1</td><td>控制 PWM0/PWM1 的右边的死区的宽度</td></tr><tr><td>PWMDIV3</td><td>控制 PWM2/PWM3 的左边的死区的宽度</td></tr><tr><td>PWMDUT3</td><td>控制 PWM2/PWM3 的右边的死区的宽度</td></tr><tr><td>PWMDIV5</td><td>控制 PWM4/PWM5 的左边的死区的宽度</td></tr><tr><td>PWMDUT5</td><td>控制 PWM4/PWM5 的右边的死区的宽度</td></tr></table></td></tr>
</table>

表 18-4-9 寄存器 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
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7	PWM1TIF	PWM1 计数器顶点中断标志位，写 1 清 0
6	PWM1ZIF	PWM1 计数器最低点中断标志位，写 1 清 0
5	PWM1PIF	PWM1 上升沿中断标志位，写 1 清 0
4	PWM1NIF	PWM1 下降沿中断标志位，写 1 清 0
3	PWM0TIF	PWM0 计数器顶点中断标志位，写 1 清 0
2	PWM0ZIF	PWM0 计数器最低点中断标志位，写 1 清 0
1	PWM0PIF	PWM0 上升沿中断标志位，写 1 清 0
0	PWM0NIF	PWM0 下降沿中断标志位，写 1 清 0

表 18-4-10 寄存器 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
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7	PWM3TIF	PWM3 计数器顶点中断标志位，写 1 清 0
6	PWM3ZIF	PWM3 计数器最低点中断标志位，写 1 清 0
5	PWM3PIF	PWM3 上升沿中断标志位，写 1 清 0
4	PWM3NIF	PWM3 下降沿中断标志位，写 1 清 0
3	PWM2TIF	PWM2 计数器顶点中断标志位，写 1 清 0
2	PWM2ZIF	PWM2 计数器最低点中断标志位，写 1 清 0
1	PWM2PIF	PWM2 上升沿中断标志位，写 1 清 0
0	PWM2NIF	PWM2 下降沿中断标志位，写 1 清 0

表 18-4-11 寄存器 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
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7	PWM5TIF	PWM5 计数器顶点中断标志位，写 1 清 0
6	PWM5ZIF	PWM5 计数器最低点中断标志位，写 1 清 0
5	PWM5PIF	PWM5 上升沿中断标志位，写 1 清 0
4	PWM5NIF	PWM5 下降沿中断标志位，写 1 清 0
3	PWM4TIF	PWM4 计数器顶点中断标志位，写 1 清 0
2	PWM4ZIF	PWM4 计数器最低点中断标志位，写 1 清 0
1	PWM4PIF	PWM4 上升沿中断标志位，写 1 清 0
0	PWM4NIF	PWM4 下降沿中断标志位，写 1 清 0

表 18-4-12 寄存器 PWMPS

8098H	7	6	5	4	3	2	1	0
PWMPS	-	-	-	PWMPS[4:0]				
R/W	-	-	-	R/W				
初始值	-	-	-	?	?	?	?	?

备注：

1. PWMPS 为索引寄存器，由 INDEX0~5 分别指向 PWMPS0~5，控制 PWM0~5 通道。

2. PWMPS0[4:0]初值为 01010，选择 P1.2；
PWMPS1[4:0]初值为 01001，选择 P1.1；
PWMPS2[4:0]初值为 01000，选择 P1.0；
PWMPS3[4:0]初值为 00000，选择 P0.0；
PWMPS4[4:0]初值为 00001，选择 P0.1；
PWMPS5[4:0]初值为 00011，选择 P0.3。

3. PWM0~5 同时打开时，PWMPS0~5 必须分别选择不同引脚，否则将导致出错; 如果两路以上的 PWM 使能，并且同时选中某个引脚，被选择的引脚将输出 0。

位编号	位符号	说明
7~5	-	-
4~0	PWMPS	PWM 引脚选择位域 00000：选择 P0.0 00001：选择 P0.1 00010：选择 P0.2 00011：选择 P0.3 00100：选择 P0.4 00101：选择 P0.5 00110：选择 P0.6 00111：选择 P0.7 01000：选择 P1.0 01001：选择 P1.1 01010：选择 P1.2 01011：选择 P1.3 01100：选择 P1.4 01101：选择 P1.5 01110：选择 P1.6 01111：选择 P1.7 10000：选择 P2.0 其他：选择 P3.0

表 18-4-13 寄存器 PWMHS

8099H	7	6	5	4	3	2	1	0
PWMHS	-	-	-	PWMHS4	-	PWMHS2	-	PWMHS0
R/W	-	-	-	R/W	-	R/W	-	R/W
初始值	-	-	-	0	-	0	-	0

位编号	位符号	说明
7~5	-	-
4	PWMHS4	PWM4/PWM5 保持使能位，1 有效
3	-	-
2	PWMHS2	PWM2/PWM3 保持使能位，1 有效
1	-	-
0	PWMHS0	PWM0/PWM1 保持使能位，1 有效

表 18-4-14 寄存器 PWMFBC

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

位编号	位符号	说明
7	PWMFBIF	PWM Fault Pin 刹车中断标志，1 有效，写 1 清 0
6	-	-
5	PWMFBIE	PWM Fault Pin 刹车中断使能，1 有效
4~2	-	-
5	PWMFBL	PWM Fault Pin 刹车电平选择 0：选择高电平刹车 1：选择低电平刹车
0	PWMFBE	PWM Fault Pin 刹车使能，1 有效

表 18-4-15 寄存器 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
初始值	-	-	0	0	0	0	0	0

位编号	位符号	说明
7~6	-	-
5	PWMFBS5	PWM5 Fault Pin 刹车选择位 0：PWM5 不受 Fault Pin 信号影响 1：Fault Pin 信号有效时，PWM5 将处于刹车状态
4	PWMFBS4	PWM4 Fault Pin 刹车选择位 0：PWM4 不受 Fault Pin 信号影响 1：Fault Pin 信号有效时，PWM4 将处于刹车状态
3	PWMFBS3	PWM3 Fault Pin 刹车选择位 0：PWM3 不受 Fault Pin 信号影响 1：Fault Pin 信号有效时，PWM3 将处于刹车状态
2	PWMFBS2	PWM2 Fault Pin 刹车选择位 0：PWM2 不受 Fault Pin 信号影响 1：Fault Pin 信号有效时，PWM2 将处于刹车状态
1	PWMFBS1	PWM1 Fault Pin 刹车选择位 0：PWM1 不受 Fault Pin 信号影响 1：Fault Pin 信号有效时，PWM1 将处于刹车状态
0	PWMFBS0	PWM0 Fault Pin 刹车选择位 0：PWM0 不受 Fault Pin 信号影响 1：Fault Pin 信号有效时，PWM0 将处于刹车状态

表 18-4-16 寄存器 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
初始值	-	-	0	0	0	0	0	0

位编号	位符号	说明
7~6	-	-
5	PWMSBE5	PWM5 软件刹车使能，1 刹车
4	PWMSBE4	PWM4 软件刹车使能，1 刹车
3	PWMSBE3	PWM3 软件刹车使能，1 刹车
2	PWMSBE2	PWM2 软件刹车使能，1 刹车
1	PWMSBE1	PWM1 软件刹车使能，1 刹车
0	PWMSBE0	PWM0 软件刹车使能，1 刹车

表 18-4-17 寄存器 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
初始值	-	-	1	0	1	0	1	0

备注：
软件和硬件刹车，电平都受本寄存器控制。

位编号	位符号	说明
7~6	-	-
5	PWMBD5	PWM5 刹车电平选择 0：刹车时保持低电平 1：刹车时保持高电平
4	PWMBD4	PWM4 刹车电平选择 0：刹车时保持低电平 1：刹车时保持高电平
3	PWMBD3	PWM3 刹车电平选择 0：刹车时保持低电平 1：刹车时保持高电平
2	PWMBD2	PWM2 刹车电平选择 0：刹车时保持低电平 1：刹车时保持高电平
1	PWMBD1	PWM1 刹车电平选择 0：刹车时保持低电平 1：刹车时保持高电平
0	PWMFBD0	PWM0 刹车电平选择 0：刹车时保持低电平 1：刹车时保持高电平

表 18-4-18 寄存器 ADPWMTRIG

808EH	7	6	5	4	3	2	1	0
ADPWMTRIG	-	-	ADTRIG5	ADTRIG4	ADTRIG3	ADTRIG2	ADTRIG1	ADTRIG0
R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W
初始值	-	-	0	0	0	0	0	0

位编号	位符号	说明
7~6	-	-
5	ADTRIG5	PWM5 的中断触发 ADC 使能，1 有效

4	ADTRIG4	PWM4 的中断触发 ADC 使能，1 有效
3	ADTRIG3	PWM3 的中断触发 ADC 使能，1 有效
2	ADTRIG2	PWM2 的中断触发 ADC 使能，1 有效
1	ADTRIG1	PWM1 的中断触发 ADC 使能，1 有效
0	ADTRIG0	PWM0 的中断触发 ADC 使能，1 有效

表 18-4-19 寄存器 LEDAT

E1H	7	6	5	4	3	2	1	0
LEDAT0	LEDAT0[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
E2H	7	6	5	4	3	2	1	0
LEDAT1	LEDAT1[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~0	LEDATx	LED 驱动数据 ***备注：*** *1. x 表示 0，LED0 对应 PWM1；* *2. LEDAT 的数据按照从 MSB 到LSB 的顺序发送。*

表 18-4-20 寄存器 LEDUTL、LEDUTH

E3H	7	6	5	4	3	2	1	0
LEDUTL	LEDUT[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
E4H	7	6	5	4	3	2	1	0
LEDUTH	LEDUT[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

备注：LEDUT 是带索引的寄存器，设置 INDEX=6~7 分别对应 LEDUT6~LEDUT7

位编号	位符号	说明
15~0	LEDUT	LED 发送数据“1”占空比配置寄存器 ***备注：*** *1. 级联 LED 的驱动波形中，每 1 位数据的周期都由对应的 PWM1DIV 决定，而数据“1”的占空比由 LEDUT 决定，数据“0”的占空比由 PWMmDUT 决定；* *2.如果 LEDAT=01010101B，同时对应的 PWMPOL=1，那么实际发送的数据按照 BIT7-BIT6-BIT5-BIT4-BIT3-BIT2-BIT1-BIT0 顺序就是 1-0-1-0-1-0-1-0，而且 BIT7/BIT5/BIT3/BIT1 的占空比由 LEDUT 决定， BIT6/BIT4/BIT2/BIT0 的占空比由对应的 PWMDUT 决定，即 LEDUT 的起效在 PWMPOL 之后；* *3. LED1 的数据“1”的占空比都由同一个 LEDUT 决定。*

表 18-4-21 寄存器 LEWTML、LEWTMH

C6H	7	6	5	4	3	2	1	0
LEWTML	LEWTM[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
C7H	7	6	5	4	3	2	1	0
LEWTMH	LEWTM[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
备注：LEWTM 是带索引的寄存器，设置 INDEX=6~7 分别对应 LEWTM6~LEWTM7								
位编号	位符号	说明						
15~0	LEWTM	LED 暂停时间配置寄存器，结合 PWMMOD 配置寄存器使用 ***备注：*** *1. 每发送 PWMMOD 字节数据之后，暂停（LEWTM+1）个 PWM 的工作时钟后进入下一次传输。* *2. LED 的暂停时间都由同一个 LEWTM 决定。*							

表 18-4-22 寄存器 LEFLG

E5H	7	6	5	4	3	2	1	0
LEFLG	LEF1	LEBSY1	-	-	LEF0	LEBSY0	-	-
R/W	R	R	-	-	R	R	-	-
初始值	0	0	-	-	0	0	-	-
位编号	位符号	说明						
7	LEF1	LEDAT1 数据缓存满标志，1 表示 LEDAT1 的数据缓存处于满状态（此时如果对LEDAT1 写数据是无效的），0 表示可以向 LEDAT1 写入新的数据 ***备注：*** *LEDAT1 有 1 字节的数据缓存，所以在 LEDAT1 数据存储空间没有写数据之前，可以连续写入 2 字节数据，之后再想向 LEDAT1 内写数据，都必须在检测到 LEF1 为0 才可以，因为 LEF1 为0 表示 LEDAT1 数据缓存中至少有 1 字节已经发送完成。*							
6	LEBSY01	LEDAT1 数据发送忙标志，1 表示此时 LEDAT1 的数据缓存中的数据还没有全部发送完成，0 表示全部发送完成						
5~4	-	-						
3	LEF0	LEDAT0 数据缓存满标志，1 表示 LEDAT0 的数据缓存处于满状态（此时如果对LEDAT0 写数据是无效的），0 表示可以向 LEDAT0 写入新的数据 ***备注：*** *LEDAT0 有 1 字节的数据缓存，所以在 LEDAT0 数据存储空间没有写数据之前，可以连续写入 2 字节数据，之后再想向 LEDAT0 内写数据，都必须在检测到 LEF0 为0 才可以，因为 LEF0 为0 表示 LEDAT0 数据缓存中至少有 1 字节已经发送完成。*							
2	LEBSY0	LEDAT0 数据发送忙标志，1 表示此时 LEDAT0 的数据缓存中的数据还没有全部发送完成，0 表示全部发送完成						
1~0	-	-						

18.5 PWM 功能控制例程

PWM 单路输出例程

例如， PWM0 输出频率为 30KHZ、占空比 50%的PWM 信号，并产生PWM 上升沿中断：

```
---------------------------------------------------------------------------------------------------------------------
//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        (24000000/30000) //当 PWM 时钟为其他时钟频率时，需相应修改参数
#define PWMDUT_V        (PWMDIV_V/2) //占空比为 50%
void PWM_init(void)
{
    CKCON |= IHCKE; //打开 IRCH 时钟
    INDEX = PWM_CH0; //设置 INDEX 值对应 PWM0
    PWMCON = TIE(0) | ZIE(0) | PIE(0) | NIE(0) | MS(0) | CKS_IH ; //设置 PWM 时钟源为 IRCH
    PWMCFG = TOG(0) | 0;
    PWMPS = 0; //设置 P00 为 PWM0 输出引脚
    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 更新使能
    while(PWMUPD); //等待更新完成PWMEN
    = (1<<PWM_CH0); //PWM0 使能

    INDEX = PWM_CH0; //设置 INDEX 值对应 PWM0
    PWMCON |=TIE(0) | ZIE(0) | PIE(1) | NIE(0); //使能上升沿中断
    PWMCMAX = 0; //设置每个上升沿都触发
    INT9EN = 1;
}
void INT9_ISR(void) interrupt 14 using 1
{
    if(PWMAIF & PIF0) //PWM 上升沿中断
    {
```

```
            PWMAIF = PIF0; //清除中断标志
        }
}
```

PWM 输出时钟

例如， PWM0 输出IRCH，程序如下：

```
---------------------------------------------------------------------------------------------------------------
void PWM_init(void)
{
    CKCON |= IHCKE; //打开 IRCH 时钟
    INDEX = PWM_CH0; //设置 INDEX 值对应 PWM0
    PWMCON = TIE(0) | ZIE(0) | PIE(0) | NIE(0) | MS(0) | CKS_IH ; //设置 PWM 时钟源为 IRCH
    PWMCFG = TOG(0) | 0;
    PWMPS = 0; //设置 P00 为 PWM0 输出引脚
    P00F = 5;
    PWMDIVH = 0; //PWMDIV、 PWMDUT 都设置为 0 可直接输出时钟
    PWMDIVL = 0;
    PWMDUTH = 0;
    PWMDUTL = 0;
    PWMUPD = (1<<PWM_CH0); //PWMDIV、 PWMDUT 更新使能
    while(PWMUPD); //等待更新完成PWMEN
    = (1<<PWM_CH0); //PWM0 使能
}
---------------------------------------------------------------------------------------------------------------
```

19 模/数转换器（ADC）

19.1 功能简介

模拟/数字转换器是 12 位逐次逼近型（SAR）ADC，最多提供多达 12 个输入通道，支持标准模式和快速模式，可通过软件来配置，快速模式支持 1MHz 的转换速度。ADC 时钟源是系统时钟，可设置时钟预分频。ADC 有多种参考电压源可选，其中选择内部电压为参考电压时可用于检测芯片供电电压。内置比较器模式，可设定比较器阈值，超出阈值可产生中断。ADC 选择内部电压为参考电压时有自动校正功能，避免芯片一致性问题。ADC 和运放结合一起使用，可以把检测信号进行放大或缩小后再进行转换。

