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

CA51F4 系列 MCU
中文用户手册

REV 2.2

深圳市锦锐科技股份有限公司

电话：0755-83949938

传真：0755-83949977

http://www.cachip.com.cn

地址：中国广东省深圳市南山区沙河西路深圳湾科技生态园一区 2 栋 B 座 5 层

重要声明：本公司保留对以下所有产品在可靠性、功能和设计方面作进一步说明的权利，同时保留在未通知的情况下，对本产品所有文档做更改的权利。

客户在使用此产品时，请向我公司销售人员索取最新文档。特此声明！

目录

1 概述.....5
2 基本特性.....5
3 芯片型号功能介绍.....8
4 系统框图.....9
5 引脚封装及其描述.....10
5.1 封装定义.....10
5.2 PAD 配置.....错误!未定义书签。
5.3 引脚描述.....12
6 中央处理器（CPU）.....16
6.1CPU 简介.....16
6.2 寄存器描述.....16
7 存储器系统.....20
7.1 随机数据存储器（RAM）.....20
7.2 特殊功能寄存器（SFR）.....20
7.3 Flash 存储器.....22
7.3.1 功能简介.....22
7.3.2 Flash 存储器组织结构.....22
7.3.3 Flash 寄存器描述.....22
7.3.4 Flash 控制例程.....25
7.4 外部 RAM 映射为程序空间.....27
8 中断系统.....28
8.1 功能简介.....28
8.2 中断逻辑.....28
8.3 中断向量表.....29
8.4 中断控制寄存器.....29
8.5 外部中断.....32
8.5.1 外部中断介绍.....32
8.5.2 外部中断寄存器.....32
8.5.3 外部中断控制例程.....36
9 时钟系统.....38
9.1 时钟系统介绍.....38
9.1.1 时钟专用名称定义.....38
9.1.2 内置 16MHz RC 振荡器（IRCH）.....38
9.1.3 内置 131 KHz RC 振荡器（IRCL）.....38
9.1.4 外部 32.768KHz 晶体谐振器（XOSCL）.....39
9.2 时钟控制寄存器描述.....39
9.3 系统时钟.....42
9.3.1 系统时钟结构图.....42
9.3.2 系统时钟控制寄存器描述.....42
9.3.3 系统时钟控制方法及例程.....44
10 供电和复位系统.....45

10.1 供电系统 45
10.1.1 LDO 功能简介 45
10.1.2 LDO 控制寄存器 46
10.2 复位系统 48
11 功耗管理 50
11.1 IDLE 模式 50
11.2 STOP 模式 50
11.3 低速运行模式 51
11.4 低功耗相关寄存器描述 51
11.5 低功耗模式控制例程 53
12 通用定时器（定时器 0,定时器 1,定时器 2） 55
12.1 定时器 0 55
12.1.1 定时器 0 介绍 55
12.1.2 定时器 0 寄存器描述 56
12.2 定时器 1 58
12.2.1 定时器 1 介绍 58
12.2.2 定时器 1 寄存器描述 59
12.3 定时器 2 60
12.3.1 功能简介 60
12.3.2 定时器 2 寄存器描述 61
13 看门狗定时器（WDT） 64
13.1 看门狗定时器(WDT)功能简介 64
13.2 看门狗定时器(WDT)寄存器描述 64
13.3 看门狗定时器控制例程 66
14 实时定时器（RTC） 68
14.1 RTC 功能简介 68
14.2 RTC 寄存器描述 69
14.3 RTC 控制例程 73
15 通用输入输出口（GPIO）及复用定义 76
15.1 功能简介 76
15.2 引脚寄存器描述 77
15.3 引脚控制例程 92
16 通用串行接口（UART） 94
16.1 UART 94
16.1.1 介绍 94
16.1.2 UART 寄存器描述 95
17 I^2C 接口 97
17.1 功能简介 97
17.2 I^2C 主要特点 97
17.3 I^2C 功能描述 97
17.4 寄存器描述 99
17.6 I^2C 控制例程 102
18 PWM 108
18.1 PWM 功能简介 108

18.2 PWM 功能描述 108
18.3 PWM 寄存器描述 109
18.4 PWM 功能控制例程 113
19 模/数字转换器（ADC） 115
19.1 功能简介 115
19.2 主要特性 115
19.3 结构框图 115
19.4 功能描述 116
19.5 寄存器描述 117
19.6 ADC 控制例程 119
20 触摸按键（TOUCH KEY） 121
20.1 功能简介 121
20.2 主要特性 121
20.3 结构图 122
20.4 功能描述 122
20.4.1 手动模式和自动模式 122
20.4.2 触摸时钟预分频 122
20.4.3 低功耗模式 123
20.4.4 触摸按键共用 LED 驱动功能描述 123
20.4.5 触摸内部基准和内部运放 124
20.4.6 触摸防水补偿机制 124
20.5 寄存器描述 124
21 低电压检测（LVD） 133
21.1 功能简介 133
21.2 功能描述 133
21.3 寄存器描述 134
21.4 LVD 控制例程 135
22 LCD 驱动 136
22.1 LCD 驱动 136
22.1.1 功能简介 136
22.1.2 LCD 偏压 136
22.1.3 LCD 功能描述 138
22.2 LCD 寄存器描述 140
22.3 LCD 驱动控制例程 144
23 程序下载和仿真 147
23.1 程序下载 147
23.2 在线仿真 147
24 电气特性 149
24.1 极限参数 149
24.2 直流电气特性 149
24.3 交流电气特性 154
25 封装类型 155
26 附录 157
附录 1 指令集速查表 157

1 概述

CA51F4 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 18K 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 12Bit ADC、Touch Key、16 Bit PWM、UART、I^2C、LCD 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。其中，LCD 驱动内建电压电荷泵模式，可实现升压功能，在不同供电条件下（包括低电压供电）LCD 电压恒定输出。强大的功能配置可使其广泛应用于各种领域产品中，对于带 LCD 显示的产品中性能更加优越。产品广泛应用于遥控器、温控器及电池供电产品。

2 基本特性

- **内核**
  - CPU：1T 8051, 最高速度比传统 8051 快 10 倍
  - 兼容 8051 指令集, 双 DPTR 工作模式
- **存储器**
  - Flash: 18K 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，1024 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℃）
  - 外部 RTC 振荡器：32.768KHz
- **RTC 功能**
  - 内置 RTC 模块可计时、分、秒、星期、天数，支持闹钟功能
  - 支持毫秒、半秒中断
- **中断系统**
  - 15 个有效中断源
  - 两级中断优先级，支持中断嵌套
  - 10 个外部中断源，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 46 个 GPIO 口，支持推挽、开漏、上拉、下拉、高阻模式
- 3 个 GPIO 灌电流达到 10mA 以上(Vol=GND+0.3V)，可用于控制 LCD 的背光
- 1 个 GPIO 灌电流达到 400mA，可作为遥控载波驱动端口

◆ 触摸按键（Touch Key）

- 内置触摸感应控制器
- 最大支持 16 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

◆ 模/数转换器（ADC）

- 支持 8 通道 12 位 SAR ADC
- 支持 3 种基准电压源：VDD、内部基准、外部基准
- 选择内部电压为基准电压时可测量 VDD 电压

◆ **PWM**

- 支持 3 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和遥控专用频率 38KHz
- 支持 PWM 中断

◆ **LCD 驱动**

- 支持内建电压电荷泵模式、电荷泵分压模式和电阻分压模式，内建电压电荷泵模式可实现升压功能
- 最大可支持 5com x 31segm.、4com x 32 segm.
- 可配置占空比：1/2、1/3、1/4、1/5 Duty

◆ 低电压检测（LVD）

- 可配置四档触发电压 2.0V、2.7V、3.7V 和 4.4V
- 可设置低电压复位或中断

◆ 复位模式

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

◆ 看门狗

- 27 位看门狗定时器，16 位调节精度，可配置看门狗复位或中断

◆ 通用串行接口（UART1）

- 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

◆ 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<3uA

- IDLE 模式，电流<10uA
- 低速运行模式，电流<20uA

◆ **封装类型：**LQFP48/SOP28

3 芯片型号功能介绍

表 3-1　CA51F4 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	外部低速晶振[32.768KHz]	GPIO 数量	UART 数量	I²C	16 bit PWM 通道数量	触摸按键数量	12 bit ADC 通道数量	LCD 驱动[comx seg]	ISP	片上仿真功能	工作电压	封装形式
CA51F452L2	18K	1024	√	√	√	46	1	√	3	16	8	4X32 5X31	√	√	1.8-5.5	LQFP48
CA51F412L2	18K	1024	√	√	√	46	1	√	3	×	8	4X32 5X31	√	√	1.8-5.5	LQFP48
CA51F452S6	18K	1024	√	√	×	26	1	√	2	8	6	4X19 5X18	√	√	1.8-5.5	SOP28
CA51F412S6	18K	1024	√	√	×	26	1	√	2	×	6	4X19 5X18	√	√	1.8-5.5	SOP28

