JZ8FC5 系列 MCU

中文用户手册

Built - in 16 Bit PWM / Touch Key / 1T 8051 8K Flash MCU

目录

1 概述

JZ8FC508T 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 8K Flash 程序存储器，可多次重复编程的特性，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 Touch Key、16 Bit PWM、5Bit DAC、UART、I^2C、RGB_LED 级联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家用照明、家用音响触摸控制、家电触摸控制、蓝牙音箱、台灯和浴室镜灯、景观及氛围灯带产品中。

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: 8K 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，512 字节外部 RAM
- **工作电压**
  - 工作电压：1.8 - 5.5V 宽电压工作范围
- **时钟系统**
  - 内置低速 RC 振荡器：131KHz
  - 内置高速 RC 振荡器：16MHz，精度为±1%（3.3V@25℃）
- **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**
  - 7 个有效中断源
  - 两级中断优先级，支持中断嵌套
  - 5 个外部中断源，3 个外部中断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 14 个 GPIO 口
  - 支持推挽、开漏、强上拉、弱上拉、强下拉、弱下拉、高阻模式
  - 推挽模式下可设置不同驱动强度和翻转速度

- **触摸按键**（Touch Key）
  - 内置触摸感应控制器
  - 最大支持 13 触摸通道
  - 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
- **PWM**
  - 支持 6 通道PWM，在 16 位范围内可任意配置周期和占空比
  - 支持可直接输出内部时钟功能
  - 支持PWM 中断
  - 支持 2 路级联LED 驱动，扫描频率大于 400Hz/S, 数据发送速度 800Kbps
  - 支持直接控制WS2812 或类似的驱动芯片，符合单色或七彩LED 灯带产品的需求。
- **低电压检测**（LVD）
  - 可配置电压检测范围 1.7 - 4.8V
  - 可设置低电压复位或中断
  - 可选择检测VDD 电压或引脚输入电压
- DAC **功能**
  - 支持两路 5Bit DAC 输出，每路可配置 32 档输出电压
- **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**（UART）
  - 支持 1 个 UART 接口
  - 支持 1 字节接收缓存
- 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
  - I2C 可设置数字滤波，增强 I2C 抗干扰性能
- **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<6uA
  - IDLE 模式，电流<13uA
  - 低速运行模式，电流<20uA
- **封装类型**：SOP16/MSOP10/DFN8/SOP8

3 芯片型号功能介绍

表 3-1　JZ8FC508T 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	GPIO 数量	UART 数量	I²C	16 bit PWM 通道数量	触摸按键数量	5 位 D/A	级联 LED 驱动	ISP	片上仿真功能	工作电压	封装形式
JZ8FC508TS1	8K	512	√	√	6	1	√	3	5	1	1	√	√	1.8-5.5	SOP8
JZ8FC508TM2	8K	512	√	√	8	1	√	5	7	1	2	√	√	1.8-5.5	MSOP10
JZ8FC508TN1	8K	512	√	√	6	1	√	3	5	1	1	√	√	1.8-5.5	DFN8L 2X2MM
JZ8FC508TS3	8K	512	√	√	14	1	√	6	13	2	2	√	√	1.8-5.5	SOP16

4 系统框图

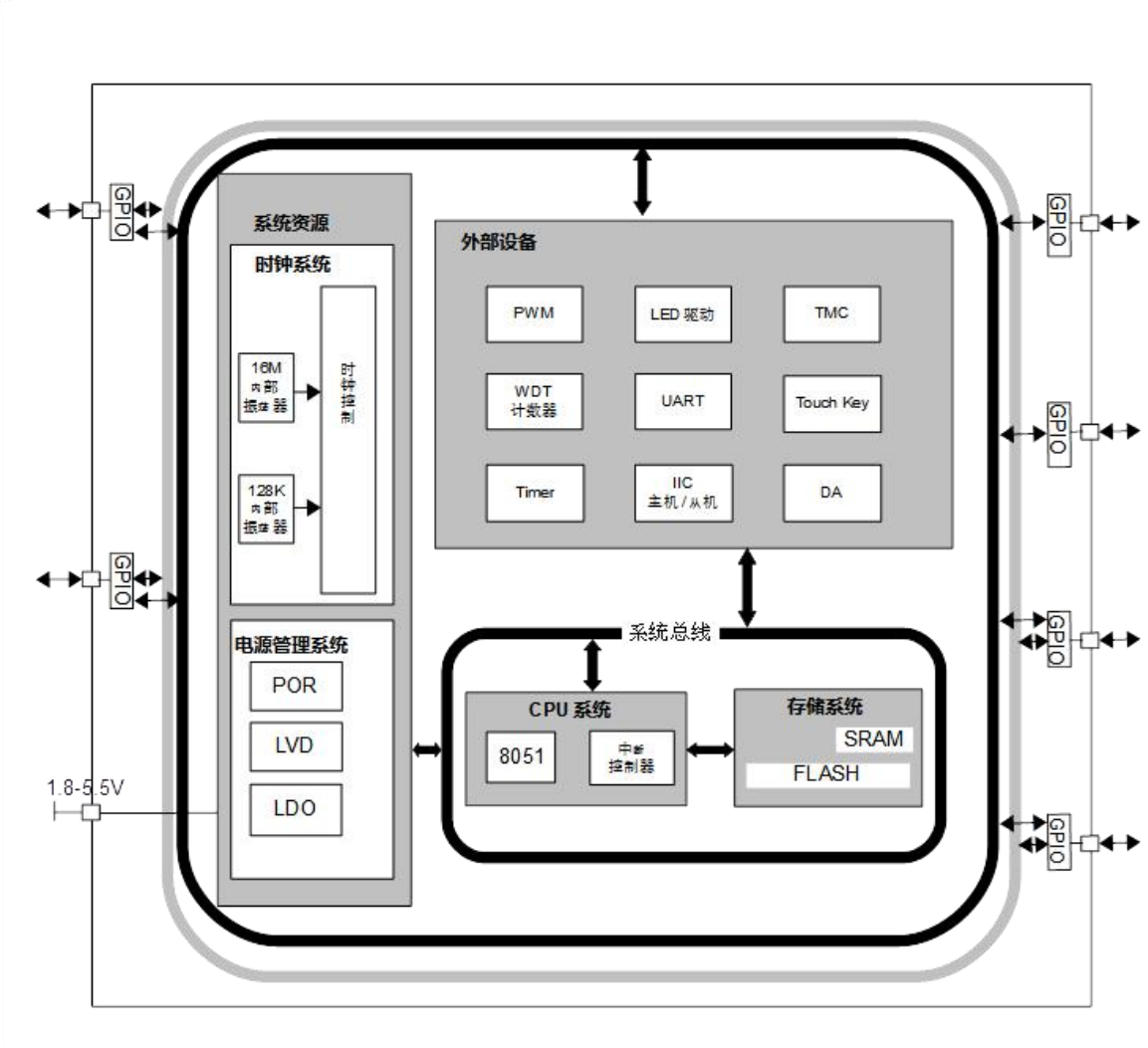


图 4-1-1 芯片框图

5 引脚封装及其描述

5.1 封装定义

型号：JZ8FC508TS1

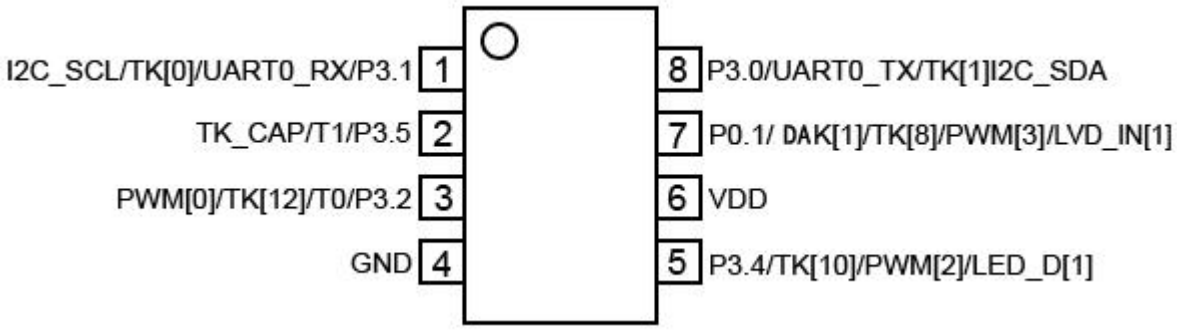


图 5-1-1　SOP8 管脚定义图

型号：JZ8FC508TM2

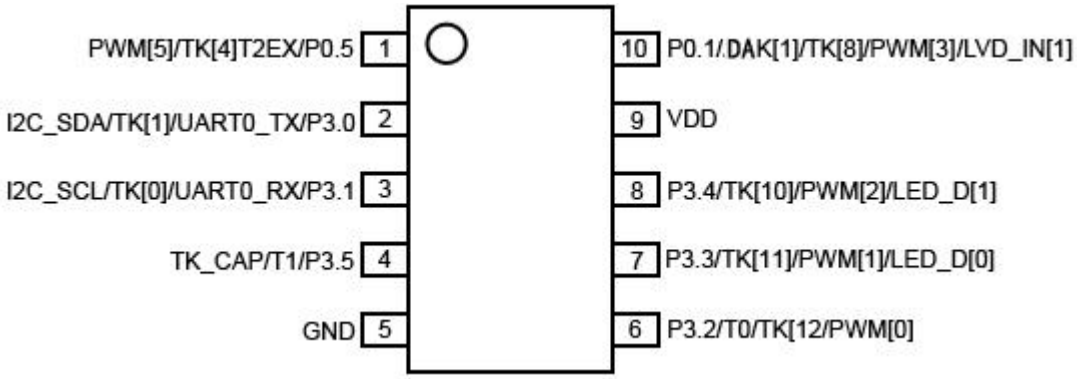


图 5-1-2MSOP10 管脚定义图

型号：**JZ8FC508TN1**

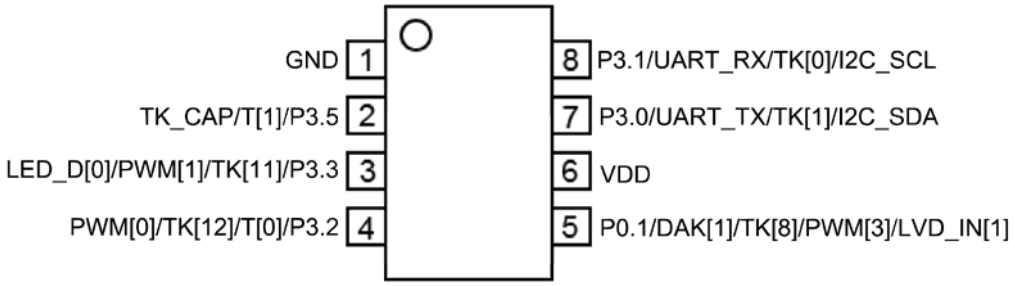


图 **5-1-3 DFN8L(2x2mm)**管脚定义图

型号：**JZ8FC508TS3**

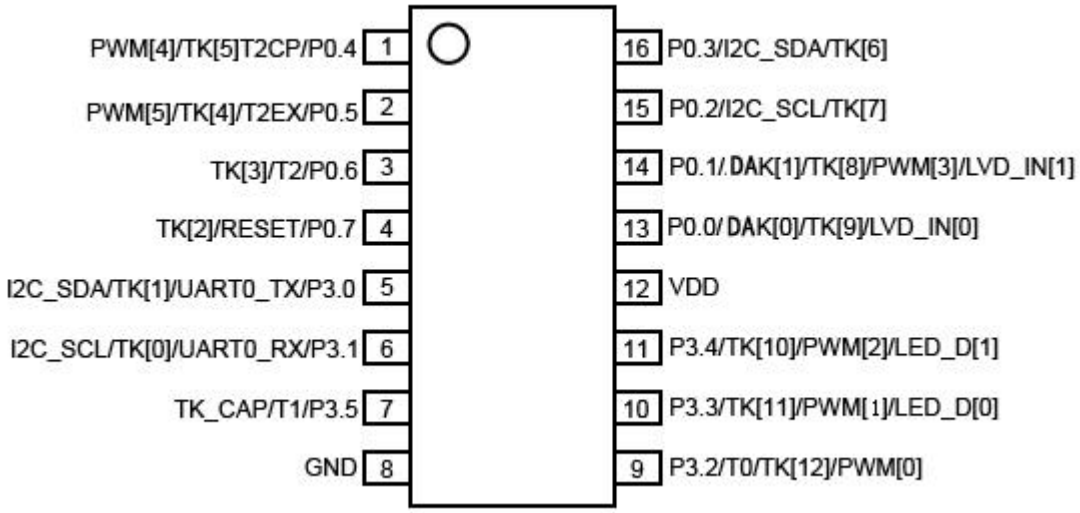


图 **5-1-4SOP16** 管脚定义图

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
SOP16	MSOP10	DFN8	SOP8			
1	-	-	-	P0.4/T2CP/TK[5]/PWM[4]	通用双向 I/O 口 T2CP 信号输入 触摸按键模拟通道输入 PWM 信号输出	通用双向 I/O 口
2	1	--	-	P0.5/T2EX/TK[4]/PWM[5]	通用双向 I/O 口 T2EX 信号输入 触摸按键模拟通道输入 PWM 信号输出	通用双向 I/O 口
3	-	--	-	P0.6/T2/TK[3]	通用双向 I/O 口 T2 信号输入 触摸按键模拟通道输入	通用双向 I/O 口
4	-	--	-	P0.7/RESET/TK[2]	通用双向 I/O 口 硬件复位引脚 触摸按键模拟通道输入	硬件复位引脚
5	2	7	8	P3.0/I2C_SDA/TK[1]/UART0_TX	通用双向 I/O 口 I^2C 数据传输口 触摸按键模拟通道输入 UART0_TX 传输口	I^2C 数据传输口
6	3	8	1	P3.1/I2C_SCL/TK[0]/UART0_RX	通用双向 I/O 口 I^2C 时钟传输口 触摸按键模拟通道输入 UART0_RX 传输口	I^2C 时钟传输口
7	4	2	2	P3.5/T1/TKCAP	通用双向 I/O 口 T1 信号输入 触摸外部电容输入口	通用双向 IO 口
8	5	1	4	VSS	电源地引脚	电源地引脚
9	6	4	3	P3.2/T0/TK[12]/PWM[0]	通用双向 I/O 口 T0 信号输入 触摸按键模拟通道输入 PWM 信号输出	通用双向 I/O 口
10	7	3	-	P3.3/TK[11]/PWM[1]	通用双向 I/O 口 触摸按键模拟通道输入 PWM 信号输出	通用双向 I/O 口
11	8	--	5	P3.4/TK[10]/PWM[2]	通用双向 I/O 口 触摸按键模拟通道输入 PWM 信号输出	通用双向 IO 口
12	9	6	6	VDD	芯片供电管脚	芯片供电管脚
13	-	--	-	P0.0/DAK[0]/TK[9]/LVD_IN[0]	通用双向 I/O 口	通用双向 IO 口

					DA 模拟输出口 触摸按键模拟通道输入 低电压检测输入	
14	10	5	7	P0.1/DAK[1]/TK[8]/PWM[3]/LVD_IN[1]	通用双向 I/O 口 DA 模拟输出口 触摸按键模拟通道输入 PWM 信号输出 低电压检测输入	通用双向 IO 口
15	-	-	-	P0.2/I2C_SCL/TK[7]	通用双向 I/O 口 I^2C 时钟传输口 触摸按键模拟通道输入	通用双向 IO 口
16	-	--	-	P0.3/I2C_SDA/TK[6]	通用双向 I/O 口 I^2C 数据传输口 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 15-2-3 和表 15-2-5

6 中央处理器（CPU）

6.1 CPU 简介

JZ8FC508T 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）

JZ8FC508T 系列芯片提供了 256 字节内部 RAM 和 512 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 512 字节外部 RAM（地址：0000H ~ 01FFH）可通过MOVX 指令间接寻址。

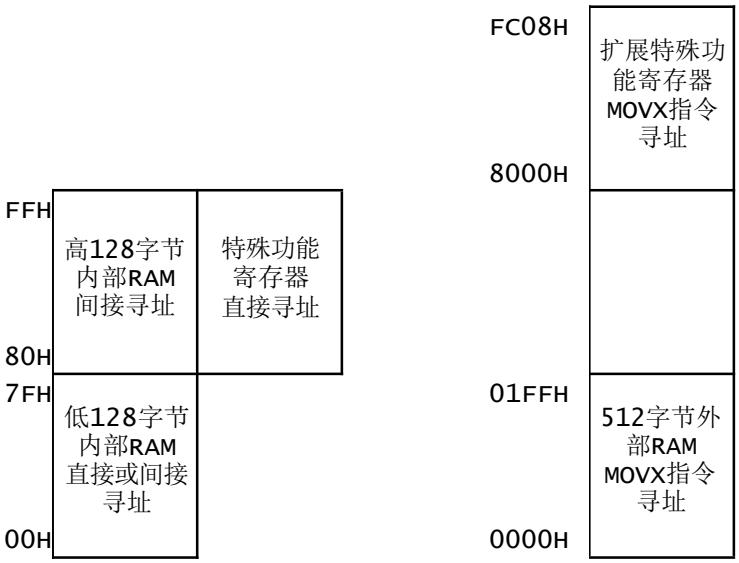


图 7-1-1　RAM 组织结构图

7.2 特殊功能寄存器（SFR）

JZ8FC508T 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	TKCON	TKCFG	TKMTS	TKIF	TK0CHS	TK1CHS	TK2CHS	TK3CHS
F0H	B	TK4CHS	TK5CHS	ATK0CL	ATK0CH	ATK1CL	ATK1CH	ATK2CL
E8H	LVDCON	ATK2CH	ATK3CL	ATK3CH	ATK4CL	ATK4CH	ATK5CL	ATK5CH
E0H	ACC	TK0MSL	TK0MSH	TK1MSL	TK1MSH	TK2MSL	TK2MSH	TK3MSL
D8H	UDCKS	TK3MSH	TK4MSL	TK4MSH	TK5MSL	TK5MSH	AK0CON	AK1CON

D0H	PSW	EP0CON	EP1CON	EP2CON	EPIF	TMCON	TMSNU	LEDAT0
C8H	T2CON	T2MOD	T2CL	T2CH	TL2	TH2	LEWTML	LEWTMH
C0H	I2CCON	I2CADR	I2CADM	I2CCCR	I2CDAT	I2CSTA	I2CFLG	LEDAT1
B8H	IP	PWM0CON	PWM1CON	PWM2CON	PWM3CON	PWM4CON	PWM5CON	LEFLG
B0H	P3	PWM0CKD	PWM1CKD	PWM2CKD	PWM3CKD	PWM4CKD	PWM5CKD	PWMIF
A8H	IE	PWM0DIVL	PWM0DIVH	PWM1DIVL	PWM1DIVH	PWM2DIVL	PWM2DIVH	PWM3DIVL
A0H	WDCON	WDFLG	WDVTHL	WDVTHH	PWM3DIVH	PWM4DIVL	PWM4DIVH	PWM5DIVL
98H	S0CON	S0BUF	PWM5DIVH	PWM0DUTL	PWM0DUTH	PWM1DUTL	PWM1DUTH	PWM2DUTL
90H	PWMEN	PWM2DUTH	PWM3DUTL	PWM3DUTH	PWM4DUTL	PWM4DUTH	PWM5DUTL	PWM5DUTH
88H	TCON	TMOD	TL0	TL1	TH0	TH1	IDLST	STPST
80H	P0	SP	DP0L	DP0H	DP1L	DP1H	PWCON	PCON