19.2 主要特性

- 12 位的分辨率
- ADC 转换速度 1MHz，可用于电机产品
- 最多提供多达 12 个输入通道
- 支持ADC 中断
- 可设置ADC 时钟预分频
- 多种参考电压可选：内部参考电压、VDD、外部参考电压。
- 选择内部参考电压时，支持自动数据校正功能
- 支持可设置的比较器模式
- 输入电压范围：VSS<=VIN<=VDD。

19.3 结构框图

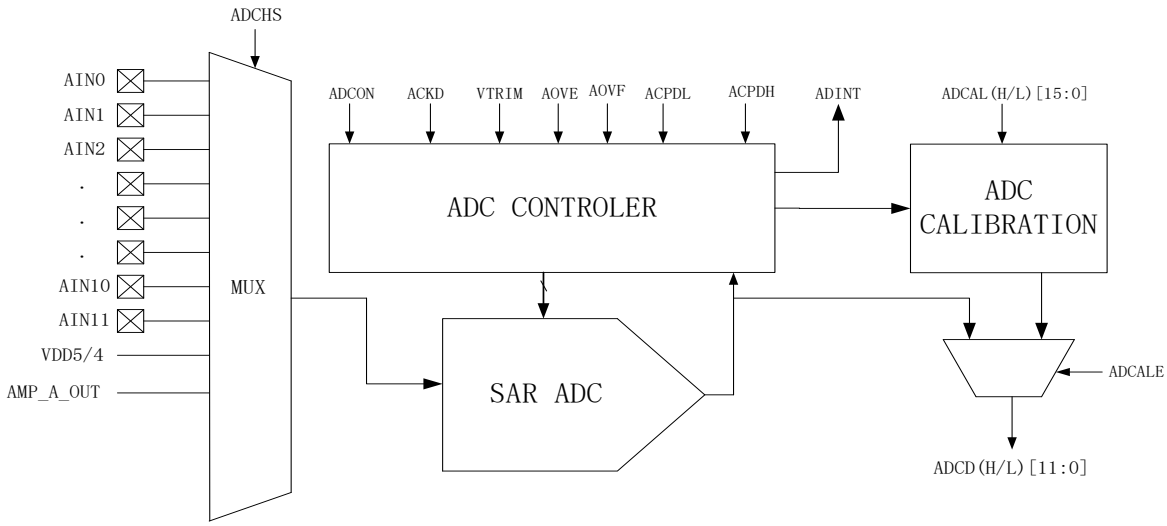


图 19-3-1 ADC 结构示意图

19.4 功能描述

ADC 的启动通过 AST 位使能，设置 AST=1 后，ADC 模块对 ADCHS 选择的输入电压源进行模/数转换。ADC 可通过 ACKD 设置时钟预分频，由系统时钟预分频后的时钟作为 ADC 转换时钟。在 ADC 时钟不变的条件下，ADC 的单次转换时间是由 HTME 设置的，转换时间为(13+2^HTME)个 ADC 时钟周期。当转换结束后，12 位的 A/D 值会被加载到寄存器 ADCDH、ADCDL，转换完后的 2.5 个时钟周期，AST 位自动清 0，同时中断标志ADIF 位会置 1，如果ADC 中断使能，会产生ADC 中断。

图 19-4-1 为 ADC 的转换时序图。

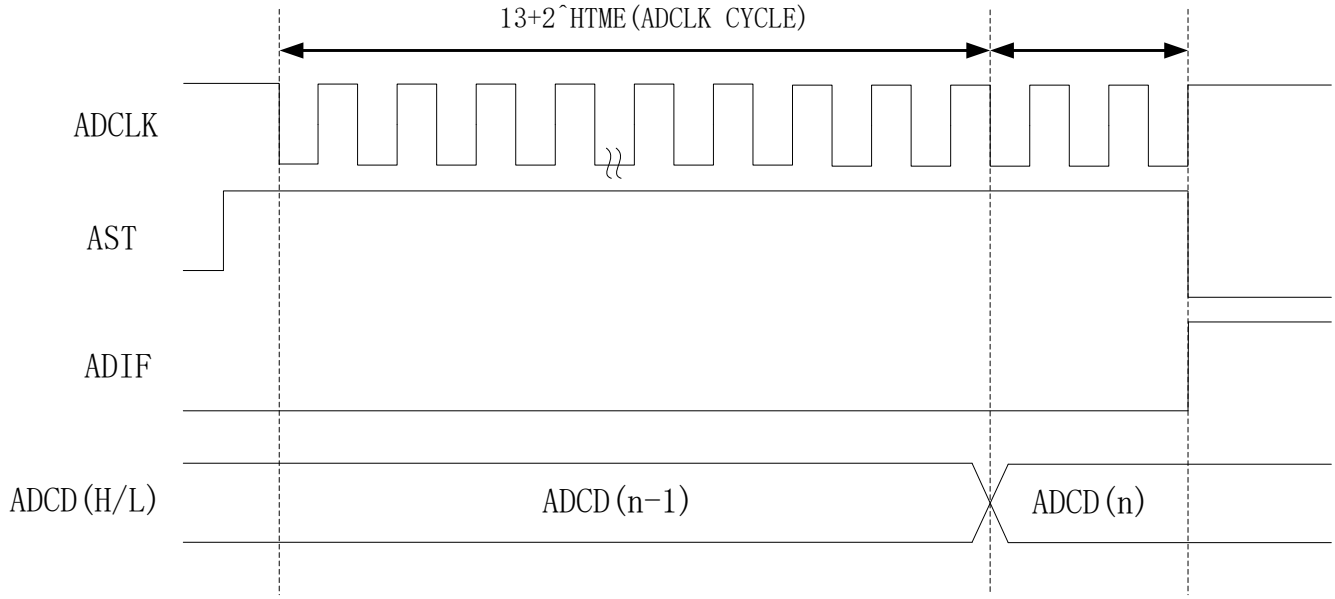


图 19-4-1 ADC 时序示意图

- **比较模式**

比较模式通过 AOVE 位使能， 当 AOVE=1 ，ADC 转换结束后， ADC 转换结果 ADCD 与比较阈值 ADCPDL、ADCPDH 比较，当 ADCD 超出比较阈值范围时，比较中断标志 AOVF 置 1，如果 ADC 中断使能，将产生中断。

- **ADC 数据校正**

当选择内部 1.5V 作为参考电压时，由于芯片的离散性，每个芯片的内部电压不一定完全相同，导致每个芯片的 ADC 转换结果也有一定的偏差，所以在 ADC 转换完后，有必要对 AD 值进行校正。芯片在出厂时，会对每个芯片的内部电压进行测试，得出与内部电压成反比例的校正值，在芯片上电启动时，自动将此校正值加 载到寄存器 ADCALL、ADCALH，当 ADC 转换完成后自动将 AD 值根据校正寄存器 ADCALL、ADCALH 的值进行等比例换算，得出准确的 AD 值，最终的 AD 值也是存放在寄存器 ADCD 中。此功能通过 ADCALE 使能， 对于用户来说，在应用时只需要设置 ADCALE=1 即可，校正过程是自动完成的。

- **ADC 和运放结合使用**

ADC 的检测信号可通过运放A 进行缩小，详见运放部分描述。

19.5 寄存器描述

表 19-5-1 寄存器 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	
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7	AST	ADC 转换开始控制位，写 1 启动转换，转换后硬件自动清 0
6	ADIE	ADC 中断使能位，1 有效
5	ADCIF	ADC 中断标志位，写 1 清 0
4~2	HTME	采样保持周期数为 2 的 HTME 次幂
1~0	VSEL	ADC 参考电压选择位 00: 内部 1.5V(INNER_VREF)作为参考电压 01: 外部 VDD 10: 外部 VREF 11: 内部 1.5V(INNER_VREF)作为参考电压 *备注：当参考电压选择为外部 VREF 时，VREF 的电压必须大于 1.1V。*

表 19-5-2 寄存器 ADCFGL

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

位编号	位符号	说明
7~5	ACKD	ADC 时钟分频设置 000：不分频 001：2 分频 010：4 分频 … 111：14 分频
4	ADCALE	ADC 校准使能位，1 有效 此位只有选择参考电压为内部 1.5V 时才有效，当 ADCALE=1，ADC 的转换结果将根据 ADCAL 寄存器的数值进行校准。具体参考寄存器 ADCAL 说明。
3~0	ADCHS	ADC 通道使能选择位域 0000：通道关闭 0001：通道 AD_CH[0](P12)使能 0010：通道 AD_CH[1](P11)使能 0011：通道 AD_CH[2](P01)使能 0100：通道 AD_CH[3](P03)使能 0101：通道 AD_CH[4](P04)使能 0110：通道 AD_CH[5](P05)使能 0111：通道 AD_CH[6](P06)使能 1000：通道 AD_CH[7](P07)使能 1001：通道 AD_CH[8](P30)使能 1010：通道 AD_CH[9](P17)使能 1011：通道 AD_CH[10](P16)使能 1100：通道 AD_CH[11](P15)使能 1101：检测 VDD 的 1/4 使能 1110：检测 AMP_A_OUT 其他：通道关闭

表 19-5-3 寄存器 ADCFGH

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

位编号	位符号	说明
7	AOVF	比较模式溢出标志位
6	AOVE	比较模式使能位，1 有效
5~0	VTRIM	内部 1.5V 参考电压校正寄存器，校正精度±1mV

表 19-5-4 寄存器 ADCAL

8088H	7	6	5	4	3	2	1	0
ADCALL	ADCAL[7:0]							
R/W	R/W							
初始值	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							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
15~0	ADCAL	ADC 校准寄存器，只有 ADCALE=1 并且选择参考电压为内部 1.5V 才有效。有效时，ADC 的输出按照如下公式： ADCDL=（ADC 转换结果*ADCAL）/32768

表 19-5-5 寄存器 ADCPDL

808AH	7	6	5	4	3	2	1	0
ADCPDLL	ADCPDL[3:0]				-	-	-	-
R/W	R/W				-	-	-	-
初始值	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							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
15~0	ADCPDL	比较模式阈值下限值设定寄存器

表 19-5-6 寄存器 ADCPDH

808CH	7	6	5	4	3	2	1	0
ADCPDHL	ADCPDH[3:0]				-	-	-	-
R/W	R/W				-	-	-	-
初始值	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							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
15~0	ADCPDH	比较模式阈值上限值设定寄存器

表 19-5-7 寄存器 ADCD

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

位编号	位符号	说明
11~0	ADCD	ADC 转换值

表 19-5-8 寄存器 ADOPC

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

位编号	位符号	说明
7~2	-	-
1~0	GAIN	运放缩放倍数选择 00：无缩放 01：缩小 4 倍 10：缩小 3 倍 11：缩小 2 倍

表 19-5-8 寄存器 TPCTL

808FH	7	6	5	4	3	2	1	0
TPCTL	TPCTL [7:0]							
R/W	-	-	-	-	-	-	R/W	
初始值	-	-	-	-	-	-	0	0

位编号	位符号	说明
7~0	TPCTL	当TPCTL 设置为 0x63 时，开启 ADC 高速模式。TPCTL 为其他值时，为低速模式。 *备注：当 ADC 设置为高速模式时， P0.0 不能用。*

19.6 ADC 控制例程

例如，设置ADC 参考电压为外部VDD，采集ADC 通道 1，ADC 中断开启，程序如下：

```
---------------------------------------------------------------------------------------------
//ADCON 定义
#define AST(N) (N<<7)              //ADC 启动， AST=0 结束
#define ADIE(N) (N<<6)             //中断使能
#define ADIF (1<<5)                //中断标志
#define HTME(N) (N<<2)             //N=0-7 //采样时间设置, 时间为 2 的 HTME 次方的时钟周期
#define VSEL(N) (N) //N=0-3        //选择参考电压 0-内部 1-VDD 2-外部
//ADCFGL 定义
#define ACKD(N) (N<<5) //N=0-7  //ADC 时钟分频 分频倍数=（ACKD+1)
#define ADCALE(N) (N<<4)           //ADC 校正， 选择内部参考电压才有效
#define ADCHS(N) (N)               //N=0-15 //ADC 通道选择， 1-13 对应通道 0-12
void ADC_init(void)
{
    P11F = 3; //设置 P11 为 ADC 引脚功能
    ADCON = AST(0) | ADIE(1) | HTME(7) | VSEL(1); //设置 ADC 参考电压为 VDD
    ADCFGL = ACKD(7) | ADCALE(1) | ADCHS(2); //选择 ADC1 通道
    ADCON |= AST(1); //启动 AD 转换
    INT2EN = 1; //INT2 中断使能
}
void ADC_ISR (void) interrupt 7
{
    unsigned int AD_Value;
    if(ADCON & ADIF)
    {
        ADCON |= ADIF; //清中断标志
        AD_Value = ADCDH*256 + ADCDL; //读取 AD 值
        AD_Value >>= 4;
        ADCON |= AST(1); //启动下一次 AD 转换
    }
}
---------------------------------------------------------------------------------------------
```

20 电容式触摸按键（Touch Key）

20.1 功能简介

JZ8FC005T 系列芯片的触摸功能模块具有优越的抗干扰性能，可通过 EFT、CS 等测试。触摸模块最大可支持多达 17 个通道，在应用时 TK_CAP 引脚需接一个 Cx 电容，容值范围 10nF~47nF，电容精度 10%以内，建议使用涤纶电容、X7R 材质电容或 NPO 材质贴片电容。Cx 可直接影响触摸灵敏度，Cx 容值越小，灵敏度越低，容值越大，灵敏度越高。

针对有低功耗需求的应用，还设计了芯片在STOP 模式时仍能正常工作的机制。

20.2 主要特性

- 高抗干扰性能，符合 EMC(CS)标准
- 最大支持 17 个通道
- 支持低功耗模式
- 支持触摸中断
- 支持充放电时钟预分频
- 支持手动和自动启动模式
- 比较器阈值有多级可选
- 触摸可设置内部充电和内部基准，可有效抑制电源低频干扰
- 内置防水补偿机制

20.3 结构图

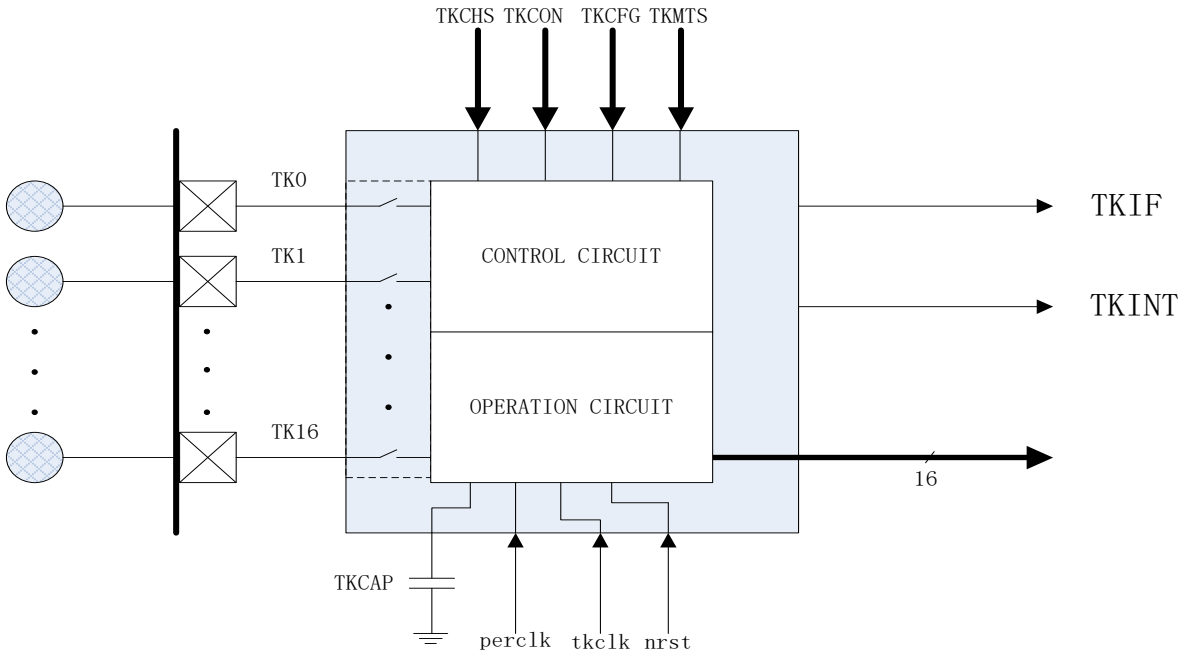


图 20-3-1 触摸模块结构图

20.4 功能描述

20.4.2 手动模式和自动模式

在手动模式下，触摸数据采集通过 TKST 位启动。当设置 TKST=1 后，触摸控制开始采集选定通道的触摸数据。通道的选择是以最多 6 个通道为一组的，通过寄存器 TKnCHS 进行设置，每次启动会一次采集完一组通道。当数据采集完成后，TKST 位自动清 0，相应通道的中断标志位 TKIF 置 1，此时可从寄存器 TKnMS 读取触摸数据。