4 系统框图

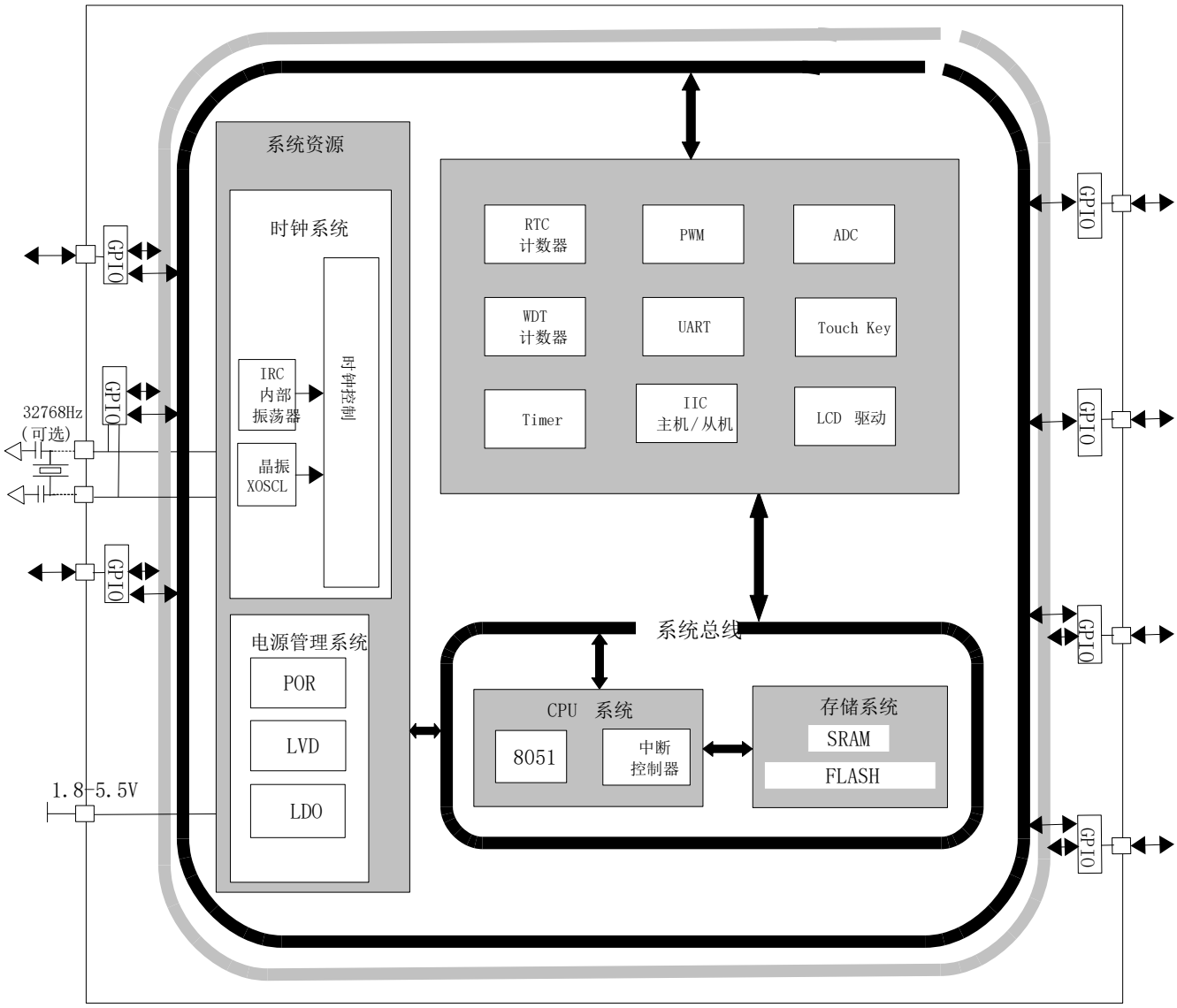


图 4-1-1 芯片框图

5 引脚封装及其描述

5.1 封装定义

型号：**CA51F4L2**

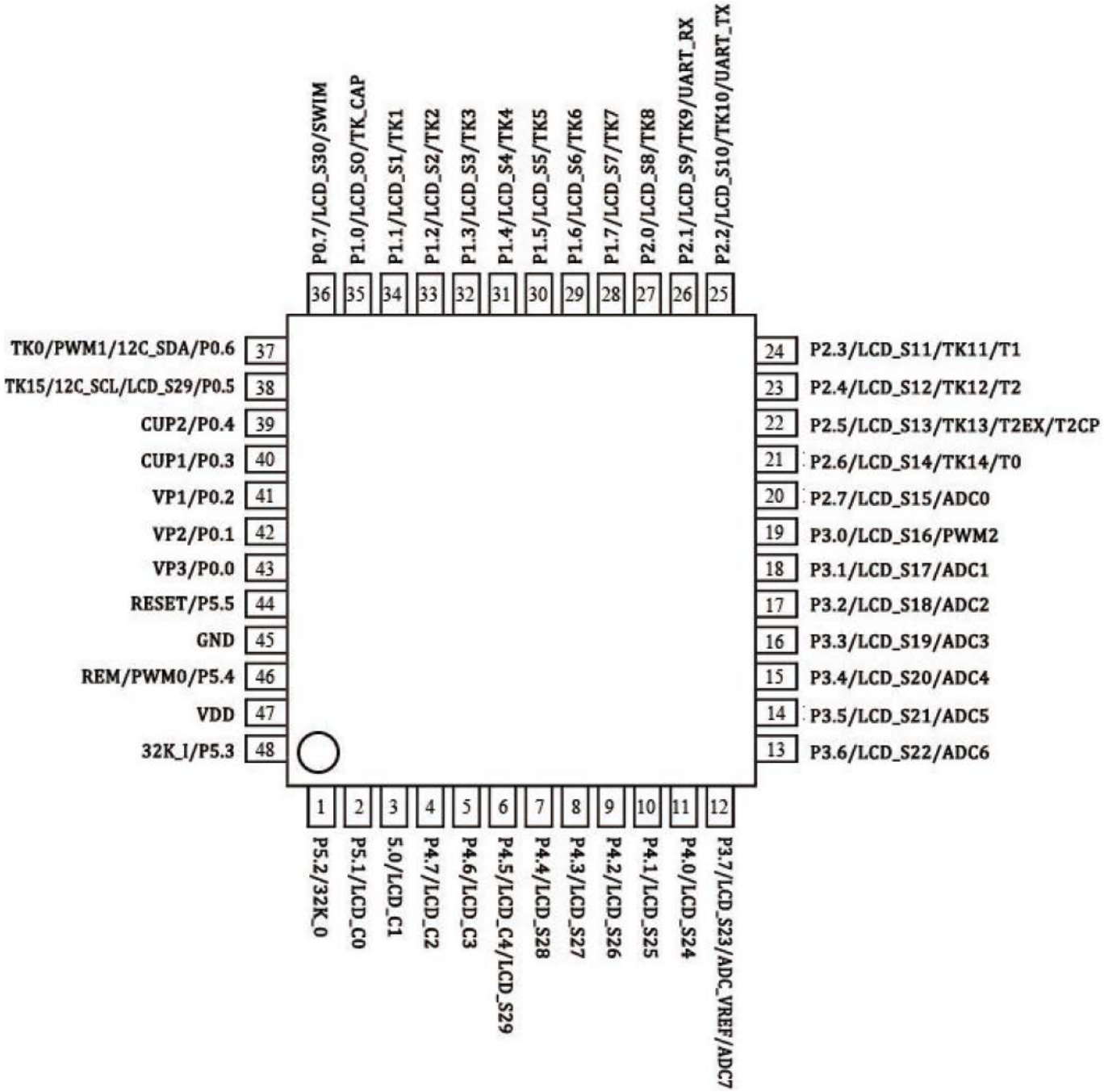


图 **5-1-1** 封装图

型号：**CA51F4S6**

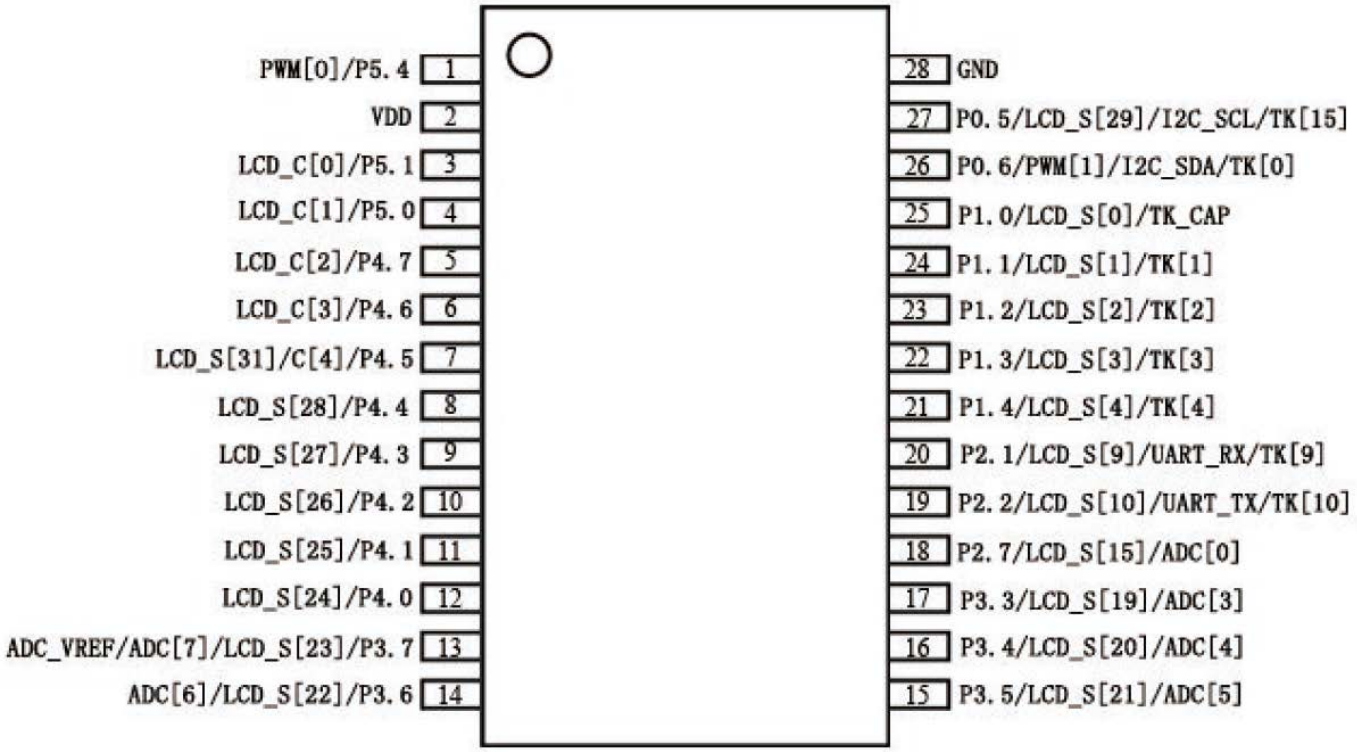


图 **5-1-2** 封装图

5.2 引脚描述

表 5-2-1 CA51F4 引脚描述

引脚序号		管脚名称	管脚功能	默认功能
LQFP48	SOP28			
1		P5.2/32K_O	通用双向 I/O 口 外部 32K 时钟输出口	通用双向 I/O 口
2	3	P5.1/LCD_C0	通用双向 I/O 口 LCD 驱动输出	通用双向 I/O 口
3	4	P5.0/LCD_C1	通用双向 I/O 口 LCD 驱动输出	通用双向 I/O 口
4	5	P4.7/LCD_C2	通用双向 I/O 口 LCD 驱动输出	通用双向 IO 口
5	6	P4.6/LCD_C3	通用双向 I/O 口 LCD 驱动输出	通用双向 IO 口
6	7	P4.5/LCD_C4/LCD_S31	通用双向 I/O 口 LCD 驱动输出	通用双向 IO 口
7	8	P4.4/LCD_S28	通用双向 I/O 口 LCD 驱动输出	通用双向 IO 口
8	9	P4.3/LCD_S27	通用双向 I/O 口 LCD 驱动输出	通用双向 IO 口
9	10	P4.2/LCD_S26	通用双向 I/O 口 LCD 驱动输出	通用双向 I/O 口
10	11	P4.1/LCD_S25	通用双向 I/O 口 LCD 驱动输出	通用双向 I/O 口
11	12	P4.0/LCD_S24	通用双向 I/O 口 LCD 驱动输出	通用双向 IO 口
12	13	P3.7/LCD_S23/ADC_VREF/ADC7	通用双向 I/O 口 LCD 驱动输出 ADC 外部参考输入 ADC 通道输入	通用双向 IO 口
13	14	P3.6/LCD_S22/ADC6	通用双向 I/O 口 LCD 驱动输出 ADC 通道输入	通用双向 IO 口
14	15	P3.5/LCD_S21/ADC5	通用双向 I/O 口 LCD 驱动输出 ADC 通道输入	通用双向 IO 口
15	16	P3.4/LCD_S20/ADC4	通用双向 I/O 口 LCD 驱动输出 ADC 通道输入	通用双向 IO 口
16	17	P3.3/LCD_S19/ADC3/INT1	通用双向 I/O 口 LCD 驱动输出 ADC 通道输入	通用双向 IO 口

17		P3.2/LCD_S18/ADC2/INT0	通用双向 I/O 口 LCD 驱动输出 ADC 通道输入	通用双向 IO 口
18		P3.1/LCD_S17/ADC1	通用双向 I/O 口 LCD 驱动输出 ADC 通道输入	通用双向 IO 口
19		P3.0/LCD_S16/PWM2/CLK_IN	通用双向 I/O 口 LCD 驱动输出 PWM2 通道输出 外部标准时钟输入	通用双向 IO 口
20	18	P2.7/LCD_S15/ADC0	通用双向 I/O 口 LCD 驱动输出 ADC 通道输入	通用双向 IO 口
21		P2.6/LCD_S14/T0/TK14	通用双向 I/O 口 T0 口 LCD 驱动输出 触摸按键通道输入	通用双向 IO 口
22		P2.5/LCD_S13/T2EX/T2CP/TK13	通用双向 I/O 口 T2EX/T2CP 口 LCD 驱动输出 触摸按键通道输入	通用双向 IO 口
23		P2.4/LCD_S12/T2/TK12	通用双向 I/O 口 T2 口 LCD 驱动输出 触摸按键通道输入	通用双向 IO 口
24		P2.3/LCD_S11/T1/TK11	通用双向 I/O 口 T1 口 LCD 驱动输出 触摸按键通道输入	通用双向 IO 口
25	19	P2.2/UART_TX/LCD_S10/TK10	通用双向 I/O 口 UART_TX 传输口 LCD 驱动输出 触摸按键通道输入	通用双向 IO 口
26	20	P2.1/UART_RX/LCD_S9/TK9	通用双向 I/O 口 UART_RX 传输口 LCD 驱动输出 触摸按键通道输入	通用双向 IO 口
27		P2.0/LCD_S8/TK8	通用双向 I/O 口 LCD 驱动输出 触摸按键通道输入	通用双向 IO 口
28		P1.7/LCD_S7/TK7	通用双向 I/O 口 LCD 驱动输出 触摸按键通道输入	通用双向 IO 口
29		P1.6/LCD_S6/TK6	通用双向 I/O 口	通用双向 IO 口

			LCD 驱动输出 触摸按键通道输入	
30		P1.5/LCD_S5/TK5	通用双向 I/O 口 LCD 驱动输出 触摸按键通道输入	通用双向 IO 口
31	21	P1.4/LCD_S4/TK4	通用双向 I/O 口 LCD 驱动输出 触摸按键通道输入	通用双向 IO 口
32	22	P1.3/LCD_S3/TK3	通用双向 I/O 口 LCD 驱动输出 触摸按键通道输入	通用双向 IO 口
33	23	P1.2/LCD_S2/TK2	通用双向 I/O 口 LCD 驱动输出 触摸按键通道输入	通用双向 IO 口
34	24	P1.1/LCD_S1/TK1	通用双向 I/O 口 LCD 驱动输出 触摸按键通道输入	通用双向 IO 口
35	25	P1.0/LCD_S0/TK_CAP	通用双向 I/O 口 LCD 驱动输出 触摸按键通道输入	通用双向 IO 口
36		P0.7/SWIM/ LCD_S30	通用双向 I/O 口 SWIM 传输口 LCD 驱动输出	SWIM 传输口
37	26	P0.6/I2C_SDA/PWM1/TK0	通用双向 I/O 口 I^2C 数据传输口 PWM1 输出 触摸按键通道输入	I^2C 数据传输口
38	27	P0.5/I2C_SCL/TK15/ LCD_S29	通用双向 I/O 口 I^2C 时钟传输口 LCD 驱动输出 触摸按键通道输入	I^2C 时钟传输口
39		P0.4/CUP2	通用双向 I/O 口 LCD 的 CUP 口	通用双向 IO 口
40		P0.3/CUP1	通用双向 I/O 口 LCD 的 CUP 口	通用双向 IO 口
41		P0.2/VP1	通用双向 I/O 口 LCD 的 VP 口	通用双向 IO 口
42		P0.1/VP2	通用双向 I/O 口 LCD 的 VP 口	通用双向 IO 口
43		P0.0/VP3	通用双向 I/O 口 LCD 的 VP 口	通用双向 IO 口
44		P5.5/RESET	通用双向 I/O 口 硬件复位输入	硬件复位引脚
45	28	GND	电源地引脚	电源地引脚

46	1	P5.4/PWM0/REM	通用双向 I/O 口 PWM0 输出 REM 输出	通用双向 IO 口
47	2	VDD	芯片供电引脚	芯片供电引脚
48		P5.3/32K_I	通用双向 I/O 口 外部 32K 时钟输入	通用双向 IO 口

备注：信号引脚复用功能设置方法详见表 15-2-7 和表 15-2-10

6 中央处理器（CPU）

6.1CPU 简介

CA51F4 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）

CA51F4 系列芯片提供了 256 字节内部 RAM 和 1024 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 1024 字节外部 RAM（地址：0000H ~ 03FFH）可通过 MOVX 指令间接寻址。

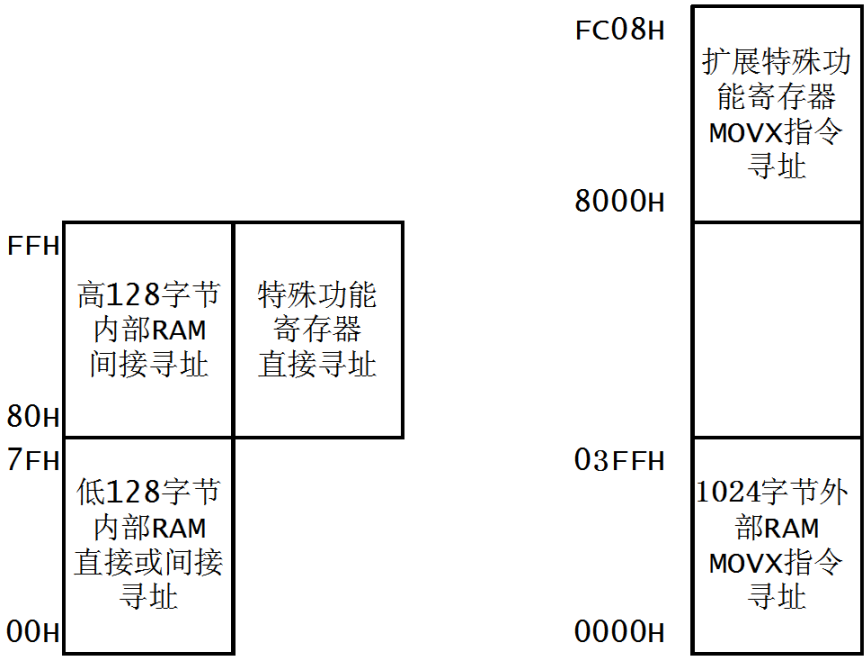


图 7-1-1　RAM 组织结构图

7.2 特殊功能寄存器（SFR）

CA51F4 系列芯片提供了兼容传统 8051 的 SFR 分布，SFR 和高 128 字节内部 RAM 共用地址 80H ~ FFH，只能直接寻址，SFR 映射如表 7-2-1 所示。

表 7-2-1　特殊功能寄存器（SFR）映射表

	可位寻址	不可位寻址						
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8H	EXIP	EPIE	EPIF	EPCON	IDLSTL	IDLSTH	STPSTL	STPSTH
F0H	B	RTCON	RTCS	RTCM	RTCH	RTCDL	RTCDH	INDEX
E8H	EXIE	RTCSS	RTAS	RTAM	RTAH	RTMSS	RTCIF	LVDCON

E0H	ACC	LCCON	LCCFG	LCDAT	LCDIVL	LCDIVH	-	-
D8H	P5	-	PWMEN	-	PWMCMAX	PWMCON	PWMCFG	PWMDIVL
D0H	PSW	PWMDIVH	PWMDUTL	PWMDUTH	PWMIF	-	-	-
C8H	T2CON	T2MOD	T2CL	T2CH	TL2	TH2	TKMSL	TKMSH
C0H	P4	TKCON	TKCFG	TKMTS	TKCHS	ATKSL	ATKSH	TKIF
B8H	IP	ADCON	ADCFGL	ADCFGH	ADCDL	ADCDH	-	-
B0H	P3	I2CCON	I2CADR	I2CADM	I2CCCR	I2CDAT	I2CSTA	I2CFLG
A8H	IE	-	WDCON	WDFLG	WDVTHL	WDVTHH	-	-
A0H	P2	-	-	-	-	-	-	-
98H	-	-	S1CON	S1BUF	S1RELL	S1RELH	-	-
90H	P1	-	-	-	-	-	-	-
88H	TCON	TMOD	TL0	TL1	TH0	TH1	IT1CON	IT0CON
80H	P0	SP	DP0L	DP0H	DP1L	DP1H	PWCON	PCON