由于 SFR 地址空间有限，JZ8FC508T 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
8018H	P30F	P31F	P32F	P33F	P34F	P35F	-	-
8030H	CKCON	CKDIV	IHCFG	ILCFGL	ILCFGH	-	-	-
8050H	ATK0NL	ATK0NH	ATK1NL	ATK1NH	ATK2NL	ATK2NH	ATK3NL	ATK3NH
8058H	ATK4NL	ATK4NH	ATK5NL	ATK5NH	TKMAXF	TKMINF	-	-
8060H	LEDUTL	LEDUTH	-	-	-	-	-	-
8100H	SPMAX	I2CIOS	TKCKS	TKPWC	-	LVDCFG	TLEN0	TLDAT0
8108H	TLCON	TLFLG	TLCKS	TLCNTKL	TLCNTKH	TLCNTLL	TLCNTLH	TLDIV
8110H	TLCOMS	TLEN1	TLDAT1	-	-	-	-	-
8118H	-	-	-	-	-	-	-	-
8120H	P00C	P01C	P02C	P03C	P04C	P05C	P06C	P07C
8138H	P30C	P31C	P32C	P33C	P34C	P35C	-	-
FC00H	MECON	FSCMD	FSDAT	LOCK	PADRD	PTSL	PTSH	-

7.3 Flash 存储器

7.3.1 功能简介

Flash 存储器包含 8K 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 128 字节

- Flash 可以按功能划分为程序区和数据区，划分单位为 128 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　8K 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 读和写命令写入后 CMD 保持不变然后通过读写 FSDAT 完成。*

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 7-3-3-5 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	0	0	0	0	0	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 128 字节为单位进行划分，当 PADRD>0 时， 程序区的地址空间为: 0 ~ (PADRD×128 - 1), 数据区的地址空间为: (PADRD×128) ~ 1FFFH. *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 40H，PADRD 的设置值不能超过最大值。*

表 7-3-3-6 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

7.3.4 Flash 控制例程

◆ **Flash 划分程序区和数据区**

例如，8K 的Flash 空间划分最后 128 字节为数据空间，其余为程序空间，程序如下：

```
--------------------------------------------------------------------------------
PADRD = 63; //程序区空间地址为：0~0x1F7F,数据区空间地址为：0x1F80~0x1FFF

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

备注：以上设置数据区在 FLASH 中的物理地址是 0x1F80~0x1FFF，但是逻辑地址是 0x0000~0x007F，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*0x80)>>8); //设置扇区高位地址
PTSL = (unsigned char)(n*0x80);          //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n~(n+100)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)(n>>8); //设置数据首地址高 8 位
PTSL = (unsigned char)n;          //设置数据首地址低 8 位
FSCMD = 2;          //设置写命令
for(i=0;i<100;i++)
{
    FSDAT = 0xAA;     //连续写入数据
}
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。

◆ **数据空间读出数据**

例如，从数据空间地址为n~(n+100)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)(n>>8); //设置数据首地址高 8 位
PTSL = (unsigned char)n;          //设置数据首地址低 8 位
FSCMD = 1;   //设置读命令
for(i=0;i<100;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

512 字节的外部 RAM 可以映射为程序空间使用，映射地址为 2000H~21FFH，映射图如图 7-5-1 所示。用户可以下载程序到外部 RAM 空间，当程序运行时直接执行跳转指令跳到映射程序区执行。同样效果，也可把 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~01FFH）。映射程序区用来实现IAP 等功能特别方便。

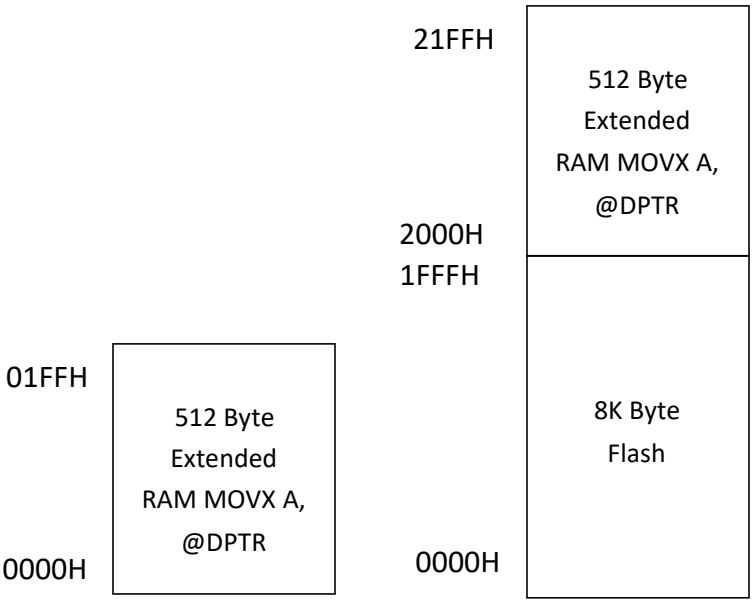


图 7-4-1 XRAM 地址映射图

8 中断系统

8.1 功能简介

JZ8FC508T 系列芯片有一个增强的中断控制系统，共有 7 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

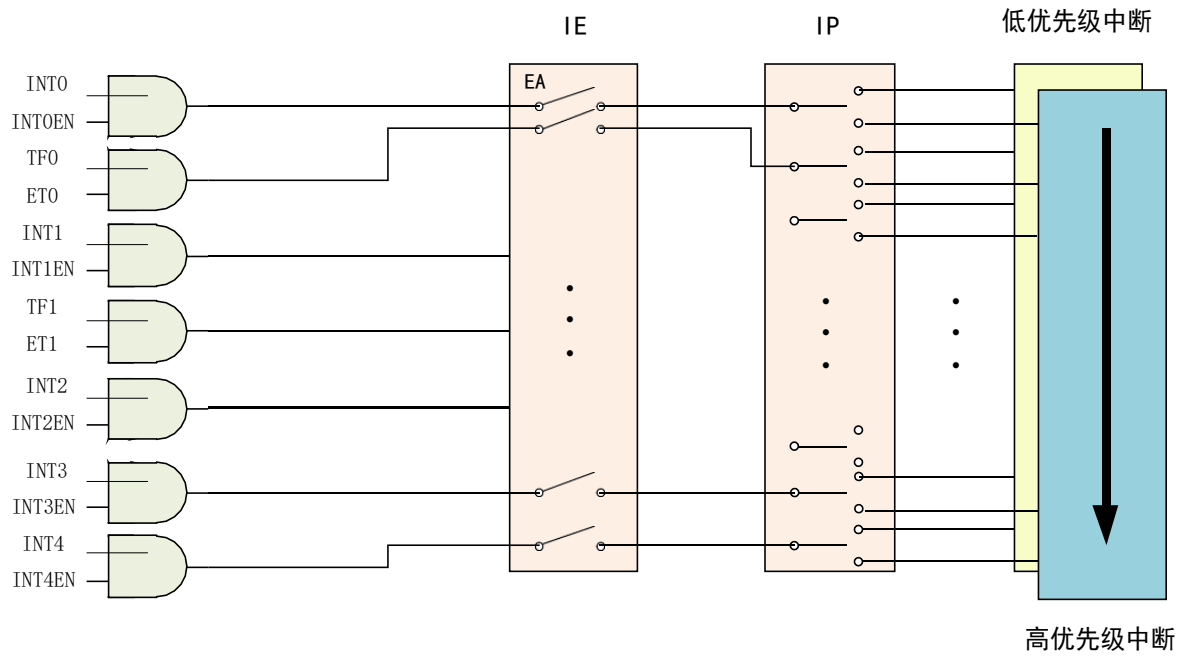


表 8-2-1 中断逻辑图

8.3 中断向量表

表 8-3-1 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
INT2	UART0/外部中断 2/PWM 中断	23H	4
INT3	定时器 2/外部中断 3/触摸按键中断/TMC 中断	2BH	5
INT4	外部中断 4/WDT 中断/I2C 中断/LVD 中断	33H	6

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	INT4EN	INT3EN	INT2EN	ET1	INT1EN	ET0	INT0EN
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：关闭 1：打开
6	INT4EN	中断 4 使能控制位（中断 4 用于 WDT/I2C/LVD/外部中断 4） 0：关闭 1：打开
5	INT3EN	中断 3 使能控制位（中断 3 用于定时器 2/TMC/TK/外部中断 3） 0：关闭 1：打开
4	INT2EN	中断 2 使能控制位（中断 2 用于UART0/PWM/外部中断 2） 0：关闭 1：打开
3	ET1	定时器 1 中断使能控制位 0：关闭 1：打开
2	INT1EN	中断 1 使能控制位（中断 1 用于外部中断 1） 0：关闭

		1：打开
1	ET0	定时器 0 中断使能控制位 0：关闭 1：打开
0	INT0EN	中断 0 使能控制位（中断 0 用于外部中断 0） 0：关闭 1：打开

备注：IE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：要开启外部中断 2 的中断， 除了设置 INT2EN 为1，EPIE0（外部中断 2 使能位）也要设为 1。

表 8-4-2 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	0

位编号	位符号	说明
7	-	-
6	PS1	中断 INT4 优先级控制位 0：低优先级 1：高优先级
5	PT2	中断 INT3 优先级控制位 0：低优先级 1：高优先级
4	PS0	中断 INT2 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

8.5 外部中断

8.5.1 外部中断介绍

除了标准 8051 的 INT0 和 INT1 以外，系统还扩展了 3 个中断入口INT2~INT4 作为外部中断。扩展的每个外部中断都可选择任意输入口作为中断触发源，同时也可各自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~INT4 外部中断状态寄存器。INT2~INT4 对应的各个配置寄存器为 EP0CON~EP2CON。

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 8-5-1 寄存器 EPIF

D4H	7	6	5	4	3	2	1	0
EPIF	-	-	-	-	-	EPIF2	EPIF1	EPIF0
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	EPIF2	外部中断 4 中断标志位，写 1 清零
1	EPIF1	外部中断 3 中断标志位，写 1 清零
0	EPIF0	外部中断 2 中断标志位，写 1 清零

表 8-5-2 寄存器 EPCON

D1H	7	6	5	4	3	2	1	0
EP0CON	EPIE0	EPPL0	-	-	EPPS0[3:0]			
R/W	R/W	R/W	-	-	R/W			
初始值	0	0	-	-	0	0	0	0
D2H	7	6	5	4	3	2	1	0
EP1CON	EPIE1	EPPL1	-	-	EPPS1[3:0]			
R/W	R/W	R/W	-	-	R/W			
初始值	0	0	-	-	0	0	0	0

D3H	7	6	5	4	3	2	1	0
EP2CON	EPIE2	EPPL2	-	-	EPPS2[3:0]			
R/W	R/W	R/W	-	-	R/W			
初始值	0	0	-	-	0	0	0	0
备注：下表中的“n”表示 0/1/2								
位编号	位符号	说明						
7	EPIEn	外部中断使能位 0：关闭 1：打开 *备注：n=0/1/2 分别对应外部中断 2/3/4。*						
6	EPPLn	外部中断触发沿选择位 0：上升沿 1：下降沿 *备注：n=0/1/2 分别对应外部中断 2/3/4。*						
5~4	-	-						
3~0	EPPSn[3:0]	中断引脚选择位域 编号和引脚对应表参考表 8-5-3 *备注：n=0/1/2 分别对应外部中断 2/3/4。*						

表 8-5-3 中断引脚编号索引表

引脚名称	编号	引脚名称	编号
P00	0	P30	8
P01	1	P31	9
P02	2	P32	10
P03	3	P33	11
P04	4	P34	12
P05	5	P35	13
P06	6		
P07	7		

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)
{
    P32F = 1;       //外部中断 0 的中断引脚为 P32，设置 P32 为输入功能
    EX0 = 1;        //INT0 中断使能
    IE0 = 1;        //外部中断 0 使能
    IT0 = 1;        //设置为下降沿中断
    PX0 = 1;          //设置 INT0 为高优先级
    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)
{
    P33F = 1;       //外部中断 1 的中断引脚为 P33，设置 P33 为输入功能
    EX1 = 1;        //INT1 中断使能
    IE1 = 1;        //外部中断 1 使能
    IT1 = 1;        //设置为下降沿中断
    PX1 = 1;          //设置 INT1 为高优先级
    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P00 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)
{
    P00F = 1;       //设置 P00 为输入引脚
    EP0CON = (1<<7) | (0<<6) | 0;      //设置为上升沿触发并设置中断引脚索引编号，0 对应 P00，如果设置为下
                                //降沿触发则设置为 EP0CON = (1<<7) | (1<<6) | 0;
    INT2EN = 1;          //INT2 中断使能
    EA = 1;              //总中断使能
}
void INT2_ISR (void) interrupt 4
{
    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

JZ8FC508T 系列芯片共支持以下时钟源：

- 内置 16MHz RC 振荡器
- 内置 131KHz RC 振荡器

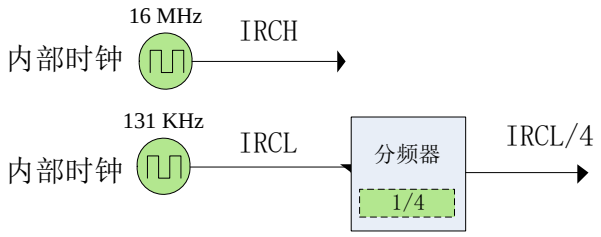


图 9-1-1 时钟源示意图

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 16MHz RC 振荡器
IRCL	内置 131KHz RC 振荡器

9.1.2 内置 16MHz 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 16MHz@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±1%。

9.2 时钟控制寄存器描述

表 9-2-1 寄存器 CKCON

8030H	7	6	5	4	3	2	1	0
CKCON	IHCKE	ILCKE	-	-	-	-	-	SCKS
R/W	R/W	R/W	-	-	-	-	-	R/W
初始值	0	0	-	-	-	-	-	0

位编号	位符号	说明
7	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。*
6	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。*
5-1	-	-
0	SCKS	系统时钟选择位 0：选择 IRCH 1：选择 IRCL

表 9-2-2 寄存器 IHCFG

8032H	7	6	5	4	3	2	1	0
IHCFG	IHCFG[1:0]		IHCFG[7:2]					
R/W	R/W		R/W					
初始值	-	-	-	-	-	-	-	-

位编号	位符号	说明
7~6	IHCFG[1:0]	IRCH 频率调整寄存器低 2 位
5~0	IHCFG[7:2]	IRCH 频率调整寄存器高 6 位

表 9-2-3 寄存器 ILCFGL、ILCFGH

8033H	7	6	5	4	3	2	1	0
ILCFGL	ILCFG[7:0]							
R/W	R/W							
初始值	-	-	-	-	-	-	-	-
8034H	7	6	5	4	3	2	1	0
ILCFGH	-	-	-	-	-	-	-	ILCFG[8]
R/W	-	-	-	-	-	-	-	R/W
初始值	-	-	-	-	-	-	-	-

位编号	位符号	说明
15~9	-	-
8~0	ILCFG	IRCL 频率调整寄存器

9.3 系统时钟

系统时钟控制由寄存器 CKCON、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。

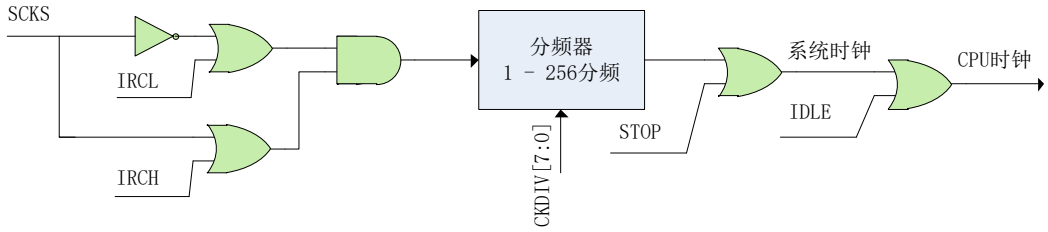


图 9-3-1 系统时钟结构图

9.3.2 系统时钟控制寄存器描述

表 9-3-2-1 寄存器 CKDIV

8031H	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**

设置系统时钟为IRCH，程序如下：

```
--------------------------------------------------------------------------------
#define IHCKE           (1<<7)
#define CKSEL_IRCH    0
void Sys_Clk_Set_IRCH(void)
{
    CKCON |= IHCKE;                             //打开 IRCH 时钟
    CKCON = (CKCON&0xFE) | CKSEL_IRCH;          //设置系统时钟为 IRCH
}
--------------------------------------------------------------------------------
```

◆ **设置系统时钟为 IRCL**

设置系统时钟为IRCL，程序如下：

```
--------------------------------------------------------------------------------
#define ILCKE       (1<<6)
#define CKSEL_IRCL    1