手动模式和自动模式通过 TMEN 位选择，和手动模式不同的是，自动模式的触摸数据采集是由定时器定时启动的，定时器的时钟源是IRCL，定时时间由寄存器 TKMTS 设置。

备注：

TKnCHS 等寄存器中的“n”表示 0/1/2/3/4/5。

20.4.3 触摸时钟预分频

触摸控制器对触摸电极充放电的时钟源是 IRCH 的 6 分频（即 4MHz），充放电的时钟频率对触摸的性能至关重要，当充放电频率太高时，有可能造成对触摸电极的充电不充分从而导致手指触摸时触摸数据变化量变小。触摸时钟预分频通过TKDIV 进行设置，通过设置合理的值可以使触摸的性能更优。

20.4.4 低功耗模式

为了实现触摸功能的低功耗应用，触摸模块设计了相应的省电机制。在 STOP 模式下，只要触摸的充放电时钟源（IRCH）和低速时钟（IRCL）处于开启状态，触摸模块就可以保持正常的充放电和计数。当触摸采集 完成后，触摸采集完成中断会唤醒 CPU，软件在 CPU 唤醒之后可以读取触摸数据，然后再次进入 STOP 模式。

20.4.5 触摸跳频功能

为提升触摸抗电压脉冲注入性能，芯片设计了触摸跳频功能。

触摸跳频功能由 FAEN 位使能，使能后，触摸的充放电时钟在触摸充放电源时钟(即 IRCH 的四分频或 IRCL)的基础上每次充放电按顺序以 2、3、4、5 分频的顺序变化，可有效降低电压脉冲注入时某频段注入干扰信号对触摸的干扰幅度。

20.5 寄存器描述

表 20-5-1 寄存器TKCON

C0H	7	6	5	4	3	2	1	0
TKCON	TKST	TKIE	TMEN	FAEN	-	VRS[2:0]		
R/W	R/W	R/W	R/W	R/W	-	R/W		
初始值	0	0	0	0	-	0	0	0

位编号	位符号	说明
7	TKST	数据采集启动使能位，1 有效，采集完后自动清 0
6	TKIE	TK 中断使能控制位，1 有效
5	TMEN	启动方式选择位 0:通过 TKST 位控制启动 1:定时器控制启动
4	FAEN	0：不使能频率调节 1：频率调节使能，触摸时钟每周期切换一次，频率有基准频率的 1-2-3-4 分频轮流切换，每周期切换一次；
3	-	–
2~0	VRS	比较器阈值电压基准选择位（阈值电压与 VDD 电压成正比例） 0：阈值电压最高 … 7：阈值电压最低

表 20-5-2 寄存器 TKPWC

C5H	7	6	5	4	3	2	1	0
TKPWC	TKPC		VDS		VIRS		TKPWS	TKCVS
R/W	R/W		R/W		R/W		R/W	R/W
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~6	TKPC	触摸按键未采样通道输出控制

		00：悬浮 01：输出低 10：输出补偿 *备注：这一功能仅对被配置为触摸按键功能的引脚有效。*
5~4	VDS	内部运放输出电压选择 00：2V 01：2.5V 10：3V 11：4V
3~2	VIRS	内部电压基准选择 00：1.0V 01：1.5V 10：2.0V 11：2.5V
1	TKPWS	充电电源选择 0：选择外部电源 1：选择内部运放输出
0	TKCVS	充电基准电压选择 0：选择外部电压基准 1：选择内部电压基准

表 20-5-3 寄存器 TKCKS

C4H	7	6	5	4	3	2	1	0
TKCKS	-	-	-	-	-	-	-	TKSIL
R/W	-	-	-	-	-	-	-	R/W
初始值	-	-	-	-	-	-	-	0

位编号	位符号	说明
7~1	-	-
0	TKSIL	触摸按键采样时钟选择 0：选择 24M 的六分频(4M)时钟 1：选择慢速(131K)时钟

表 20-5-4 寄存器TKCFG

C1H	7	6	5	4	3	2	1	0
TKCFG	TKDIV			TKTMS				
R/W	R/W			R/W				
初始值	1	1	1	1	1	1	1	1

位编号	位符号	说明
7~5	TKDIV	触摸时钟分频选择 000: 不分频 001：2 分频 010：3 分频 … 111：8 分频
4~0	TKTMS	外挂调制电容放电时间设置 放电时间= TKTMS x 128 x 充放电时钟周期 在 TKDIV=0 的条件下，放电时间范围是：32us - 992us *备注：TKTMS 不能设置为 0。*

表 20-5-5 寄存器TKMTS

C2H	7	6	5	4	3	2	1	0
TKMTS	TKMTS[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~0	TKMTS	定时模式的启动时间选择寄存器 启动时间=（TKMTS+1）x 128 x IRCL 时钟周期。因为 IRCL 时钟频率为 131KHz，所以时间范围是 1ms - 256ms。

表 20-5-6 寄存器TKCHS

91H	7	6	5	4	3	2	1	0
TK0CHS	-	-	-	TKPS0[3:0]				
R/W	-	-	-	R/W				
初始值	-	-	-	0	0	0	0	0
92H	7	6	5	4	3	2	1	0
TK1CHS	-	-	-	TKPS1[3:0]				
R/W	-	-	-	R/W				
初始值	-	-	-	0	0	0	0	0
93H	7	6	5	4	3	2	1	0
TK2CHS	-	-	-	TKPS2[3:0]				
R/W	-	-	-	R/W				
初始值	-	-	-	0	0	0	0	0
94H	7	6	5	4	3	2	1	0
TK3CHS	-	-	-	TKPS3[3:0]				
R/W	-	-	-	R/W				
初始值	-	-	-	0	0	0	0	0
95H	7	6	5	4	3	2	1	0
TK4CHS	-	-	-	TKPS4[3:0]				
R/W	-	-	-	R/W				
初始值	-	-	-	0	0	0	0	0
96H	7	6	5	4	3	2	1	0
TK5CHS	-	-	-	TKPS5[3:0]				
R/W	-	-	-	R/W				
初始值	-	-	-	0	0	0	0	0

位编号	位符号	说明
7~5	-	-
4~0	TKPSn	通道 n 选择位域 00000：TK0~TK16 关闭 00001：选择 TK0 00010：选择 TK1 00011：选择 TK2 ..... 10001：选择 TK16 10010：选择内部参考电容

表 20-5-9 寄存器TKMS

F1H	7	6	5	4	3	2	1	0
TK0MSL	TK0MS[7:0]							
R/W	R							
初始值	0	0	0	0	0	0	0	0
F2H	7	6	5	4	3	2	1	0
TK0MSH	TK0MS[15:8]							
R/W	R							
初始值	0	0	0	0	0	0	0	0
F3H	7	6	5	4	3	2	1	0
TK1MSL	TK1MS[7:0]							
R/W	R							
初始值	0	0	0	0	0	0	0	0
F4H	7	6	5	4	3	2	1	0
TK1MSH	TK1MS[15:8]							
R/W	R							
初始值	0	0	0	0	0	0	0	0
F5H	7	6	5	4	3	2	1	0
TK2MSL	TK2MS[7:0]							
R/W	R							
初始值	0	0	0	0	0	0	0	0
F6H	7	6	5	4	3	2	1	0
TK2MSH	TK2MS[15:8]							
R/W	R							
初始值	0	0	0	0	0	0	0	0
E9H	7	6	5	4	3	2	1	0
TK3MSL	TK3MS[7:0]							
R/W	R							
初始值	0	0	0	0	0	0	0	0
EAH	7	6	5	4	3	2	1	0
TK3MSH	TK3MS[15:8]							
R/W	R							
初始值	0	0	0	0	0	0	0	0
EBH	7	6	5	4	3	2	1	0
TK4MSL	TK4MS[7:0]							
R/W	R							
初始值	0	0	0	0	0	0	0	0
ECH	7	6	5	4	3	2	1	0
TK4MSH	TK4MS[15:8]							
R/W	R							
初始值	0	0	0	0	0	0	0	0
EDH	7	6	5	4	3	2	1	0
TK5MSL	TK5MS[7:0]							
R/W	R							
初始值	0	0	0	0	0	0	0	0
EEH	7	6	5	4	3	2	1	0
TK5MSH	TK5MS[15:8]							
R/W	R							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
15~0	TKnMS	触摸采样数据寄存器

表 20-5-10 寄存器TKIF

C3H	7	6	5	4	3	2	1	0
TKIF	-	-	TKIF5	TKIF4	TKIF3	TKIF2	TKIF1	TKIF0
R/W	-	-	R	R	R	R	R	R
初始值	-	-	0	0	0	0	0	0

位编号	位符号	说明
7~6	-	-
5	TKIF5	6 个选定通道中的第 6 通道触摸采集中断标志位
4	TKIF4	6 个选定通道中的第 5 通道触摸采集中断标志位
3	TKIF3	6 个选定通道中的第 4 通道触摸采集中断标志位
2	TKIF2	6 个选定通道中的第 3 通道触摸采集中断标志位
1	TKIF1	6 个选定通道中的第 2 通道触摸采集中断标志位
0	TKIF0	6 个选定通道中的第 1 通道触摸采集中断标志位

20.6 触摸控制例程

备注：触摸应用实例请参考本公司标准触摸库软件及相关文档。

21 蜂鸣器（BUZZER）

21.1 功能简介

JZ8FC005T 系列芯片内置一路BUZZER 输出，输出频率可灵活设置。BUZZER 可用于直接驱动蜂鸣器。

21.2 寄存器描述

表 21-2-1 寄存器 BZCON

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

位编号	位符号	说明
7~3	-	-
2	BZD	BUZZER 输出电平值，BZH=1 时才有效
1	BZH	BUZZER 输出固定电平使能，1 使能，BUZZER 输出 BZD 的值
0	BZE	BUZZER 使能位，1 使能 *备注：* *BUZZER 要使能，还必须对应的引脚选择作为 BUZZER 功能。*

表 21-2-2 寄存器 BZDIV

80C1H	7	6	5	4	3	2	1	0
BZDIVL	BZDIV[7:0]							
R/W	R/W							
初始值	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							
初始值	0	0	1	1	1	1	1	0

位编号	位符号	说明
15~0	BZDIV	BUZZER 频率配置寄存器 $F_{buzzer} = F_{sys}/(BZDIV+1)$ *备注：* *F_{buzzer} 为 BUZZER 的输出频率，F_{sys} 为系统时钟的频率。*

表 21-2-3 寄存器 BZDUT

80C3H	7	6	5	4	3	2	1	0
BZDUTL	BZDUT[7:0]							
R/W	R/W							
初始值	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							
初始值	0	0	0	1	1	1	1	1

位编号	位符号	说明
15~0	BZDUT	BUZZER 占空比配置寄存器 高电平时间= T_{sys} *(BZDUT+1) *备注：* *1. T_{sys} 为系统时钟的周期。* *2. 当BZDUT 的值小于 BZDIV 时，BUZERR 输出才有意义。*

22 低电压检测（LVD）

22.1 功能简介

低电压检测（LVD）用于监控芯片自身的供电 VDD，可设置检测电压范围为 1.7V~4.8V 每级 0.1V。当 VDD 小于所设定的电压值时，可设置触发中断或复位。

重要提醒：*由于生产工艺的影响，芯片之间 LVD 触发电压存在一定的差异。*

LVD 结构图如图 22-1-1 所示。

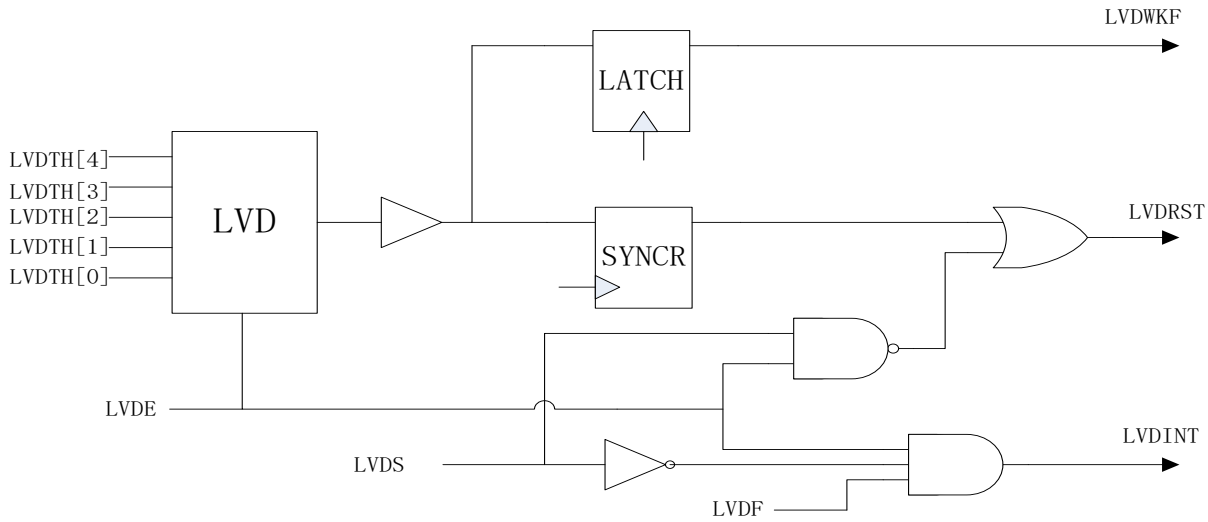


图 22-1-1 LVD 模块示意图

22.2 功能描述

LVD 功能通过 LVDE 位使能，而检测的电压则通过 LVDTH 位域设置。当芯片 VDD 小于所设置的电压时，LVD 功能产生的标志 LVDF 位将置 1，如果 LVDS=0，会产生 LVD 中断，如果 LVDS=1，会产生复位。要注意的是，LVD 复位产生之后，LVD 自身的电路并不会复位，寄存器 LVDCON 还会保持之前的状态，所以，当 LVD 复位产生之后，如果 VDD 持续低于所设定的电压，芯片将会一直处于复位状态。同样地，当 LVD 中断产生后，如果VDD 持续低于所设定的电压，LVD 中断也会重复地产生。

22.3 寄存器描述

表 22-3-1 寄存器 LVDCON

D7H	7	6	5	4	3	2	1	0
LVDCON	LVDE	LVDS	LVDF	LVDTH[3:0]				
R/W	R/W	R/W	R/W	R/W				
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7	LVDE	LVD 使能位，1 有效
6	LVDS	LVD 功能选择位 0：中断 1：复位
5	LVDF	LVD 产生标志位，写 1 清 0
4	-	-
3~0	LVDTH	LVD 触发电平选择位域 00000：1.7V 00001：1.8V 00010：1.9V …… 11101： 4.6V 11110：4.7V 11111：4.8V

22.4 LVD 控制例程

LVD 中断例程

例如，设置 LVD 为中断模式，检测电压为 3V，程序如下：

```
-------------------------------------------------------------------------------------------
#define LVDE(N)          (N<<7)    //N=0~1
#define LVDS_reset       (1<<6)
#define LVDS_int         (0<<6)
#define LVDF             (1<<5)
#define LVDTH_3V         13
void LVD_init(void)
{
    LVDCON = LVDE(1) | LVDS_int | LVDTH_3V;   //设置 LVD 使能，设置 LVD 为中断模式，检测电压为 3V
    INT4EN = 1;   //INT4 中断使能
    EA = 1;          //开启总中断
}
void INT4_ISR (void) interrupt 9
{
    if(LVDCON & LVDF)
    {
        LVDCON |= LVDF;       //清除 LVD 中断标志
        //LVD 中断服务程序
        ...

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

LVD 复位例程

例如，设置 LVD 为复位模式，检测电压为 3V，程序如下：