由于 SFR 地址空间有限，CA51F4 系列芯片在外部 RAM 地址空间增加了扩展特殊功能寄存器，扩展特殊功能寄存器映射如图表 7-2-2 所示。

表 7-2-2 扩展特殊功能寄存器映射表

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
8000H	P00F	P01F	P02F	P03F	P04F	P05F	P06F	P07F
8008H	P10F	P11F	P12F	P13F	P14F	P15F	P16F	P17F
8010H	P20F	P21F	P22F	P23F	P24F	P25F	P26F	P27F
8018H	P30F	P31F	P32F	P33F	P34F	P35F	P36F	P37F
8020H	P40F	P41F	P42F	P43F	P44F	P45F	P46F	P47F
8028H	P50F	P51F	P52F	P53F	P54F	P55F	-	-
8078H	-	-	-	-	-	-	-	RCSTA
8080H	CKCON	CKSEL	CKDIV	IHCFG	-	ILCFHL	ILCFGH	-
8088H	ADCALL	ADCALH	-	-	-	-	-	ADOPC
8090H	TKMAXF	TKMINF	ATKNL	ATKNH	-	-	-	-
8098H	-	PWMHS	-	-	PWMSBC	PWMBD	-	-
80B0H	SWICON	SWIDAT	SWISTA	SWIOVT	-	-	-	-
80F8H	TSCMD	TSSTA	STPCL	STPCH				
8100H	SPMAX	-	-	TKPWC	-	-	TLEN	TLDAT
8108H	TLCON	TLFLG	TLCKS	TLCNTKL	TLCNTKH	TLCNTLL	TLCNTLH	TLDIV
8110H	-	-	-	-	-	-	LCPMP	LCCAD
8118H	UDCKS	-	-	-	-	RMCTL	FTCTL	TPCTL
8120H	P00C	P01C	P02C	P03C	P04C	P05C	P06C	P07C
8128H	P10C	P11C	P12C	P13C	P14C	P15C	P16C	P17C
8130H	P20C	P21C	P22C	P23C	P24C	P25C	P26C	P27C
8138H	P30C	P31C	P32C	P33C	P34C	P35C	P36C	P37C
8140H	P40C	P41C	P42C	P43C	P44C	P45C	P46C	P47C
8148H	P50C	P51C	P52C	P53C	P54C	P55C	-	-
FC00H	MECON	FSCMD	FSDAT	LOCK	PADRD	PTSL	PTSH	REPSET

7.3 Flash 存储器

7.3.1 功能简介

Flash 存储器包含 18K 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 18K 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	IFEN	-	-	-	-	CMD[2:0]		
R/W	R/W	-	-	-	-	R/W		
初始值	0	-	-	-	-	0	0	0

位编号	位符号	说明
7	IFEN	信息区使能位，1 表示使能信息区操作
6~3	-	-
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、5~4	-	-
6	REPE	信息区解锁标志，有保护操作，必须先执行写 8'h56,后执行 8'hcb 操作才能
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	1	0	0	1	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) ~ 47FFH. *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 90H，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[14:8]						
R/W	-	R/W						
初始值	-	-	-	0	0	0	0	0
位编号	位符号	说明						
15	-	-						
14~0	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，18K 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 = 0x8F; //程序区空间地址为：0~0x477F,数据区空间地址为：0x4780~0x47FF
-----------------------------------------------------------------------------------
```

备注：以上设置数据区在 FLASH 中的物理地址是 0x4780~0x47FF，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

1024 字节的外部 RAM 可以映射为程序空间使用，映射地址为 4800H~4BFFH，映射图如图 7-4-1 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~03FFH）。映射程序区用来实现 IAP 等功能特别方便。

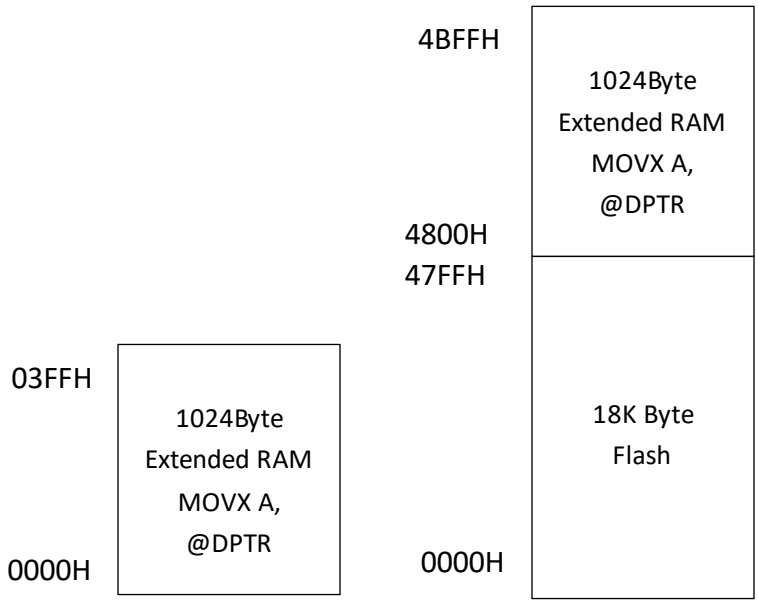


图 7-4-1 XRAM 地址映射图

8 中断系统

8.1 功能简介

CA51F4 系列芯片有一个增强的中断控制系统，共有 14 个中断入口，每个中断入口有若干中断源，每个中断源有 2 级中断优先级。每个中断源都有独立的中断向量、优先级设置位、中断使能位、中断标志。CPU 在响应中断后，进入该中断对应的中断服务程序，接到 RETI 指令后将返回中断前状态。如果同时有多个有效中断产生中断请求，CPU 将根据设置的中断优先级依次响应；如果优先级相同，则根据它们的自然优先级（中断入口地址从低到高）依次响应。

8.2 中断逻辑

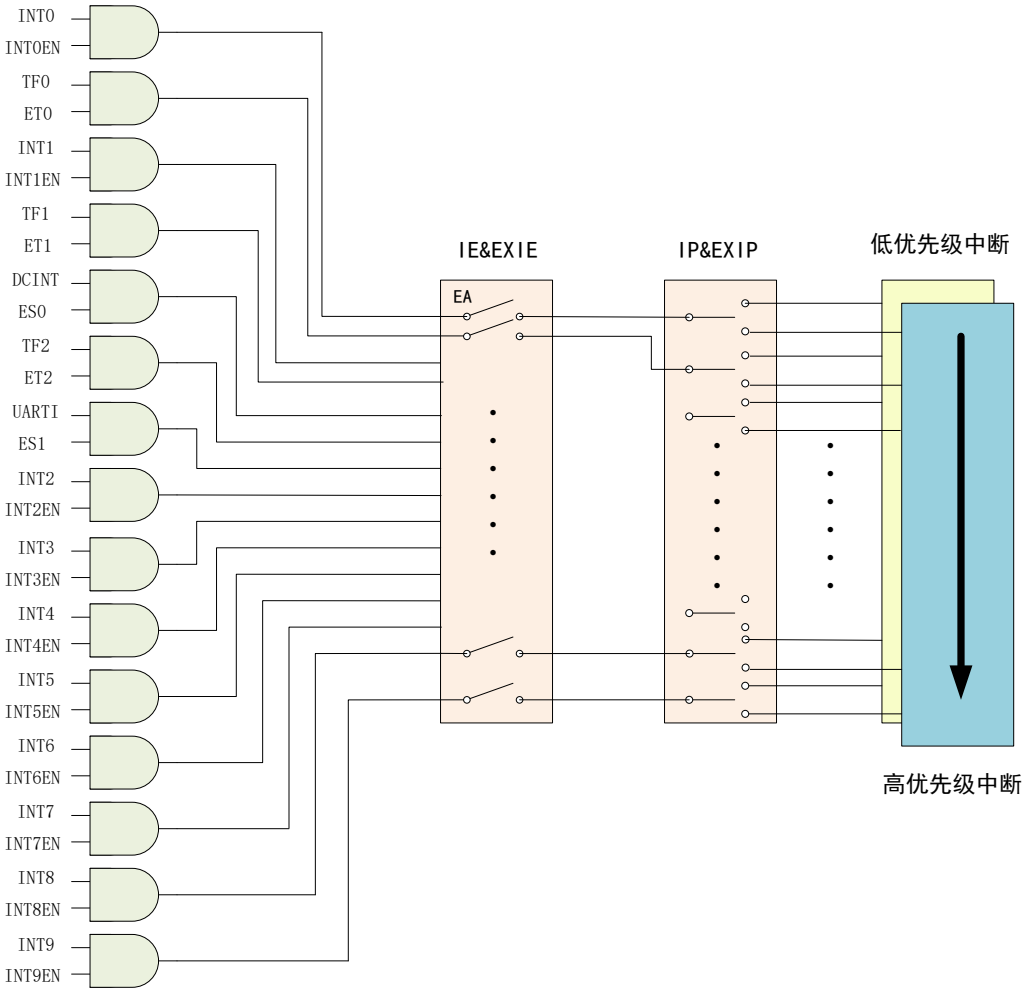


表 8-2-1 中断逻辑图

8.3 中断向量表

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

8.4 中断控制寄存器

表 8-4-1 寄存器 IE

A8H	7	6	5	4	3	2	1	0
IE	EA	ES1	ET2	-	ET1	EX1	ET0	EX0
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	EA	全局中断使能控制位 0：全局中断关闭 1：全局中断打开
6	ES1	UART1 中断使能控制位 0：UART1 中断关闭 1：UART1 中断打开
5	ET2	定时器 2 中断使能控制位 0：定时器 2 中断关闭 1：定时器 2 中断打开
4	-	-
3	ET1	定时器 1 中断使能控制位

		0：定时器 1 中断关闭 1：定时器 1 中断打开
2	EX1	外部中断 1 使能控制位 0：外部中断 1 中断关闭 1：外部中断 1 中断打开
1	ET0	定时器 0 中断使能控制位
0	EX0	外部中断 0 使能控制位 0：外部中断 0 中断关闭 1：外部中断 0 中断打开

表 8-4-2 寄存器 EXIE

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

位编号	位符号	说明
7	INT9EN	中断 9 使能控制位（中断 9 用于 PWM/外部中断 9） 0：关闭 1：打开
6	INT8EN	中断 8 使能控制位（中断 8 用于 RTC /外部中断 8） 0：关闭 1：打开
5	INT7EN	中断 7 使能控制位（中断 7 用于 WDT /外部中断 7） 0：关闭 1：打开
4	INT6EN	中断 6 使能控制位（中断 6 用于 I2C/SWI /外部中断 6） 0：关闭 1：打开
3	INT5EN	中断 5 使能控制位（中断 5 用于外部中断 5） 0：关闭 1：打开
2	INT4EN	中断 4 使能控制位（中断 4 用于 LVD/外部中断 4） 0：关闭 1：打开
1	INT3EN	中断 3 使能控制位（中断 3 用于 TK/外部中断 3） 0：关闭 1：打开
0	INT2EN	中断 2 使能控制位（中断 2 用于 ADC/外部中断 2） 0：关闭 1：打开

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

表 8-4-3 寄存器 IP

B8H	7	6	5	4	3	2	1	0
IP	-	PS1	PT2	PS0	PT1	PX1	PT0	PX0
R/W	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初始值	-	0	0	0	0	0	0	0

位编号	位符号	说明
7	-	-
6	PS1	UART1 优先级控制位 0：低优先级 1：高优先级
5	PT2	定时器 2 优先级控制位 0：低优先级 1：高优先级
4	-	-
3	PT1	定时器 1 优先级控制位 0：低优先级 1：高优先级
2	PX1	外部中断 1 优先级控制位 0：低优先级 1：高优先级
1	PT0	定时器 0 优先级控制位 0：低优先级 1：高优先级
0	PX0	外部中断 0 优先级控制位 0：低优先级 1：高优先级

表 8-4-4 寄存器 EXIP

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

位编号	位符号	说明
7	PX9	中断 INT9 优先级控制位 0：低优先级

		1：高优先级
6	PX8	中断 INT8 优先级控制位 0：低优先级 1：高优先级
5	PX7	中断 INT7 优先级控制位 0：低优先级 1：高优先级
4	PX6	中断 INT6 优先级控制位 0：低优先级 1：高优先级
3	PX5	中断 INT5 优先级控制位 0：低优先级 1：高优先级
2	PX4	中断 INT4 优先级控制位 0：低优先级 1：高优先级
1	PX3	中断 INT3 优先级控制位 0：低优先级 1：高优先级
0	PX2	中断 INT2 优先级控制位 0：低优先级 1：高优先级

8.5 外部中断

8.5.1 外部中断介绍

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

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

8.5.2 外部中断寄存器

表 8-5-2-1 寄存器 IT0CON

8FH	7	6	5	4	3	2	1	0

IT0CON	-	-	IT0PS[4:0]					
R/W	-	-	R/W					
初始值	-	-	0	1	1	0	1	0

位编号	位符号	说明
7~5	-	-
4~0	IT0PS[4:0]	INT0 中断管脚选择位 编号和管脚对应表参考**表 8-5-2-6**

表 8-5-2-2 寄存器 IT1CON

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

位编号	位符号	说明
7~5	-	-
4~0	IT1PS[4:0]	INT1 中断引脚选择位 编号和管脚对应表参考**表 8-5-2-6**

表 8-5-2-3 寄存器 EPIE

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

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

表 8-5-2-4 寄存器 EPIF

FAH	7	6	5	4	3	2	1	0

EPIF	EPIF9	EPIF8	EPIF7	EPIF6	EPIF5	EPIF4	EPIF3	EPIF2
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0

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

表 8-5-2-5 寄存器 EPCON

FBH	7	6	5	4	3	2	1	0
EPCON	EPPL	-	-	EPPS[4:0]				
R/W	R/W	-	-	R/W				
初始值	0	-	-	0	0	0	0	0
备注：EPCON 为带索引寄存器，设置 INDEX=0~7 分别对应 EPCON0~EPCON7								

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

表 8-5-2-6 中断引脚编号索引表

引脚名称	编号	引脚名称	编号
P00	0	P30	24
P01	1	P31	25
P02	2	P32	26
P03	3	P33	27
P04	4	P34	28
P05	5	P35	29
P06	6	P36	30
P07	7	P47	31
P10	8	P40	32
P11	9	P41	33

P12	10	P42	34
P13	11	P43	35
P14	12	P44	36
P15	13	P45	37
P16	14	P46	38
P17	15	P47	39
P20	16	P50	40
P21	17	P51	41
P22	18	P52	42
P23	19	P53	43
P24	20	P54	44
P25	21	P55	45
P26	22		
P27	23		

8.5.3 外部中断控制例程

◆ **外部中断 0/1 控制例程**

例如，使能外部中断 0，程序如下:

```
-----------------------------------------------------------------------------------
void INT0_init(void)
{
    P10F = 1;           //P10 设置为输入功能
    IT0CON = 8;         //设置 P10 为 INT0 中断引脚
    EX0 = 1;            //INT0 中断使能
    IE0 = 1;            //外部中断 0 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)
{
    P10F = 1;           //P10 设置为输入功能
    IT1CON = 8;         //设置 P10 为 INT1 中断引脚
    EX1 = 1;            //INT1 中断使能
    IE1 = 1;            //外部中断 1 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 2~9 控制例程**

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

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

9 时钟系统

9.1 时钟系统介绍

CA51F4 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
- 支持外部 32.768KHz 晶体振荡器

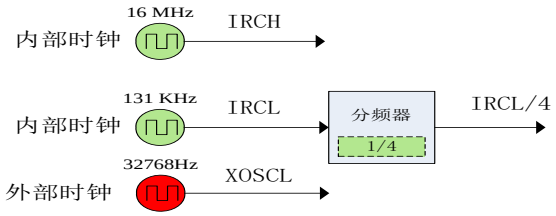


图 9-1-1 时钟源示意图

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
XOSCL	外部 32768Hz 晶体振荡器

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%。

9.1.4 外部 32.768KHz 晶体谐振器（XOSCL）

XOSCL 主要是作为 RTC 的时钟源，用于实时计时，实现产品的时钟功能。XOSCL 设置为系统时钟时主要应用在低功耗的场合，最低功耗可小于 9uA。XOSCL 通过寄存器 CKCON 的 XLCKE 位打开或关闭，要注意的是，XOSCL 起振时间比较长，大约需要 1 秒左右才能达到稳定，在应用时需要等待 XOSCL 时钟稳定后才可以使用，寄存器 CKCON 的 XLSTA 位是 XOSCL 时钟稳定标志。