void Sys_Clk_Set_IRCL(void)
{
    CKCON |= ILCKE;                             //打开 IRCL 时钟
    Delay_ms(1);                                //延时 1ms，等待 IRCL 时钟稳定
    CKCON = (CKCON&0xFE) | CKSEL_IRCL;          //设置系统时钟为 IRCL
}
```

备注：如设置 IRCL 为系统时钟，必须在使能 IRCL 时钟后等待约 1ms 再切换为系统时钟，否则可能会出现异常。

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

10 供电和复位系统

10.1 供电系统

在JZ8FC508T 系列芯片VDD 和VSS 引脚间接入 1.8V - 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。

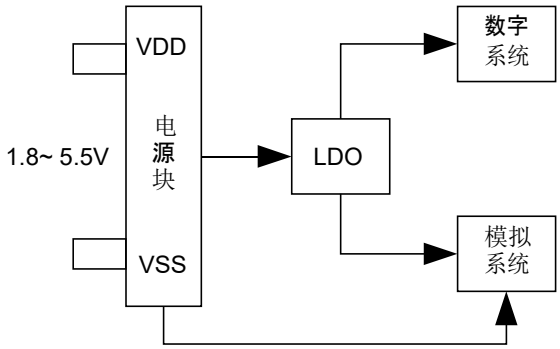


图 10-1-1 供电系统示意图

图 10-1-2 为芯片供电典型电路图。

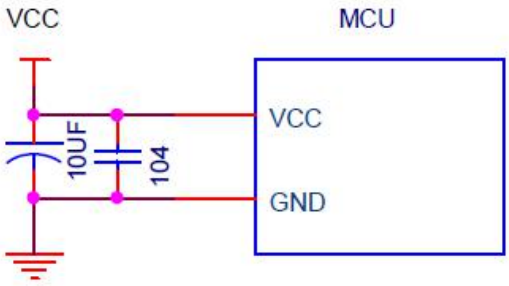


图 10-1-2 芯片供电典型电路图

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

JZ8FC508T 系列芯片有一个内置的低压差线性稳压器（LDO）。LDO 模块为芯片提供内核电压。LDO 的输出电压通过 VLEVEL 位（PWCON[2:0]）设置，VLEVEL 默认值为 5，对应的输出电压为 1.61V。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。

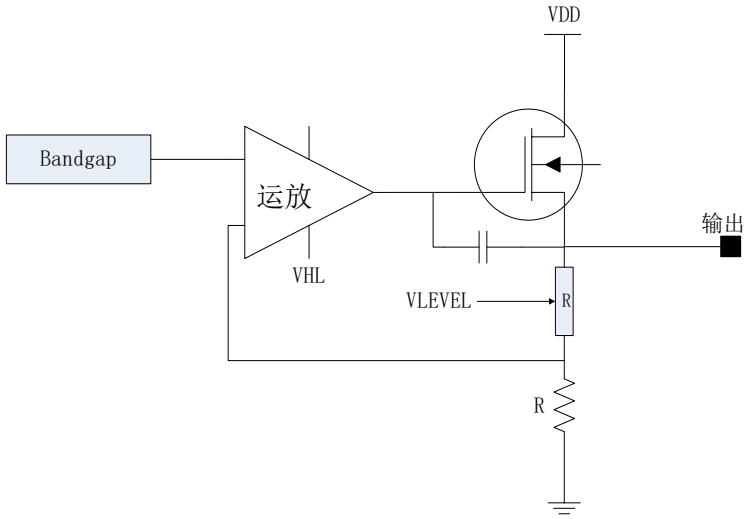


图 10-1-1 LDO 模块示意图

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 *备注：内部基准电压上电时由系统自动加载，用户不允许修改。*
3	VHL	LDO 工作模式控制位 1：高功率模式 0：低功率模式
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 *备注：* *1 内部时钟电路由LDO供电，改变LDO输出电压会引起内部时钟频率的变化，一般情况下，LDO电压保持默认值即可，不建议修改。* *2 不允许设置LDO输出电压小于1.5V，否则有可能引起异常。*

10.2 复位系统

JZ8FC508T 系列芯片有多个内部和外部复位源，如图 10-2-1 所示。

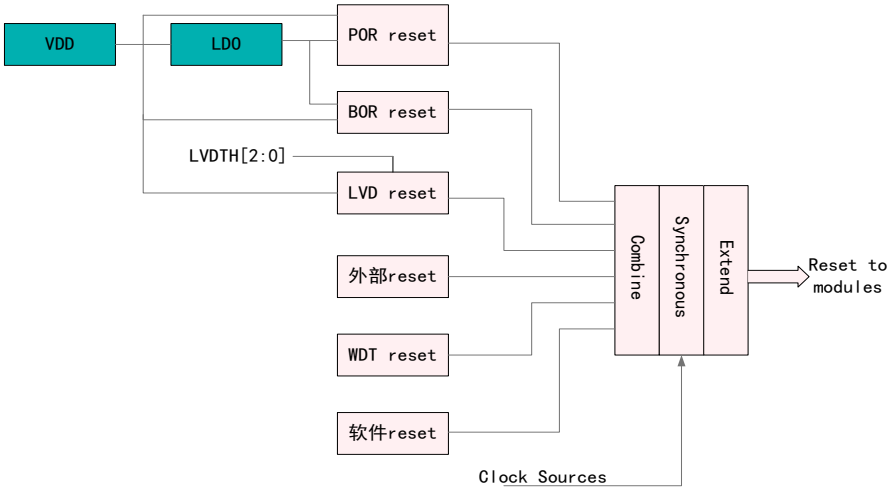
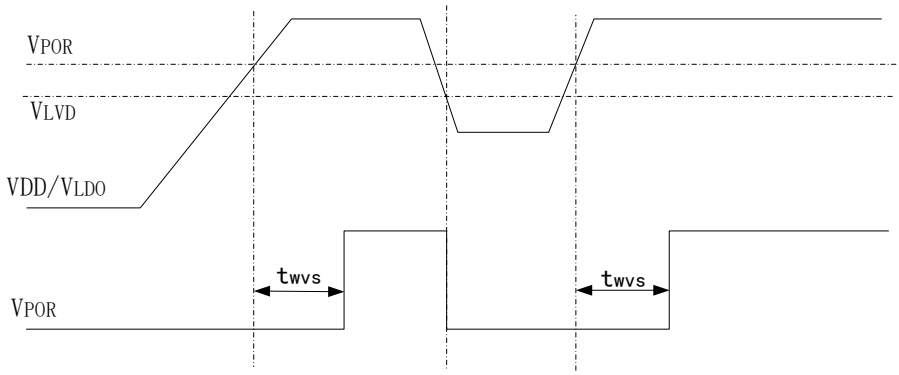


图 10-2-1 复位系统结构图

- **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。在上电掉电复位及外部硬复位产生后，程序将从 Mask ROM 开始运行。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

JZ8FC508T 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 12uA，STOP 模式时系统功耗小于 6uA，低速运行时功耗小于 20uA。

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。

重要提醒：

1 *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。*

2 *如果系统时钟选择为 IRCL，进入 STOP 时，IRCL 不可关闭，否则从 STOP 唤醒时可能会发生异常。*

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 20uA。

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	SMOD	-	SWRST	-	-	TSMODE	STOP	IDLE
R/W	R/W	-	W	-	-	R	W	W
初始值	0	-	0	-	-	0	0	0

位编号	位符号	说明
7	SMOD	UART0 波特率倍频控制位 在 UART0 工作于模式 1,2,3 时，设置 SMOD=1 会使波特率倍频，与标准 8051 相同。
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

8EH	7	6	5	4	3	2	1	0
IDLST	-	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	-	-
6	I2CINT/WDIF/LVDINT/EPIF[2]	IDLE 模式时，I²C/WDT/LVD/外部中断 4 的中断状态
5	T2INT/TKINT/TMINT/EPIF[1]	IDLE 模式时，定时器 2/触摸按键/TMC/外部中断 3 的中断状态
4	TI/RI/PWMINT/EPIF[0]	IDLE 模式时，UART0/PWM/外部中断 2 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

表 11-4-2 寄存器 STPST

8FH	7	6	5	4	3	2	1	0
STPST	-	STPSTL [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	WDTWKF/LVDWKF/I2CWKF	STOP 模式时，WDT/LVD/I²C 的中断状态
5	TKWKF/TMWKF	STOP 模式时，触摸按键/TMC 的中断状态
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

11.5 低功耗模式控制例程

◆ **STOP 模式例程**

STOP 模式程序如下：

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

#define CKSEL_IRCH    0
#define CKSEL_IRCL    1

void Stop(void)
{
    bit IE_EA;
    I2CCON = 0;  //关闭 I2C 功能，因为 I2C 默认是使能的，如果 I2C 不关闭将无法关闭 IRCH 时钟
    CKCON = 0;   //关闭所有时钟
    PWCON &=0xf7; //设置 LDO 进入低功率模式
    MECON |= (1<<6); //设置 FLASH 进入深度睡眠状态
    IE_EA = EA;  //保存全局中断使能位状态
    EA = 0;          /*为保证从 STOP 模式唤醒后执行三条 NOP 指令，关闭中断后再进 STOP，执行三条 NOP 指
                     令后再开中断。注意：关闭全局中断并不会影响中断唤醒 STOP。*/
    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 后，必须把 LDO 设置回高功率模式
}
```

◆ **IDLE 模式例程**

IDLE 模式程序如下：

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

#define CKSEL_IRCH    0
#define CKSEL_IRCL    1

void Idle(void)
{
    bit IE_EA;
    I2CCON = 0;  //关闭 I2C 模块，因为 I2C 默认是使能的，如果 I2C 不关闭将无法关闭 IRCH 时钟
    CKCON |= ILCKE; //打开 IRCL 时钟
    CKCON |= CKSEL_IRCL;    //切换系统时钟为 IRCL
    CKCON&=~ IHCKE;         //关闭 IRCH 时钟

    PWCON &=0xf7; //设置 LDO 进入低功率模式
    MECON |= (1<<6); //设置 FLASH 进入深度睡眠状态
    IE_EA = EA;  //保存全局中断使能位状态
    EA = 0;         /*为保证从 IDLE 模式唤醒后执行三条 NOP 指令，关闭中断后再进 IDLE，执行三条 NOP 指
                    令后再开中断。注意：关闭全局中断并不会影响中断唤醒 IDLE。*/

    PCON |= 0x01;       //进入 IDLE 模式
    _nop_();
    _nop_();
    _nop_();
    EA = IE_EA;         //恢复原全局中断开关状态
    PWCON |= 0x08; //退出 IDLE 后，必须把 LDO 设置回高功率模式
}
```

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

#define CKSEL_IRCH    0
#define CKSEL_IRCL    1

void LowSpeedMode(void)
{
    I2CCON = 0; //关闭 I2C 模块，因为 I2C 默认是使能的，如果 I2C 不关闭将无法关闭 IRCH 时钟
    CKCON |= ILCKE;      //打开 IRCL 时钟
    CKCON |= CKSEL_IRCL;   //切换系统时钟为 IRCL
    CKCON&=~ IHCKE;   //关闭 IRCH 时钟
    PWCON &=0xf7; //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。

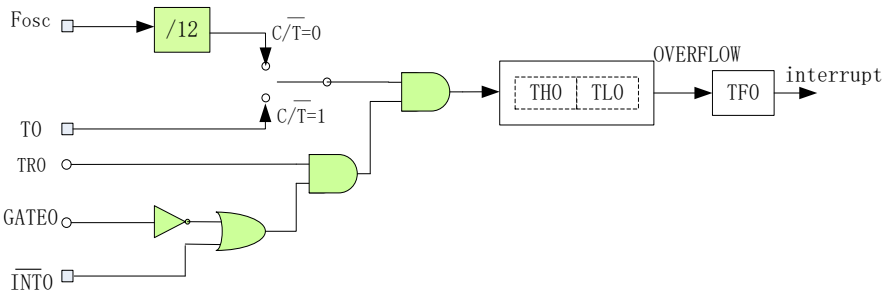


图 12-1-1-1 定时器 0 的模式 0 与 1

- **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。

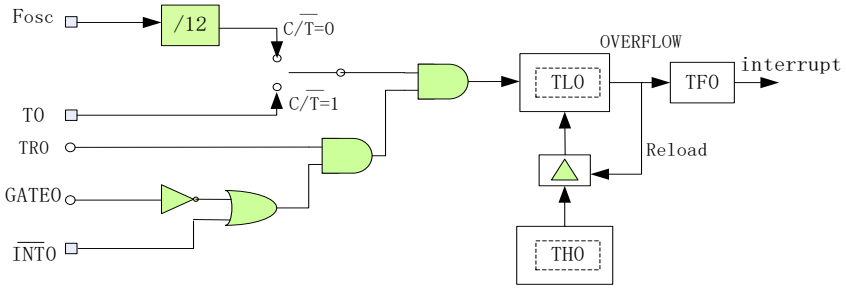


图 12-1-1-2 定时器 0 的模式 2

- **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。

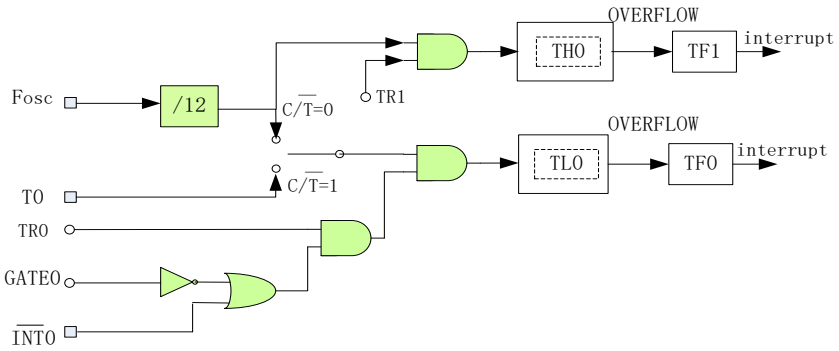


图 12-1-1-3 定时器 0 的模式 3

1212 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。

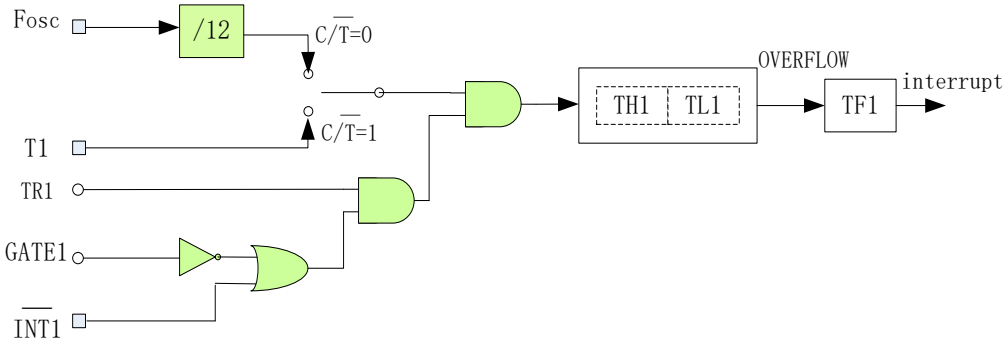


图 12-2-1 定时器 1 的模式 0 和 1

- **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。

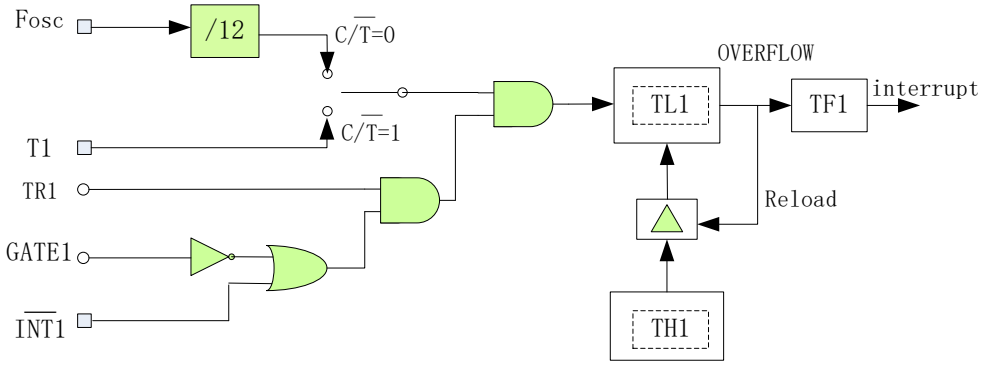


图 12-2-2 定时器 1 的模式 2

- **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=3 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。

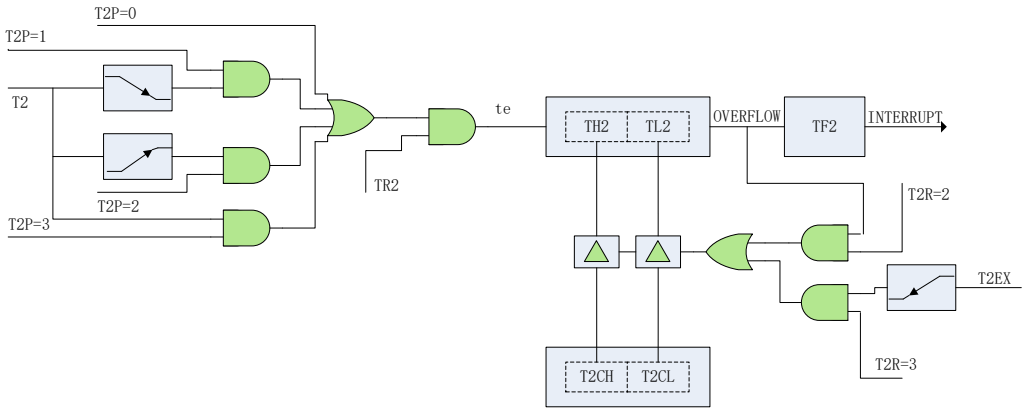


图 12-3-1-1 定时器 2 的重载模式

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。

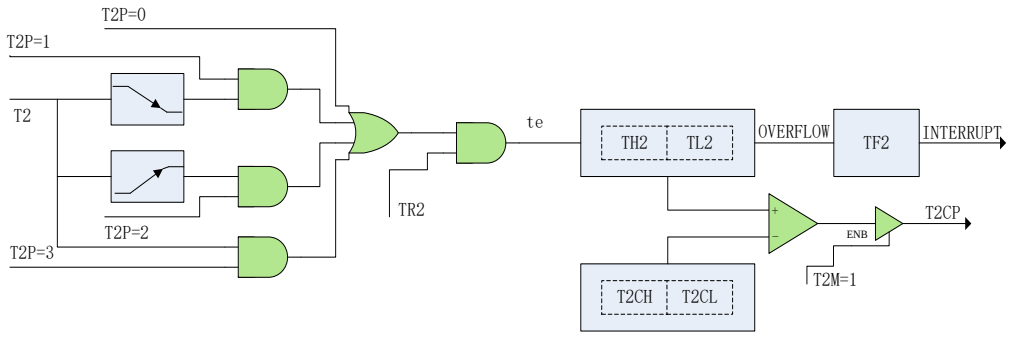


图 12-3-1-2 定时器 2 的比较模式

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。

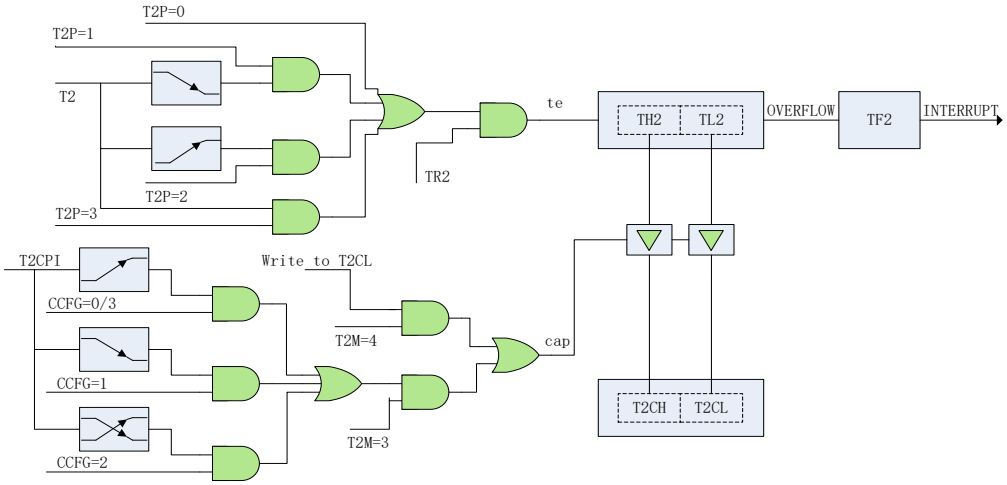


图 12-3-1-3 定时器 2 的抓取模式

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	UCKS	UART0 时钟选择位 0：UART0 使用定时器 1 溢出脉冲 1：UART0 使用定时器 2 溢出脉冲
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
0	T2P0	

		11：定时器 2 使用内部系统时钟计数，通过 T2 门控

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＝2 或 T2M＝3 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

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 16MHz 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。

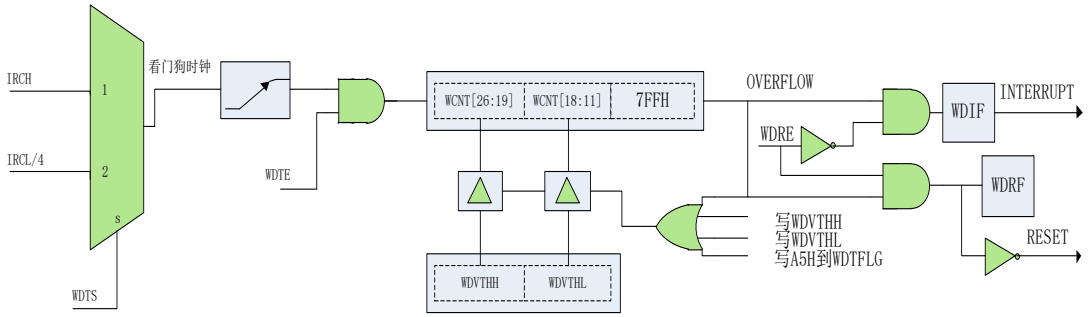


图 13-1-1 看门狗模块结构图

13.2 看门狗定时器(WDT)寄存器描述

表 13-2-1 寄存器 WDCON

A0H	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 其他：WDT 关闭