```
-------------------------------------------------------------------------------------------
#define LVDE(N)          (N<<7)    //N=0~1
#define LVDS_reset       (1<<6)
#define LVDS_int         (0<<6)
#define LVDF             (1<<5)
#define LVDTH_3V         13
void LVD_init(void)
{
    LVDCON = LVDE(1) | LVDS_reset | LVDTH_3V;//设置 LVD 使能，设置 LVD 为复位模式，检测电压为 3V
}
-------------------------------------------------------------------------------------------
```

23 运放（AMP）

23.1 功能介绍

JZ8FC005T 系列芯片内置两个运放：运放A 和运放B。

运放 A 为通用运放，使能位为 AMP_A_EN，经校准后，失调电压低于 0.5mv.运放 A 也可设置为数字输出，数字输出使能位为CODE_DETA，当AMP_A_OUT 输出电压高于逻辑 1 的电平时，AMP_A_SIG 为1，反之为 0。

运放 B 为无线充专用运放，用于 ASK 解码。运放B 的放大倍数内部共有 4 档可选，经运放 B 放大后，如电平高于逻辑 1 电平，输出为 1，反之为 0. 运放 B 的输出转换为数字信号后主要作为无线充 ASK 的输入信号进行解码。也可选择运放 B 输出的数字信号从 AMP_B_OUT 输出，使能位为 AMP_B_T_EN。与运放 A 类似，AMP_B_SIG 表示运放B 的输出数字逻辑。

23.2 结构图

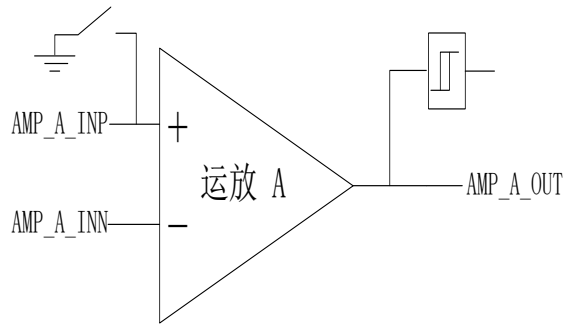


图 23-2-1 运放 A 结构图

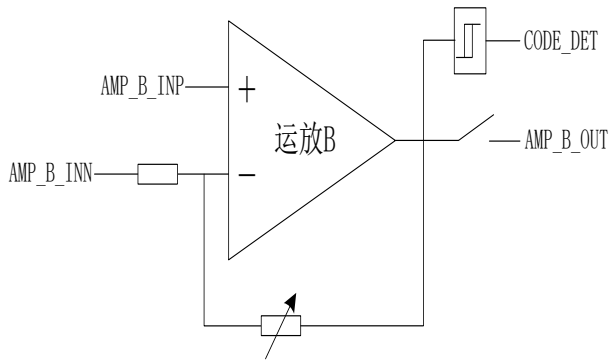


图 23-2-2 运放 B 结构图

23.3 寄存器描述

表 23-3-1 寄存器 AMPCON

80B8H	7	6	5	4	3	2	1	0
AMPCON	AMP_A_EN	AMP_A_INP_PD	AMP_B_EN	AMP_B_SM_EN	AMP_B_OUT_EN	CODE_DETA	AMP_A_SM_EN	-
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	-
初始值	0	0	0	1	0	-	0	0

位编号	位符号	说明
7	AMP_A_EN	AMP_A 的使能信号，1 使能，0 不使能
6	AMP_A_INP_PD	AMP_A 的正端输入下拉到地控制，1 下拉到地，0 接到 PAD
5	AMP_B_EN	AMP_B 的使能信号，1 使能，0 不使能
4	AMP_B_SM_EN	AMP_B 的输出施密特控制，1 为施密特输出，0 为反相器输出
3	AMP_B_T_EN	AMP_B 的输出使能，为 1 时，AMP_B 输出连到 P01 脚，为 0 时，不输出
2	CODE_DETA	AMP_A 数字信号输出，
1	AMP_A_SM_EN	AMP_A 的输出施密特控制，1 为施密特输出，0 为反相器输出
0	-	-

表 23-3-2 寄存器 AMPAOS

80B9H	7	6	5	4	3	2	1	0
AMPAOS	AMP_B_GAIN		OS_TRIM					
R/W	R/W		R/W					
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~6	AMP_B_GAIN	AMP_B 的增益选择位 00：33 倍 01：50 倍 10：75 倍 11：100 倍
5~0	OS_TRIM	运放 A 的校正位： OS_TRIM<5>选择方向，为 0 时，增加负端电压；为 1 时，减小负端电压。 调整电压=0.4*OS_TRIM<5:0> mV

表 23-3-3 寄存器 AMPOUT

80BAH	7	6	5	4	3	2	1	0
AMPOUT	-	-	-	-	-	-	AMP_A_SIG	AMP_B_SIG
R	-	-	-	-	-	-	R	R
初始值	-	-	-	-	-	-	0	0

位编号	位符号	说明
7~2	-	-
1	AMP_A_SIG	AMP_A 数字信号
0	AMP_B_SIG	AMP_B 数字信号

24 无线解码模块

24.1 功能简介

JZ8FC005T 系列芯片内置两路 QI 硬件解码器，支持QI 无线充标准通信协议硬件解码，开发产品更便捷。

24.2 寄存器描述

表 24-2-1 寄存器 DCCR / DC1CR

80D0H	7	6	5	4	3	2	1	0
DCCR	EN	WBIE	ERRIE	SSEL	P2SF	WBF	ERRF	OVESEL
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0

80E0H	7	6	5	4	3	2	1	0
DC1CR	EN	WBIE	ERRIE	SSEL	P2SF	WBF	ERRF	OVESEL
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7	EN	ASK 码使能位 0：ASK 解码禁止 1：ASK 解码使能
6	WBIE	一字节 ASK 解码完成中断使能位 0：一字节数据 ASK 解码完成中断禁止 1：一字节数据 ASK 解码完成中断使能和从 Preamble 接收到 start 位中断使能
5	ERRIE	解码错误中断使能位 0：解码错误中断禁止 1：解码错误中断使能
4	SSEL	ASK 信号来源选择： 0：ASK 信号来自内部运放 1：ASK 信号来自引脚（引脚的选择设置在 DCFCR/DC1FCR 寄存器）
3	P2SF	从 Preamble 位接收到 start 位（表示开始接收数据）标志 手动写 1 清 0 (注：对应的中断使能位与 WBIE 合并)
2	WBF	一字节 data 解码完成标志位 手动写 1 清 0
1	ERRF	解码错误标志： 手动写 1 清 0
0	OVESEL	超时产生错误设置： 0：超时不产生错误 1：超时会产生错误 （注：无论如何，只要出现超时，硬件内部都会自动复位，等待新一次数据传输）

表 24-2-2 寄存器 DCDS / DC1DS

80D1H	7	6	5	4	3	2	1	0
DCDS	SYNC_NUM					ERRSF[2:0]		
R	R/W					R		
初始值	0	0	0	0	0	0	0	0

80E1H	7	6	5	4	3	2	1	0
DC1DS	SYNC_NUM					ERRSF[2:0]		
R	R/W					R		
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~3	SYNC_NUM	ASK 通信时，允许识别最少的同步头数
2~0	ERRSF[2:0]	错误原因标志： 出现错误时硬件会自动更新该标志，并且会在新的一次传输开始时自动清 0 错误原因译码： 000：无错误 001：超时错误（任何情况下只要出现超时都会表示出来） 010：信号脉冲宽度超出阈值范围错误 011：Parity 校验错误 100：START 错误（START 接收到 1） 101：STOP 错误（STOP 接收到 0）

表 24-2-3 寄存器 DCDB

80D2H	7	6	5	4	3	2	1	0
DCDB	DCDB							
R	R							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~0	DCDB	接收到的一字节数据

表 24-2-4 寄存器 DCBIT0HH

80D3H	7	6	5	4	3	2	1	0
DCBIT0HH	DCBIT0HH							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~0	DCBIT0HH	BIT0 的高阈值的高 8 位 注：脉冲宽度的设置: 假设bit0 的脉冲宽度为X BIT0L * T（perclk）< X ± 2^(DCFCR[3:0]+2) * $T_{(perclk)}$< BIT0H * T（perclk）

表 24-2-5 寄存器 DCBIT0HL

80D4H	7	6	5	4	3	2	1	0
DCBIT0HL	DCBIT0HL							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~0	DCBIT0HL	BIT0 的高阈值的低 8 位

表 24-2-6 寄存器 DCBIT0LH

80D5H	7	6	5	4	3	2	1	0
DCBIT0LH	DCBIT0LH							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
7~0	DCBIT0LH	BIT0 的低阈值的高 8 位						

表 24-2-7 寄存器 DCBIT0LL

80D6H	7	6	5	4	3	2	1	0
DCBIT0LL	DCBIT0LL							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
7~0	DCBIT0LL	BIT0 的低阈值的低 8 位						

表 24-2-8 寄存器 DCBIT1HH

80D7H	7	6	5	4	3	2	1	0
DCBIT1HH	DCBIT1HH							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
7~0	DCB1HH	BIT1 的高阈值的高 8 位 注：BIT1 脉冲宽度的设置: 假设bit1 的脉冲宽度为X BIT1L * T（perclk）< X ± 2^(DCFCR[3:0]+2) * $T_{(perclk)}$< BIT1H * T（perclk）						

表 24-2-9 寄存器 DCBIT1HL

80D8H	7	6	5	4	3	2	1	0
DCBIT1HL	DCBIT1HL							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
7~0	DCBIT1HL	BIT1 的高阈值的低 8 位						

表 24-2-10 寄存器 DCBIT1LH

80D9H	7	6	5	4	3	2	1	0
DCBIT1LH	DCBIT1LH							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
7~0	DCBIT1LH	BIT1 的低阈值的高 8 位						

表 24-2-11 寄存器 DCBIT1LL

80DAH	7	6	5	4	3	2	1	0
DCBIT1LL	DCBIT1LL							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~0	DCBIT1LL	BIT1 的低阈值的低 8 位

表 24-2-12 寄存器 DCOVRH

80DBH	7	6	5	4	3	2	1	0
DCOVRH	DCOVRH							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~0	DCOVRH	超时阈值寄存器高 8 位

表 24-2-13 寄存器 DCOVRL

80DCH	7	6	5	4	3	2	1	0
DCOVRL	DCOVRL							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~0	DCOVRL	超时阈值寄存器低 8 位

表 24-2-14 寄存器 DCFCR

80DDH	7	6	5	4	3	2	1	0
DCFCR	SPIN_SEL[7:6]		-	-	FCR[3:0]			
R/W	R/W		-	-	R/W			
初始值	0	0	-	-	0	0	0	0

位编号	位符号	说明
7~6	SPIN_SEL[7:6]	ASK 信号来源引脚设置： 00：ASK 信号来自 P01 01：ASK 信号来自 P14 10：ASK 信号来自 P16 11：无信号
5	-	-
4	-	-

3~0	FCR[3:0]	对 ASK 信号的采样点的间隔设置（用于滤波）： 0000: 采样点间隔 2 个 perclk 周期 0001：采样点间隔 4 个 perclk 周期 0010：采样点间隔 8 个 perclk 周期 0011：采样点间隔 16 个 perclk 周期 0100：采样点间隔 32 个 perclk 周期 0101：采样点间隔 64 个 perclk 周期 0110：采样点间隔 128 个 perclk 周期 0111：采样点间隔 256 个 perclk 周期 1000：采样点间隔 512 个 perclk 周期 1001：采样点间隔 1024 个 perclk 周期

表 24-2-15 寄存器 DC1DB

80E2H	7	6	5	4	3	2	1	0
DC1DB	DC1DB							
R	R							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
7~0	DC1DB	接收到的一字节数据						

表 24-2-16 寄存器 DC1BIT0HH

80E3H	7	6	5	4	3	2	1	0
DC1BIT0HH	DC1BIT0HH							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
7~0	DC1BIT0HH	BIT0 的高阈值的高 8 位 注：脉冲宽度的设置: 假设bit0 的脉冲宽度为X BIT0L * T（perclk）< X ± 2^(DC1FCR[3:0]+2) * $T_{(perclk)}$< BIT0H * T（perclk）						

表 24-2-17 寄存器 DC1BIT0HL

80E4H	7	6	5	4	3	2	1	0
DC1BIT0HL	DC1BIT0HL							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
7~0	DC1BIT0HL	BIT0 的高阈值的低 8 位						

表 24-2-18 寄存器 DC1BIT0LH

80E5H	7	6	5	4	3	2	1	0
DC1BIT0LH	DC1BIT0LH							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~0	DC1BIT0LH	BIT0 的低阈值的高 8 位

表 24-2-19 寄存器 DC1BIT0LL

80E6H	7	6	5	4	3	2	1	0
DC1BIT0LL	DC1BIT0LL							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~0	DC1BIT0LL	BIT0 的低阈值的低 8 位

表 24-2-20 寄存器 DC1BIT1HH

80E7H	7	6	5	4	3	2	1	0
DC1BIT1HH	DC1BIT1HH							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~0	DC1BIT1HH	BIT1 的高阈值的高 8 位 注：BIT1 脉冲宽度的设置: 假设bit1 的脉冲宽度为X BIT1L * T（perclk）< X ± 2^(DC1FCR[3:0]+2) * $T_{(perclk)}$< BIT1H * T（perclk）

表 24-2-21 寄存器 DC1BIT1HL

80E8H	7	6	5	4	3	2	1	0
DC1BIT1HL	DC1BIT1HL							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~0	DC1BIT1HL	BIT1 的高阈值的低 8 位

表 24-2-22 寄存器 DC1BIT1LH

80E9H	7	6	5	4	3	2	1	0
DC1BIT1LH	DC1BIT1LH							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~0	DC1BIT1LH	BIT1 的低阈值的高 8 位

表 24-2-23 寄存器 DC1BIT1LL

80EAH	7	6	5	4	3	2	1	0
DC1BIT1LL	DC1BIT1LL							

R/W	R/W							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
7~0	DC1BIT1LL	BIT1 的低阈值的低 8 位						

表 24-2-24 寄存器 DC1OVRH

80EBH	7	6	5	4	3	2	1	0
DC1OVRH	DC1OVRH							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
7~0	DC1OVRH	超时阈值寄存器高 8 位						

表 24-2-25 寄存器 DC1OVRL

80ECH	7	6	5	4	3	2	1	0
DC1OVRL	DC1OVRL							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
7~0	DC1OVRL	超时阈值寄存器低 8 位						

表 24-2-25 寄存器 DC1FCR

80EDH	7	6	5	4	3	2	1	0
DC1FCR	SPIN_SEL[7:6]		-	-	FCR[3:0]			
R/W	R/W		-	-	R/W			
初始值	0	0	-	-	0	0	0	0
位编号	位符号	说明						
7~6	SPIN_SEL[7:6]	ASK 信号来源引脚设置： 00：ASK 信号来自 P02 01：ASK 信号来自 P13 10：ASK 信号来自 P15 11：无信号						
5	-	-						
4	-	-						
3~0	FCR[3:0]	对 ASK 信号的采样点的间隔设置（用于滤波）： 0000: 采样点间隔 2 个 perclk 周期 0001：采样点间隔 4 个 perclk 周期 0010：采样点间隔 8 个 perclk 周期 0011：采样点间隔 16 个 perclk 周期 0100：采样点间隔 32 个 perclk 周期 0101：采样点间隔 64 个 perclk 周期 0110：采样点间隔 128 个 perclk 周期 0111：采样点间隔 256 个 perclk 周期 1000：采样点间隔 512 个 perclk 周期 1001：采样点间隔 1024 个 perclk 周期						

25 乘除法器（MDU）

25.1 功能简介

乘除法器（MDU）可实现 32 位数除以 32 位数、16 位数乘以 16 位数、左移位、右移位共 4 种运算。其中乘法运算以及移位操作时间为 1 个时钟周期，除法运算为 8 个时钟周期。

25.2 结构图

图 25-2-1 为乘法电路原理示意图。

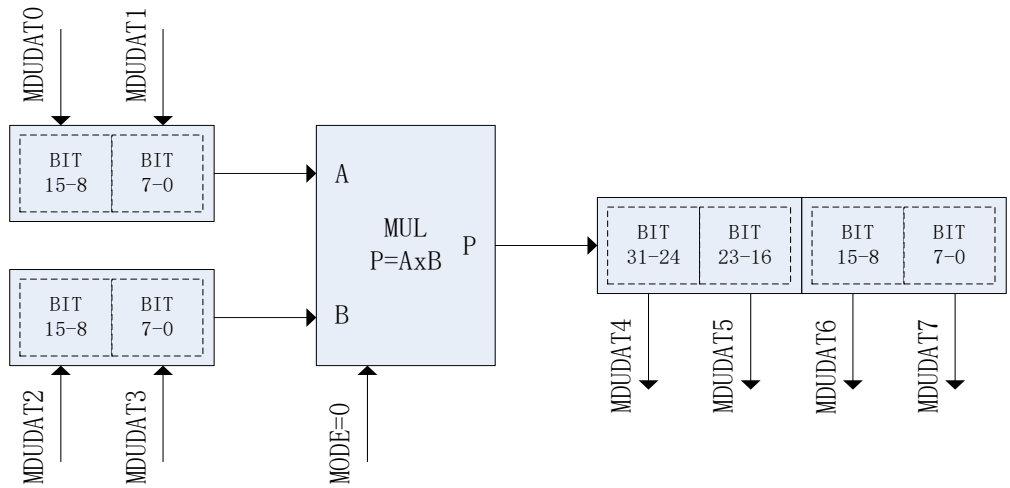


图 25-2-1 乘法电路原理示意图

图 25-2-2 为除法电路原理示意图。

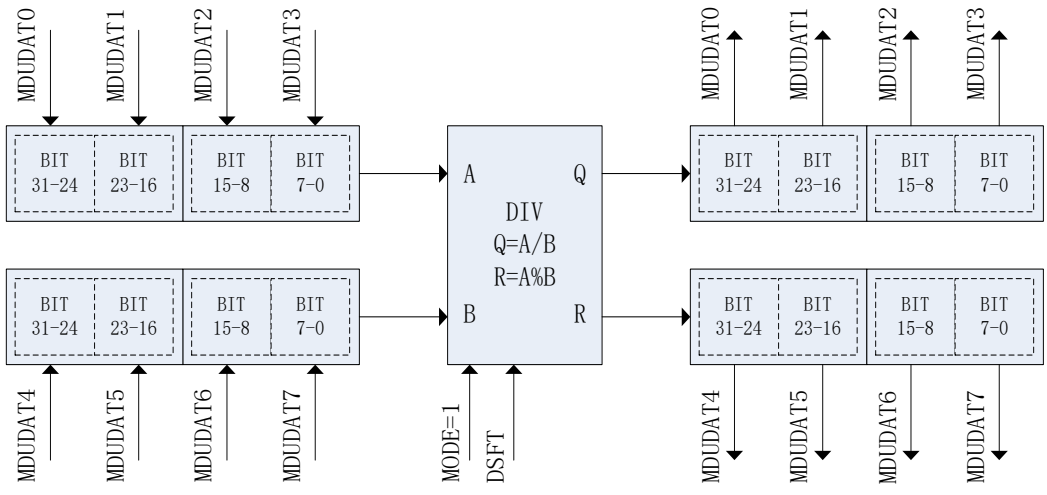


图 25-2-2 除法电路原理示意图

图 25-2-3 为移位电路原理示意图。

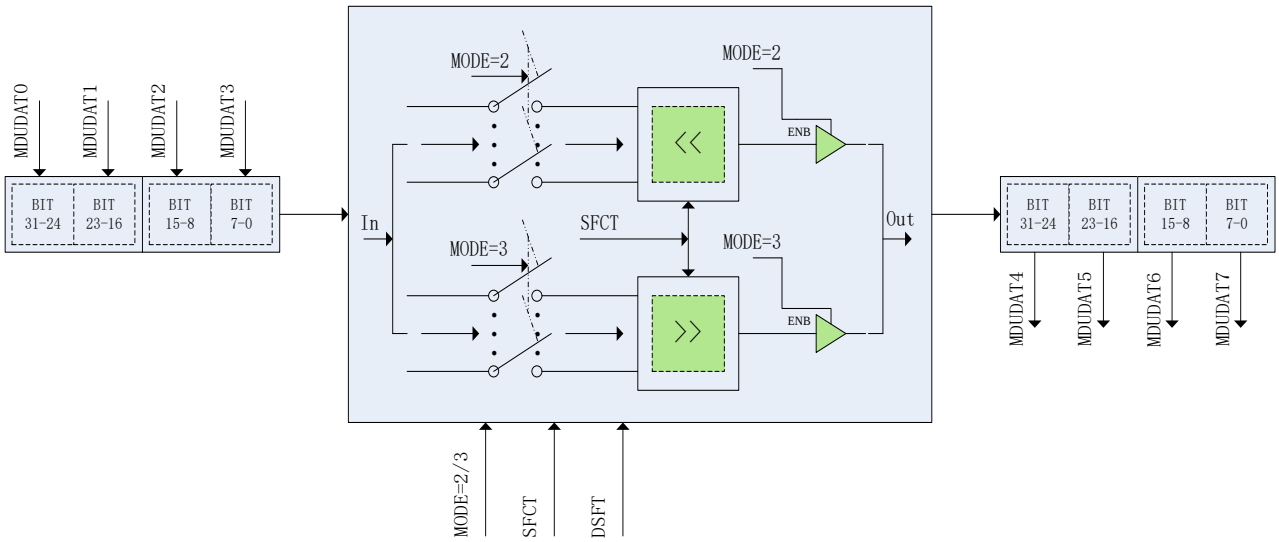


图 25-2-3 移位电路原理示意图

25.3 功能描述

25.3.1 乘法器

当设置 MODE=0 时，MDU 设置为 16 位×16 位乘法器。其中被乘数写入寄存器 MDUDAT0、MDUDAT1，乘数写入寄存器 MDUDAT2、MDUDAT3。由于乘法器运算只需要 1 个时钟周期，被乘数和乘数写入寄存器后立即就可以得出乘积，乘积存放在寄存器MDUDAT4、MDUDAT5、MDUDAT6、MDUDAT7。

25.3.2 除法器

当设置 MODE=1 时，MDU 设置为 32 位÷32 位除法器。其中被除数写入寄存器 MDUDAT0、MDUDAT1、MDUDAT2、MDUDAT3，除数写入 MDUDAT4、MDUDAT5、MDUDAT6、MDUDAT7。被除数和除数写入寄存器后，需要设置 DSFT 位才能启动运算，当运算完成后，DSFT 自动清 0，商数存放在寄存器 MDUDAT0、MDUDAT1、MDUDAT2、MDUDAT3，余数存放在 MDUDAT4、MDUDAT5、MDUDAT6、MDUDAT7。由于除法运算需要 8 个时钟周期，所以在启动除法运算后需要等待 8 个时钟周期或等待 DSFT 清 0 后才能读取运算结果。

25.3.3 移位运算

当设置 MODE=2 或 MODE=3 时，MDU 设置为移位运算器，其中 MODE=2 时为左移位，MODE=3 时为右移位。移位的位数通过 SFCT 位域设置，而被操作的 32 位数写入寄存器 MDUDAT0 、MDUDAT1 、MDUDAT2、MDUDAT3。被操作数写入后设置 DSFT=1 启动运算，由于移位运算为 1 个时钟周期，所以设置 DSFT=1 后立即可以读取运算结果。运算结果存放于寄存器 MDUDAT4、MDUDAT5、MDUDAT6、MDUDAT7。

25.4 寄存器描述

表 25-4-1 寄存器 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				
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~6	MODE	操作模式选择位域 00：乘法运算 01：除法运算 10：左移位操作 11：右移位操作
5	DSFT	除法运算和移位操作启动位，1 有效。在乘法运算中，此位无效。
4~0	SFCT	移位操作移位次数，移位次数为（SFCT + 1）。在乘除法运算中，此位无效。

表 25-4-2 寄存器 MDUDAT

E7H	7	6	5	4	3	2	1	0
MDUDAT	MDUDAT[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

备注：MDUDAT 为带索引寄存器，设置 INDEX=0~7 分别对应 MDUDAT0～MDUDAT7

位编号	位符号	说明
7~0	MDUDAT	MDU 数据存储寄存器。对 MDUDAT0～MDUDAT7 读写没有顺序要求。 在乘法运算中， MDUDAT0：被乘数 15～8 位 MDUDAT1：被乘数 7～0 位 MDUDAT2：乘数 15～8 位 MDUDAT3：乘数 7～0 位 乘法运算结果： MDUDAT4：乘积 31～24 位 MDUDAT5：乘积 23～16 位 MDUDAT6：乘积 15～ 8 位 MDUDAT7：乘积 7～ 0 位 在除法运算中， MDUDAT0：被除数 31～24 位； MDUDAT1：被除数 23～16 位； MDUDAT2：被除数 15～ 8 位； MDUDAT3：被除数 7～ 0 位。 MDUDAT4：除数 31～24 位； MDUDAT5：除数 23～16 位； MDUDAT6：除数 15～ 8 位； MDUDAT7：除数 7～ 0 位。 除法运算结果： MDUDAT0：商数 31～24 位； MDUDAT1：商数 23～16 位； MDUDAT2：商数 15～ 8 位； MDUDAT3：商数 7～ 0 位。 MDUDAT4：余数 31～24 位；

		MDUDAT5：余数 23～16 位； MDUDAT6：余数 15～ 8 位； MDUDAT7：余数 7～ 0 位。 在移位操作中， MDUDAT0：源操作数 15～8 位； MDUDAT1：源操作数 7～0 位； MDUDAT2：源操作数 15～8 位； MDUDAT3：源操作数 7～0 位。 移位操作结果： MDUDAT4：目标操作数 31～24 位； MDUDAT5：目标操作数 23～16 位； MDUDAT6：目标操作数 15～ 8 位； MDUDAT7：目标操作数 7～ 0 位。 备注： 除法运算时，除数如果为 0，模块将不进行运算。同时，DSFT 为 1 时，如果对被除数或者除数改写将不会影响正在进行的运算的结果。

25.5 MDU 控制例程

先定义以下联合体：