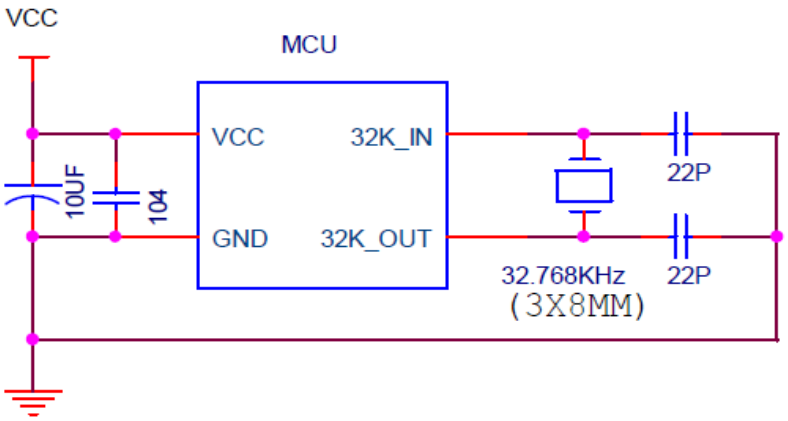


图 9-1-4-1 XOSCL 典型电路图

重要提醒： *1. 硬件设计时晶振负载电容地必须和芯片地连接，晶振补偿电容尽量靠近芯片 GND 引脚。32.768KHz 石英晶振要求使用直径 3mmx8mm 的晶振规格。*

2. 以上电路及元件参数仅供参考，使用不同厂家晶振在电路使用中参数可能需要修改。

9.2 时钟控制寄存器描述

表 9-2-1 寄存器 CKCON

8080H	7	6	5	4	3	2	1	0
CKCON	ILCKE	IHCKE	-	-	XLCKE	XLSTA	-	-
R/W	R/W	R/W	-	-	R/W	R	-	-
初始值	0	0	-	-	0	0	-	-

位编号	位符号	说明
7	ILCKE	IRCL 使能控制位 1：打开 0：关闭 *备注：* *该位为 1 时，时钟模块打开，但是该位为 0 时，如果系统或者其他模块选择了该时钟源，*

		该时钟仍然会被打开。
6	IHCKE	IRCH 使能控制位 1：打开 0：关闭 *备注：* *该位为 1 时，时钟模块打开，但是该位为 0 时，如果系统或者其他模块选择了该时钟源，该时钟仍然会被打开。*
5~4	-	-
3	XHCKE	XOSCH 使能控制位 1：打开 0：关闭 *备注：* *1 位有效时，时钟模块开启，但是该位为 0 时，如果系统或者其他模块选择了该时钟源，该时钟仍然会被打开。* *2 由于 XOSCH 为外部时钟，所以若想所以启动 XOSCH 还需要把对应的引脚功能配置为 XOSCH 的功能。*
2	XHSTA	XOSCH 时钟稳定信号标志，1 有效
1~0	-	-

表 9-2-2 寄存器 RCSTA

807FH	7	6	5	4	3	2	1	0
RCSTA	-	IHSTA	-	-	-	-	-	-
R/W	-	R	-	-	-	-	-	-
初始值	-	0	-	-	-	-	-	-

位编号	位符号	说明
7	-	-
6	IHSTA	IRCH 时钟稳定信号标志，1 有效
5~0	-	-

表 9-2-3 寄存器 IHCFG

8083H	7	6	5	4	3	2	1	0
IHCFG	IHCFG[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	IHCFG	内部 16MHz 时钟配置寄存器

		备注：此寄存器上电后自动加载值对应频率 16MHz，除特殊应用外，不建议修改此数值。

表 9-2-4 寄存器 ILCFGL、ILCFGH

8085H	7	6	5	4	3	2	1	0
ILCFGL	ILCFG[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
8086H	7	6	5	4	3	2	1	0
ILCFGH	-	-	-	-	-	-	-	ILCFG[8]
R/W	-	-	-	-	-	-	-	R/W
初始值	-	-	-	-	-	-	-	0

位编号	位符号	说明
8~0	ILCFG	内部 131KHz 时钟配置寄存器 *备注：此寄存器上电后自动加载值对应频率 131KHz，除特殊应用外，不建议修改此数值。*

9.3 系统时钟

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

9.3.1 系统时钟结构图

系统时钟结构图见图 9-3-1。

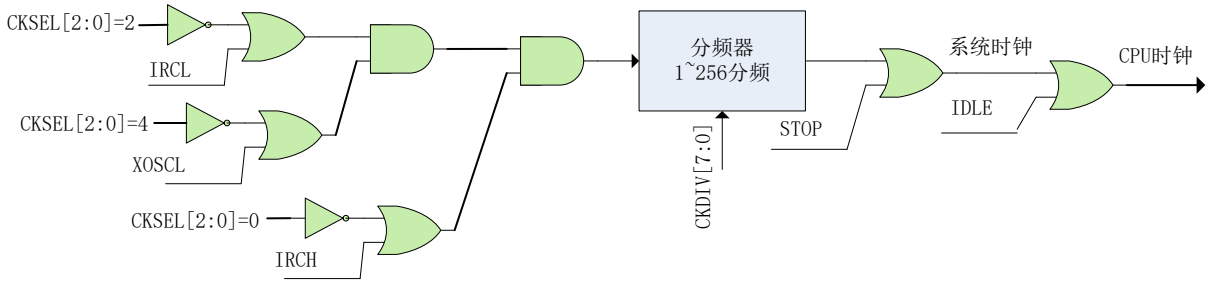


图 9-3-1 系统时钟结构图

9.3.2 系统时钟控制寄存器描述

表 9-3-2-1 寄存器 CKSEL

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

位编号	位符号	说明
7	RTCKS	RTC 时钟源选择位 0：XOSCL 1：IRCL *备注：当设置为 IRCL 时，RTC 的时钟为 IRCL 的 4 分频*
6~3	-	-
2~0	CKSEL	系统时钟选择位 000：IRCH 001： 预留 010：IRCL 100：XOSCL

		其他：IRCH *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。*

表 9-3-2-2 寄存器 CKDIV

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

位编号	位符号	说明
7~0	CKDIV	系统时钟分频： 00H：不分频 01H：2 分频 02H：3 分频 03H：4 分频 …… FFH：256 分频

9.3.3 系统时钟控制方法及例程

◆ **设置系统时钟为 IRCH**

设置系统时钟为 IRCH，程序如下：

```
-----------------------------------------------------------------------------
#define IHCKE           (1<<6)
#define CKSEL_IRCH      0
void Sys_Clk_Set_IRCH(void)
{
    CKCON |= IHCKE;                         //打开 IRCH 时钟
    CKSEL = (CKSEL&0xF8) | 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<<7)
#define CKSEL_IRCL      2
void Sys_Clk_Set_IRCL(void)
{
    CKCON |= ILCKE;                         //打开 IRCL 时钟
    Delay_ms(1);                            //使能 IRCL 后延时 1ms，等待 IRCL 时钟稳定
    CKSEL = (CKSEL&0xF8) | CKSEL_IRCL;      //设置系统时钟为 IRCL
}
-----------------------------------------------------------------------------
```

◆ **设置系统时钟为 XOSCL**

设置系统时钟为 XOSCL，程序如下：

```
-----------------------------------------------------------------------------
#define XLCKE           (1<<3)
#define XLSTA           (1<<2)
#define CKSEL_XOSCL 4
void Sys_Clk_Set_XOSCL(void)
{
    P52F = 3;
    P53F = 3;
    CKCON |= XLCKE;
    while(!(CKCON & XLSTA));
    CKSEL = (CKSEL&0xF8) | CKSEL_XOSCL;
}
-----------------------------------------------------------------------------
```

10 供电和复位系统

10.1 供电系统

在 CA51F4 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。

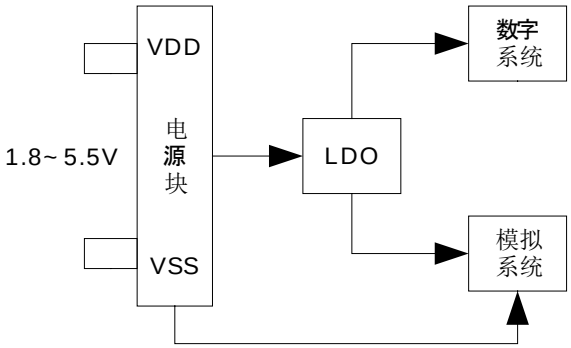


图 10-1-1 供电系统示意图

图 10-1-2 为芯片供电典型电路图。

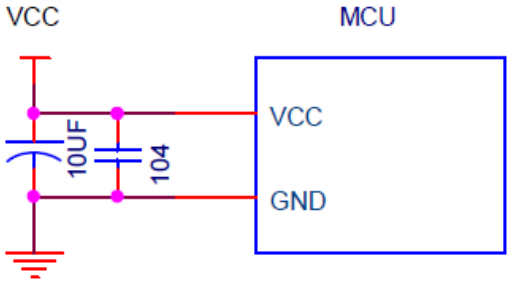


图 10-1-2 芯片供电典型电路图

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

CA51F4 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。

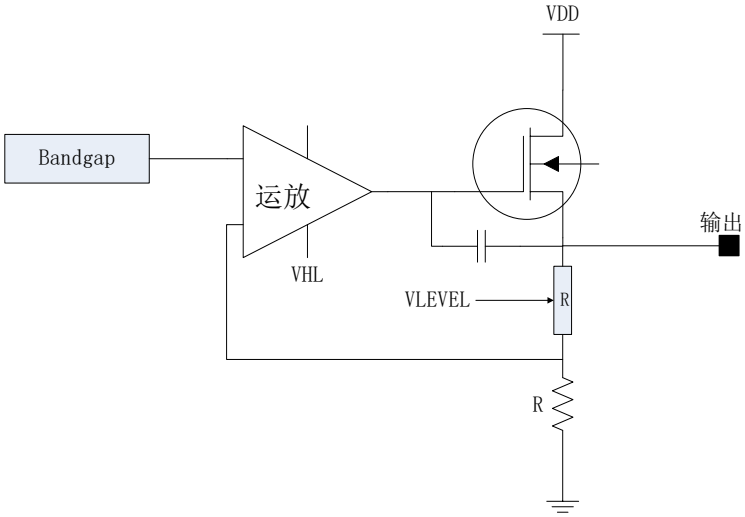


图 10-1-3 LDO 模块示意图

10.1.2 LDO 控制寄存器

表 10-1-2-1 寄存器 PWCON

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

位编号	位符号	说明
7~4	FLEVEL	内部基准电压（Bandgap）输出调整位域 0000：0.825V 0001：0.850V 0010：0.875V 0011：0.900V 0100：0.925V 0101：0.950V 0110：0.975V 0111：1.000V 1000：1.025V 1001：1.050V 1010：1.075V 1011：1.100V 1100：1.125V 1101：1.150V

		1110：1.175V 1111：1.200V *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

CA51F4 系列芯片有多个内部和外部复位源，如图 10-2-1 所示。

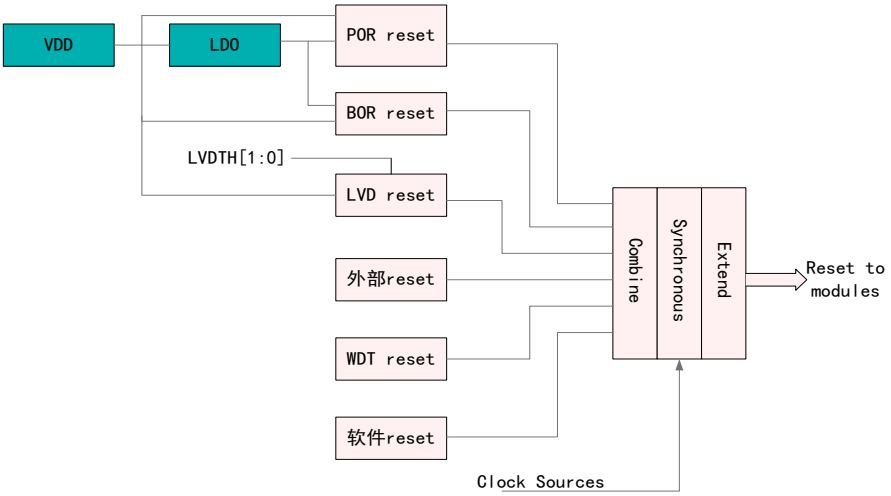
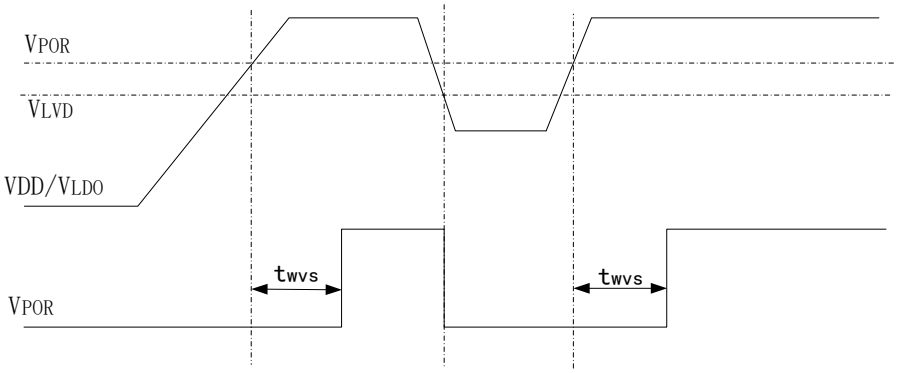


图 10-2-1 复位系统结构图

- **上电复位（POR）**

系统上电呈现逐渐上升的曲线形式，需要一定时间才能达到正常的工作电压。上电复位是基于电源电压 VDD 和内部 LDO 的输出电压，当电压低于检测阈值时，上电复位信号有效。

上电复位电路能够保证芯片在上电过程中处于复位状态，芯片上电后能够从一个已知的稳定的状态开始运行。上电复位信号也会被芯片内部的计数器展宽，以保证上电后内部的各种模拟模块能够进入稳定的工作状态。



twvs：等待电压稳定时间

图 10-2-2 上电复位电路示例及上电过程

- **掉电复位（BOR）**

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

- **低电压复位**

低电压检测（LVD）可以在多种工作模式下持续监控电源电压 VDD。当 VDD 低于 LVD 设定的域值电压超过 20us 就可以产生复位信号（前提是 LVD 设置为复位模式）。

- **外部复位**

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

- **看门狗复位**

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

- **软复位**

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

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

11 功耗管理

CA51F4 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 10uA，STOP 模式时系统功耗小于 3uA，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。

备注：

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

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

表 11-4-2 寄存器 IDLSTL、IDLSTH

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

位编号	位符号	说明
15	-	-
14	PWMINT/EPIF[7]	IDLE 模式时， PWM/外部中断 9 的中断状态
13	RTCINT/EPIF[6]	IDLE 模式时，RTC/外部中断 8 的中断状态
12	WDFLG[1]/EPIF[5]	IDLE 模式时， WDT/外部中断 7 的中断状态
11	I2CINT/SWIINT/EPIF[4]	IDLE 模式时，I2C/SWI/外部中断 6 的中断状态

10	EPIF[3]	IDLE 模式时，外部中断 5 的中断状态
9	LVDINT/EPIF[2]	IDLE 模式时，LVD/外部中断 4 的中断状态
8	TKINT/TLINT/EPIF[1]	IDLE 模式时，TK/ /外部中断 3 的中断状态
7	ADCINT/EPIF[0]	IDLE 模式时，ADC/外部中断 2 的中断状态
6	U1INT	IDLE 模式时，UART1 中断的中断状态
5	T2INT	IDLE 模式时，定时器 2 的中断状态
4	-	-
3	TCON[7]	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	TCON[5]	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 STPSTL、STPSTH