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

A1H	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

A2H	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
A3H	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 16MHz，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<<6)
#define WDTS_IRCL           (2<<6)

#define WDRE_reset          (1<<0)
#define WDRE_int            (0<<0)
void WDT_init(void)
{
    WDCON = WDTS_IRCH | WDRE_int;     //设置看门时钟为 IRCH, 看门狗中断模式
    WDVTHH = 0x1E;                    //设置看门狗时间为 1 秒
    WDVTHL = 0x83;
    WDFLG = 0xA5;                     //刷新看门狗
    INT4EN = 1;                       //开启看门狗中断
    EA = 1;                          //开启总中断
}
void WDT_ISR (void) interrupt 6
{
    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 16MHz，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<<6)
#define WDTS_IRCL           (2<<6)

#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 = 0x1e;                        //设置看门狗时间为 1 秒
```

```
    WDVTHL = 0x83;
    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

D5H	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

D6H	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<<7)
#define ILCKE           (1<<6)

void INT3_ISR (void) interrupt 5
{
    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;    //打开 IRCL 时钟
    TMCON = TME(1);  //TMC 使能
    TMSNU = 0;          //设置 1 个最小单位时间（即 512 个 IRCL 时钟周期）产生中断
    INT3EN = 1;         //开启 TMC 中断
    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

通用输入/输出口用于芯片和外部进行数据传输，JZ8FC508T 系列芯片最大封装有 14 个 I/O 引脚，每个 I/O 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。每个引脚都分配了功能设置寄存器 PnxF 和 PnxC（分别对应引脚 Pnx，其中 n=0、3, 代表 P0、P3，x=0~7, 代表 Pn.0~Pn.7），用户可通过寄存器 PnxF 和 PnxC 配置引脚的主功能和其他选项。

每个 I/O 可通过 PnxPUP/PnxPDP(PnxF[7]/PnxF[6]) 使能上/下拉电阻， 强/弱上/下拉电阻由 PU_SEL/PD_SEL(PnxC[5]/PnxC[4])选择，当 PU_SEL/PD_SEL 为 1 时，选择为强上/下拉，否则为弱上/下拉，默认为强上/下拉。

当I/O 设置为输出模式时，当 PnxOPR(PnxF[5])为 1 时，I/O 为开漏输出模式。

I/O 为推挽输出时，通过 DRV(PnxC[3:2])可设置 IO 推挽输出驱动强度，通过 SR(PnxC[1:0])可设置 IO 输出翻转斜率，当I/O 输出电平翻转时，由于电感效应，会在I/O 端口产生过冲信号，此过冲信号对芯片系统有可能造成一定影响，降低I/O 输出强度及I/O 速度可有效降低过冲信号幅度，在应用时，此两项参数可灵活配置。

I/O 为输入模式时，可通过 SMIT_EN(PnxC[6])位选择斯密特模式或反相器模式，选择为反相器模式时，高低电平的触发门限值为 1/2 VDD，默认为斯密特模式。

GPIO 的主要特性如下：

- 可配置为高阻模式
- I/O 结构可独立设置强上拉、弱上拉、强下拉、弱下拉电阻
- 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 1.8~5.5V 宽电压范围
- 设为推挽输出时，可独立设置IO 驱动强度
- 设为推挽输出时，可独立设置IO 输出速度

重要提醒：所有 GPIO 引脚输入电压不可高于 VDD 引脚电压，否则可能会引起芯片工作异常。

GPIO 推挽模式结构图如图 15-1-1 所示。

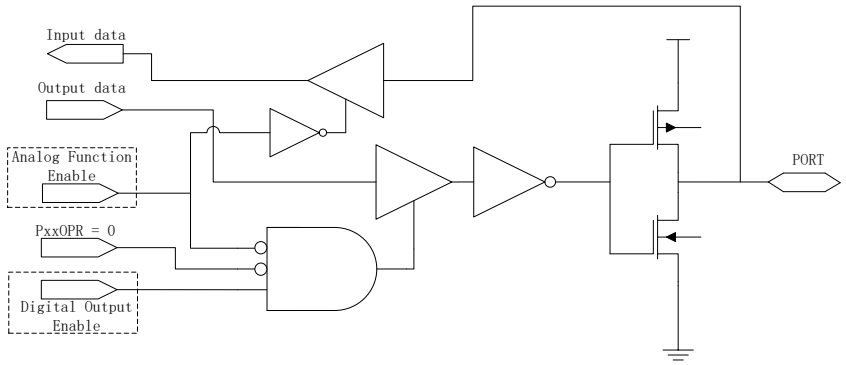


图 15-1-1 I/O 推挽模式结构示意图

GPIO 开漏模式结构图如图 15-1-2 所示。

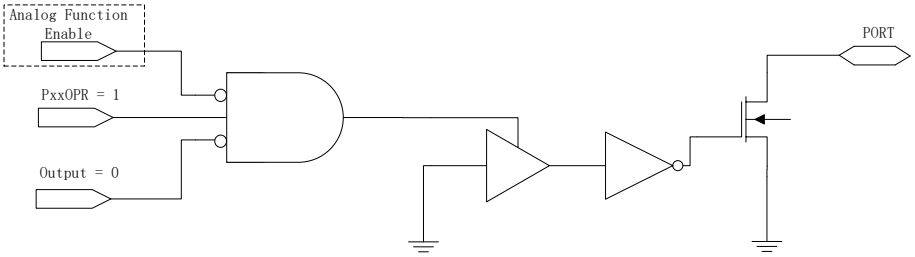


图 15-1-2 I/O 开漏模式结构示意图

GPIO 下拉结构图如图 15-1-3 所示。

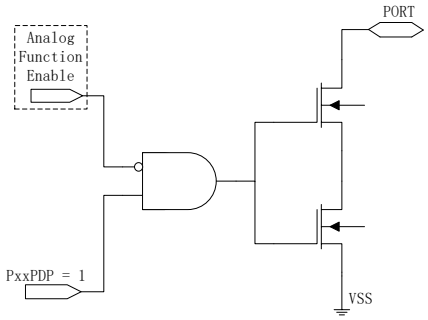


图 15-1-3 I/O 下拉模式结构示意图

GPIO 上拉结构图如图 15-1-4 所示。

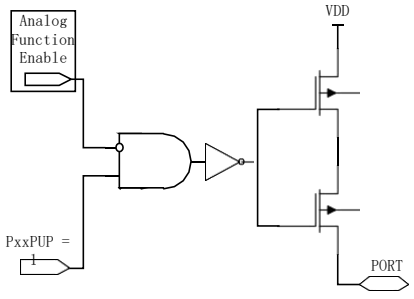


图 15-1-4 I/O 上拉模式结构示意图

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 P3

B0H	7	6	5	4	3	2	1	0
P3	-	-	P35	P34	P33	P32	P31	P30
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	-	-
5~0	P3x	引脚 P3x 的数据寄存器，管脚功能设置为 GPIO 时有效 0：设为输入时 P3x 电平为低，设为输出时 P3x 输出低电平 1：设为输入时 P3x 电平为高，设为输出时 P3x 输出高电平

表 15-2-3 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
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
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	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
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
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
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	-	-	0	0	0
8007H	7	6	5	4	3	2	1	0
P07F	P07PUP	P07PDP	P07OPR	-	-	P07S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
8018H	7	6	5	4	3	2	1	0

P30F	P30PUP	P30PDP	P30OPR	-	-	P30S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
8019H	7	6	5	4	3	2	1	0
P31F	P31PUP	P31PDP	P31OPR	-	-	P31S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
801AH	7	6	5	4	3	2	1	0
P32F	P32PUP	P32PDP	P32OPR	-	-	P32S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
801BH	7	6	5	4	3	2	1	0
P33F	P33PUP	P33PDP	P33OPR	-	-	P33S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
801CH	7	6	5	4	3	2	1	0
P34F	P34PUP	P34PDP	P34OPR	-	-	P34S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
801DH	7	6	5	4	3	2	1	0
P35F	P35PUP	P35PDP	P35OPR	-	-	P35S		
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	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

备注：Pnx → n=0、3, 代表 P0、P3
x=0~7, 代表 Pn.0~Pn.7

表 15-2-4 寄存器 PnxC

8120H	7	6	5	4	3	2	1	0
P00C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
初始值	-	1	1	1	1	1	1	1
8121H	7	6	5	4	3	2	1	0
P01C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
初始值	-	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
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
8139H	7	6	5	4	3	2	1	0
P31C	-	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
813AH	7	6	5	4	3	2	1	0
P32C	-	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
813BH	7	6	5	4	3	2	1	0
P33C	-	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
813CH	7	6	5	4	3	2	1	0
P34C	-	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
813DH	7	6	5	4	3	2	1	0
P35C	-	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

位编号	位符号	说明
7	-	-
6	SMIT_EN	IO为输入功能时模式选择位 0：反相器模式 1：SMIT 模式
5	PU_SEL	上拉电阻选择位 0：弱上拉（上拉电阻为45K） 1：强上拉（上拉电阻为 10K）
4	PD_SEL	下拉电阻选择位 0：弱下拉（上拉电阻为45K） 1：强下拉（上拉电阻为 15K）
3~2	DRV	输出强度选择位，范围：0~3，数值越大，驱动能力越强
1~0	SR	输出斜率控制位，范围：0~3，数值越大，IO 翻转斜率越高（速度越快）

表 15-2-5 引脚复用功能映射表

取值 名称	0	1	2	3	4	5	6	7
P00S	高阻	数字输入	数字输出	DAK[0]	TK[9]	高阻	高阻	LVD_IN0
P01S	高阻	数字输入	数字输出	DAK[1]	TK[8]	PWM[3]	TLED_COM	LVD_IN1
P02S	高阻	数字输入	数字输出	I2C_SCL	TK[7]	高阻	高阻	高阻
P03S	高阻	数字输入	数字输出	I2C_SDA	TK[6]	高阻	高阻	高阻
P04S	高阻	数字输入	数字输出	T2CP	TK[5]	PWM[4]	高阻	高阻
P05S	高阻	数字输入	数字输出	T2EX	TK[4]	PWM[5]	高阻	高阻
P06S	高阻	数字输入	数字输出	T2	TK[3]	高阻	高阻	高阻
P07S	高阻	数字输入	数字输出	RESET	TK[2]	高阻	高阻	高阻
P30S	高阻	数字输入	数字输出	I2C_SDA	TK[1]	UART0_RX	高阻	高阻
P31S	高阻	数字输入	数字输出	I2C_SCL	TK[0]	UART0_TX	高阻	高阻

P32S	高阻	数字输入/INT0	数字输出	T0	TK[12]	PWM[0]	高阻	高阻
P33S	高阻	数字输入/INT1	数字输出	高阻	TK[11]	PWM[1]	高阻	高阻
P34S	高阻	数字输入	数字输出	高阻	TK[10]	PWM[2]	高阻	高阻
P35S	高阻	数字输入	数字输出	T1	TKCAP	高阻	高阻	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，并且打开强上拉，程序如下：