```
---------------------------------------------------------------------------------------------
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;
---------------------------------------------------------------------------------------------
```

◆ **乘法运算操作例程**

例如，被乘数为 65535，乘数为 1000，程序如下：

```
---------------------------------------------------------------------------------------------
#define MOD_MULT (0<<6)
void Mult(void)
{
    WORD_UNION Faciend;          //被乘数
    WORD_UNION Multiplier;       //乘数
    DWORD_UNION Product;         //乘积

    Faciend.wVal = 65535;
    Multiplier.wVal = 1000;

    MDUCON      = MOD_MULT;//设置 MDU 为乘法运算模式

    INDEX = 0;
    MDUDAT = Faciend.bVal[0]; //填写被乘数高 8 位
    INDEX = 1;
    MDUDAT = Faciend.bVal[1]; //填写被乘数低 8 位
    INDEX = 2;
    MDUDAT = Multiplier.bVal[0];//填写乘数高 8 位
    INDEX = 3;
    MDUDAT = Multiplier.bVal[1];//填写乘数低 8 位
```

```
    INDEX = 4;
    Product.bVal[0] = MDUDAT;  //读取乘积 24~31 位
    INDEX = 5;
    Product.bVal[1] = MDUDAT;  //读取乘积 16~23 位
    INDEX = 6;
    Product.bVal[2] = MDUDAT;  //读取乘积 8~15 位
    INDEX = 7;
    Product.bVal[3] = MDUDAT;  //读取乘积 0~7 位
}
```

◆ **除法运算操作例程**

例如，被除数为 0xFFFFFFFF，除数为 0x10000000，程序如下：

```
#define MOD_DIV  (1<<6)
#define DSFT (1<<5)
void Divid(void)
{
    DWORD_UNION Dividend;        //被除数
    DWORD_UNION Divisor;         //除数
    DWORD_UNION Quotient;        //商
    DWORD_UNION Remainder;       //余数

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

    MDUCON = MOD_DIV; //设置 MDU 为除法运算模式

    INDEX = 0;
    MDUDAT = Dividend.bVal[0];      //填写被除数 24~31 位
    INDEX = 1;
    MDUDAT = Dividend.bVal[1];      //填写被除数 16~23 位
    INDEX = 2;
    MDUDAT = Dividend.bVal[2];      //填写被除数 8~15 位
    INDEX = 3;
    MDUDAT = Dividend.bVal[3];      //填写被除数 0~7 位

    INDEX = 4;
    MDUDAT = Divisor.bVal[0]; //填写除数 24~31 位
    INDEX = 5;
    MDUDAT = Divisor.bVal[1]; //填写除数 16~23 位
    INDEX = 6;
    MDUDAT = Divisor.bVal[2]; //填写除数 8~15 位
    INDEX = 7;
    MDUDAT = Divisor.bVal[3]; //填写除数 0~7 位
```

```
    MDUCON |=  DSFT;      //启动除法运算
    while(MDUCON & DSFT);    //等待除法运算结束

    INDEX = 0;
    Quotient.bVal[0] = MDUDAT;//读取商 24~31 位
    INDEX = 1;
    Quotient.bVal[1] = MDUDAT;//读取商 16~23 位
    INDEX = 2;
    Quotient.bVal[2] = MDUDAT;//读取商 8~15 位
    INDEX = 3;
    Quotient.bVal[3] = MDUDAT;//读取商 0~7 位

    INDEX = 4;
    Remainder.bVal[0]= MDUDAT;     //读取余数 24~31 位
    INDEX = 5;
    Remainder.bVal[1]= MDUDAT;     //读取余数 16~23 位
    INDEX = 6;
    Remainder.bVal[2]= MDUDAT;     //读取余数 8~15 位
    INDEX = 7;
    Remainder.bVal[3]= MDUDAT;     //读取余数 0~7 位
}
-------------------------------------------------------------------------------------------
```

◆ **移位运算操作例程**

例如，被操作数为 0x88880001，向左（或向右）移位 8 位，程序如下：

```
-------------------------------------------------------------------------------------------
#define MOD_SHIFT_LEFT          (2<<6)
#define MOD_SHIFT_RIGHT         (3<<6)
#define DSFT          (1<<5)
void Shift(void)
{
    DWORD_UNION SourceData;              //源数据
    DWORD_UNION DestinationData; //目标数据

    MDUCON = MOD_SHIFT_LEFT|7;      //设置向左移位操作及移位位数
    //MDUCON = MOD_SHIFT_RIGHT|7; //设置向右移位操作及移位位数

    SourceData.dwVal = 0x88880001;

    INDEX = 0;
    MDUDAT = SourceData.bVal[0];  //填写源数据 24~31 位
    INDEX = 1;
    MDUDAT = SourceData.bVal[1];  //填写源数据 16~23 位
    INDEX = 2;
```

```
    MDUDAT = SourceData.bVal[2];  //填写源数据 8~15 位
    INDEX = 3;
    MDUDAT = SourceData.bVal[3];  //填写源数据 0~7 位