FEH	7	6	5	4	3	2	1	0
STPSTL	STPST[7:0]							
R/W	R							
初始值	0	0	0	0	0	0	0	0
FFH	7	6	5	4	3	2	1	0
STPSTH	STPST[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	RTCWKF	STOP 模式时，RTC 中断状态
14	WDTWKF	STOP 模式时，WDT 中断状态
13	I2CWKF/SWIWKF	STOP 模式时，I2C/SWI 的中断状态
12	-	-
11	LVDWKF	STOP 模式时，LVD 中断状态
10	TKWKF	STOP 模式时，触摸按键的中断状态
9	EPWKF[7]	STOP 模式时，外部中断 9 的中断状态
8	EPWKF[6]	STOP 模式时，外部中断 8 的中断状态
7	EPWKF[5]	STOP 模式时，外部中断 7 的中断状态
6	EPWKF[4]	STOP 模式时，外部中断 6 的中断状态
5	EPWKF[3]	STOP 模式时，外部中断 5 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：

```
-----------------------------------------------------------------------------------------
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
    SWICON |= 0x01;  //关闭单线通信功能，否则无法关闭主时钟
    CKCON = 0;       //关闭所有时钟
    PWCON &= ~0x08;//设置 LDO 进入低功率模式
    MECON |= (1<<6); //设置 FLASH 进入深度睡眠状态
    while(STPSTH|STPSTL); //如果有中断未响应,等待中断被响应
    IE_EA = EA;  //保存全局中断使能位状态
    EA = 0;
    PCON |= 0x02;    //进入 STOP 模式
    _nop_();
    _nop_();
    _nop_();
    EA = IE_EA;
    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：

```
-----------------------------------------------------------------------------------------
void Idle(void)
{
    CKCON |= (1<<7);            //打开 IRCL
    CKSEL = (CKSEL&0xF8) | 2; //系统时钟设置为 IRCL
    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
    SWICON |= 0x01;  //关闭单线通信功能，否则无法关闭主时钟
    CKCON = 0;       //关闭所有时钟
    PWCON &= ~0x08;//设置 LDO 进入低功率模式
    MECON |= (1<<6);
    while(IDLSTH|IDLSTL); //如果有中断未响应,等待中断被响应
    PCON |= 0x01;    //进入 IDLE 模式
    _nop_();
    _nop_();
    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：

```
void LowSpeedMode(void)
{
    CKCON |= (1<<7);            //打开 IRCL
    CKSEL = (CKSEL&0xF8) | 2; //系统时钟设置为 IRCL
    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
    SWICON |= 0x01;  //关闭单线通信功能，否则无法关闭主时钟
    CKCON = 0;       //关闭所有时钟
    PWCON &= ~0x08;//设置 LDO 进入低功率模式
}
```

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

---

12 通用定时器（定时器 0, 定时器 1, 定时器 2）

12.1 定时器 0

12.1.1 定时器 0 介绍

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

- **模式 0**

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

- **模式 1**

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

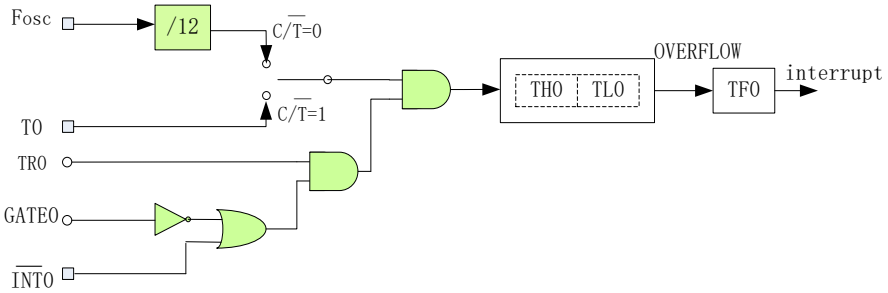


图 12-1-1-1 定时器 0 的模式 0 与 1

- **模式 2**

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

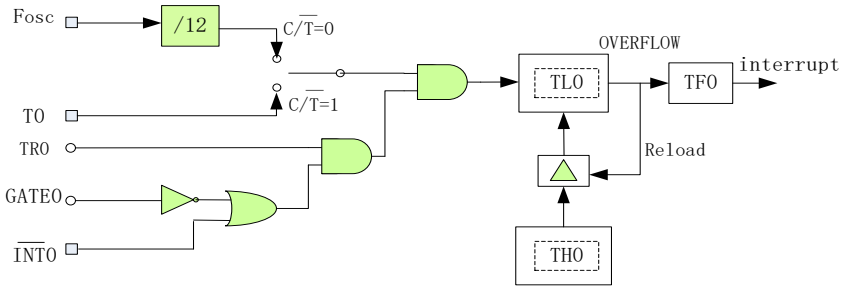


图 12-1-1-2 定时器 0 的模式 2

- **模式 3**

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

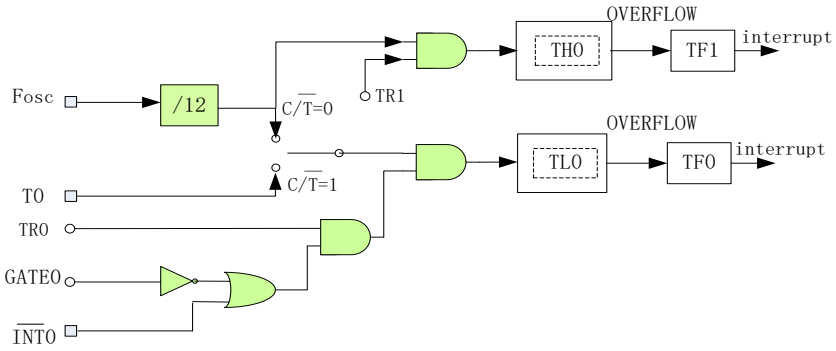


图 12-1-1-3 定时器 0 的模式 3

12.1.2 定时器 0 寄存器描述

表 12-1-2-1 寄存器 TCON

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

位编号	位符号	说明
7	TF1	定时器 0 模式 3 的 TH0 溢出/定时器 1 溢出标志位，中断被响应后自动清 0. 定时器 1 溢出标志位。
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
4	T1M0	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
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
0	T0M0	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

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。

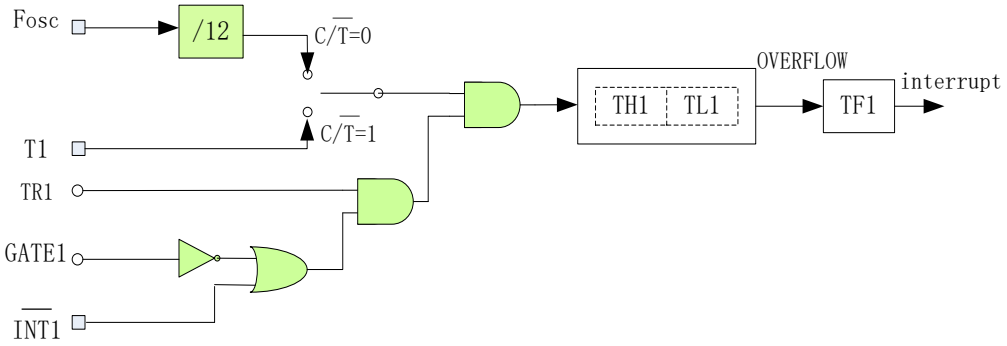


图 12-2-1 定时器 1 的模式 0 和 1

- **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。

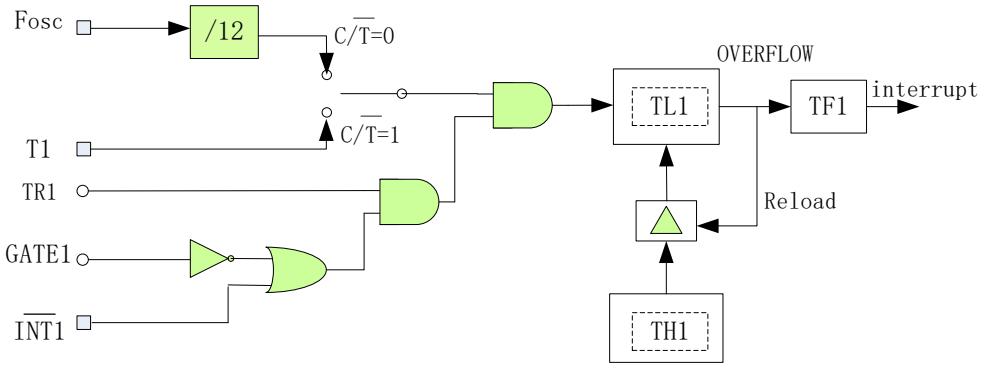


图 12-2-2 定时器 1 的模式 2

- **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。

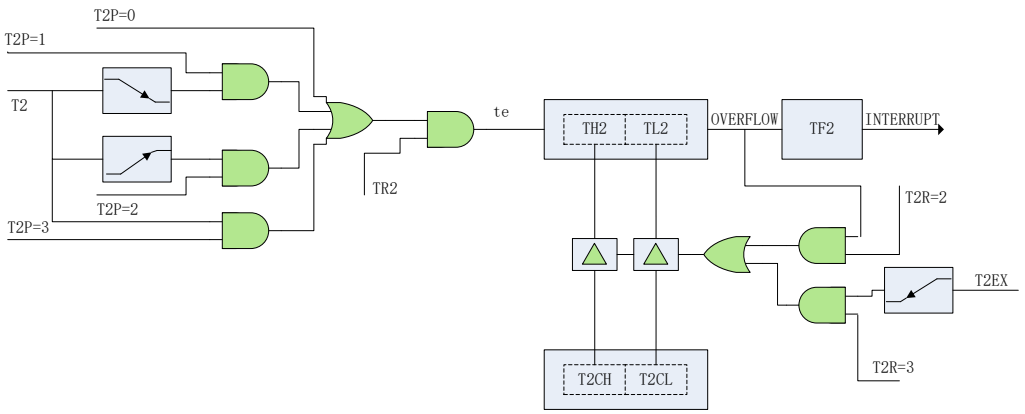


图 12-3-1-1 定时器 2 的重载模式

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。

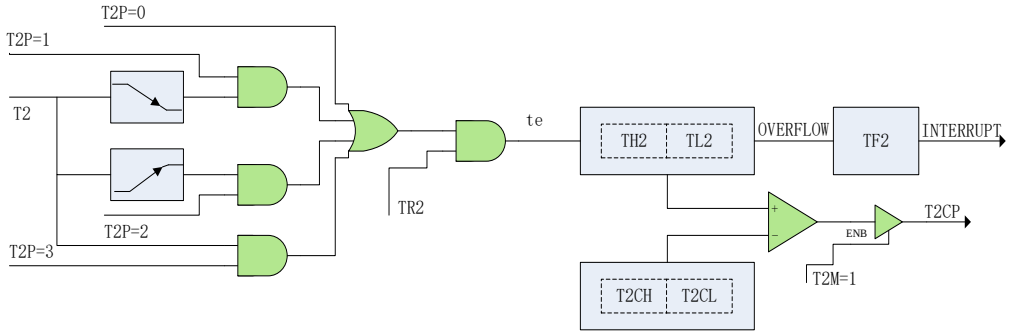


图 12-3-1-2 定时器 2 的比较模式

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。

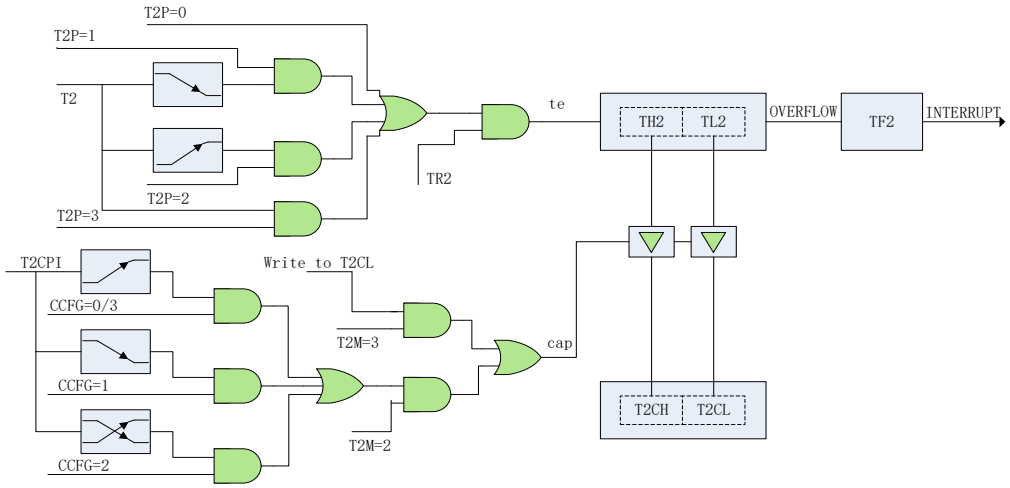


图 12-3-1-3 定时器 2 的抓取模式

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 11：定时器 2 使用内部系统时钟计数，通过 T2 门控
0	T2P0	

表 12-3-2-2 寄存器 T2MOD

C9H	7	6	5	4	3	2	1	0
T2MOD	TF2	CF2	RF2	CCFG1	CCFG0	-	T2M1	T2M0
R/W	R/W	R/W	R/W	R/W	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	TF2	Timer2 计数器溢出中断标志，写 1 清 0
6	CF2	抓取中断标志，写 1 清 0
5	RF2	自动重载中断标志，写 1 清 0
4	CCFG1	[CCFG1,CCFG0]抓取模式触发沿选择位，在 T2M＝3 或 T2M＝4 时有效 01：下降沿 10：上升或下降沿 其它值：上升沿
3	CCFG0	
2	-	-
1	T2M1	工作模式选择位 00：定时器/计数器模式 01：比较模式 10：抓取模式 0 11：抓取模式 1
0	T2M0	

表 12-3-2-3 寄存器 T2CL

CAH	7	6	5	4	3	2	1	0
T2CL	T2CL							
R/W	R/W							

初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
7~0	T2CL	在重载模式，T2CL 是重载值的低字节 在比较模式，T2CL 是比较值的低字节 在抓取模式，T2CL 保存捕获值的低字节						

表 12-3-2-4 寄存器 T2CH

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

表 12-3-2-5 寄存器 TL2

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

表 12-3-2-6 寄存器 TH2

CDH	7	6	5	4	3	2	1	0
TH2	TH2							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
7~0	TH2	定时器 2 计数值的高字节						

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 – 8.389s，有 16 位调节精度。看门狗主要用于监控系统，避免 CPU 因为外界干扰出现死机。如果软件不能在溢出前刷新看门狗定时器，看门狗将产生内部复位或者中断。写 A5H 到寄存器 WDFLG 将刷新看门狗，读 WDFLG 可得到看门狗状态。在 STOP 模式下，如果看门狗处于使能状态，则看门狗所选的时钟源正常工作，此时如果看门狗设为中断，看门狗中断可唤醒 CPU。