```
-----------------------------------------------------------------------------------------
P00C |= (1<<5);
P00F = (1<<7) | (1<<5) | 2;
-----------------------------------------------------------------------------------------
```

P00 设置为输入功能，并且打开强上拉，程序如下：

```
-----------------------------------------------------------------------------------------
P00C |= (1<<5);
P00F = (1<<7) | 1;
-----------------------------------------------------------------------------------------
```

16 通用串行接口（UART0）

16.1 功能简介

UART0 是一个全双工同步/异步串行数据收发器（全双工意味着可以同时发送和接收数据），与标准 8051 基本兼容。UART0 接收器有一字节的缓存，也就是接收完的一个字节数据会被送到缓存寄存器，同时接收器可以接收新的数据，当然，在新的一字节数据接收完之前，前面接收的一字节数据必须被读取，否则会被新数据覆盖。寄存器 S0BUF 是 UART0 的发送/接收数据寄存器，在物理上，S0BUF 实际是两个寄存器，一个是数据发送寄存器，另一个是数据接收寄存器，写 S0BUF 会将数据写入发送寄存器并启动数据发送，而读 S0BUF 会读取接收寄存器中接收到的一字节数据。

UART0 有 4 种工作模式，如表 16-1-1 所示。

表 16-1-1 UART0 通信工作模式

SM00	SM10	模式	描述	波特率
0	0	0	同步移位模式	Fclk/12
0	1	1	8 位异步模式	波特率为(2^ SMOD)*CPUCLK*(定时器 1/2 溢出率)/32，详见 T2CON 中UCKS
1	0	2	9 位异步模式	当 SMOD＝0 时，波特率为 Fclk/64 当 SMOD＝1 时，波特率为 Fclk/32
1	1	3	9 位异步模式	波特率为(2^ SMOD)*CPUCLK*(定时器 1/2 溢出率)/32，详见 T2CON 中UCKS

备注：

1 以上波特率的前提条件是 UDE=0，当 UDE=1 时，波特率由 DNUM 值决定。

2 由于定时器 2 的时钟是直接来自系统时钟，没有经过分频，所以选择定时器 2 作为 UART0 时钟发生器会有更高的波特率。

- **模式 0**

在模式 0，UART0 同步收发数据。引脚 TX 输出移位时钟，引脚 RX 用于输出或接收数据。传输数据为 8 位数据，从最低位开始传输，波特率固定为主时钟频率的 1/12。写入数据到寄存器 S0BUF 会启动 UART0 发送。作为接收器，需要设置寄存器S0CON 的 REN=1 和清除 RI0 标志，当接收到一字节数据时，RI0 会置 1。

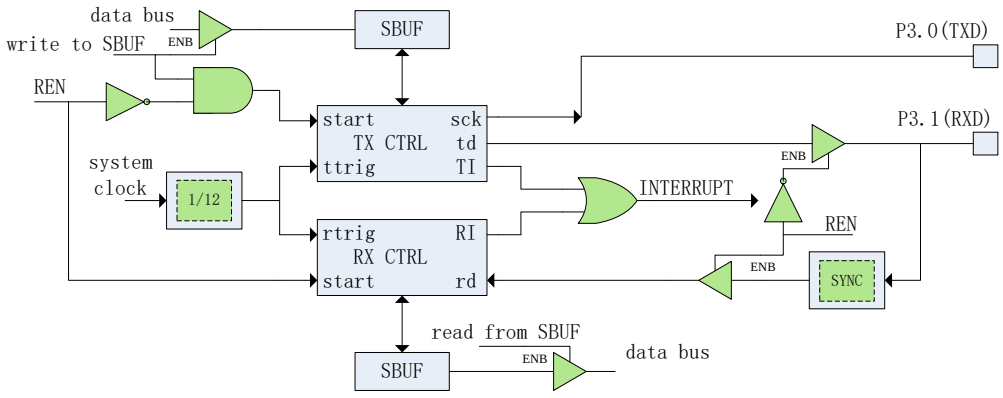


图 16-1-1　UART0 模式 0 示意图

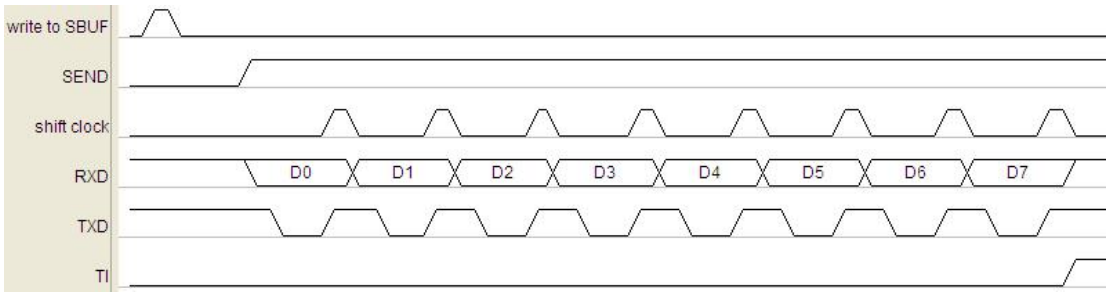


图 16-1-2 UART0 模式 0 发送数据波形

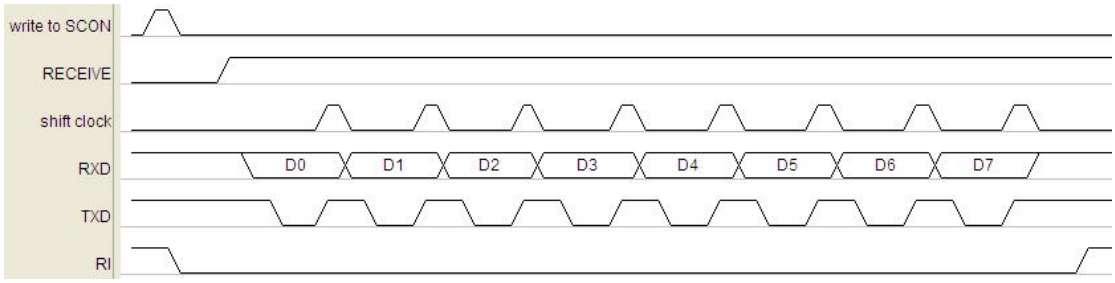


图 16-1-3 串口 0 模式 0 接收数据波形

- **模式 1**

在模式 1，UART0 可异步同时收发 8 位数据。可通过设置 UCKS 位（详见寄存器T2CON）来选择定时器 1 或定时器 2 的溢出信号作为 UART0 的时钟，相应地，设置定时器的溢出率也就可以调整 UART0 的波特率。另外，可通过SMOD 位（详见寄存器PCON）来选择波特率倍频。

写入数据到寄存器 S0BUF 会启动 UART0 发送。第一个传送的位是开始位（为 0），然后是 8 位数据（低位先发），最后传送的是停止位（为 1）。

在接收状态，UART0通过检测引脚 RX 的下降沿来同步。传送过程完成后，8位数据存放在寄存器 S0BUF，有效停止位值存放在 RB80 位（S0CON[2]）。

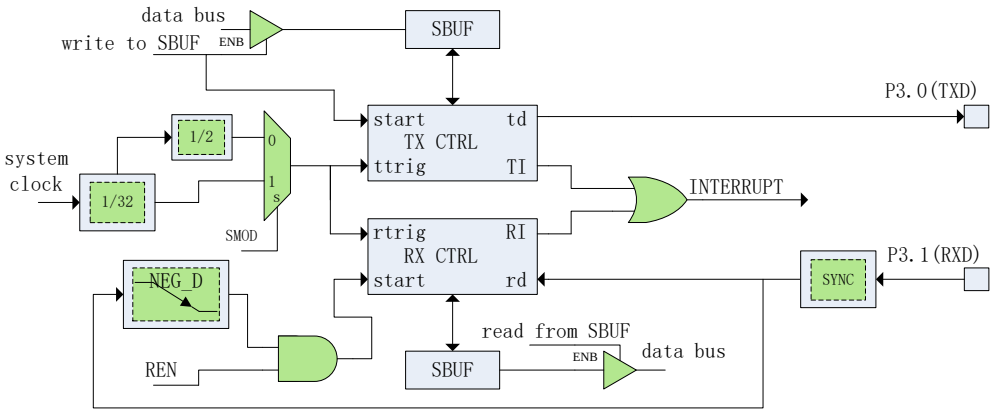


图 16-1-4　UART0 模式 1 示意图

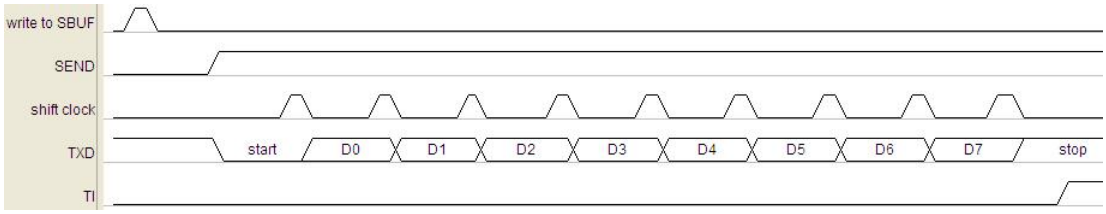


图 16-1-5　UART0 模式 1 发送数据波形

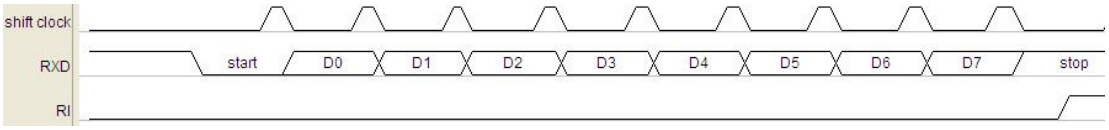


图 16-1-6　UART0 模式 1 接收数据波形

- **模式 2**

在模式 2，UART0 可异步同时收发 9 位数据，通过设置寄存器 PCON 的 SMOD 位可选择波特率固定为 Fsys/32 或Fsys/64。

写入数据到寄存器 S0BUF 会启动 UART0 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S0CON 的 TB80 位，最后传送的是停止位（为 1）。

传送过程通过写 S0BUF 寄存器开启。首先传送的是开始位（一直为 0），然后是 9 位数据，首先是传输数据的最低位，第 9 位是S0CON 寄存器的TB80，最后是传送停止位（一直为 1）。

在接收状态，UART0 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 S0BUF，第 9 位数据存放在 RB80 位（S0CON[2]）。

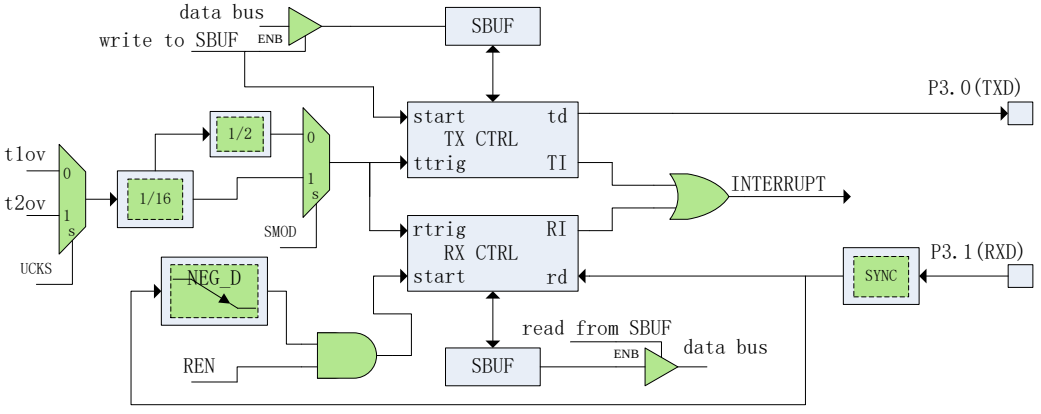


图 16-1-7 UART0 模式 2 示意图

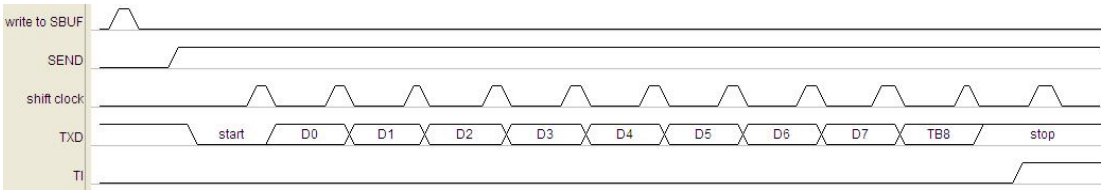


图 16-1-8 UART0 模式 2 发送数据波形

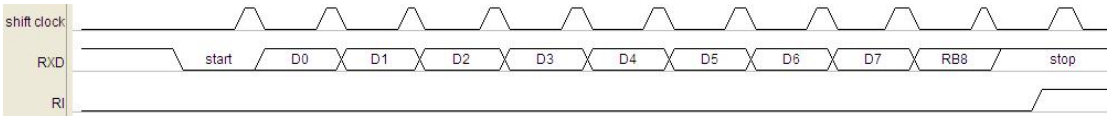


图 16-1-9 UART0 模式 2 接收数据波形

- **模式 3**

模式 2 和模式 3 的唯一不同的是模式 3 的波特率可通过定时器 1 或定时器 2 来产生，可参考模式 1 的示意图。波特率设置可参考模式 1 介绍，而其他功能描述参考模式 2。

- **UART0 多机通信**

在 UART0 模式 2 和 3 中有一个专门适用于多机通信的机制。当寄存器S0CON 的 SM20 位置 1，只有接收到第 9 位数据为 1（RB80=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 SM20 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 SM20=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 SM20 仍然设为 1，这样就只有被寻址的从机才会产生接收中断。

- **快速波特率设置**

在标准 51 单片机的 UART0 中, UART 的波特率固定为定时器溢出率的 32 分频，由于 JZ8FC508T 系列MCU 的主时钟为 16MHz（或 16MHz 的分频），配置的波特率和标准波特率相比有较大误差，所以在 JZ8FC508T 系列 MCU 中，设计了校正波特率的机制，UART 的波特率不固定为定时器溢出率的 32 分频，而是可以由寄存器

UDCKS 设置。例如：当 UART 的波特率固定为定时器溢出率的 32 分频时，选择定时器 2 作为 UART 的波特率发生器，若要配置波特率为 115200，计算公式为：16000000÷32÷115200=4.34, 由于定时器计数只能取整数，所以取 4（即每 4 个系统时钟周期定时器溢出一次），误差率约为 8.5%，误差率太大会导致通信不正常。由于系统时钟是固定的，要达到更准确的波特率，只能通过修改分频系数来实现。如果设置定时器 5 个时钟周期溢出， 那么：16000000÷115200÷5=27.78。取分频数为 28，那么波特率为 114285，和115200 相比，误差率约为 0.8%，一般情况下不会影响 UART 通信。另外，更小的分频数也可以实现更高的波特率配置。

芯片默认的分频系数为 32，与标准 51 相同。如果要更改分频系数，通过设置 UDE=1 来使能，DNUM 的数值表示不同的分频系数，详见寄存器 UDCKS 描述。

要注意的是，当 UDE=1 时，SMOD 位（PCON[7]）将不起作用。

16.2 寄存器描述

表 16-2-1 寄存器 S0CON

98H	7	6	5	4	3	2	1	0
S0CON	SM00	SM10	SM20	REN0	TB80	RB80	TI0	RI0
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	SM00	串口 0 模式选择位，详见表 17-1-1-1
6	SM10	
5	SM20	多机通信使能位，1 有效
4	REN0	串行接收使能位，1 有效
3	TB80	发送的第 9 位数据 在模式 2 和 3，这个位用于UART0 发送数据，对应发送数据的第 9 位 （例如奇偶校验或多机通信），由软件控制
2	RB80	接收的第 9 位数据 在模式 2 和 3，这个位用于UART0 接收数据，对应接收数据第 9 位；模式 1 时该位为停止位；如果 SM2=1，该位为多主机令牌位；在模式 0 这个位没有使用
1	TI0	发送中断标志，1 有效，写 0 清除
0	RI0	接收中断标志，1 有效，写 0 清除

表 16-2-2 寄存器 S0BUF

99H	7	6	5	4	3	2	1	0
S0BUF	S0BUF[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	S0BUF	发送接收缓冲器 写 S0BUF 将启动发送所写的数据 读 S0BUF 将读取已经接收的数据

表 16-2-3 寄存器 UDCKS

D8H	7	6	5	4	3	2	1	0
UDCKS	UDE	-	-	DNUM[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	UDE	快速波特率配置使能控制位，1 有效 ***备注：*** *UDE=0 时，UART0 波特率按照原来的配置，UDE=1，UART0 波特率由 DNUM 来配置。*
6~5	-	-
4~0	DNUM	快速波特率配置寄存器，仅在 UDE=1 时有效 模式 0：发送时，须满足 DNUM>=1；接收时，DNUM>=1 BR = (F_{sys}/2)*(1/(DNUM+1)) 模式 1：发送时，须满足 DNUM>=0；接收时，DNUM>=6 BR = F_{sys}*(1/(DNUM+1)) 模式 2：发送时，须满足 DNUM>=0；接收时，DNUM>=6 BR = (F_{sys}/2)*(1/(DNUM+1)) 模式 3：发送时，须满足 DNUM>=0；接收时，DNUM>=6 BR = F_{sys}*(1/(DNUM+1))

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 19-3-1 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。

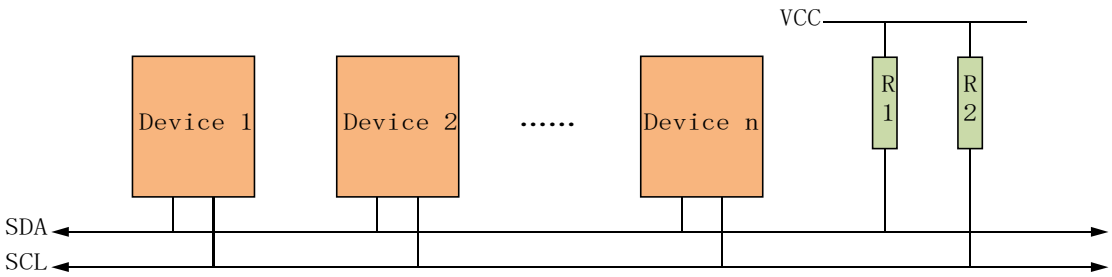


图 17-3-1 I2C 总线互连图

I²C 模块原理示意图如图图 17-3-2 所示。

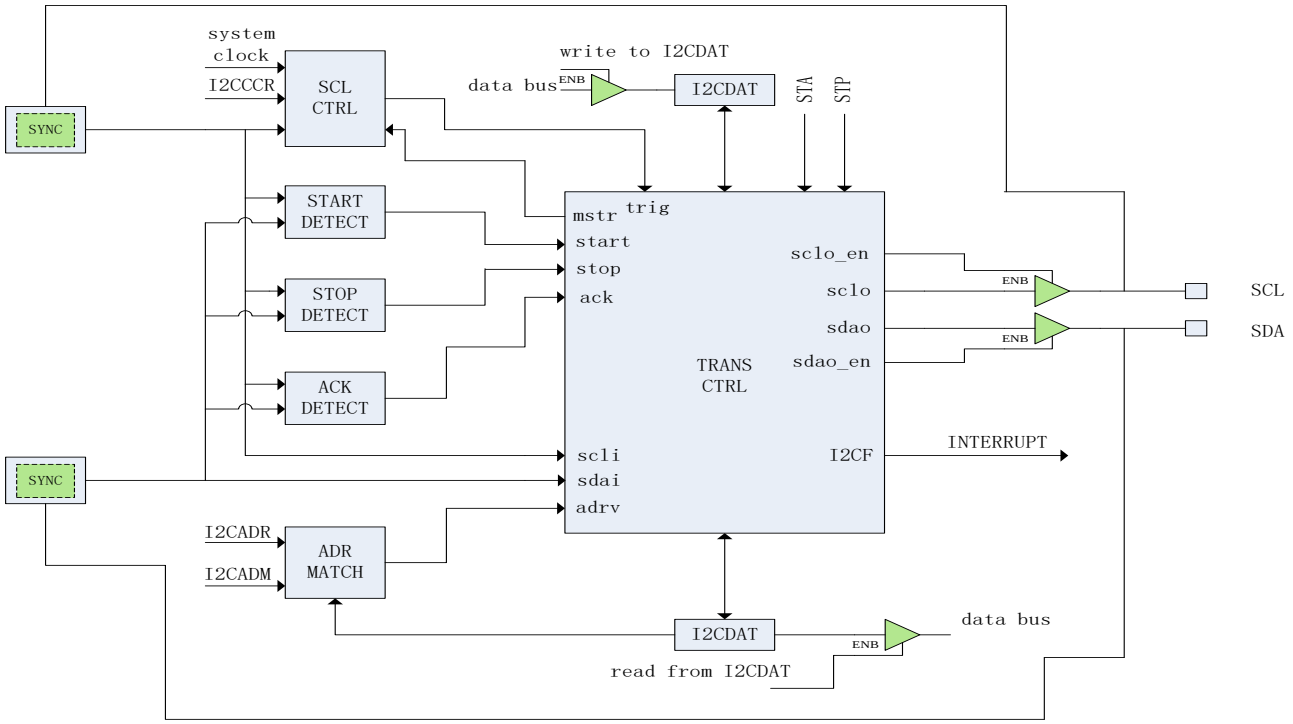


图 17-3-2　I²C 模块原理示意图

- **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。

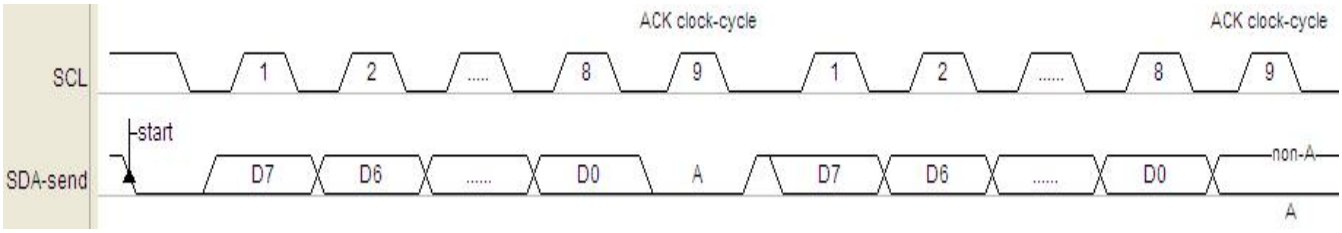


图 17-1-3　I2C 总线数据传输格式

- **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。

- **I2C**时钟设置

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，I2C 的采样时钟由SMPDIV(I2CCCR[7:5])设置，当 SMPDIV 不为 0 时，滤波功能自动启动。作为主机时，SCL 的输出时钟频率由SMPDIV和I2CCKD(I2CCCR[4:0])决定（具体请查看寄存器部分介绍）。

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

C0H	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

C1H	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

C2H	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	SMPDIV[2:0]			I2CCKD[4:0]				
R/W	R/W							
初始值	0	0	1	0	0	0	0	0

位编号	位符号	说明
7~5	SMPDIV	I2C采样时钟设置，I2C采样时钟为I_2C工作时钟的2的smpdiv次幂分频，即： 000：$F_{sample}=F_{i2cclk}$ 001：$F_{sample}=F_{i2cclk}/2$ 010：$F_{sample}=F_{i2cclk}/4$ … 111：$F_{sample}=F_{i2cclk}/128$
4~0	I2CCKD	I2C SCL输出时钟频率设置，SCL输出时钟频率为采样频率的(I2CCKD +1)分频，即： $F_{scl}= F_{sample}$ /(I2CCKD +1))

		备注： *1.* 当*SMPDIV= 0*时，如果设置*I2CCKD*小于*9*，将自动按*9*计算。 *2.* 当 *SMPDIV>0* 时，如果设置 *I2CCKD* 小于 *7*，将自动按 *7* 计算。 ***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

C4H	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

C5H	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	I^2C 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

C6H	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	-	-	-	-	-	-	-	I2CS
R/W	-	-	-	-	-	-	-	R/W
初始值	-	-	-	-	-	-	-	0

位编号	位符号	说明
7~1	-	-
0	I2CS	I²C 引脚选择位 0：SCL 在引脚 P31，SDA 在引脚 P30 1：SCL 在引脚 P02，SDA 在引脚 P03

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(N)        (N<<7) //N = 0~1
//I2CFLG 定义
#define I2CF             (1<<0)

#define I2C_ADDR         0xCA          //定义 I2C 从机地址
unsigned char xdata WriteBuffer[20]={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19};
void main(void)
{
    unsigned char i;
    EA = 1;                                          //开全局中断
/**********选择 I2C 端口*****************************************************************/
//  I2CIOS = 0;             //选择 P30,P31 作为 I2C 通信引脚
//  P30F = 3 | (1<<7);      //设置 P30 作为 I2C SDA，并打开上位
//  P31F = 3 | (1<<7);      //设置 P31 作为 I2C SCL，并打开上位

    I2CIOS = 1;             //选择 P03,P02 作为 I2C 通信引脚
    P03F = 3|(1<<7);        //设置 P03 作为 I2C SDA，并打开上位
    P02F = 3|(1<<7);        //设置 P02 作为 I2C SCL，并打开上位
//以上两组端口二选一
/*********************************************************************************/
    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(100);
        }
    }
```

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
unsigned char xdata ReadBuffer[20];
void main(void)
{
    unsigned char i;
    EA = 1;                                                    //开全局中断
/**********选择 I2C 端口**********************************************************/
//  I2CIOS = 0;                 //选择 P30,P31 作为 I2C 通信引脚
```

```
//    P30F = 3 | (1<<7);          //设置 P30 作为 I2C SDA，并打开上位
//    P31F = 3 | (1<<7);          //设置 P31 作为 I2C SCL，并打开上位

      I2CIOS = 1;                 //选择 P03,P02 作为 I2C 通信引脚
      P03F = 3|(1<<7);            //设置 P03 作为 I2C SDA，并打开上位
      P02F = 3|(1<<7);            //设置 P02 作为 I2C SCL，并打开上位
//以上两组端口二选一
/*****************************************************************************/
      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;
          }

          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 SEND_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 SEND_STOP;
        }
        I2CCON |= AAK(1);               //设置应答位

        for(i = 0; i < 20; 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 SEND_STOP;
            }
            ReadBuffer[i] = I2CDAT;     //读取数据到数据寄存器
            if(i < 19)
            {
                I2CCON |= AAK(1);       //如果不是最后一字节，预设 ACK 状态
            }
            else
            {
                I2CCON &= ~AAK(1);  //如果是最后一字节，不发送 ACK
            }
        }
SEND_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 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)
{
        EA = 1;                                         //开全局中断
/**********选择 I2C 端口*****************************************************************/
//      I2CIOS = 0;                 //选择 P30,P31 作为 I2C 通信引脚
//      P30F = 3 | (1<<7);          //设置 P30 作为 I2C SDA，并打开上位
//      P31F = 3 | (1<<7);          //设置 P31 作为 I2C SCL，并打开上位