    MDUCON |= DSFT;       //启动移位操作

    INDEX = 4;
    DestinationData.bVal[0] = MDUDAT;    //读取目标数据 24~31 位
    INDEX = 5;
    DestinationData.bVal[1] = MDUDAT;    //读取目标数据 16~23 位
    INDEX = 6;
    DestinationData.bVal[2] = MDUDAT;    //读取目标数据 8~15 位
    INDEX = 7;
    DestinationData.bVal[3] = MDUDAT;    //读取目标数据 0~7 位
}
--------------------------------------------------------------------------------
```

26 SPI 接口

26.1 功能简介

SPI 接口能够实现芯片与其他设备以半/全双工同步传输数据。外围设备可以是其它的 MCU,ADC,传感器,或闪速存储器等。SPI 可以是三线或者四线，有以下特点。

- 支持主机或从机操作
- 可选择最低位或最高位优先传输
- 4 种可编程的比特率
- 可编程的极性和相位
- 发送结束中断标志
- 写入冲突标志保护机制
- 支持主模式故障出错中断

图 26-1-1 和图 26-1-2 分别是 SPI 主机模式和从机模式的原理示意图。

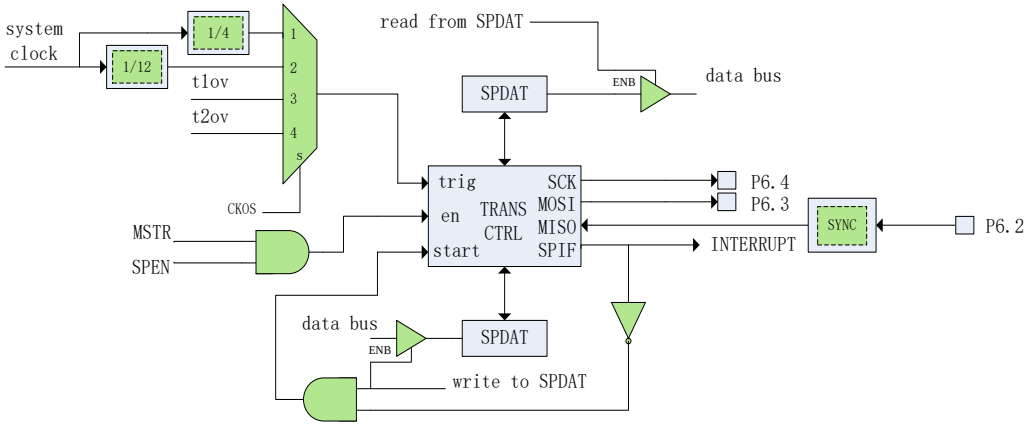


图 26-1-1 SPI 主机模式示意图

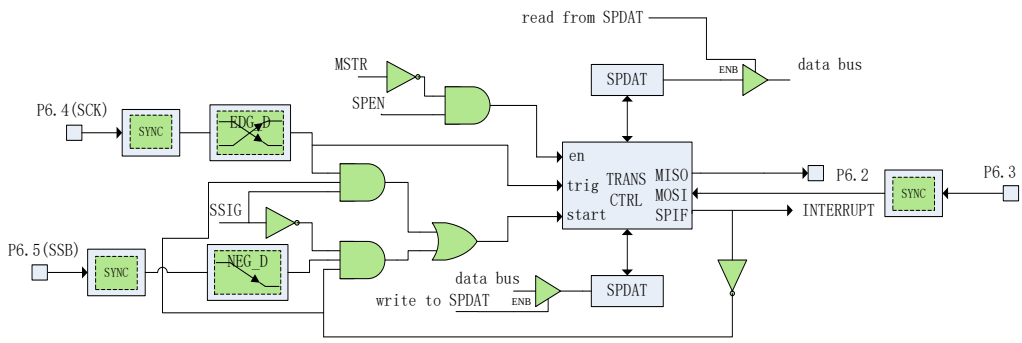


图 26-1-2 SPI 从机模式示意图

SPI 工作模式如下表所示。

表 26-1-1 SPI 工作模式

名称	描述
主机模式	所有的传输行为都由主机发起，包括 SCK 和 SSB 信号的产生等。 当设置 MSTR（SPCON[4]）位为 1，SPI 处于主机模式。用户需要另选择一个 GPIO 作为片选引脚，连接从机 SSB，数据传输开始前，主机拉低这个引脚，传输结束后拉高。 在主机模式，写入寄存器 SPDAT 的会启动数据传输。数据在时钟有效沿从 MOSI 移位输出。
从机模式	当设置 MSTR 位为 0，SPI 处于从机模式。 当 SSIG（SPCON[5]）为 1，则 SSB 引脚无效，SPI 为三线通信，从机默认片选有效；当 SSIG 为 0，SSB 引脚有效，SSB 为低电平表示从机被片选。

表 26-1-2 SPI 接口引脚描述

名称	描述
MOSI	主机输出，从机输入 当 SPI 作为主机时该引脚为主机数据输出端口，作为从机时为从机数据输入端口
MISO	主机输入，从机输出 当 SPI 作为主机时该引脚为主机数据输入端口，作为从机时为从机数据输出端口
SCK	串行时钟 当 SPI 作为主机时该引脚为串行时钟输出端口，作为从机时为串行时钟输入端口
SSB	从机选择 当 SPI 引脚主机时该引脚为从机选择输入端口，作为从机时为从机选择输入端口

表 26-1-3 SPI 相位与极性

名称	描述
CPHA	相位控制位 0：表示在 SCK 奇数边缘（1,3,5,…,15）采样数据 1：表示在 SCK 偶数边缘（2,4,6,…,16）采样数据
CPOL	极性控制位 0：表示 SCK 空闲时处于低电平 1：表示 SCK 空闲时处于高电平

结合表 26-1-3，实际传输时的波形如图 26-1-3 和图 26-1-4 所示。

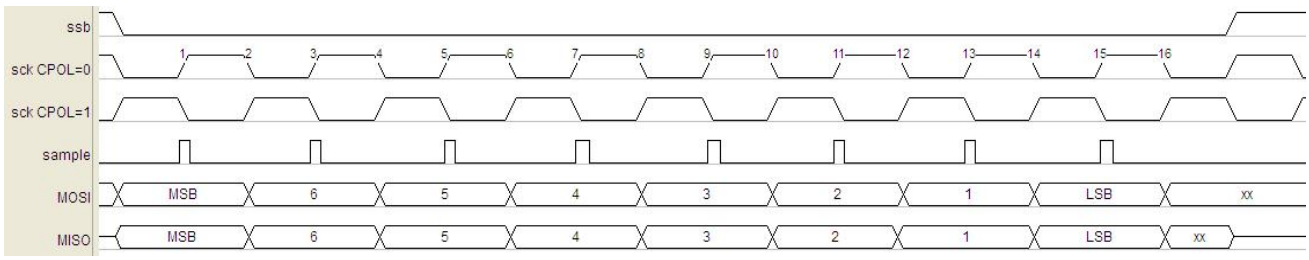


图 26-1-3 CPHA=0 时 SPI 时序图

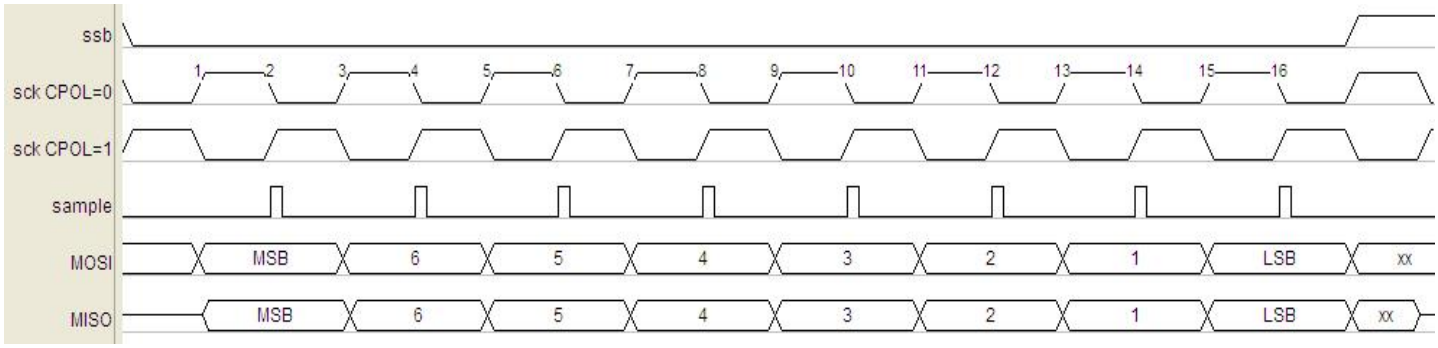


图 26-1-4 CPHA=1 时 SPI 时序图

26.2 寄存器描述

表 26-2-1 寄存器 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	
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7	SPEN	SPI 模块使能位，1 有效
6	LSBF	低位或高位优先发送/接收选择位 0：高位先发 1：低位先发
5	SSIG	SSB 引脚无效控制位，默认为 0，此时 SSB 信号有效
4	MSTR	主机/从机选择位 0：从机 1：主机
3	CPOL	时钟极性选择位 0：默认情况下时钟为低 1：默认情况下时钟为高
2	CPHA	时钟相位选择位 0：在时钟离开默认情况时采样数据 1：在时钟回到默认情况时采样数据
1~0	CKOS	SPI 输出时钟选择位 00：1/8 系统时钟 01：1/24 系统时钟 10：使用定时器 1 溢出标志，每两次溢出传输一次数据 11：使用定时器 2 溢出标志，每两次溢出传输一次数据

表 26-2-2 寄存器 SPDAT

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

位编号	位符号	说明
7~0	SPDAT	写 SPDAT 时，写入内部的 TBUF，读 SPDAT 时，从 RBUF 读出

表 26-2-3 寄存器 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
初始值	0	-	-	-	-	0	0	0

位编号	位符号	说明
7	SPIE	SPI 中断使能位，1 有效
6~3	-	-
2	WCOL	写入冲突标志位，在数据正在发送时，如有软件有写 SPDAT 的操作，此时数据无法写入，即产生写入冲突标志。该位 1 有效，写 1 清 0，有效时不会产生中断
1	MODF	故障模式标志位，1 有效，表明 SSB 在不正确的逻辑电平下，写 1 清 0，有效时会产生中断
0	SPIF	数据传输完成标志位，1 有效，写 1 清 0，有效时会产生中断

26.3 SPI 控制例程

SPI 主机例程

SPI 作为主机，向从机发送 10 个字节数据，程序如下：

```
---------------------------------------------------------------------------------------------------------------------------
#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 CPOS(N)         (N)     //3:系统时钟选择为定时器 2 溢出
                                //2:系统时钟选择为定时器 1 溢出
                                //1:系统时钟选择为 1/24 系统时钟
                                //0:系统时钟选择为 1/8 系统时钟
//SPSTA 位定义
#define SPIE        (1<<7)  //SPI 中断使能
#define WCOL        (1<<2)  //写入冲突标志
#define MODF        (1<<1)  //故障模式
#define SPIF        (1<<0)  //传输完成
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;//设置 P10 为 SPI SCK
    P01F = 6;//设置 P01 为 SPI MISO
    P00F = 6;//设置 P00 为 SPI MOSI
    P15F = 2;//设置 P15 为输出功能，作为 SPI 片选引脚
    P15 = 1; //设置片选引脚为高
    SPCON = SPEN(1) | LSBF(0) | SSIG(1) | MSTR(1) | CPOL(0) |CPHA(0) |CPOS(1);     //高位先发, SSB 有效,
主机, 1/24 系统时钟
    SPSTA |= SPIE;
    INT5EN = 1;
    spi_int_flag=0;
}
void INT5_ISR (void) interrupt 10
{
    if(SPSTA & SPIF)                    //SPI 中断
    {
        SPSTA |= SPIF;                  //清除 SPI 中断标志
        spi_int_flag = 1;
    }
    if(SPSTA&MODF)
```

```
    {
        SPSTA|=MODF;
    }
}
void main(void)
{
    CKSEL = 3;

    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 从机例程

SPI 作为从机，接收主机发送的数据，程序如下：

```
#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 CPOS(N)         (N)     //3:系统时钟选择为定时器 2 溢出
                                //2:系统时钟选择为定时器 1 溢出
                                //1:系统时钟选择为 1/24 系统时钟
                                //0:系统时钟选择为 1/8 系统时钟
//SPSTA 位定义
#define SPIE        (1<<7)  //SPI 中断使能
#define WCOL        (1<<2)  //写入冲突标志
#define MODF        (1<<1)  //故障模式
#define SPIF        (1<<0)  //传输完成
```

```
unsigned char spi_int_flag;
void SPI_init(void)
{
    P10F = 6;//设置 P10 为 SPI SCK
    P01F = 6;//设置 P01 为 SPI MISO
    P00F = 6;//设置 P00 为 SPI MOSI
    P15F = 6;//设置 P15 为输出功能，作为 SPI 片选引脚
    SPCON = SPEN(1) | LSBF(0) | SSIG(0) | MSTR(0) | CPOL(0) |CPHA(0) |CPOS(0);     //高位先发, SSB 有效，
从机, 1/8 系统时钟
    SPSTA |= SPIE;
    INT5EN = 1;
}
void INT5_ISR (void) interrupt 10
{
    if(SPSTA & SPIF)                                    //SPI 中断
    {
        SPSTA |= SPIF;                                  //清除 SPI 中断标志
        spi_int_flag = 1;
    }
    if(SPSTA&MODF)
    {
        SPSTA|=MODF;
    }
}
void main(void)
{
    CKSEL = 3;

    Uart2_Initial(115200);
    SPI_init();
    EA = 1;
    while(1)
    {
        if(spi_int_flag)
        {
            spi_int_flag = 0;
            Uart2_PutChar(0xaa);
            Uart2_PutChar(SPDAT);
        }
    }
}
```

---

27 程序下载和仿真

27.1 程序下载

JZ8FC005T 系列芯片主要采用 ISP 方式下载程序，芯片通过 I2C 接口与下载工具相连接，默认的升级接口为 P1.4(I2C SDA),P1.3(I2C SCL)。

更多关于程序下载步骤的细节请参考“JZCHIP 开发下载工具使用说明”。

27.2 在线仿真

JZ8FC005T 系列芯片支持在线仿真，芯片与仿真器之间通过 IIC 接口进行通信，出厂默认的 I2C 接口是 P1.4(I2C SDA)和 P1.3(I2C SCL)。要注意的是，由于芯片与仿真器间通过 IIC 通信，所以与仿真器连接的 I2C 接口引脚不能设置为其他功能，并且应用程序里不能使用 IIC 功能，否则将无法进入仿真模式。另外，由于 I2C 的通信速度是由主时钟决定，所以应用程序里不能将主时钟设置为低速时钟，也不能进入省电模式，否则都会影响芯片与仿真器间的通信。

当TSME=0(PCON[3])时，芯片禁止进入仿真模式。当芯片进入仿真模式后，TSMODE 位(PCON[2]])置 1，应用程序可通过判断此位状态来决定是否切换至低速时钟或进入省电模式。

更多关于仿真功能的细节可参考仿真器的相关文档介绍。

28 电气特性

28.1 极限参数

参数	最小值	最大值	单位
直流供电电压	-0.3	6	V
I/O 引脚输入电压	-0.3	VDD+0.3	V
工作环境温度	-40	85	℃
储存温度	-55	125	℃
CPU 工作频率	-	24	MHz

*备注：超过“**极限参数**”范围有可能对芯片造成损坏，无法预期芯片在上述范围外的工作状态，若长期在标示范围外工作，可能会影响芯片的可靠性。*

28.2 直流电气特性

芯片参数	符号	工作电压	最小值	典型值	最大值	单位	测试条件
工作电流	Iop1	VDD=2.0V		3.18		mA	系统时钟为 IRCH(24MHz)，其他时钟关闭，LDO 设置为默认值（高功率模式，输出电压为 1.61V），所有输出引脚无负载，所有数字输入引脚不浮动，所有外设关闭，CPU 执行NOP 指令
		VDD=3.3V		3.44			
		VDD=5V		3.47			
	Iop2	VDD=2.0V		80.1		uA	系统时钟为 IRCL(131kHZ)，其他时钟关闭，LDO 设置为低功率模式，输出电压为 1.61V，所有输出引脚无负载，所有数字输入引脚不浮动，所有外设关闭，CPU 执行NOP 指令
		VDD=3.3V		82.1			
		VDD=5V		83.3			
STOP 模式电流	I_{stp}	VDD=2.0V		5.9		uA	所有时钟关闭，所有输出引脚无负载，所有数字输入引脚不浮动，所有外设关闭，LDO 设置为低功率模式，Flash 进入睡眠模式，CPU 进入 STOP 模式。
		VDD=3.3V		6.1			
		VDD=5V		6.2			
IDLE 模式电流	I_{idl1}	VDD=2.0V		1.00		mA	系统时钟设为 IRCH（24MHz），其他时钟关闭，所有输出引脚无负载，所有数字输入引脚不浮动，所有外设关闭，LDO 设置为低功率模式，Flash 进入睡眠模式，CPU 进入IDLE 模式。
		VDD=3.3V		1.10			
		VDD=5V		1.12			
	I_{idl2}	VDD=2.0V		12.8		uA	系统时钟设为 IRCL（131KHz），其他时钟关闭，所有输出引脚无负载，所有数字输入引脚不浮动，所有外设关闭，LDO 设置为低功率模式，CPU 进入IDLE 模式。
		VDD=3.3V		13.2			
		VDD=5V		13.4			
IO 端口输入高电压（斯密特模式开启）	Vhi1	VDD=2.0V	1.00	-	2.0	V	-
		VDD=3.3V	1.75		3.3		
		VDD=5V	2.60		5		
IO 端口输入高	Vhi2	VDD=2.0V		0.5*VDD	VDD	V	-