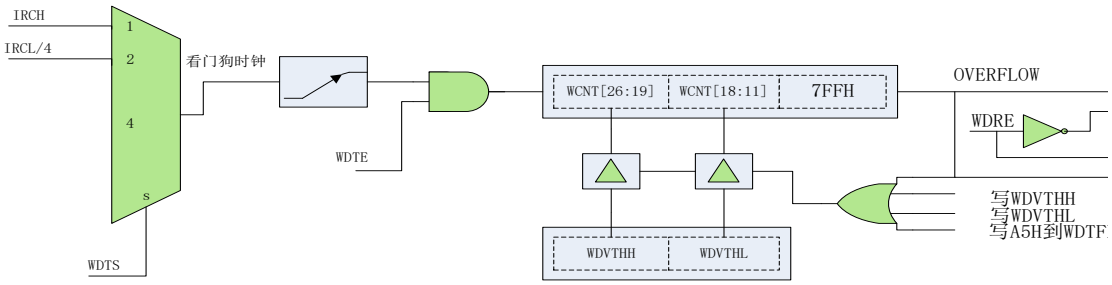


图 13-1-1 看门狗模块结构图

13.2 看门狗定时器(WDT)寄存器描述

表 13-2-1 寄存器 WDCON

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

位编号	位符号	说明
7~5	WDTS	WDT 时钟选择位 001：选择 IRCH 010：选择 IRCL 四分频 100：无效 其他：WDT 关闭
4~1	-	
0	WDRE	WDT 功能选择位 0：WDT 溢出后产生中断 1：WDT 溢出后产生复位

表 13-2-2 寄存器 WDFLG

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

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

表 13-2-3 寄存器 WDVTHL、WDVTHH

ACH	7	6	5	4	3	2	1	0
WDVTHL	WDVTH[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
ADH	7	6	5	4	3	2	1	0
WDVTHH	WDVTH[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	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<<5)
#define WDTS_IRCL          (1<<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 = 0x84;
    WDFLG = 0xA5;                     //刷新看门狗
    INT7EN = 1;                       //开启看门狗中断
    EA = 1;                           //开启总中断
}
void WDT_ISR (void) interrupt 12
{
    if(WDFLG & 0x02)
    {
        //看门狗中断服务程序
        WDFLG = 0xA5;//刷新看门狗
    }
}
-----------------------------------------------------------------------------------
```

◆ **看门狗复位模式例程**

例如，看门狗时钟设置为 IRCH，IRCH 的频率为 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<<5)
#define WDTS_IRCL          (1<<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 = 0x84;
    WDFLG = 0xA5;                    //刷新看门狗
}
```

---

14 实时定时器（RTC）

14.1 RTC 功能简介

内置 RTC 是一个实时时钟模块，主要时钟源是外部 32.768KHz 晶体振荡器，它包含毫秒、秒、分、时、天和星期寄存器，同时，还内置了闹钟功能，当 RTC 时间和设定的闹钟时间匹配时，会产生中断，这对于包含时钟和闹钟功能的产品来说特别方便。另外，RTC 还可以设置毫秒级中断、半秒中断，其中，毫秒中断中断时间可设置。RTC 在不外挂 32.768KHz 晶振的情况下，也可以设置 IRCL 的四分频作为 RTC 的时钟源，应用于对计时精度要求不高的场合。在 STOP/IDLE 模式，RTC 也可以开启并作为 STOP/IDLE 模式唤醒的触发源。RTC 结构图如图 14-1-1 所示。

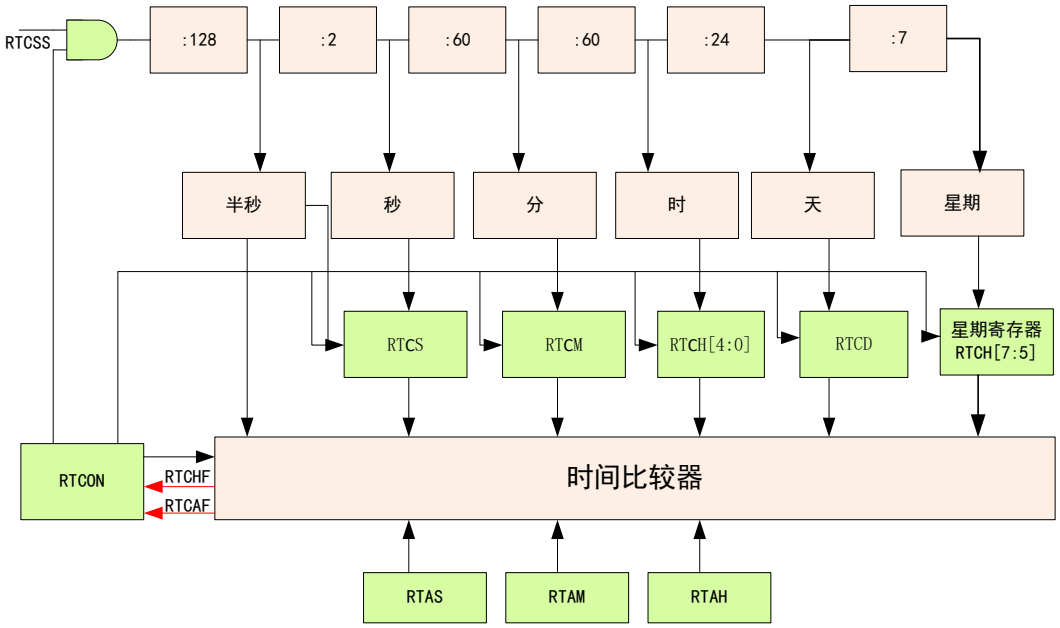


图 14-1-1　RTC 结构图

- **RTC 使能和关闭**

RTC 使能和关闭由 RTCE 位（RTCON[7]）控制。设置 RTCE=1 后，RTC 开始计时；设置 RTCE=0 后，RTC 模块所有寄存器的状态都会被锁存。RTC 使能后，需要等待 300us 才能写 RTC 时间寄存器，否则写入值无效。值得注意的是，由于 RTC 的时钟源主要是外挂 32.768KHz 晶振，必须要等待 32.768KHz 晶振正常起振后再使能 RTC 模式，否则可能会操作无效。

- **RTC 寄存器读写**

RTC 寄存器（RTCSS、RTCS、RTCM、RTCH、RTCDL、RTCDH）写入由 RTCWE 位（RTCON[1]）控制。RTCWE 置 1 后，需要等待 50us 才能改写 RTC 寄存器，改写后等待 50us 后 RTCWE 才变为 0。写一个非法时间（超过了有效秒、分或时范围的值）到 RTC 寄存器将会被当作该寄存器的最大有效值写入。在写秒、分

、时和星期时，微秒寄存器 RTCSS 会被清零，从而能精确的计时。RTC 寄存器可直接进行读取。

- **RTC 闹钟功能**

当 RTC 时间和闹钟时间匹配时，将产生闹钟中断，标志位为 RTCAF。用户可以通过寄存器 RTAS、RTAM、RTAH 来设置闹铃时间，不需要设置 RTCWE 位。写一个非法的值(超过了有效秒、分或时范围的值)将会当作该寄存器最大有效值来写入。用户可以设置相应的比较使能位（HCE、MCE、SCE）去比较 RTAS、RTAM 和 RTAH 寄存器的一个或多个值。如果相应比较使能位设为 0，相应的时间项比较会被忽略（例如设置 HCE=1、MCE=0、SCE=1，只会比较小时、秒寄存器，分寄存器会认为已匹配）。这样就允许闹铃可以在一天的特定时间发生一次（所有的比较位使能），或者周期性的一每秒一次、每分钟一次、每小时一次（通过比较使能位的组合设置来实现）。

14.2 RTC 寄存器描述

表 14-2-1 寄存器 RTCON

F1H	7	6	5	4	3	2	1	0
RTCON	RTCE	MSE	HSE	SCE	MCE	HCE	RTCWE	-
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	RTCE	RTC 时钟使能，1 有效
6	MSE	毫秒中断使能信号，1 有效
5	HSE	半秒中断使能信号，1 有效
4	SCE	闹钟秒比较使能，1 有效
3	MCE	闹钟分钟比较使能，1 有效
2	HCE	闹钟小时比较使能，1 有效
1	RTCWE	时钟写使能，1 有效
0	-	-

表 14-2-2 寄存器 RTCSS

E9H	7	6	5	4	3	2	1	0
RTCSS	RTCSS[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	RTCE	RTC 微秒计数器，每 1/256 秒加 1

表 14-2-3 寄存器 RTCS

F2H	7	6	5	4	3	2	1	0
RTCS	-	-	RTCS[5:0]					
R/W	-	-	R/W					
初始值	-	-	0	0	0	0	0	0

位编号	位符号	说明
7~6	-	-
5~0	RTCS	秒计数器，每 1 秒加 1，计数范围为 0~59

表 14-2-3 寄存器 RTCM

F3H	7	6	5	4	3	2	1	0
RTCM	-	-	RTCM[5:0]					
R/W	-	-	R/W					
初始值	-	-	0	0	0	0	0	0

位编号	位符号	说明
7~6	-	-
5~0	RTCM	分计数器，每分钟加 1，计数范围为 0~59

表 14-2-4 寄存器 RTCH

F4H	7	6	5	4	3	2	1	0
RTCH	RTCW[2:0]			RTCH[4:0]				
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
7~5	RTCW	星期计数器，计数范围为 1~7，代表星期一到星期日，当设置为 0 时，星期计数功能关闭
4~0	RTCH	小时计数器，每小时该加 1，计数范围为 0~23

表 14-2-5 寄存器 RTCDL、RTCDH

F5H	7	6	5	4	3	2	1	0
RTCDL	RTCD[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
RTCDH	RTCD[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	RTCD	天数计数器，每天计数加 1						

表 14-2-6 寄存器 RTAS

EAH	7	6	5	4	3	2	1	0
RTAS	-	-	RTAS[5:0]					
R/W	-	-	R/W					
初始值	-	-	0	0	0	0	0	0
位编号	位符号	说明						
7~6	-	-						
5~0	RTAS	闹钟秒值设定，取值范围为 0~59						

表 14-2-7 寄存器 RTAM

EBH	7	6	5	4	3	2	1	0
RTAM	-	-	RTAM[5:0]					
R/W	-	-	R/W					
初始值	-	-	0	0	0	0	0	0
位编号	位符号	说明						
7~6	-	-						
5~0	RTAM	闹钟分钟值设定，取值范围为 0~59						

表 14-2-8 寄存器 RTAH

ECH	7	6	5	4	3	2	1	0
RTAH	-	-	-	RTAH[4:0]				
R/W	-	-	-	R/W				
初始值	-	-	-	0	0	0	0	0
位编号	位符号	说明						
7~5	-	-						
4~0	RTAH	闹钟小时值设定，取值范围为 0~23						

表 14-2-9 寄存器 RTMSS

EDH	7	6	5	4	3	2	1	0

RTMSS	RTMSS[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	RTMSS	RTC 毫秒中断阈值寄存器，毫秒中断时间=（RTMSS+1）x128xRTC 时钟周期。如果 RTC 时钟是 32.768KHz，那设置的时间单位是 128x（1/32.768）=3.90625ms。

表 14-2-10 寄存器 RTCIF

EEH	7	6	5	4	3	2	1	0
RTCIF	-	-	-	-	-	RTCMF	RTCHF	RTCAF
R/W	-	-	-	-	-	R	R	R
初始值	-	-	-	-	-	0	0	0

位编号	位符号	说明
7~3	-	-
2	RTCMF	RTC 毫秒中断标志，写 1 清 0
1	RTCHF	RTC 半秒中断标志，写 1 清 0
0	RTCAF	RTC 闹钟中断标志，写 1 清 0

14.3 RTC 控制例程

◆ **RTC 写入时间**

写时、分、秒程序如下：

```
-----------------------------------------------------------------------------------------
//CKCON 寄存器定义
#define XLCKE           (1<<3)
#define XLSTA           (1<<2)
//RTCON 定义
#define RTCE(N)         (N<<7)
#define MSE(N)          (N<<6)
#define HSE(N)          (N<<5)
#define SCE(N)          (N<<4)
#define MCE(N)          (N<<3)
#define HCE(N)          (N<<2)
#define RTCWE(N)        (N<<1)
//RTCIF 定义
#define RTC_MF          (1<<2)
#define RTC_HF          (1<<1)
#define RTC_AF          (1<<0)
void RTC_WriteHour(unsigned char hour)        //hour=0~23
{
    RTCON |= RTCWE(1);
    RTCH = (RTCH&0xE0)|hour;
    Delay_50us(1);
    RTCON &= ~RTCWE(1);
}
void RTC_WriteMinute(unsigned char minute) //minute=0~59
{
    RTCON |= RTCWE(1);
    RTCM = minute;
    Delay_50us(1);
    RTCON &= ~RTCWE(1);
}
void RTC_WriteSecond(unsigned char second) //second=0~59
{
    RTCON |= RTCWE(1);
    RTCS = second;
    Delay_50us(1);
    RTCON &= ~RTCWE(1);
}
-----------------------------------------------------------------------------------------
```

◆ **RTC 设置闹钟时间**

例如，设置闹钟时间为 11:30:00，时、分、秒比较全使能，程序如下：

```
-----------------------------------------------------------------------------------
void RTM_init(void)
{
    RTAH = 11;    //设置闹钟小时
    RTAM = 30;    //设置闹钟分
    RTAS = 00;    //设置闹钟秒
    RTCON |= SCE(1)|MCE(1)|HCE(1);    //时、分、秒比较使能
}
void RTC_ISR (void) interrupt 13
{
    if(RTCIF & RTC_AF)                //闹钟中断
    {
        RTCIF = RTC_AF;
        //闹钟中断服务程序
    }
}
-----------------------------------------------------------------------------------
```

◆ **RTC 初始化**

RTC 初始化程序如下：

```
-----------------------------------------------------------------------------------
void RTC_init(void)
{
    P52F = 3;
    P53F = 3;
    CKCON |= XLCKE;
    while(!(CKCON & XLSTA));
    RTCON = RTCE(1) | MSE(1) | HSE(1);  //RTC 使能，毫秒中断使能，半秒中断使能
    Delay_50us(6);      //RTC 使能后必须延时 300us 再写入时间，  否则写入时间可能无效。
    RTC_WriteHour(11);     //写入小时
    RTC_WriteMinute(29);   //写入分
    RTC_WriteSecond(0);    //写入秒
    RTM_init();    //设置闹钟
    RTMSS = 0;    //设置毫秒中断时间
    INT8EN = 1;   //开启 RTC 中断
}
void RTC_ISR (void) interrupt 13
{
    if(RTCIF & RTC_MF)              //毫秒中断
    {
        RTCIF = RTC_MF;
```

```
            //毫秒中断服务程序

        }
        if(RTCIF & RTC_HF)              //半秒中断
        {
            RTCIF = RTC_HF;
            //半秒中断服务程序
        }
        if(RTCIF & RTC_AF)              //闹钟中断
        {
            RTCIF = RTC_AF;
            //闹钟中断服务程序
        }
}
-----------------------------------------------------------------------------------------
```

15 通用输入输出口（GPIO）及复用定义

15.1 功能简介

CA51F4 系列芯片最大封装有 46 个 I/O 引脚，每个引脚都是复用功能引脚，不仅能独立编程为输入/输出口，而且还能设置为其他功能引脚。每个引脚都分配了一个功能设置寄存器 PnxF（分别对应引脚 Pnx，其中 n=0、1、2、3、4、5，代表 P0、P1、P2、P3、P4、P5，x=0~7，代表 Pn.0~Pn.7，当 n=5 时，x=0~5），用户可通过寄存器 PnxF 配置引脚的主功能和其他选项。详见寄存器部分介绍。

GPIO 的主要特性如下：

- 可配置为高阻模式
- I/O 结构可独立设置上拉电阻
- 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