        I2CIOS = 1;                 //选择 P03,P02 作为 I2C 通信引脚
        P03F = 3|(1<<7);            //设置 P03 作为 I2C SDA，并打开上位
        P02F = 3|(1<<7);            //设置 P02 作为 I2C SCL，并打开上位
//以上两组端口二选一
/*******************************************************************************************/
        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
        INT4EN = 1;                             //I2C 中断开启
        while(1)
        {
        }
}
```

---

18 PWM

18.1 PWM 功能简介

JZ8FC508T 系列芯片最多有 6 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 PWM1（LED_D0）/PWM2(LED_D1)还支持级联 LED 驱动，扫描频率大于 400Hz/S, 数据发送速度 800Kbps,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 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 PWMnDIV 来设置，而寄存器 PWMnDUT 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 PWMnTOG 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 PWMnCON 的 PWMnCKS。另外，每路 PWM 的时钟分频可通过 PWMnCKD 独立设置。

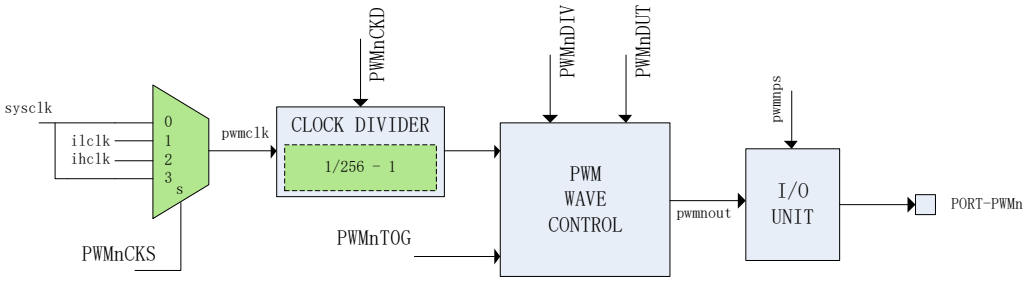


图 18-2-1 PWM 原理示意图

备注：

PWMnDIV，PWMnDUT 等带“n”的寄存器名称，其中“n”表示 0/1/2/3/4/5，分别代表 PWM 通道 0/1/2/3/4/5 这 6 个通道的控制或者配置寄存器。

- **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 PWMnDUT 时，PWM 引脚输出高电平（PWMnTOG=0），当计数值大于 PWMnDUT 时，PWM 引脚输出低电平（PWMnTOG=0）。当计数值与 PWMnDIV 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 PWMnDIV>PWMnDUT>0 时，PWM 波形如图所示。

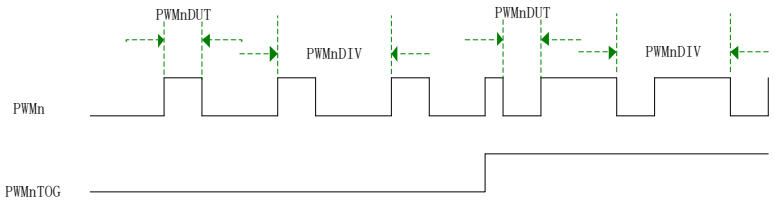


图 18-2-2 PWM 输出波形

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

- **PWM 中断**

PWM 中断通过寄存器 PWMnCON 的 PWMnIE 位使能，当 PWM 计数器计数到顶点（即等于 PWMnDIV）时产生的中断。寄存器 PWMIF 包含 6 个通道的中断标志位。

- **单线级联LED 驱动**

PWM1/PWM2 通道支持单线级联LED 驱动，级联LED 的典型驱动时序图如图 18-2-3 所示。

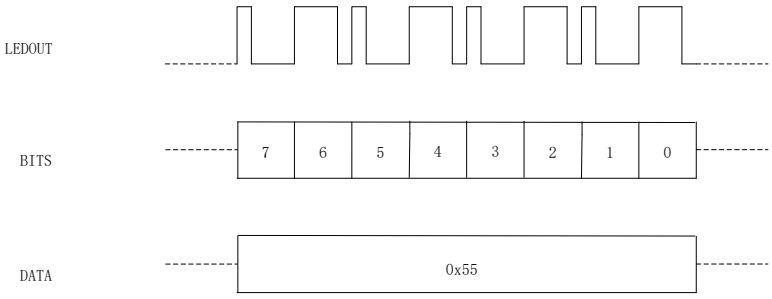


图 18-2-2 级联 LED 时序图

位码示意图如图 18-2-3 所示。

图 18-2-3 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 PWMnDUT(n=1/2)配置，位码 1 的高电平时间宽度由 LEDUTH 配置，而位周期时间由PWMnDIV 配置。当PWMnMOD 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=0/1)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=0/1)位置 0，LEFn(n=0/1)=0 表示可以重新向LEDAT0/LEDAT1 装载数据。当 PWMnMOD 不为 0 时，PWMnMOD 也同时表示发送完PWMnMOD 个字节后插入等待时间，等待时间由LEWTM 来配置。

当 PWMnPOL=1(n=1/2)时，LEDAT0/LEDAT1 的数据反相，即：例如写入 01010101B，实际发送出来的是 10101010B。

18.3 PWM 寄存器描述

表 18-3-1 寄存器 PWMEN

90H	7	6	5	4	3	2	1	0
PWMEN	-	-	PWM5EN	PWM4EN	PWM3EN	PWM2EN	PWM1EN	PWM0EN
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	PWM5EN	PWM5 使能控制位，1 有效
4	PWM4EN	PWM4 使能控制位，1 有效
3	PWM3EN	PWM3 使能控制位，1 有效
2	PWM2EN	PWM2 使能控制位，1 有效
1	PWM1EN	PWM1 使能控制位，1 有效
0	PWM0EN	PWM0 使能控制位，1 有效

表 18-3-2 寄存器 PWMCON

B9H	7	6	5	4	3	2	1	0
PWM0CON	PWM0IE	PWM0TOG	-	-	-	-	PWM0CKS[1:0]	
R/W	R/W	R/W	-	-	-	-	R/W	

初始值	0	0	-	-	-	-	0	0

BAH	7	6	5	4	3	2	1	0
PWM1CON	PWM1IE	PWM1TOG	PWM1MOD[2:0]			PWM1POL	PWM1CKS[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

BBH	7	6	5	4	3	2	1	0
PWM2CON	PWM2IE	PWM2TOG	PWM2MOD[2:0]			PWM2POL	PWM2CKS[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

BCH	7	6	5	4	3	2	1	0
PWM3CON	PWM3IE	PWM3TOG	-	-	-	-	PWM3CKS[1:0]	
R/W	R/W	R/W	-	-	-	-	R/W	
初始值	0	0	-	-	-	-	0	0

BDH	7	6	5	4	3	2	1	0
PWM4CON	PWM4IE	PWM4TOG	-	-	-	-	PWM4CKS[1:0]	
R/W	R/W	R/W	-	-	-	-	R/W	
初始值	0	0	-	-	-	-	0	0

BEH	7	6	5	4	3	2	1	0
PWM5CON	PWM5IE	PWM5TOG	-	-	-	-	PWM5CKS[1:0]	
R/W	R/W	R/W	-	-	-	-	R/W	
初始值	0	0	-	-	-	-	0	0

位编号	位符号	说明
备注：PWM1/PWM2 可驱动级联 LED，所以增加与 LED 驱动有关的控制位。		
7	PWMnIE	PWMn 计数器溢出中断使能控制位，1 有效
6	PWMnTOG	PWMn 输出取反使能控制位，1 有效
5~3	PWMmMOD	PWMm 作为 LED 驱动时，连续发送字节数配置寄存器，0 表示 PWMm 不作为LED 驱动使用，1~7 表示 PWMm 每发送 1~7 字节数据就暂停 1 次 ***备注：*** *1. m 表示 1/2 。* *2. 详细使用参考 LEWTM。*
2	PWMmPOL	PWMm 作为 LED 驱动时，发送数据取反使能控制位，1 有效 ***备注：*** *1 当 PWMmMOD！=0 时，对应的 PWMmPOL 的值才有意义；* *2 当 PWMmPOL=1 时，如果对应的 LEDAT=01010101B，那么实际上发送的将会是 10101010B。*
1~0	PWMnCKS	PWM 工作时钟选择位

		01：IRCL 10：IRCH 其他：系统时钟

表 18-3-3 寄存器 PWMCKD

B1H	7	6	5	4	3	2	1	0
PWM0CKD	PWM0CKD[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
B2H	7	6	5	4	3	2	1	0
PWM1CKD	PWM1CKD[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
B3H	7	6	5	4	3	2	1	0
PWM2CKD	PWM2CKD[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
B4H	7	6	5	4	3	2	1	0
PWM3CKD	PWM3CKD[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
B5H	7	6	5	4	3	2	1	0
PWM4CKD	PWM4CKD[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
B6H	7	6	5	4	3	2	1	0
PWM5CKD	PWM5CKD[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	PWMnCKD	PWM 工作时钟预分频配置寄存器 00H：不分频 01H：2 分频 02H：3 分频 …… FEH：255 分频

		FFH：256 分频

表 18-3-4 寄存器 PWMDIVL、PWMDIVH

A9H	7	6	5	4	3	2	1	0
PWM0DIVL	PWM0DIV[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
AAH	7	6	5	4	3	2	1	0
PWM0DIVH	PWM0DIV[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
ABH	7	6	5	4	3	2	1	0
PWM1DIVL	PWM1DIV[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
ACH	7	6	5	4	3	2	1	0
PWM1DIVH	PWM1DIV[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
ADH	7	6	5	4	3	2	1	0
PWM2DIVL	PWM2DIV[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
AEH	7	6	5	4	3	2	1	0
PWM2DIVH	PWM2DIV[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
AFH	7	6	5	4	3	2	1	0
PWM3DIVL	PWM3DIV[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
PWM3DIVH	PWM3DIV[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

A5H	7	6	5	4	3	2	1	0
PWM4DIVL	PWM4DIV[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
A6H	7	6	5	4	3	2	1	0
PWM4DIVH	PWM4DIV[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
A7H	7	6	5	4	3	2	1	0
PWM5DIVL	PWM5DIV[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
9AH	7	6	5	4	3	2	1	0
PWM5DIVH	PWM5DIV[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	PWMnDIV	PWMn 周期配置寄存器						

表 18-3-5 寄存器 PWMDUTL、PWMDUTH

9BH	7	6	5	4	3	2	1	0
PWM0DUTL	PWM0DUT[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
9CH	7	6	5	4	3	2	1	0
PWM0DUTH	PWM0DUT[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
9DH	7	6	5	4	3	2	1	0
PWM1DUTL	PWM1DUT[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
9EH	7	6	5	4	3	2	1	0

PWM1DUTH	PWM1DUT[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
9FH	7	6	5	4	3	2	1	0
PWM2DUTL	PWM2DUT[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
91H	7	6	5	4	3	2	1	0
PWM2DUTH	PWM2DUT[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
92H	7	6	5	4	3	2	1	0
PWM3DUTL	PWM3DUT[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
93H	7	6	5	4	3	2	1	0
PWM3DUTH	PWM3DUT[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
94H	7	6	5	4	3	2	1	0
PWM4DUTL	PWM4DUT[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
PWM4DUTH	PWM4DUT[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
96H	7	6	5	4	3	2	1	0
PWM5DUTL	PWM5DUT[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
97H	7	6	5	4	3	2	1	0
PWM5DUTH	PWM5DUT[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	PWMnDUT	PWMn 占空比配置寄存器

表 18-3-6 寄存器 PWMIF

B7H	7	6	5	4	3	2	1	0
PWMIF	-	-	PWMIF5	PWMIF4	PWMIF3	PWMIF2	PWMIF1	PWMIF0
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	PWMIF5	PWM5 中断标志位，1 有效，写 1 清 0
4	PWMIF4	PWM4 中断标志位，1 有效，写 1 清 0
3	PWMIF3	PWM3 中断标志位，1 有效，写 1 清 0
2	PWMIF2	PWM2 中断标志位，1 有效，写 1 清 0
1	PWMIF1	PWM1 中断标志位，1 有效，写 1 清 0
0	PWMIF0	PWM0 中断标志位，1 有效，写 1 清 0

表 18-3-7 寄存器 LEDAT

D7H	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

C7H	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 0/1，LED0/LED1 对应 PWM1/PWM2；* *2. LEDAT 的数据按照从 MSB 到LSB 的顺序发送。*

表 18-3-8 寄存器 LEDUTL、LEDUTH

8060H	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
8061H	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
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 PWMmDIV 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/LED2 的数据“1”的占空比都由同一个 LEDUT 决定。*							

表 18-3-9 寄存器 LEWTML、LEWTMH

CEH	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
CFH	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
位编号	位符号	说明						
15~0	LEWTM	LED 暂停时间配置寄存器，结合 PWMmMOD 配置寄存器使用 ***备注：*** *1 每发送 PWMmMOD 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 LED1/LED2 的暂停时间都由同一个 LEWTM 决定。*							

表 18-3-10 寄存器LEFLG

BFH	7	6	5	4	3	2	1	0
LEFLG	-	LEBSY1	-	-	-	LEBSY0	-	-
R/W	-	R	-	-	-	R	-	-
初始值	-	0	-	-	-	0	-	-

位编号	位符号	说明
7	-	
6	LEBSY1	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	-	
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	-	-

18.4 PWM 功能控制例程

◆ **PWM 单路输出例程**

以 PWM0 为例，PWM 时钟源为 IRCH（IRCH 频率为 16MHz），输出频率为 30K 的时钟，占空比为 30%，程序如下：

```
-------------------------------------------------------------------------------------------
//PWMxCON
#define PWMIE(N)            (N<<7)      //N=0-1
#define PWMTOG(N)           (N<<6)      //N=0-1
#define PWMMOD(N)           (N<<3)      //N=0-7
#define PWMPOL(N)           (N<<2)      //N=0-1
#define PWMCKS_SYS          0
#define PWMCKS_IL           1
#define PWMCKS_IH           2

#define IHCKE           (1<<7)

void PWM_init(void)
{
  P32F = 5;       //设置 P32 为 PWM 引脚功能
  CKCON |= IHCKE;   //打开 IRCH 时钟

  PWM0DIVH = 0x02;      //设置 DIV 值，16000000/30000=0x215
  PWM0DIVL = 0x15;

  PWM0DUTH = 0x00;      //设置 DUT 值，占空比为 30%
  PWM0DUTL = 0xA0;

  PWM0CON = PWMIE(0) | PWMTOG(0) | PWMCKS_IH;       //选择 PWM 时钟源为IRCH，不开启中断
  PWM0CKD = 0;                  //设置预分频系数，设置为 0 表示不分频

  PWMEN |= (1<<0);              //PWM0 使能
}
-------------------------------------------------------------------------------------------
```

◆ **单线级联LED驱动灯带控制例程**

以 PWM1 为例，驱动 8 级 RGB LED ，使 RGB LED 循环变色，程序如下：

```
---------------------------------------------------------------------------------------------
    //PWMxCON
  #define PWMIE(N)              (N<<7)      //N=0-1
  #define PWMTOG(N)             (N<<6)      //N=0-1
  #define PWMMOD(N)             (N<<3)      //N=0-7
  #define PWMPOL(N)             (N<<2)      //N=0-1
  #define PWMCKS_SYS      0
  #define PWMCKS_IL             1
  #define PWMCKS_IH             2



  //LEFLG
  #define LEF0      (1<<3)
  #define LEBSY0   (1<<2)

  void PWM_init(void)
  {
     P33F = 5;              //设置 P33 为 PWM 引脚功能
     CKCON |= IHCKE;    //打开 IRCH 时钟

     PWM1DIVH = 0;     //设置位周期时间
     PWM1DIVL = 20;

     PWM1DUTH = 0;     //设置位码 0 时间
     PWM1DUTL = 6;

     LEDUTH = 0;          //设置位码 1 时间
     LEDUTL = 13;

     PWM1CON = PWMIE(0) | PWMTOG(0) | PWMMOD(3) | PWMPOL(0) | PWMCKS_IH;      //设置IRCH 为 PWM 时钟
                                                                                //源，发送 3 个字节后插入暂停时间
     PWM1CKD = 0;      //设置预分频系数，设置为 0 表示不分频

     LEDWTMH = 0;      //设置暂停时间
     LEDWTML = 50;

     PWMEN |= (1<<1);        //PWM1 使能
  }
  code unsigned char LED_DAT[][3] = //LED 数据表
  {
     {0xff,0x00,0x00},
     {0xff,0xff,0x00},
```

```
    {0x00,0xff,0x00},
    {0x00,0xff,0xff},
    {0x00,0x00,0xff},
    {0xff,0x00,0xff},
  };
void main(void)
  {
    unsigned char i;
    unsigned char color_index=0;
    PWM_init();
    while(1)
    {
        for(i=0;i<24;i++)     //每级 RGB LED 为 3 个字节数据，8 级为 24 字节
        {
            while(LEFLG & LEBSY0); //等待数据全部发送

            LEDAT0 = LED_DAT[color_index][i%3];  //写入 LED 数据到LEDAT 寄存器
        }
        color_index ++;              //循环变色
        if(color_index> 6) color_index =0;
        Delay1S();                          //等待 1 秒
        …….
    }
  }
```

---

19 数模转换器（DAK）

19.1 功能简介

此模块包含两路 DA 输出，每路 DA 输出可以设置 32 档电压。设计此模块的主要目的是，在JZ8FC508T芯片作为触摸芯片时，可通过 DA 输出不同电压来表示不同的键值信息，以供主控芯片通过 ADC 来检测。

DAK 通过 AKnCON 中的 AKnE 位使能，设置 AKnE=1 且对应引脚选择 DAK 功能后，DAK 模块就可以根据用户配置的 AKnS 的值，在对应的引脚上输出相应的电压。

备注：n=0/1，分别对应 DAK0/DAK1。

19.2 寄存器描述

表 19-2-1 寄存器 AKCON

DEH	7	6	5	4	3	2	1	0
AK0CON	AK0E	-	-	AK0S[4:0]				
R/W	R/W	-	-	R/W				
初始值	0	-	-	0	0	0	0	0
DFH	7	6	5	4	3	2	1	0
AK1CON	AK1E	-	-	AK1S[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	AKnE	DAKn 通道使能寄存器，1 有效
6~5	-	-
4~0	AKnS	DAKn 输出电压配置寄存器 DAKn 输出电压为： (AKnS+1)÷ 32)×Vdd, 误差率为±1%

19.4 DAK 控制例程

以DAK0 为例，设置DAK0 引脚输出 1/2 Vdd，程序如下：

```
---------------------------------------------------------------------------------------------
//AKxCON 定义
#define AKE(N)              (N<<7)
#define AKS(N)              N                   //N=0-31