电压（斯密特模式关闭）		VDD=3.3V VDD=5V					
IO 端口输入低电压（斯密特模式开启）	Vlo1	VDD=2.0V VDD=3.3V VDD=5V	0 0 0	- - -	0.65 1.18 2.0	V	-
IO 端口输入低电压（斯密特模式关闭）	Vlo2	VDD=2.0V VDD=3.3V VDD=5V	0	0.5*VDD		V	-
IO 端口推电流	Ipu	VDD=3.3V VDD=5V	- -	7 9	- -	mA	IO 设为推挽输出模式，驱动能力设为最大，Vol=VDD - 0.3V
IO 端口灌电流	I_{ol}	VDD=3.3V VDD=5V	- -	12 17	- -	mA	IO 设为推挽输出模式，驱动能力设为最大，Vol=GND + 0.3V
P0.0、 P0.1、P1.0~P1.5 强灌电流	I_{si}	VDD=3.3V VDD=5V	- -	72 95	- -	mA	IO 设为推挽输出模式，驱动能力设为最大，SINK_EN 设置为 1，Vol=GND+0.3V
IO 端口强下拉电阻	Rd1	VDD=2.0~5.5V		15		KΩ	-
IO 端口弱下拉电阻	Rd2	VDD=2.0~5.5V	-	45	-	KΩ	-
IO 端口强上拉电阻	Ru1	VDD=2.0~5.5V	-	10	-	KΩ	-
IO 端口弱上拉电阻	Ru2	VDD=2.0~5.5V		45		KΩ	

备注： 最小值的数据测量条件：VDD=2.0V, TA=25℃，除非另有说明。典型值的数据测量条件：VDD=3.3V, TA=25℃，除非另有说明。最大值的数据测量条件：VDD=5.5V, TA=25℃，除非另有说明。

28.3 交流电气特性

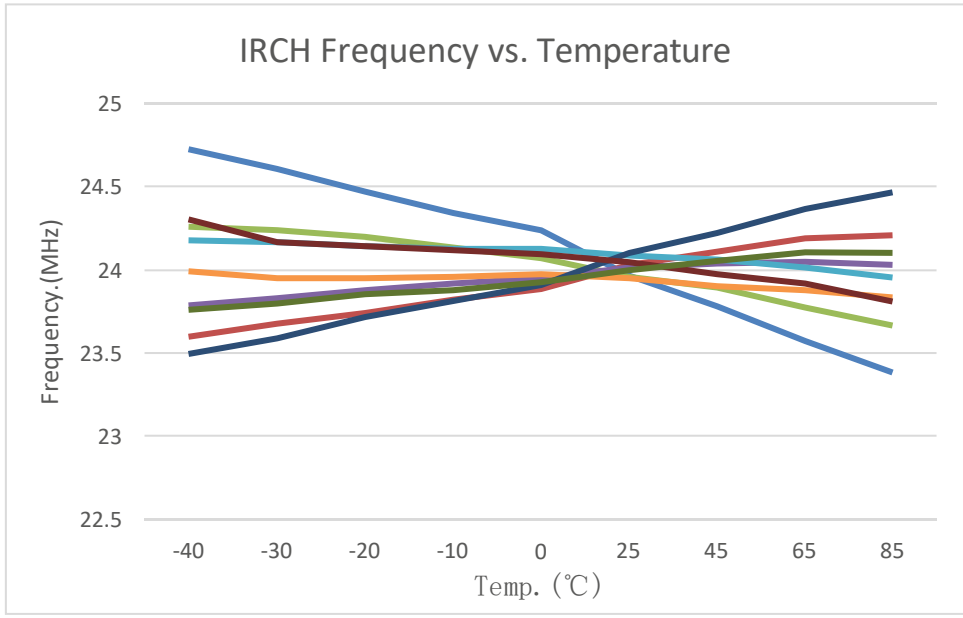
交流电气特性（VDD=2.0-5.5V, TA=25℃，除非其它说明）

芯片参数	符号	最小值	典型值	最大值	单位	条件
内部低速时钟（IRCL）起振时间	Trc1	-	50	-	us	IRCL 频率为 131K
内部高速时钟（IRCH）起振时间	Trc2	-	10	-	us	IRCH 频率为 24MHz
复位脉冲时间	Trst	-	0.5	-	us	

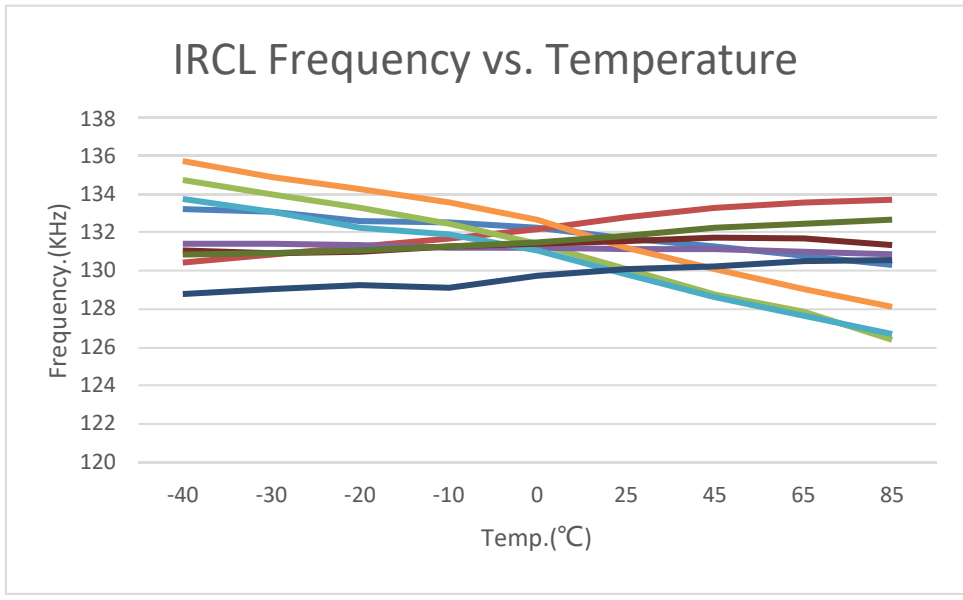
备注：VDD=3.3V,TA=25℃,内部高速时钟出厂频率为 24MHz，精度为±1%.

28.4 内部高速 RC 温度特性

IRCH 温度特性



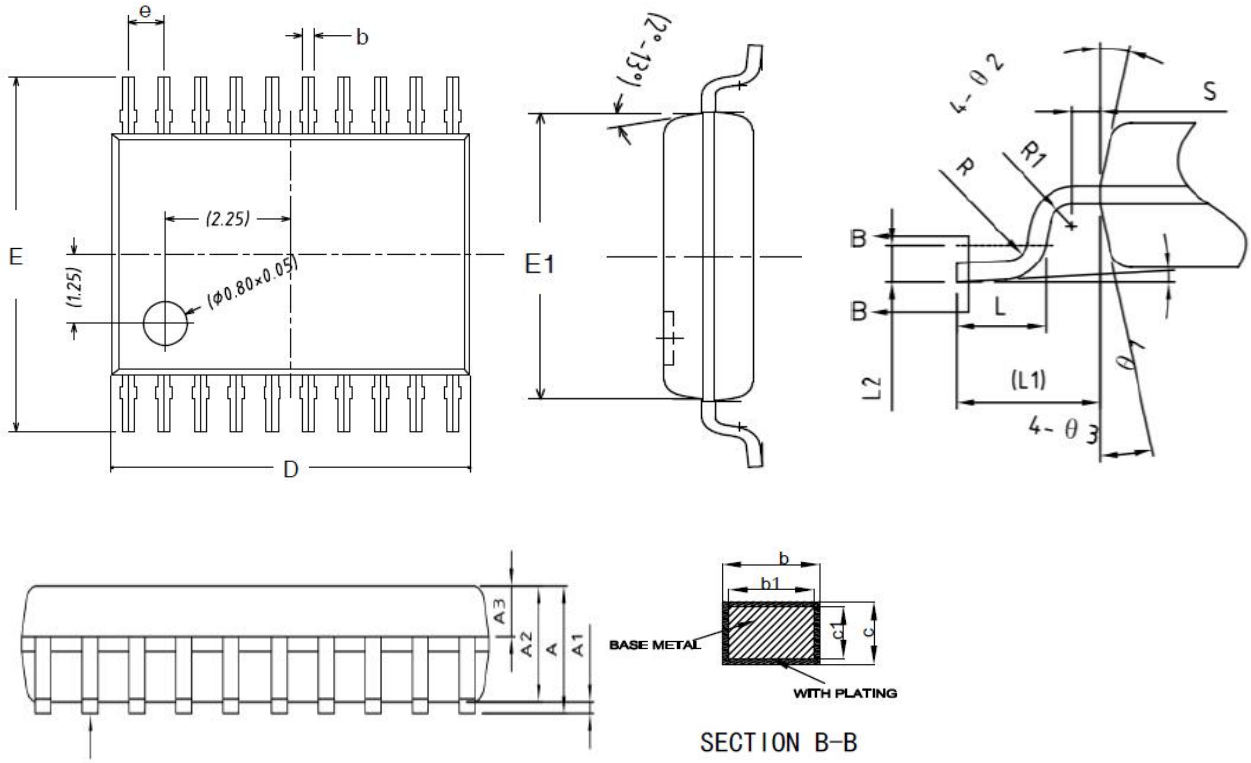
IRCL 温度特性



备注：以上图形数据为随机抽取的部分芯片实测数据，仅供参考。

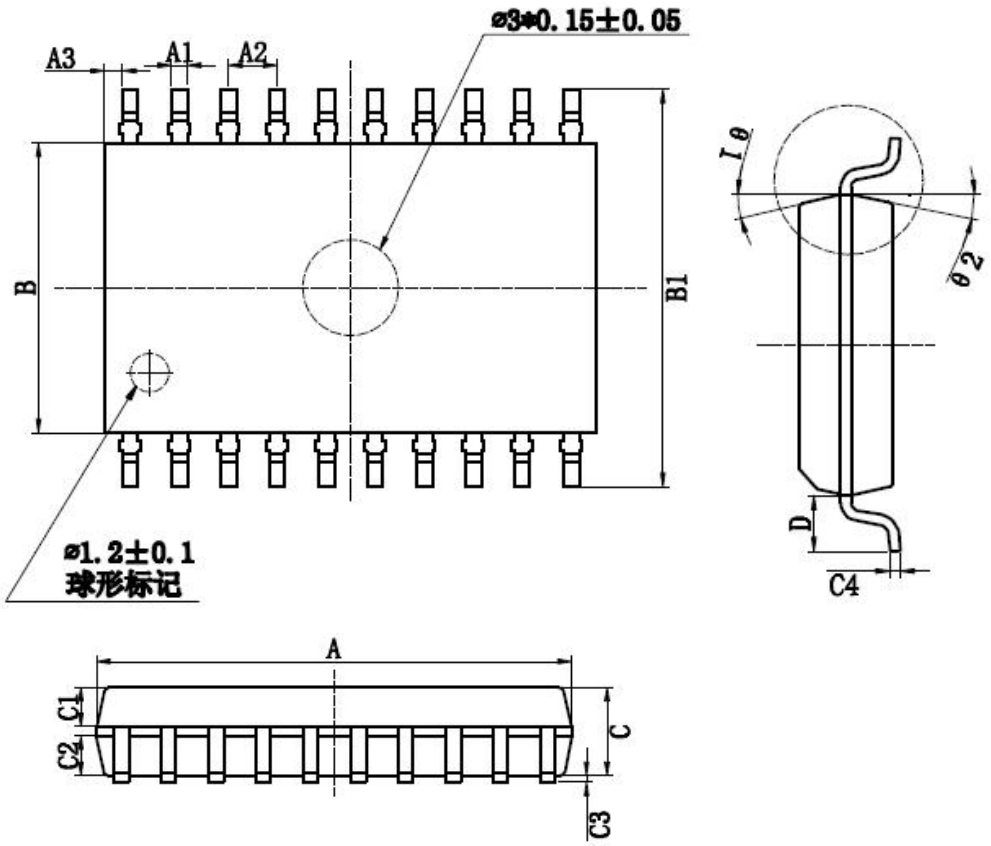
29 封装类型

封装形式：TSSOP20



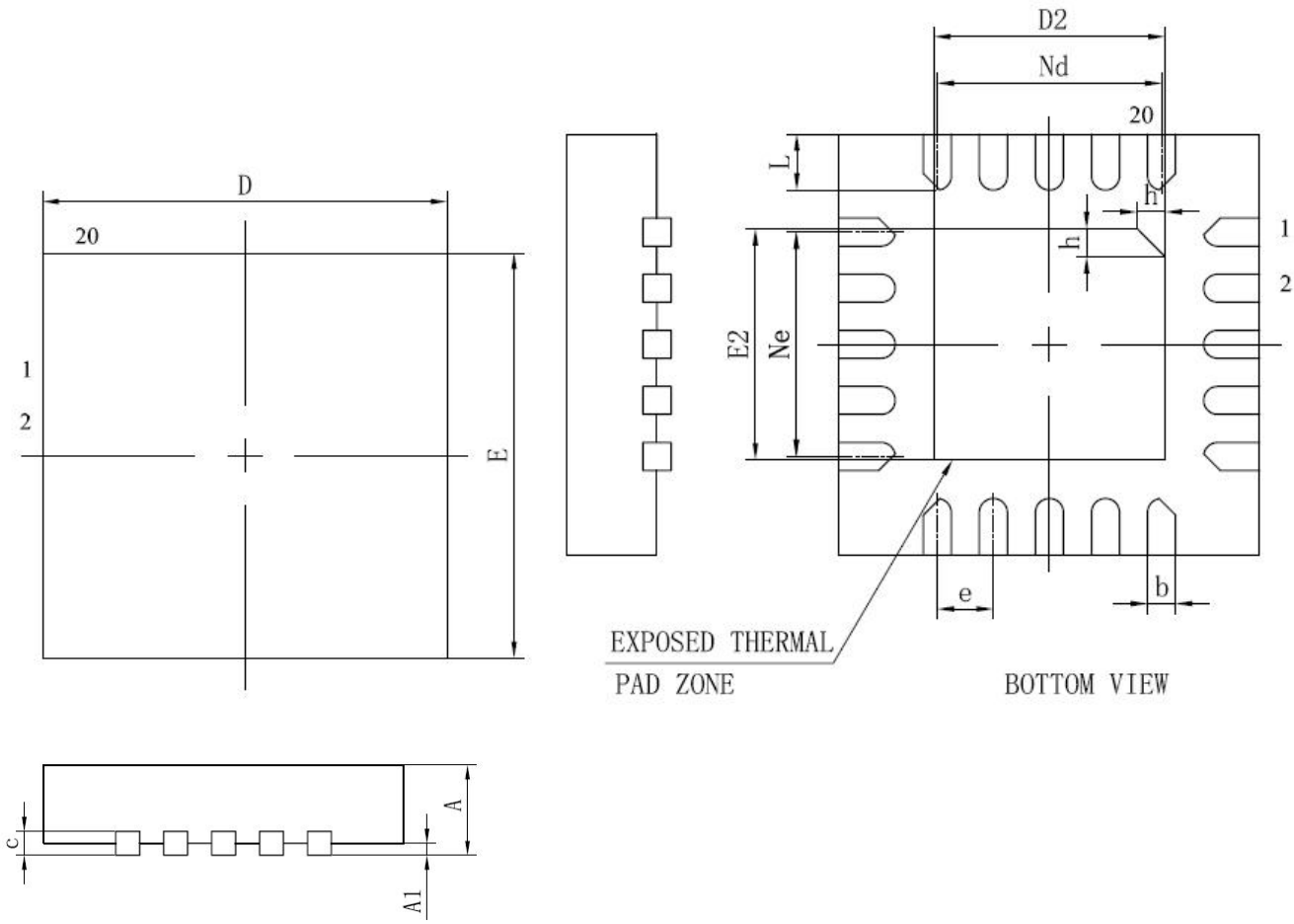
序号	最小值(mm)	标准值(mm)	最大值(mm)
A	1.0	---	1.1
A1	0.05	---	0.15
A2	---	---	0.95
A3	0.39	---	0.40
b	0.20	0.22	0.24
c	0.10	---	0.19
c1	0.10		0.15
D	6.40	6.45	6.50
E	6.25	6.40	6.55
E1		4.35	4.40
L	0.50	0.60	0.70
e	0.55	0.65	0.75
L2	0.25BSC		
R	0.09		
L1	1.0REF		