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。

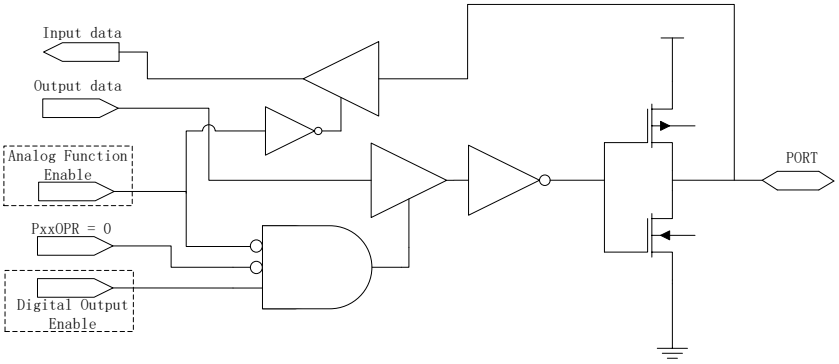


图 15-1-1 I/O 推挽模式结构示意图

GPIO 开漏模式结构图如图 15-1-2 所示。

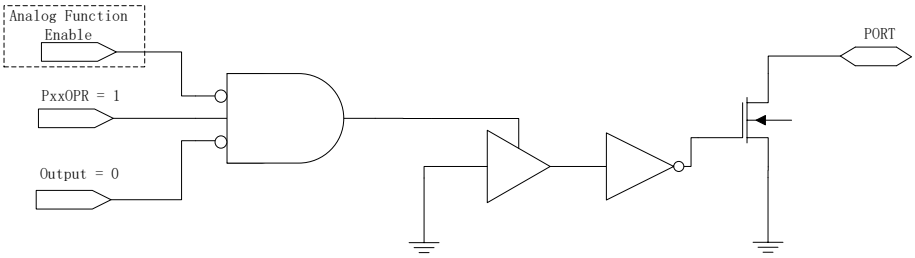


图 15-1-2 I/O 开漏模式结构示意图

GPIO 上拉结构图如图 15-1-3 所示。

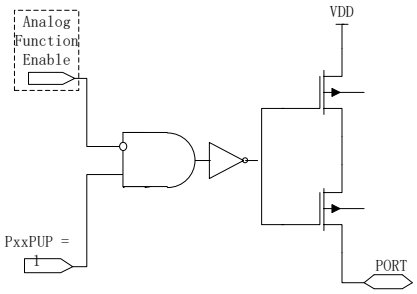


图 15-1-3 I/O 上拉模式结构示意图

15.2 引脚寄存器描述

表 15-2-1 寄存器 P0

80H	7	6	5	4	3	2	1	0
P0	P07	P06	P05	P04	P03	P02	P01	P00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0

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

表 15-2-2 寄存器 P1

90H	7	6	5	4	3	2	1	0
P1	P17	P16	P15	P14	P13	P12	P11	P10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0

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

表 15-2-3 寄存器 P2

A0H	7	6	5	4	3	2	1	0
P2	P27	P26	P25	P24	P23	P22	P21	P20
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
7~1	-	-
0	P2x	引脚 P2x 的数据寄存器，管脚功能设置为 GPIO 时有效 0：设为输入时 P2x 电平为低，设为输出时 P2x 输出低电平 1：设为输入时 P2x 电平为高，设为输出时 P2x 输出高电平

表 15-2-4 寄存器 P3

B0H	7	6	5	4	3	2	1	0
P3	P37	P36	P35	P34	P33	P32	P31	P30
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
7~1	-	-
0	P3x	引脚 P30 的数据寄存器，管脚功能设置为 GPIO 时有效 0：设为输入时 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 15-2-5 寄存器 P4

C0H	7	6	5	4	3	2	1	0

P4	P47	P46	P45	P44	P43	P42	P41	P40
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	P4x	引脚 P4x 的数据寄存器，管脚功能设置为 GPIO 时有效 0：设为输入时 P4x 电平为低，设为输出时 P4x 输出低电平 1：设为输入时 P4x 电平为高，设为输出时 P4x 输出高电平

表 15-2-6 寄存器 P5

D8H	7	6	5	4	3	2	1	0
P5	-	-	P55	P54	P53	P52	P51	P50
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
7~0	P5x	引脚 P5x 的数据寄存器，管脚功能设置为 GPIO 时有效 0：设为输入时 P5x 电平为低，设为输出时 P5x 输出低电平 1：设为输入时 P5x 电平为高，设为输出时 P5x 输出高电平

表 15-2-7 引脚功能控制寄存器

8000H	7	6	5	4	3	2	1	0
P00F	P00PUP	-	P00OPR	-	-	P00S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
8001H	7	6	5	4	3	2	1	0
P01F	P01PUP	-	P01OPR	-	-	P01S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
8002H	7	6	5	4	3	2	1	0
P02F	P02PUP	-	P02OPR	-	-	P02S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
8003H	7	6	5	4	3	2	1	0
P03F	P03PUP	-	P03OPR	-	-	P03S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
8004H	7	6	5	4	3	2	1	0
P04F	P04PUP	-	P04OPR	-	-	P04S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
8005H	7	6	5	4	3	2	1	0
P05F	P05PUP	-	P05OPR	-	-	P05S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	1	1
8006H	7	6	5	4	3	2	1	0
P06F	P06PUP	-	P06OPR	-	-	P06S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	1	1
8007H	7	6	5	4	3	2	1	0
P07F	P07PUP	-	P07OPR	-	-	P07S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	1	1
8008H	7	6	5	4	3	2	1	0
P10F	P10PUP	-	P10OPR	-	-	P10S		

R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
8009H	7	6	5	4	3	2	1	0
P11F	P11PUP	-	P11OPR	-	-	P11S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
800AH	7	6	5	4	3	2	1	0
P12F	P12PUP	-	P12OPR	-	-	P12S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
800BH	7	6	5	4	3	2	1	0
P13F	P13PUP	-	P13OPR	-	-	P13S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
800CH	7	6	5	4	3	2	1	0
P14F	P14PUP	-	P14OPR	-	-	P14S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
800DH	7	6	5	4	3	2	1	0
P15F	P15PUP	-	P15OPR	-	-	P15S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
800EH	7	6	5	4	3	2	1	0
P16F	P16PUP	-	P16OPR	-	-	P16S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
800FH	7	6	5	4	3	2	1	0
P17F	P17PUP	-	P17OPR	-	-	P17S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
8010H	7	6	5	4	3	2	1	0
P20F	P20PUP	-	P20OPR	-	-	P20S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0

8011H	7	6	5	4	3	2	1	0
P21F	P21PUP	-	P21OPR	-	-	P21S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0

8012H	7	6	5	4	3	2	1	0
P22F	P22PUP	-	P22OPR	-	-	P22S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0

8013H	7	6	5	4	3	2	1	0
P23F	P23PUP	-	P23OPR	-	-	P23S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0

8014H	7	6	5	4	3	2	1	0
P24F	P24PUP	-	P24OPR	-	-	P24S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0

8015H	7	6	5	4	3	2	1	0
P25F	P25PUP	-	P25OPR	-	-	P25S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0

8016H	7	6	5	4	3	2	1	0
P26F	P26PUP	-	P26OPR	-	-	P26S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0

8017H	7	6	5	4	3	2	1	0
P27F	P27PUP	-	P27OPR	-	-	P27S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0

8018H	7	6	5	4	3	2	1	0
P30F	P30PUP	-	P30OPR	-	-	P30S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0

8019H	7	6	5	4	3	2	1	0
P31F	P31PUP	-	P31OPR	-	-	P31S		
R/W	R/W	-	R/W	-	-	R/W		

初始值	0	-	0	-	-	0	0	0
801AH	7	6	5	4	3	2	1	0
P32F	P32PUP	-	P32OPR	-	-	P32S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
801BH	7	6	5	4	3	2	1	0
P33F	P33PUP	-	P33OPR	-	-	P33S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
801CH	7	6	5	4	3	2	1	0
P34F	P34PUP	-	P34OPR	-	-	P34S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
801DH	7	6	5	4	3	2	1	0
P35F	P35PUP	-	P35OPR	-	-	P35S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
8020H	7	6	5	4	3	2	1	0
P40F	P40PUP	-	P40OPR	-	-	P40S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
8021H	7	6	5	4	3	2	1	0
P41F	P41PUP	-	P41OPR	-	-	P41S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
8022H	7	6	5	4	3	2	1	0
P42F	P42PUP	-	P42OPR	-	-	P42S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
8023H	7	6	5	4	3	2	1	0
P43F	P43PUP	-	P43OPR	-	-	P43S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
8024H	7	6	5	4	3	2	1	0

P44F	P44PUP	-	P44OPR	-	-	P44S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
8025H	7	6	5	4	3	2	1	0
P45F	P45PUP	-	P45OPR	-	-	P45S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
8026H	7	6	5	4	3	2	1	0
P46F	P46PUP	-	P46OPR	-	-	P46S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
8027H	7	6	5	4	3	2	1	0
P47F	P47PUP	-	P47OPR	-	-	P47S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
8028H	7	6	5	4	3	2	1	0
P50F	P50PUP	-	P50OPR	-	-	P50S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
8029H	7	6	5	4	3	2	1	0
P51F	P51PUP	-	P51OPR	-	-	P51S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
802AH	7	6	5	4	3	2	1	0
P52F	P52PUP	-	P52OPR	-	-	P52S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
802BH	7	6	5	4	3	2	1	0
P53F	P53PUP	-	P53OPR	-	-	P53S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0
802CH	7	6	5	4	3	2	1	0
P54F	P54PUP	-	P54OPR	-	-	P54S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	0	0

802DH	7	6	5	4	3	2	1	0
P55F	P55PUP	-	P55OPR	-	-	P55S		
R/W	R/W	-	R/W	-	-	R/W		
初始值	0	-	0	-	-	0	1	1

备注：

PXnF 代表 P00F~P55F，其中 X=0,1,2,3,4,5 代表 P0~P5，n=0,1,2,...,7(当 X=5 时 n<=5)。

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	-	-
5	PnxOPR	开漏使能控制位，引脚设为数字输出时才有效 0: 开漏关闭 1: 开漏打开

表 15-2-8 寄存器 PXnC

8120H	7	6	5	4	3	2	1	0
P00C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
8121H	7	6	5	4	3	2	1	0
P01C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
8122H	7	6	5	4	3	2	1	0
P02C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
8123H	7	6	5	4	3	2	1	0
P03C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
8124H	7	6	5	4	3	2	1	0
P04C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-

8125H	7	6	5	4	3	2	1	0
P05C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-

8126H	7	6	5	4	3	2	1	0
P06C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-

8127H	7	6	5	4	3	2	1	0
P07C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-

8128H	7	6	5	4	3	2	1	0
P10C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-

8129H	7	6	5	4	3	2	1	0
P11C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-

812AH	7	6	5	4	3	2	1	0
P12C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-

812BH	7	6	5	4	3	2	1	0
P13C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-

812CH	7	6	5	4	3	2	1	0
P14C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-

812DH	7	6	5	4	3	2	1	0
P15C	-	SMIT_EN	-	-	-	-	-	-

R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
812EH	7	6	5	4	3	2	1	0
P16C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
812FH	7	6	5	4	3	2	1	0
P17C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
8130H	7	6	5	4	3	2	1	0
P20C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
8131H	7	6	5	4	3	2	1	0
P21C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
8132H	7	6	5	4	3	2	1	0
P22C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
8133H	7	6	5	4	3	2	1	0
P23C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
8134H	7	6	5	4	3	2	1	0
P24C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
8135H	7	6	5	4	3	2	1	0
P25C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-

8136H	7	6	5	4	3	2	1	0
P26C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
8137H	7	6	5	4	3	2	1	0
P27C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
8138H	7	6	5	4	3	2	1	0
P30C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
8139H	7	6	5	4	3	2	1	0
P31C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
813AH	7	6	5	4	3	2	1	0
P32C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
813BH	7	6	5	4	3	2	1	0
P33C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
813CH	7	6	5	4	3	2	1	0
P34C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
813DH	7	6	5	4	3	2	1	0
P35C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
813EH	7	6	5	4	3	2	1	0
P36C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-

初始值	-	1	-	-	-	-	-	-
813FH	7	6	5	4	3	2	1	0
P37C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
8140H	7	6	5	4	3	2	1	0
P40C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
8141H	7	6	5	4	3	2	1	0
P41C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
8142H	7	6	5	4	3	2	1	0
P42C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
8143H	7	6	5	4	3	2	1	0
P43C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
8144H	7	6	5	4	3	2	1	0
P44C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
8145H	7	6	5	4	3	2	1	0
P45C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
8146H	7	6	5	4	3	2	1	0
P46C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
8147H	7	6	5	4	3	2	1	0

P47C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
8148H	7	6	5	4	3	2	1	0
P50C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
8149H	7	6	5	4	3	2	1	0
P51C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
814AH	7	6	5	4	3	2	1	0
P52C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
814BH	7	6	5	4	3	2	1	0
P53C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-
814CH	7	6	5	4	3	2	1	0
P54C	SINK_EN	SMIT_EN	-	-	-	-	-	-
R/W	R/W	R/W	-	-	-	-	-	-
初始值	0	1	-	-	-	-	-	-
814DH	7	6	5	4	3	2	1	0
P55C	-	SMIT_EN	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
初始值	-	1	-	-	-	-	-	-

备注：

PXnC 代表 P00C~P55C，其中 X=0,1,2,3,4,5 代表 P0~P5，n=0,1,2,...,7(当 X=5 时 n=0,1,2,...,5)。

位编号	位符号	说明
7	SINK_EN	为 1 时大灌电流使能，为 0 不使能
6	SMIT_EN	为 1 输入的 SMIT 使能，为 0 输入是反相器使能
5~0	-	-

表 15-2-9 寄存器 RMCTL

811DH	7	6	5	4	3	2	1	0

RMCTL	-	-	-	-	-	-	SINK_SEL	
R/W	-	-	-	-	-	-	R/W	R/W
初始值	-	-	-	-	-	-	0	0

位编号	位符号	说明
7~2	-	-
1~0	SINK_SEL	引脚 P54 的灌电流能力选择位： 00：125mA 的灌电流 01：225mA 的灌电流 10：315mA 的灌电流 11：400mA 的灌电流

表 15-2-10 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	高阻	高阻	高阻	VP3	高阻
P01S	高阻	数字输入	数字输出	高阻	高阻	高阻	VP2	高阻
P02S	高阻	数字输入	数字输出	高阻	高阻	高阻	VP1	高阻
P03S	高阻	数字输入	数字输出	高阻	高阻	高阻	CUP1	高阻
P04S	高阻	数字输入	数字输出	高阻	高阻	高阻	CUP2	高阻
P05S	高阻	数字输入	数字输出	I2C_SCL	高阻	TK15	LCD_S29	高阻
P06S	高阻	数字输入	数字输出	I2C_SDA	PWM1	TK0	TLCOM	高阻
P07S	高阻	数字输入	数字输出	SWIM	高阻	高阻	LCD_S30	高阻
P10S	高阻	数字输入	数字输出	高阻	高阻	TK_CAP	LCD_S0	高阻
P11S	高阻	数字输入	数字输出	高阻	高阻	TK1	LCD_S1	高阻
P12S	高阻	数字输入	数字输出	高阻	高阻	TK2	LCD_S2	高阻
P13S	高阻	数字输入	数字输出	高阻	高阻	TK3	LCD_S3	高阻
P14S	高阻	数字输入	数字输出	高阻	高阻	TK4	LCD_S4	高阻
P15S	高阻	数字输入	数字输出	高阻	高阻	TK5	LCD_S5	高阻
P16S	高阻	数字输入	数字输出	高阻	高阻	TK6	LCD_S6	高阻
P17S	高阻	数字输入	数字输出	高阻	高阻	TK7	LCD_S7	高阻
P20S	高阻	数字输入	数字输出	高阻	高阻	TK8	LCD_S8	高阻
P21S	高阻	数字输入	数字输出	UART1_RX	高阻	TK9	LCD_S9	高阻
P22S	高阻	数字输入	数字输出	UART1_TX	高阻	TK10	LCD_S10	高阻
P23S	高阻	数字输入/T1	数字输出	高阻	高阻	TK11	LCD_S11	高阻
P24S	高阻	数字输入/T2	数字输出	高阻	高阻	TK12	LCD_S12	高阻
P25S	高阻	数字输入/T2EX	数字输出	T2CP	高阻	TK13	LCD_S13	高阻
P26S	高阻	数字输入/T0	数字输出	高阻	高阻	TK14	LCD_S14	高阻
P27S	高阻	数字输入	数字输出	ADC0	高阻	高阻	LCD_S15	高阻
P30S	高阻	数字输入	数字输出	高阻	PWM2	CLK_IN	LCD_S16	高阻
P31S	高阻	数字输入	数字输出	ADC1	高阻	高阻	LCD_S17	高阻