void DAK0_init(void)
{
    P00F = 3;
    AK0CON = AKE(1) | AKS(15);
}
---------------------------------------------------------------------------------------------
```

20 电容式触摸按键（Touch Key）

20.1 功能简介

JZ8FC508T 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 13 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 13 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
- 支持触摸引脚与LED 驱动引脚复用
- 内置防水补偿机制
- STOP 模式下可设自动唤醒阈值

20.3 结构图

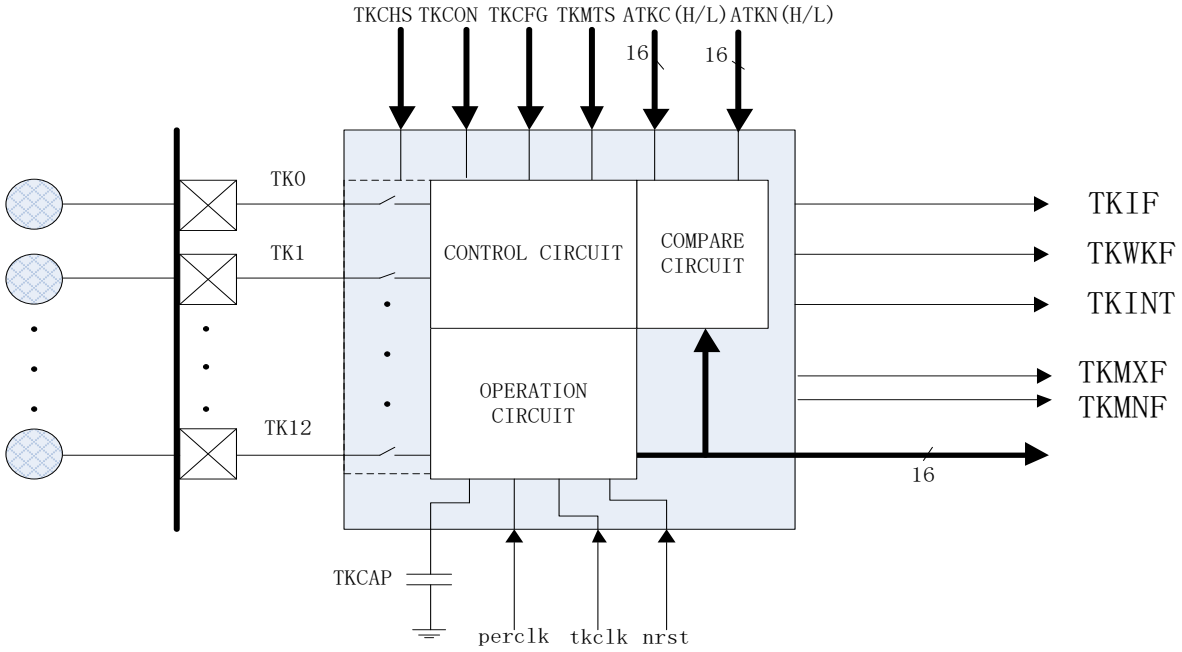


图 20-3-1 触摸模块结构图

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 4 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，如果 TWKE=0，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。另外，触摸模块还设计了触摸阈值自动比较功能，用户可通过阈值设置寄存器设置通道的触发阈值，在 STOP 模式，触摸控制器仍然可以将采集的触摸数据和阈值进行比较，当触摸数据超过阈值时，如果 TWKE=1，会产生阈值触发中断并唤醒 CPU，CPU 被唤醒后就可以进行正常的触摸采集和判断。

20.4.5 触摸按键共用 LED 驱动

触摸按键共用 LED 驱动可实现 N 个触摸按键和 N 个触摸指示灯控制只需要(N+1)个引脚。其中，触摸按键和 LED 驱动正端控制共用引脚，LED 负端接COM，触摸和 LED 控制采用分时的方式实现。

每一路触摸都有单独的控制位TLENx(x=0~12,对应TK0~TK12)来使能共用LED 驱动功能，需要注意的是，对应的触摸引脚功能必须开启。共用 LED 使能后，TLDATx(x=0~12，对应 LED0~LED12)可独立控制每路LED 灯亮灭。COM 引脚为P01。

触摸数据采集和LED 控制采用分时的方式实现，其中触摸数据采集的时间由 TLCNTK 定义，而 LED 扫描的时间由 TLCNTL 定义。注意，TLCNTK 定义的时间是每组触摸采集的总时间，每组触摸通道数量为 1~6，当实际触摸的时间大于定义的时间时，会产生TLERR 中断。当计数器计数到 TLCNTK 定义的时间时，产生TLKOV 中断。触摸采集阶段完成后，便进入 LED 扫描阶段。TLCNTL 定义了LED 扫描阶段的时间，此时间会影响 LED 扫描的占空比，也就是会影响LED 的亮度，在应用时可以根据需要调整。在 LED 扫描阶段，当计数器计到 TLCNTL 定义的时间时，会产生TLLOV 中断，至此，一个完整的触摸共用LED 周期完成。以下为工作阶段示意图。

重要提醒：*在触摸引脚与 LED 驱动引脚复用模式应用中，由于 LED 灯本身存在二极管结电容，不同种类的 LED 灯的结电容存在较大差异，并且此结电容在 LED 灯亮与灭时表现有可能不一致(尤其是白色 LED 灯比较明显)，此结电容及其不一致性会对触摸造成不良影响，所以在应用此模式时应对所使用的 LED 灯严格挑选，并且量产之后不能随便更换 LED 灯品种。*

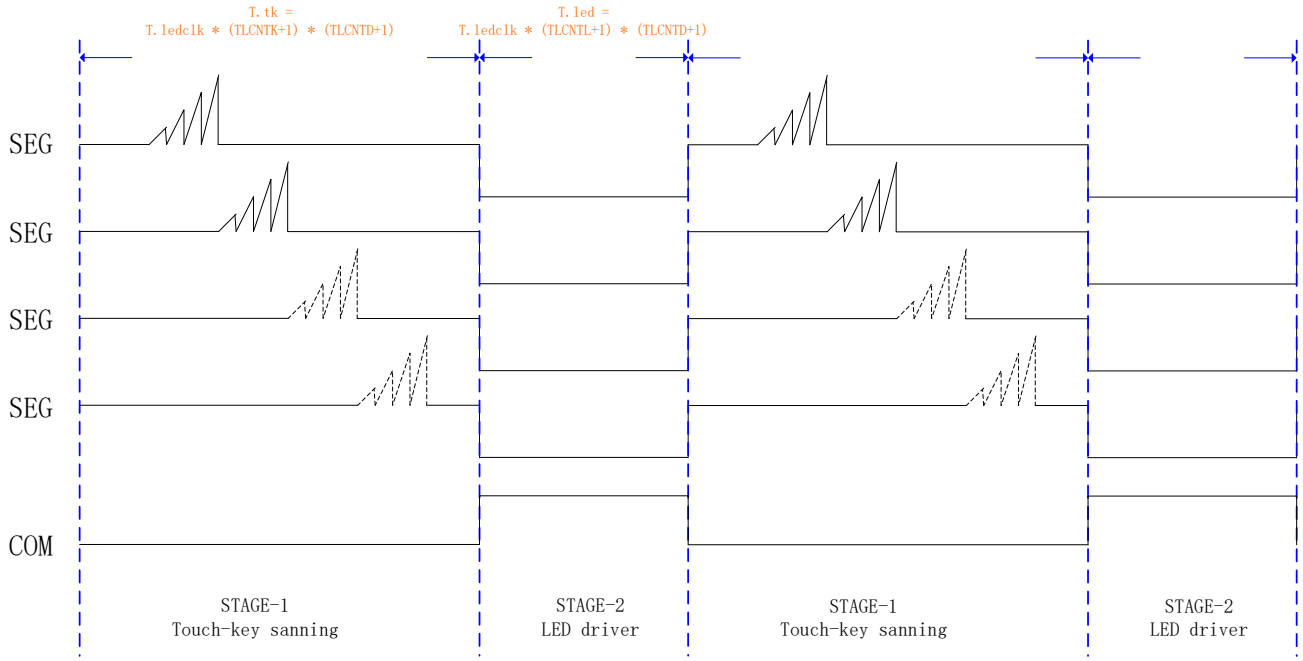


图 20-4-5 触摸共用 LED 驱动示意图

20.5 寄存器描述

表 20-5-1 寄存器TKCON

F8H	7	6	5	4	3	2	1	0
TKCON	TKST	TKIE	TMEN	TWKE	-	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	TWKE	中断触发方式选择位 0:采样完成触发中断 1:超出阈值范围触发中断
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

8103H	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：* *1. 这一功能仅对被配置为触摸按键功能的引脚有效。* *2. 如果共用 LED 驱动功能使能，那么被选中作为 LED 驱动的引脚，这一功能将失效。*
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

8102H	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 16M 的四分频(4M)时钟 1：选择慢速(131K)时钟

表 20-5-4 寄存器TKCFG

F9H	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

FAH	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

FCH	7	6	5	4	3	2	1	0
TK0CHS	POL0	NPOL0	-	-	TKPS0[3:0]			

R/W	R/W	R/W	-	-	R/W			
初始值	0	0	-	-	0	0	0	0
FDH	7	6	5	4	3	2	1	0
TK1CHS	POL1	NPOL1	-	-	TKPS1[3:0]			
R/W	R/W	R/W	-	-	R/W			
初始值	0	0	-	-	0	0	0	0
FEH	7	6	5	4	3	2	1	0
TK2CHS	POL2	NPOL2	-	-	TKPS2[3:0]			
R/W	R/W	R/W	-	-	R/W			
初始值	0	0	-	-	0	0	0	0
FFH	7	6	5	4	3	2	1	0
TK3CHS	POL3	NPOL3	-	-	TKPS3[3:0]			
R/W	R/W	R/W	-	-	R/W			
初始值	0	0	-	-	0	0	0	0
F1H	7	6	5	4	3	2	1	0
TK4CHS	POL4	NPOL4	-	-	TKPS4[3:0]			
R/W	R/W	R/W	-	-	R/W			
初始值	0	0	-	-	0	0	0	0
F2H	7	6	5	4	3	2	1	0
TK5CHS	POL5	NPOL5	-	-	TKPS5[3:0]			
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	POLn	ATKnC 阈值比较方向设置位 0:触摸数据小于阈值时触发阈值比较中断 1:触摸数据大于或等于阈值时触发阈值比较中断
6	NPOLn	ATKnN 阈值比较方向设置位 0:触摸数据小于阈值时触发阈值比较中断 1:触摸数据大于或等于阈值时触发阈值比较中断
5~4	-	-
3~0	TKPSn	通道 n 选择位域 0000：TK0~TK12 关闭 0001：选择 TK0 0010：选择 TK1 0011：选择 TK2 …… 1101：选择 TK12 1110：选择内部参考电容

表 20-5-7 寄存器 ATKC

F3H	7	6	5	4	3	2	1	0

ATK0CL	ATK0C[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
F4H	7	6	5	4	3	2	1	0
ATK0CH	ATK0C[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
F5H	7	6	5	4	3	2	1	0
ATK1CL	ATK1C[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
F6H	7	6	5	4	3	2	1	0
ATK1CH	ATK1C[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
F7H	7	6	5	4	3	2	1	0
ATK2CL	ATK2C[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
E9H	7	6	5	4	3	2	1	0
ATK2CH	ATK2C[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
EAH	7	6	5	4	3	2	1	0
ATK3CL	ATK3C[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
EBH	7	6	5	4	3	2	1	0
ATK3CH	ATK3C[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
ECH	7	6	5	4	3	2	1	0
ATK4CL	ATK4C[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
EDH	7	6	5	4	3	2	1	0
ATK4CH	ATK4C[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
EEH	7	6	5	4	3	2	1	0
ATK5CL	ATK5C[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

EFH	7	6	5	4	3	2	1	0
ATK5CH	ATK5C[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	ATKnC	比较阈值设置寄存器，当 TWKE=1 时，ATK0C~ATK5C 自动与 TK0MS~TK5MS 比较

表 20-5-8 寄存器 ATKN

8050H	7	6	5	4	3	2	1	0
ATK0NL	ATK0N[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
8051H	7	6	5	4	3	2	1	0
ATK0NH	ATK0N[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
8052H	7	6	5	4	3	2	1	0
ATK1NL	ATK1N[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
8053H	7	6	5	4	3	2	1	0
ATK1NH	ATK1N[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
8054H	7	6	5	4	3	2	1	0
ATK2NL	ATK2N[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
8055H	7	6	5	4	3	2	1	0
ATK2NH	ATK2N[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
8056H	7	6	5	4	3	2	1	0
ATK3NL	ATK3N[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
8057H	7	6	5	4	3	2	1	0
ATK3NH	ATK3N[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

8058H	7	6	5	4	3	2	1	0
ATK4NL	ATK4N[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
8059H	7	6	5	4	3	2	1	0
ATK4NH	ATK4N[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
805AH	7	6	5	4	3	2	1	0
ATK5NL	ATK5N[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
805BH	7	6	5	4	3	2	1	0
ATK5NH	ATK5N[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	ATKnN	比较阈值设置寄存器，当 TWKE=1 时，ATK0N~ATK5N 自动与 TK0MS~TK5MS 比较

表 20-5-9 寄存器TKMS

E1H	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
E2H	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
E3H	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
E4H	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
E5H	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

E6H	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
E7H	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
D9H	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
DAH	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
DBH	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
DCH	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
DDH	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

FBH	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，当 TWKE=1 时，TKIF5

		表示 TK5MS 超出 ATK5C 或者 ATK5N 的阈值范围，写 1 清 0
4	TKIF4	6 个选定通道中的第 5 通道触摸采集中断标志位，当 TWKE=1 时，TKIF4 表示 TK4MS 超出 ATK4C 或者 ATK4N 的阈值范围，写 1 清 0
3	TKIF3	6 个选定通道中的第 4 通道触摸采集中断标志位，当 TWKE=1 时，TKIF3 表示 TK3MS 超出 ATK3C 或者 ATK3N 的阈值范围，写 1 清 0
2	TKIF2	6 个选定通道中的第 3 通道触摸采集中断标志位，当 TWKE=1 时，TKIF2 表示 TK2MS 超出 ATK2C 或者 ATK2N 的阈值范围，写 1 清 0
1	TKIF1	6 个选定通道中的第 2 通道触摸采集中断标志位，当 TWKE=1 时，TKIF1 表示 TK1MS 超出 ATK1C 或者 ATK1N 的阈值范围，写 1 清 0
0	TKIF0	6 个选定通道中的第 1 通道触摸采集中断标志位，当 TWKE=1 时，TKIF0 表示 TK0MS 超出 ATK0C 或者 ATK0N 的阈值范围，写 1 清 0

表 20-5-11 寄存器TKMAXF

805CH	7	6	5	4	3	2	1	0
TKMAXF	-	-	TKMXF5	TKMXF4	TKMXF3	TKMXF2	TKMXF1	TKMXF0
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	TKMXF5	1 表示 TK5MS 超出 ATK5C 的阈值范围，0 表示TK5MS 不超出 ATK5C 的阈值，极性由 POL5 决定；如果 TWKE=1，那么置位 TKMXF5 的同时会置位 TKIF5；软件无法对其操作
4	TKMXF4	1 表示 TK4MS 超出 ATK4C 的阈值范围，0 表示TK4MS 不超出 ATK4C 的阈值，极性由 POL4 决定；如果 TWKE=1，那么置位 TKMXF4 的同时会置位 TKIF4；软件无法对其操作
3	TKMXF3	1 表示 TK3MS 超出 ATK3C 的阈值范围，0 表示TK3MS 不超出 ATK3C 的阈值，极性由 POL3 决定；如果 TWKE=1，那么置位 TKMXF3 的同时会置位 TKIF3；软件无法对其操作
2	TKMXF2	1 表示 TK2MS 超出 ATK2C 的阈值范围，0 表示TK2MS 不超出 ATK2C 的阈值，极性由 POL2 决定；如果 TWKE=1，那么置位 TKMXF2 的同时会置位 TKIF2；软件无法对其操作
1	TKMXF1	1 表示 TK1MS 超出 ATK1C 的阈值范围，0 表示TK1MS 不超出 ATK1C 的阈值，极性由 POL1 决定；如果 TWKE=1，那么置位 TKMXF1 的同时会置位 TKIF1；软件无法对其操作
0	TKMXF0	1 表示 TK0MS 超出 ATK0C 的阈值范围，0 表示TK0MS 不超出 ATK0C 的阈值，极性由 POL5 决定；如果 TWKE=1，那么置位 TKMXF0 的同时会置位 TKIF0；软件无法对其操作

表 20-5-12 寄存器TKMINF

805DH	7	6	5	4	3	2	1	0

TKMINF	-	-	TKMNF5	TKMNF4	TKMNF3	TKMNF2	TKMNF1	TKMNF0
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	TKMNF5	1 表示 TK5MS 超出 ATK5N 的阈值范围，0 表示 TK5MS 不超出 ATK5N 的阈值，极性由 NPOL5 决定；如果 TWKE=1，那么置位 TKMNF5 的同时会置位 TKIF5；软件无法对其操作
4	TKMNF4	1 表示 TK4MS 超出 ATK4N 的阈值范围，0 表示 TK4MS 不超出 ATK4N 的阈值，极性由 NPOL4 决定；如果 TWKE=1，那么置位 TKMNF4 的同时会置位 TKIF4；软件无法对其操作
3	TKMNF3	1 表示 TK3MS 超出 ATK3N 的阈值范围，0 表示 TK3MS 不超出 ATK3N 的阈值，极性由 NPOL3 决定；如果 TWKE=1，那么置位 TKMNF3 的同时会置位 TKIF3；软件无法对其操作
2	TKMNF2	1 表示 TK2MS 超出 ATK2N 的阈值范围，0 表示 TK2MS 不超出 ATK2N 的阈值，极性由 NPOL2 决定；如果 TWKE=1，那么置位 TKMNF2 的同时会置位 TKIF2；软件无法对其操作
1	TKMNF1	1 表示 TK1MS 超出 ATK1N 的阈值范围，0 表示 TK1MS 不超出 ATK1N 的阈值，极性由 NPOL1 决定；如果 TWKE=1，那么置位 TKMNF1 的同时会置位 TKIF1；软件无法对其操作
0	TKMNF0	1 表示 TK0MS 超出 ATK0N 的阈值范围，0 表示 TK0MS 不超出 ATK0N 的阈值，极性由 NPOL5 决定；如果 TWKE=1，那么置位 TKMNF0 的同时会置位 TKIF0；软件无法对其操作

表 20-5-13 寄存器 TLEN

8106H	7	6	5	4	3	2	1	0
TLEN0	TLEN7	TLEN6	TLEN5	TLEN4	TLEN3	TLEN2	TLEN1	TLEN0
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
8111H	7	6	5	4	3	2	1	0
TLEN1	-	-	-	TLEN12	TLEN11	TLEN10	TLEN9	TLEN8
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W
初始值	-	-	-	0	0	0	0	0

备注：

1. TLENx=1 (x=0,1,2,...,12)，LEDx 要使能，必须对应的 TKx 引脚选择触摸按键功能。

2. 用户可以在触摸按键引脚范围内任意选择若干或全部作为共用 LED 驱动的引脚。

位编号	位符号	说明
7~5 (TLEN1)	-	-
4~0 (TLEN1)	TLEN12~TLEN8	LED12~LED8 使能，1 有效
7~0 (TLEN0)	TLEN7~TLEN0	LED7~LED0 使能，1 有效

表 20-5-14 寄存器 TLDAT

8107H	7	6	5	4	3	2	1	0
TLDAT0	TLDAT7	TLDAT6	TLDAT5	TLDAT4	TLDAT3	TLDAT2	TLDAT1	TLDAT0
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
8112H	7	6	5	4	3	2	1	0
TLDAT1	-	-	-	TLDAT12	TLDAT11	TLDAT10	TLDAT9	TLDAT8
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W
初始值	-	-	-	0	0	0	0	0

位编号	位符号	说明
7~0 (TLDAT1)	-	-
7~0 (TLDAT1)	TLDAT12~TLDAT8	LED12~LED8 数据，1 表示驱动有效电平
7~0 (TLDAT0)	TLDAT7~TLDAT0	LED7~LED0 数据，1 表示驱动有效电平

表 20-5-15 寄存器 TLCON

8108H	7	6	5	4	3	2	1	0
TLCON	TLEIE	TLKIE	TLLIE	-	-	TLLVS		TLPOL
R/W	R	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	TLEIE	TLERR 中断使能，1 有效
6	TLKIE	TLKOV 中断使能，1 有效
5	TLLIE	TLLOV 中断使能，1 有效
4~3	-	-
2~1	TLLVS	触摸按键扫描阶段，COM 引脚电平选择 00：输出高 01：输出低 10：上拉高 其他：保留
0	TLPOL	LED 驱动阶段，有效驱动电平选择 0：驱动电平高有效 1：驱动电平低有效

表 20-5-16 寄存器 TLFLG

8109H	7	6	5	4	3	2	1	0
TLFLG	TLERR	TLKOV	TLLOV	-	-	-	-	-

R/W	R/W	R/W	R/W	-	-	-	-	-
初始值	0	0	0	-	-	-	-	-

位编号	位符号	说明
7	TLERR	触摸按键扫描阶段时间设置过短标志 0：表示用户设置的时间足够触摸按键扫描的时间 1：表示用户设置的时间不足，定时器计数完还有通道没扫描完 *备注：* *此位为硬件自动置位和自动清零位，考虑到用户使用的便捷性，此位也可以用软件对其写 1 清 0。*
6	TLKOV	触摸按键扫描阶段定时器计满标志，1 表示计满，软件写 1 清 0
5	TLLOV	LED 驱动阶段定时器计满标志，1 表示计满，软件写 1 清 0
4~0	-	-

表 20-5-17 寄存器 TLCKS

810AH	7	6	5	4	3	2	1	0
TLCKS	-	-	-	-	-	TLCKS[2:0]		
R/W	-	-	-	-	-	R/W		
初始值	-	-	-	-	-	0	0	0

位编号	位符号	说明
7~3	-	-
2~0	TLCKS	LED 驱动工作时钟选择 001：IRCL 010：IRCH/4 其他：关闭

表 20-5-18 寄存器 TLCNTK

810BH	7	6	5	4	3	2	1	0
TLCNTKL	TLCNTK[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
810CH	7	6	5	4	3	2	1	0
TLCNTKH	TLCNTK[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	TLCNTK	触摸按键扫描阶段时间配置寄存器

		$T_{.tk} = T_{.ledclk}$ x (TLCNTK+1) x (TLDIV+1) ***备注：*** *1. T.tk，表示触摸按键扫描阶段的时间；T.ledclk，表示 LED 驱动电路的工作时钟的周期。* *2. 触摸按键扫描阶段的时间，最小必须满足 1 路触摸按键扫描的时间，最大只需满足 6 路触摸按键扫描的时间，哪怕用户选择了全部的 20 路通道，但是电路每次最多只扫描 6 路，然后再分时扫描其他路。* *3. 触摸按键每通道扫描的时间大约需要 1ms。*

表 20-5-19 寄存器 TLCNTL

810DH	7	6	5	4	3	2	1	0
TLCNTLL	TLCNTL[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
810EH	7	6	5	4	3	2	1	0
TLCNTLH	TLCNTL[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	TLCNTL	LED 驱动阶段时间配置寄存器 $T_{.led} = T_{.ledclk}$x (TLCNTL+1) x (TLDIV+1) xN ***备注：*** *1. T.led，表示 LED 驱动阶段的时间；N，表示所有使能的 LED 驱动的个数，即有效的 COM 的总数。* *2. 用户配触摸按键扫描阶段的时间和 LED 驱动阶段的时间，除了保证各个阶段有足够时间，还必须考虑触摸按键扫描灵敏度以及 LED 驱动波形的频率，一般 LED 驱动的频率必须在 64Hz 以上。* *3. 接第 2. 点，用户一般不要设置太多路 LED 驱动，但理论上可以选择全部的 20 路，只要控制好时间。*

表 20-5-20 寄存器 TLDIV

810FH	7	6	5	4	3	2	1	0
TLDIV	-	-	-	-	TLDIV[3:0]			
R/W	-	-	-	-	R/W			
初始值	-	-	-	-	0	0	0	0

位编号	位符号	说明

7~4	-	-
3~0	TLDIV	$T_{.ledclk}$ 时钟分频寄存器 分频倍数为（TLDIV+1）

表 20-5-21 寄存器 TLCOMS

8110H	7	6	5	4	3	2	1	0
TLCOMS	-	-	-	-	TLCMS[3:0]			
R/W	-	-	-	-	R/W			
初始值	-	-	-	-	0	0	0	0

位编号	位符号	说明
7~4	-	-
3~0	TLCMS	COM 引脚选择控制 0001：选择 P0.1 其他：关闭

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 低电压检测（LVD）

21.1 功能简介

低电压检测（LVD）除了可监控芯片自身的供电 VDD 外，还可以用来检测外部引脚输入电压，可设置检测电压范围为 1.7V~4.8V。当检测的电压低于所设定的电压值时，可设置触发中断或复位。

LVD 结构图如图 21-1-1 所示。

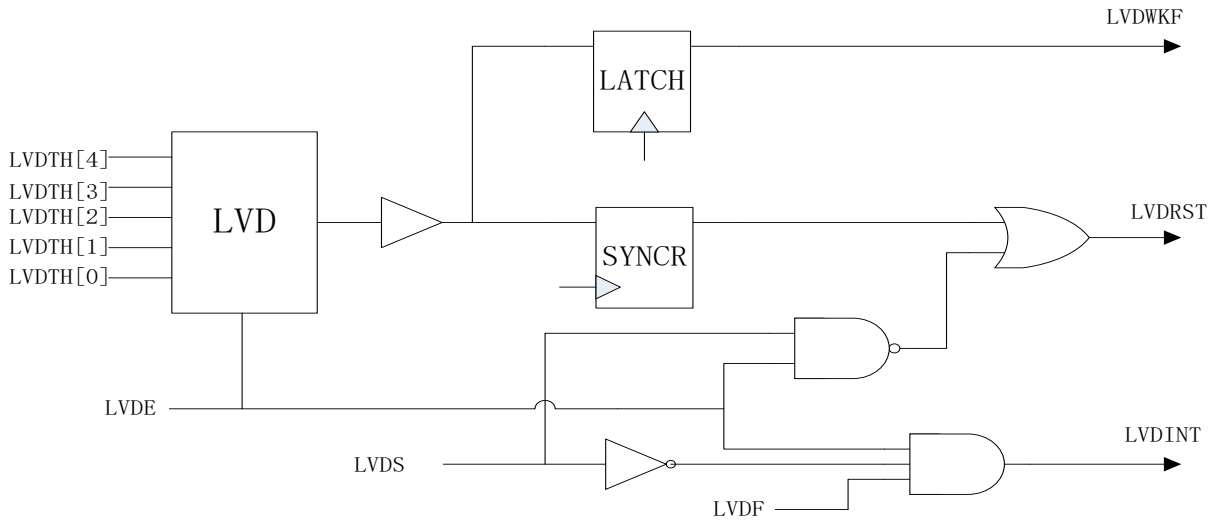


图 21-1-1 LVD 模块示意图

21.2 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。LVD 检测电压源由LVSEL 位选择，LVSEL=0 时，LVD 检测VDD 电压，当 LVSEL=1，LVD 检测外部引脚输入电压，输入引脚由XLVSEL 选择为P00 或 P01。当检测的电压低于所设置的电压时，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，如果检测的电压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，如果检测的电压持续低于所设定的电压，LVD 中断也会重复地产生。

21.3 寄存器描述

表 21-3-1 寄存器 LVDCON

E8H	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
4~0	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

表 21-3-2 寄存器LVDCFG

8105H	7	6	5	4	3	2	1	0
LVDCFG	LVSEL	XLVSEL	-	-	-	-	-	-
R/W	R/W	R/W	-	-	-	-	-	-
初始值	0	0	-	-	-	-	-	-

位编号	位符号	说明
7	LVSEL	LVD 检测电压选择 0：检测 VDD 电压 1：检测外部电压
6	XLVSEL	LVD 检测外部电压引脚选择 0：检测 P0.0 电压 1：检测 P0.1 电压
5~0	-	-

21.4 LVD 控制例程

LVD 中断例程

例如，设置 LVD 为中断模式，检测VDD，电压为 3V，程序如下：