封装形式： **SOP20**



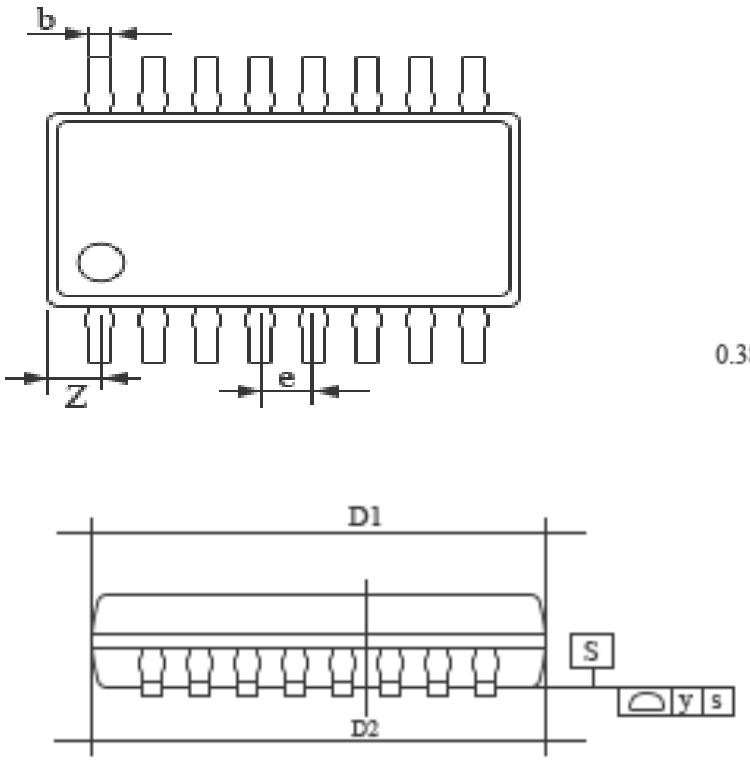
序号	最小值(mm)	标准值(mm)	最大值(mm)
A	12.65	12.70	12.80
A1	0.381	0.40	0.431
A2	1.24	1.27	1.30
A3	0.45	0.455	0.46
B	7.40	7.50	7.60
B1	10.206	10.30	10.406
C	2.18	2.23	2.28
C1	0.938	1.0	1.038
C2	0.938	1.0	1.038
C3	0.145	0.175	0.205
D	1.353	1.40	1.453
C4	0.246	0.25	0.262

封装形式：QFN20(3X3MM)



序号	最小值(mm)	标准值(mm)	最大值(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

封装形式 (SOP16)



序号	最小值(mm)	标准值(mm)	最大值(mm)
A	1.500	1.600	1.700
A2	1.400	1.450	1.500
b	0.356	0.406	0.456
D1	9.70	9.90	10.10
D2	9.75	9.95	10.15
E	5.90	6.000	6.100
E1	3.800	3.900	4.000
E2	3.850	3.950	4.050
e	————	1.27	————
Z	————	0.505	————

30 附录

附录 1 指令集速查表

指令	描述	说明	周期
数据传送指令			
MOV A,Rn	寄存器内容送入累加器	(A) ← (Rn)	1
MOV A,direct	直接地址单元中的数据送入累加器	(A) ← (direct)	1
MOV A,@Ri	间接 RAM 中的数据送入累加器	(A) ← ((Ri))	1
MOV A,#data8	8 位立即数送入累加器	(A) ← #data	1
MOV Rn,A	累加器内容送入寄存器	(Rn) ← (A)	1
MOV Rn,direct	直接地址单元中的数据送入寄存器	(Rn) ← (direct)	2
MOV Rn,#data8	8 位立即数送入寄存器	(Rn) ← #data	1
MOV direct,A	累加器内容送入直接地址单元	(direct) ← (A)	1
MOV direct,Rn	寄存器内容送入直接地址单元	(direct) ← (Rn)	2
MOV direct,direct	直接地址单元中的数据送入直接地址单元	(direct) ← (direct)	2
MOV direct,@Ri	间接 RAM 中的数据送入直接地址单元	(direct) ← ((Ri))	2
MOV direct,#data8	8 位立即数送入直接地址单元	(direct) ← #data	2
MOV @Ri,A	累加器内容送入间接 RAM 单元	((Ri)) ← (A)	1
MOV @Ri,direct	直接地址单元中的数据送入间接 RAM 单元	((Ri)) ← (direct)	2
MOV @Ri,#data8	8 位立即数送入间接 RAM 单元	((Ri)) ← #data	1
MOV DPTR,#data16	16 位立即数地址送入地址寄存器	(DPTR) ← #data116	2
MOV A,@A+DPTR	以 DPTR 为基地址变址寻址单元中的数据送入累加器	(A) ← ((A)) + (DPTR)	2
MOV A,@A+PC	以 PC 为基地址变址寻址单元中的数据送入累加器	(PC) ← (PC) + 1 (A) ← ((A) + (PC))	2
MOVX A,@Ri	外部RAM(8 位地址)送入累加器	(A) ← ((Ri))	2
MOVX A,@DPTR	外部RAM(16 位地址)送入累加器	(A) ← ((DPTR))	2
MOVX @Ri,A	累加器送入外部RAM(8 位地址)	((Ri)) ← (A)	2
MOVX @DPTR,A	累加器送入外部RAM(16 位地址)	(DPTR) ← (A)	2
PUSH direct	直接地址单元中的数据压入堆栈	(SP) ← (SP) + 1 ((SP)) ← (direct)	2
POP DIRECT	堆栈中的数据弹出到直接地址单元	(direct) ← ((SP)) (SP) ← (SP) - 1	2
XCH A,Rn	寄存器与累加器交换	(A) ↔ (Rn)	1
XCH A,direct	直接地址单元与累加器交换	(A) ↔ (direct)	1
XCH A,@Ri	间接 RAM 与累加器交换	(A) ↔ ((Ri))	1
XCHD A,@Ri	间接 RAM 与累加器进行低半字节交换	(A.3,...,A.0) ↔ ((Ri).3,...,(Ri).0)	1
SWAP A	累加器半字节交换	(A.3,...,A.0) ↔ (A.7,...,A.4)	1
算术操作类指令			
ADD A, Rn	寄存器内容加到累加器	(A) ← (A) + (Rn)	1

ADD A, direct	直接地址单元加到累加器	(A) ← (A) + (direct)	1
ADD A, @Ri	间接 RAM 内容加到累加器	(A) ← (A) + ((Ri))	1
ADD A, #data8	8 位立即数加到累加器	(A) ← (A) + #data	1
ADDC A, Rn	寄存器内容带进位加到累加器	(A) ← (A) + (C) + (Rn)	1
ADDC A, direct	直接地址单元带进位加到累加器	(A) ← (A) + (C) + (direct)	1
ADDC A, @Ri	间接 RAM 内容带进位加到累加器	(A) ← (A) + (C) + ((Ri))	1
ADDC A, #data8	8 位立即数带进位加到累加器	(A) ← (A) + (C) + #data	1
SUBB A, Rn	累加器带借位减寄存器内容	(A) ← (A) - (C) - (Rn)	1
SUBB A, direct	累加器带借位减直接地址单元	(A) ← (A) - (C) - (direct)	1
SUBB A, @Ri	累加器带借位减间接 RAM 内容	(A) ← (A) - (C) - ((Ri))	1
SUBB A, #data8	累加器带借位减 8 位立即数	(A) ← (A) - (C) - #data	1
INC A	累加器加 1	(A) ← (A) + 1	1
INC Rn	寄存器加 1	(Rn) ← (Rn) + 1	1
INC direct	直接地址单元内容加 1	(direct) ← (direct) + 1	1
INC @Ri	间接 RAM 内容加 1	((Ri)) ← ((Ri)) + 1	1
INC DPTR	DPTR 加 1	(DPTR) ← (DPTR) + 1	2
DEC A	累加器减 1	(A) ← (A) – 1	1
DEC Rn	寄存器减 1	(Rn) ← (Rn) - 1	1
DEC direct	直接地址单元内容减 1	(direct) ← (direct) - 1	1
DEC @Ri	间接 RAM 内容减 1	((Ri)) ← ((Ri)) - 1	1
MUL AB	A 乘以B	temp16 ← (A) X (B) (A)←(temp.7,temp.6,...,temp.0) (B)←(temp.15,temp.14,...,temp.8)	4
DIV AB	A 除以B	QUO ← (A) / (B) .....REM (A) ← QUO (B) ← REM	4

DA A	累加器进行十进制转换	IF (A.3,...,A.0) > 9 || AC = 1
		THEN
		temp16 ← (A) + 0x06
		(A) ← (temp.7,...,temp.0)
		
		IF (temp16) > 0xFF
		THEN
		CY ← 1
			1

		IF (A.7,...,A.4) > 9
		|| CY = 1
		THEN
		temp16 ← (A) + 0x60
		(A) ← (temp.7,...,temp.0)
		
		IF (temp16) > 0xFF
		THEN
		CY ← 1	

逻辑操作类指令			
ANL A, Rn	累加器与寄存器相“与”	(A) ← (A) & (Rn)	1
ANL A, direct	累加器与直接地址单元相“与”	(A) ← (A) & (direct)	1
ANL A, @Ri	累加器与间接 RAM 内容相“与”	(A) ← (A) & ((Ri))	1
ANL A, #data8	累加器与 8 位立即数相“与”	(A) ← (A) & #data	1
ANL direct, A	直接地址单元与累加器相“与	(direct) ← (direct) & (A)	1
ANL direct, #data8	直接地址单元与 8 位立即数相“与”	(direct) ← (direct) & #data	2
ORL A, Rn	累加器与寄存器相“或”	(A) ← (A) | (Rn)	1
ORL A, direct	累加器与直接地址单元相“或”	(A) ← (A) | (direct)	1
ORL A, @Ri	累加器与间接 RAM 内容相“或”	(A) ← (A) | ((Ri))	1
ORL A, #data8	累加器与 8 位立即数相“或”	(A) ← (A) | #data	1
ORL direct, A	直接地址单元与累加器相“或”	(direct) ← (direct) | (A)	1
ORL direct, #data8	直接地址单元与 8 位立即数相“或”	(direct) ← (direct) | #data	2
XRL A, Rn	累加器与寄存器相“异或”	(A) ← (A) ^ (Rn)	1
XRL A, direct	累加器与直接地址单元相“异或”	(A) ← (A) ^ (direct)	1
XRL A, @Ri	累加器与间接 RAM 内容相“异或”	(A) ← (A) ^ ((Ri))	1
XRL A, #data8	累加器与 8 位立即数相“异或”	(A) ← (A) ^ #data	1
XRL direct, A	直接地址单元与累加器相“异或”	(direct) ← (direct) ^ (A)	1
XRL direct, #data8	直接地址单元与 8 位立即数相“异或”	(direct) ← (direct) ^ #data	2
CLR A	累加器清 0	(A) ← 0	1
CPL A	累加器求反	(A) ← /(A)	1
RL A	累加器循环左移	(A) ← (A.6,A.5,...,A.0,A.7)	1
RLC A	累加器带进位循环左移	C ← A.7 (A) ← (A.6,A.5,...,A.0,C)	1
RR A	累加器循环右移	(A) ← (A.0,A.7,...,A.2,A.1)	1
RRC A	累加器带进位循环右移	C ← A.0 (A) ← (C,A.7,...,A.2,A.1)	1
控制转移类指令			
ACALL addr11	绝对短调用子程序	(PC) ← (PC) + 2 (SP) ← (SP) + 1	2

		((SP)) ← (PC7-0) (SP) ← (SP) + 1 ((SP)) ← (PC15-8) (PC10-0) ← page address	
LACLL addr16	长调用子程序	(PC) ← (PC) + 3 (SP) ← (SP) + 1 ((SP)) ← (PC7-0) ((SP)) ← (PC15-8) (PC) ←addr15-0	2
RET	子程序返回	(PC15-8) ← ((SP)) (SP) ← (SP) - 1 (PC7-0) ← ((SP)) (SP) ← (SP) - 1	2
RETI	中断返回	(PC15-8) ← ((SP)) (SP) ← (SP) - 1 (PC7-0) ← ((SP)) (SP) ← (SP) - 1	2
AJMP addr11	绝对短转移	(PC) ← (PC) + 2 (PC10-0) ← page address	2
LJMP addr16	长转移	(PC) ← (PC) + 3 (SP) ← (SP) + 1 ((SP)) ← (PC7-0) (SP) ← (SP) + 1 ((SP)) ← (PC15-8) (PC10-0) ←addr15-0	2
SJMP rel	相对转移	(PC) ← (PC) + 2 (PC) ← (PC) + rel	2
JMP @A+DPTR	相对于 DPTR 的间接转移	(PC) ← (A) + (DPTR)	2
JZ rel	累加器为零转移	(PC) ← (PC) + 2 IF (A) = 0 THEN (PC) ← (PC) + rel	2
JNZ rel	累加器非零转移	(PC) ← (PC) + 2 IF (A) <> 0 THEN (PC) ← (PC) + rel	2

CJNE A, direct, rel	累加器与直接地址单元比较，不等则转移	(PC) ← (PC) + 3
		IF (A) <> (direct)
		THEN
		(PC) ← (PC) + relative offset
		IF (A) < (direct)
		THEN
		(C) ← 1
		ELSE
		(C) ← 0
			2

CJNE A, #data8, rel	累加器与 8 位立即数比较，不等则转移	(PC) ← (PC) + 3
		IF (A) <> data
		THEN
		(PC) ← (PC) + relative offset
		IF (A) < data
		THEN
			2

		(C) ← 1 ELSE (C) ← 0	

CJNE Rn, #data8, rel	寄存器与 8 位立即数比较，不等则转移	(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	间接 RAM 单元，不等则转移	(PC) ← (PC) + 3
		IF ((Ri)) <> data
		THEN
		(PC) ← (PC) +
		relative offset
		IF ((Ri)) < data
		THEN
		(C) ← 1
		ELSE
		(C) ← 0
			2

DJNZ Rn, rel	寄存器减 1，非零转移	(PC) ← (PC) + 2 (Rn) ← (Rn) - 1 IF (Rn) <> 0 THEN (PC) ← (PC) + rel	2
DJNZ direct, rel	直接地址单元减 1，非零转移	(PC) ← (PC) + 2 (direct) ← (direct) - 1 IF (direct) <> 0 THEN (PC) ← (PC) + rel	2
NOP	空操作	(PC) ← (PC) + 1	1
布尔变量操作类指令			
CLR C	清进位位	(C) ← 0	1
CLR bit	清直接地址位	(bit) ← 0	1
SETB C	置进位位	(C) ← 1	1
SETB bit	置直接地址位	(bit) ← 1	1
CPL C	进位位求反	(C) ← /(C)	1
CPL bit	直接地址位求反	(bit) ← /(bit)	1
ANL C, bit	进位位和直接地址位相“与”	(C) ← (C) & (bit)	2
ANL C, /bit	进位位和直接地址位的反码相“与”	(C) ← (C) & /(bit)	2
ORL C, bit	进位位和直接地址位相“或”	(C) ← (C) | (bit)	2
ORL C, /bit	进位位和直接地址位的反码相“或”	(C) ← (C) | /(bit)	2
MOV C, bit	直接地址位送入进位位	(C) ← (bit)	1
MOV bit, C	进位位送入直接地址位	(bit) ← (C)	2
JC rel	进位位为 1 则转移(CY=O 不转移，=1 转移)	(PC) ← (PC) + 2 IF (C) = 1 THEN (PC) ← (PC) + rel	2
JNC rel	进位位为 0 则转移	(PC) ← (PC) + 2 IF (C) = 0 THEN (PC) ← (PC) + rel	2
JB bit, rel	直接地址位为 1 则转移	(PC) ← (PC) + 3	2

		IF (bit) = 1 THEN (PC) ← (PC) + rel	
JNB bit, rel	直接地址位为 0 则转移	(PC) ← (PC) + 3 IF (bit) = 0 THEN (PC) ← (PC) + rel	2
JBC bit, rel	直接地址位为 1 则转移，该位清零	(PC) ← (PC) + 3 IF (bit) = 1 THEN (bit) ← 0 (PC) ← (PC) + rel	2
伪指令			
ORG	设置程序起始地址		
END	标志源代码结束		
EQU	定义常数		
SET	定义整型数		
DATA	给数据地址定值		
BYTE	给字节类型符号定值		
WROD	给字类型符号定值		
BIT	给位地址取名		
ALTNAME	用自定义名取代保留字		
DB	给一块连续的存储区装载字节型数据		
DW	给一块连续的存储区装载字型数据		
DS	预留一个连续的存储区或装入指定字节		
INCLUDE	将一个源文件插入程序中		
TITLE	列表文件中加入标题行		
NOLIST	汇编时不产生列表文件		
NOCODE	条件汇编时，条件为假的不产生清单		