```
-------------------------------------------------------------------------------------------
#define LVSEL(N)        (N<<7)    //N=0~1
#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)
{
    LVDCFG = LVSEL(0);          //设置为检测 VDD
    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 6
{
    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VDD，电压为 3V，程序如下：

```
-------------------------------------------------------------------------------------------
#define LVSEL(N)        (N<<7)    //N=0~1
#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)
```

```
{
    LVDCFG = LVSEL(0);          //设置为检测 VDD
    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
}
```

22 程序下载和仿真

22.1 程序下载

JZ8FC508T 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 P30(I2C SDA),P31(I2C SCL)。

更多关于程序下载步骤的细节请参考“ 开发下载工具使用说明”。

22.2 在线仿真

JZ8FC508T 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P30(I2C SDA）和 P31(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。

23 电气特性

23.1 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	℃
最高频率		16	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。*

23.2 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=1.8V		1.28		mA	系统时钟为 IRCH(16MHz)，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		1.50			
		VDD=5V		1.55			
	Iop3	VDD=1.8V		19.2		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		20.2			
		VDD=5V		20.9			
STOP 模式电流	I_{stp}	VDD=1.8V		5.1		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		5.3			
		VDD=5V		5.7			
IDLE 模式电流	I_{idl1}	VDD=1.8V		0.537		mA	系统时钟设为 IRCH（16MHz），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		0.629			
		VDD=5V		0.641			
	I_{idl3}	VDD=1.8V		11		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
		VDD=3.3V		11.6			
		VDD=5V		12.1			

							设置为低功率模式，CPU 进入IDLE 模式。
IO 端口输入高电压（斯密特模式开启）	Vhi1	VDD=1.8V	0.75	-	1.8	V	-
		VDD=3.3V	1.20		3.3		
		VDD=5V	1.50		5		
IO 端口输入高电压（斯密特模式关闭）	Vhi2	VDD=1.8V		0.5*VDD	VDD	V	-
		VDD=3.3V					
		VDD=5V					
IO 端口输入低电压（斯密特模式开启）	Vlo1	VDD=1.8V	0	-	0.62	V	-
		VDD=3.3V	0	-	0.85		
		VDD=5V	0	-	1.20		
IO 端口输入低电压（斯密特模式关闭）	Vlo2	VDD=1.8V	0	0.5*VDD		V	-
		VDD=3.3V					
		VDD=5V					
IO 端口推电流	Ipu	VDD=3.3V	-	4.27	-	mA	IO 设为推挽输出模式，驱动能力设为最大，Vol=VDD-0.3V
		VDD=5V	-	6.07	-		
IO 端口灌电流	I_{ol}	VDD=3.3V	-	11.33	-	mA	IO 设为推挽输出模式，驱动能力设为最大，Vol=GND+0.3V
		VDD=5V	-	16.05	-		
IO 端口强下拉电阻	Rd1	VDD=1.8~5.5V		15		KΩ	-
IO 端口弱下拉电阻	Rd2	VDD=1.8~5.5V	-	45	-	KΩ	-
IO 端口强上拉电阻	Ru1	VDD=1.8~5.5V	-	10	-	KΩ	-
IO 端口弱上拉电阻	Ru2	VDD=1.8~5.5V		45		KΩ	

说明：以上参数是随机抽取的典型芯片测试结果，仅供参考。

23.3 交流电气特性

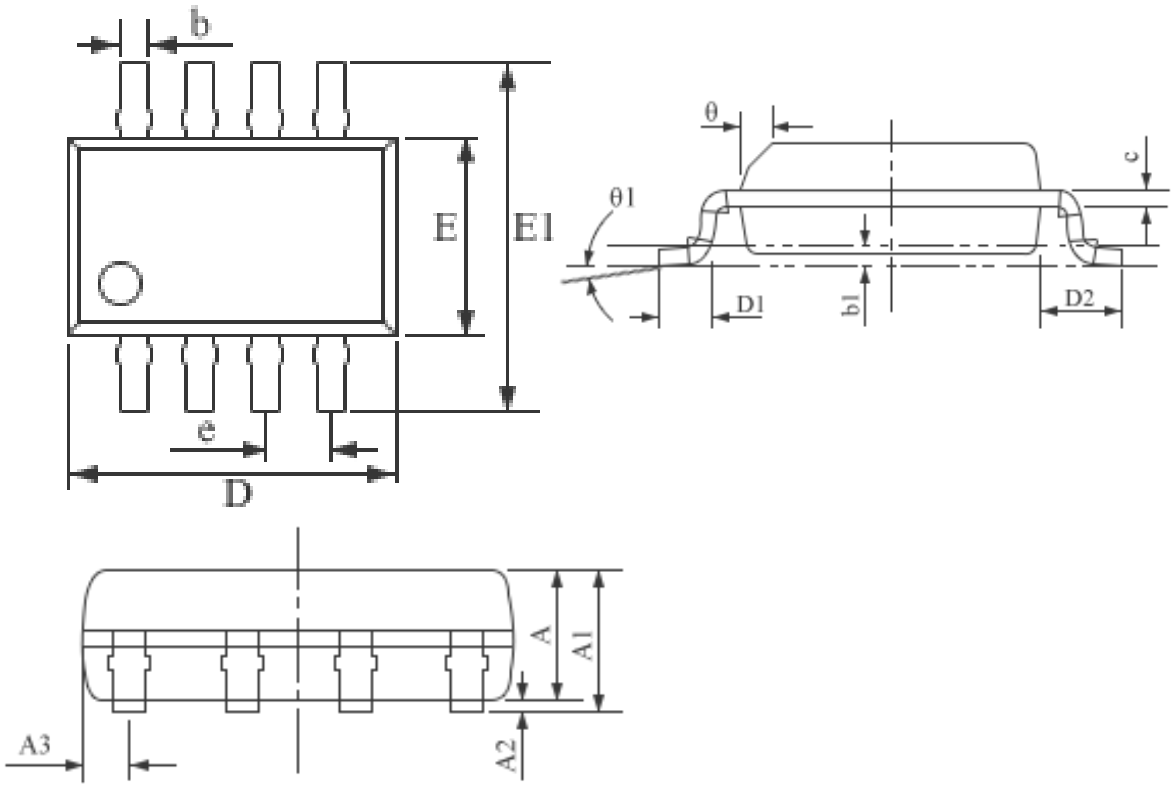
交流电气特性（VDD=1.8-5.5V, TA=25℃，除非其它说明）

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 16MHz
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 16MHz，精度为±1%.

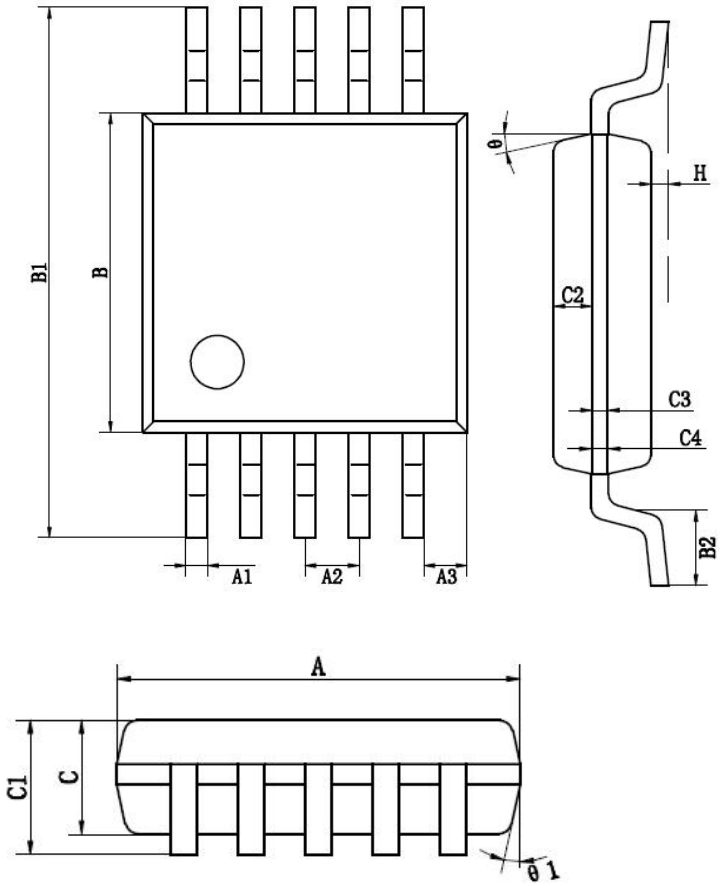
24 封装类型

封装形式（一）(SOP8)



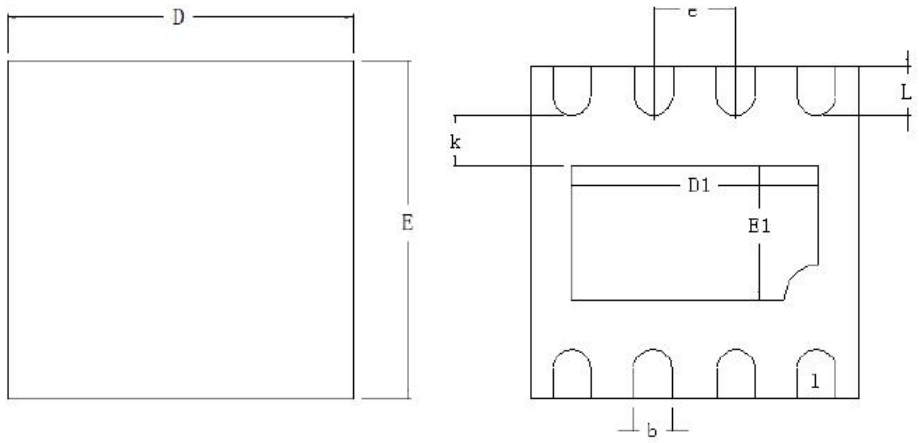
序号	最小值(mm)	标准值(mm)	最大值(mm)
A	1.40	1.45	1.50
A1	1.55	1.60	1.65
A2	0.10	0.15	0.20
A3	0.50	0.535	0.540
b	0.354	0.406	0.504
b1	0.155	0.150	0.175
c	0.20	0.203	0.210
D	4.830	4.880	4.910
D1	0.610	0.660	0.710
D2	1.045	1.050	1.0505
e	———	1.270	———
E	3.810	3.910	3.96
E1	5.900	6.000	6.10

封装形式（二）(MSOP10)

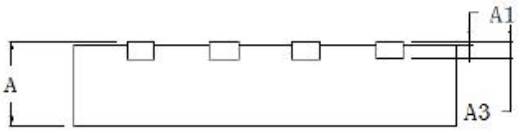


序号	最小值(mm)	标准值(mm)	最大值(mm)
A	2.90	3.00	3.10
A1	0.18	0.20	0.25
A2	0.50TYP		
A3	0.40TYP		
B	2.90	3.00	3.10
B1	4.70	4.90	5.10
B2	0.45	0.60	0.75
C	0.75	0.85	0.95
C1	---	---	1.10
C2	0.328TYP		
C3	0.152		
C4	0.15	0.19	0.23
H	0.02	---	0.15

封装形式（三）(DFN8L　2X2MM)



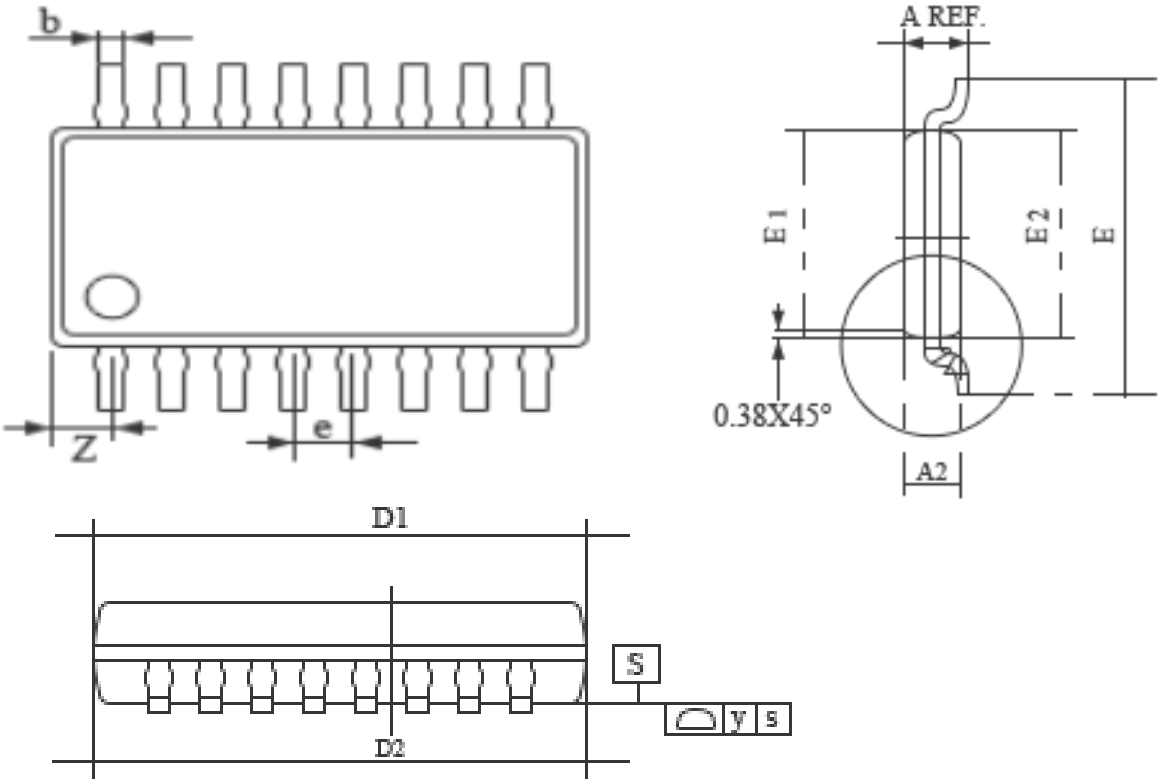
Top View　　Bottom View



Side View

序号	最小值(mm)	标准值(mm)	最大值(mm)
A	0.70	0.75	0.80
A1	0	0.02	0.05
A3		0.203 REF	
D	1.85	2.00	2.15
E	1.85	2.00	2.15
D1	1.45	1.50	1.55
E1	0.75	0.80	0.85
K		0.30 BSC	
b	0.20	0.23	0.26
e		0.50	
L	0.30	0.35	0.40

封装形式（四）(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	————

25 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	4

		.6,...,temp.0) (B)←(temp.15,temp.14,...,temp.8)	
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
		
		IF (A.7,...,A.4) > 9 || CY = 1
		THEN
		temp16 ← (A) + 0x60
		(A) ← (temp.7,...,temp.0)
		
		IF (temp16) > 0xFF
		THEN
		CY ← 1
			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 ((SP)) ← (PC7-0) (SP) ← (SP) + 1 ((SP)) ← (PC15-8) (PC10-0) ← page address	2
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))	2

		(SP) ← (SP) - 1	
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 IF (bit) = 1 THEN (PC) ← (PC) + rel	2
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		