P32S	高阻	数字输入	数字输出	ADC2	高阻	高阻	LCD_S18	高阻
P33S	高阻	数字输入	数字输出	ADC3	高阻	高阻	LCD_S19	高阻
P34S	高阻	数字输入	数字输出	ADC4	高阻	高阻	LCD_S20	高阻
P35S	高阻	数字输入	数字输出	ADC5	高阻	高阻	LCD_S21	高阻
P36S	高阻	数字输入	数字输出	ADC6	高阻	高阻	LCD_S22	高阻
P37S	高阻	数字输入	数字输出	ADC7	ADC_VREF	高阻	LCD_S23	高阻
P40S	高阻	数字输入	数字输出	高阻	高阻	高阻	LCD_S24	高阻
P41S	高阻	数字输入	数字输出	高阻	高阻	高阻	LCD_S25	高阻
P42S	高阻	数字输入	数字输出	高阻	高阻	高阻	LCD_S26	高阻
P43S	高阻	数字输入	数字输出	高阻	高阻	高阻	LCD_S27	高阻
P44S	高阻	数字输入	数字输出	高阻	高阻	高阻	LCD_S28	高阻
P45S	高阻	数字输入	数字输出	高阻	高阻	LCD_S31	LCD_C4	高阻
P46S	高阻	数字输入	数字输出	高阻	高阻	高阻	LCD_C3	高阻
P47S	高阻	数字输入	数字输出	高阻	高阻	高阻	LCD_C2	高阻
P50S	高阻	数字输入	数字输出	高阻	高阻	高阻	LCD_C1	高阻
P51S	高阻	数字输入	数字输出	高阻	高阻	高阻	LCD_C0	高阻
P52S	高阻	数字输入	数字输出	32K_O	高阻	高阻	高阻	高阻
P53S	高阻	数字输入	数字输出	32K_I	高阻	高阻	高阻	高阻
P54S	高阻	数字输入	数字输出	高阻	PWM0/REM	高阻	高阻	高阻
P55S	高阻	数字输入	数字输出	RESET	高阻	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，并且打开上拉，程序如下：

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

P00 设置为输入功能，并且打开上拉，程序如下：

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

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

---

16 通用串行接口（UART）

16.1 UART

16.1.1 介绍

UART 是全双工异步串行数据收发器，UART 有一字节的接收缓存。UART 有两种不同的工作模式，如表 16-1-1-1 所示。

表 16-1-1-1 UART 工作模式

SM1	模式	描述	波特率
0	A	9 位异步模式	CPUCLK/(32*(1024-SREL))
1	B	8 位异步模式	CPUCLK/(32*(1024-SREL))

UART 设计了专门的波特率发生器，波特率通过寄存器 SRELL、SRELH 来配置。

图 16-1-1-1 是 UART 的原理示意图。

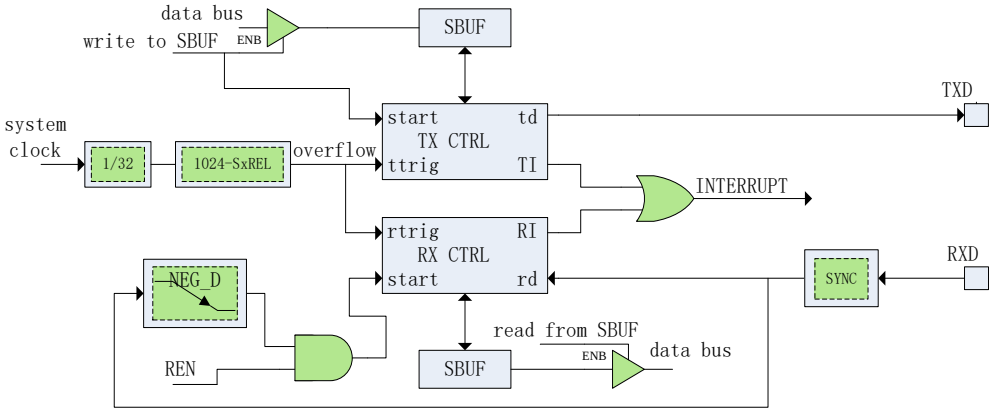


图 16-1-1-1 UART 工作原理示意图

- **模式 A**

在模式 A，UART 可异步同时收发 9 位数据。写入数据到寄存器 SBUF 会启动 UART 数据发送。第一个传送的位是开始位（为 0），然后是 9 位数据（低位先发），第 9 位数据是寄存器 SCON 的 TB81 位，最后传送的是停止位（为 1）。在接收状态，UART 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 SBUF，第 9 位数据存放在 RB81 位。

- **模式 B**

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

- **UART 多机通信**

在 UART 模式 A 中有一个专门适用于多机通信的机制。当寄存器 SCON 的 SM2x 位置 1，只有接收到第 9 位数据为 1（RB81=1）的从机才会产生接收中断，利用这个功能可进行多机通信，从机将它们的 SM2x 位都置为 1，主机传送从机的地址时将第 9 位数据设为 1，这样所有的从机都会产生接收中断；从机的软件用它们自己的地址和接收的地址进行比较，如果一致，被寻址的从机设置 SM2x=0，然后主机继续传送后面的数据时设置第 9 位为 0，因为其他的从机 SM2x 仍然设为 1，这样就只有被寻址的从机才会产生接收中断。

16.1.2 UART 寄存器描述

表 16-1-2-1 寄存器 SCON

9AH	7	6	5	4	3	2	1	0
SCON	SM1	U1IE	SM21	REN1	TB81	RB81	TI1	RI1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0

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

表 16-1-2-2 寄存器 SBUF

9BH	7	6	5	4	3	2	1	0
SBUF	SBUF[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	SBUF	UART 收发缓冲器

		写 SBUF 将开始发送所写的数据 读 SBUF 将得到已经接收的数据

表 16-1-2-3 寄存器 SRELL、SRELH

9CH	7	6	5	4	3	2	1	0
SRELL	SRELL[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
9DH	7	6	5	4	3	2	1	0
SRELH	-	-	-	-	-	-	SREL[9:8]	
R/W	-	-	-	-	-	-	R/W	
初始值	-	-	-	-	-	-	0	0

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

表 16-1-2-7 寄存器 UDCKS

8118H	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，UART 波特率按照原来的配置，UDE=1，UART 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 发送时，须满足 DNUM>=0；接收时，DNUM>=6 $BRx = F_{sys}*(1/((DNUM+1)*(1024-SRELL)))$

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。

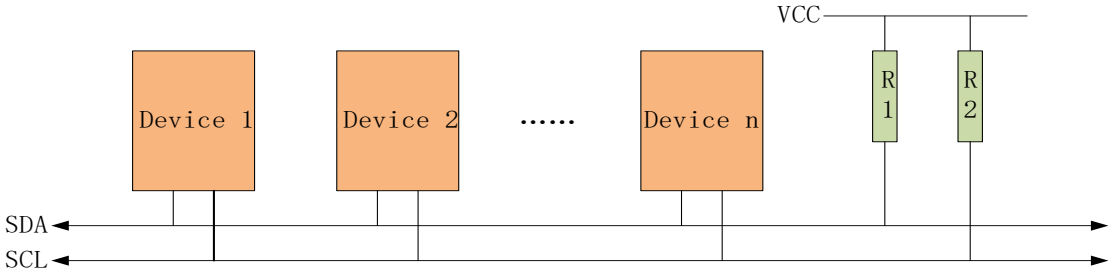


图 17-3-1 I2C 总线互连图

I²C 模块原理示意图如图图 17-3-2 所示。

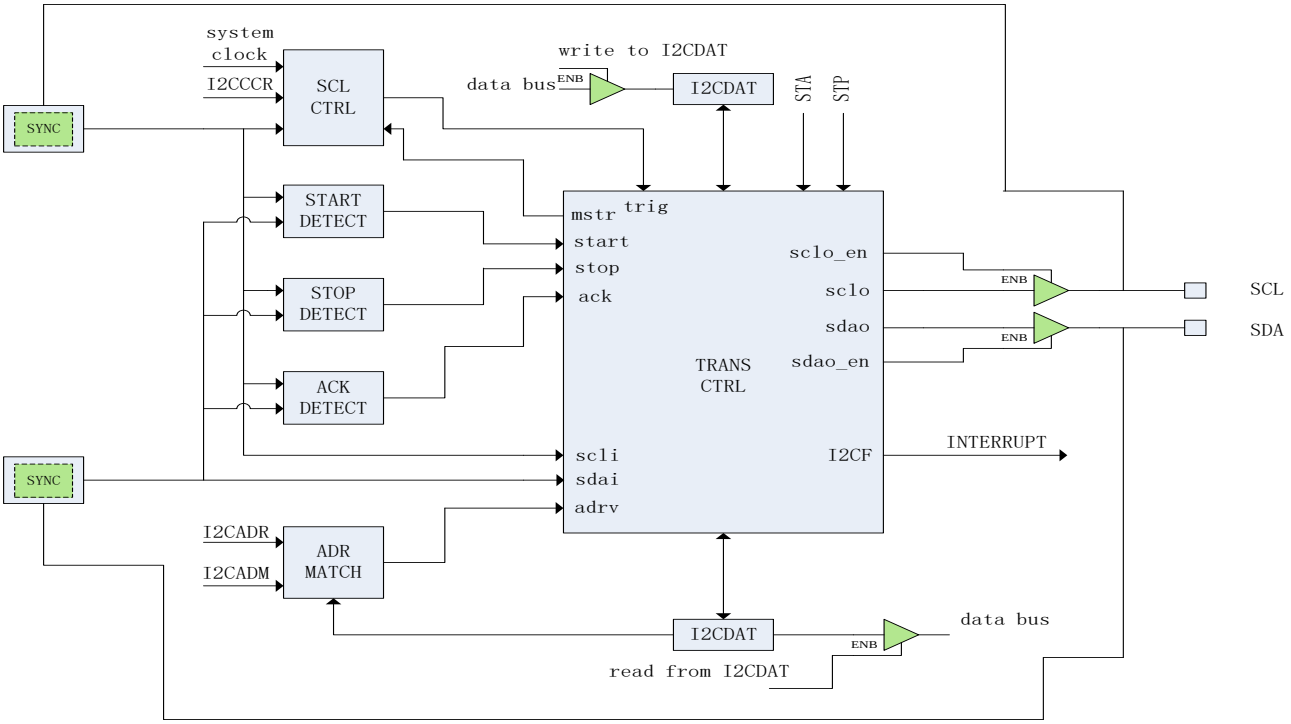


图 17-3-2　I²C 模块原理示意图

- **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。

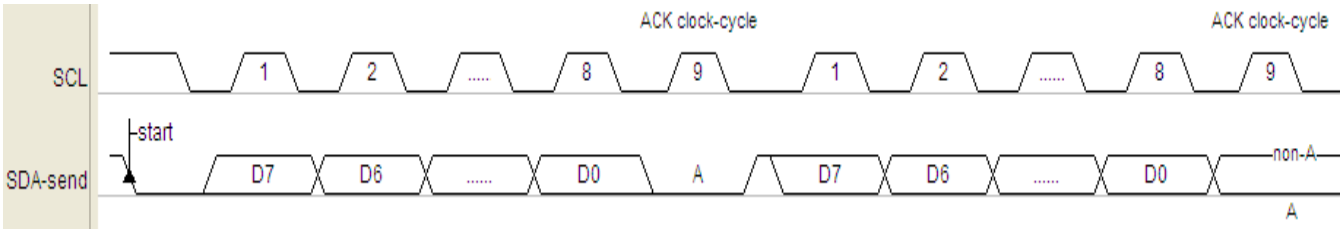


图 17-1-3　I2C 总线数据传输格式

- **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 寄存器描述

表 17-4-1 寄存器 I2CCON

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

位编号	位符号	说明
7	I2CE	I^2C 模块使能位，1 有效
6	I2CIE	I^2C 中断使能位，1 有效
5	STA	I^2C 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	I^2C 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	I^2C 发送 ACK 信号控制位，1 有效 *备注：* *当 I^2C 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，I^2C 模块检测到 START 信号时将置位 I2CF

表 17-4-2 寄存器 I2CADR

B2H	7	6	5	4	3	2	1	0

I2CADR	GCE	I2CADRL[6:0]						
R/W	R/W	R/W						
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7	GCE	识别广播地址（00H）使能位，1 有效
6~0	I2CADRL	I^2C 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 17-4-3 寄存器 I2CADM

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

位编号	位符号	说明
7	SPFE	为 1 时，I^2C 模块检测到 STOP 信号时将置位 I2CF
6~0	I2CADML	I^2C 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 17-4-4 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 17-4-5 寄存器 I2CDAT

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

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

表 17-4-6 寄存器 I2CSTA

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

位编号	位符号	说明
7~0	I2CSTA	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 17-4-7 寄存器 I2CFLG

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

位编号	位符号	说明
7~1	-	-
0	I2CF	I^2C 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，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。*

17.6 I^2C 控制例程

◆ **I^2C 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*************************************************************/
    P06F = 3| (1<<7);
    P05F = 3| (1<<7);
/*****************************************************************************/
    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*************************************************************/
    P06F = 3| (1<<7);
    P05F = 3| (1<<7);
/*****************************************************************************/
    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*************************************************************/
    P06F = 3| (1<<7);
    P05F = 3| (1<<7);
/*****************************************************************************/
    I2CCON = I2CE(1) | I2CIE(1) | STA(0) | STP(0)| CKHD(1) | AAK(1)| CBSE(0) | STFE(0);
    I2CADR = GCE(0)|(I2C_ADDR>>1);          //设置 I2C 从机地址
    I2CCCR = 0x20;                           //设置 I2C 采样时钟
    INT6EN = 1;                              //I2C 中断开启
    while(1)
    {
    }
}
```

---

18 PWM

18.1 PWM 功能简介

CA51F4 系列芯片最多有 3 通道 PWM 输出，PWM 周期和占空比可在 16 位范围内任意配置。

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

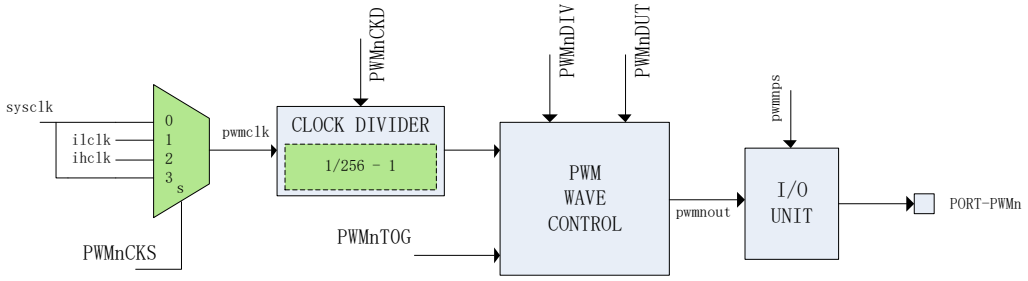


图 18-2-1 PWM 原理示意图

- **PWM 输出波形**

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

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

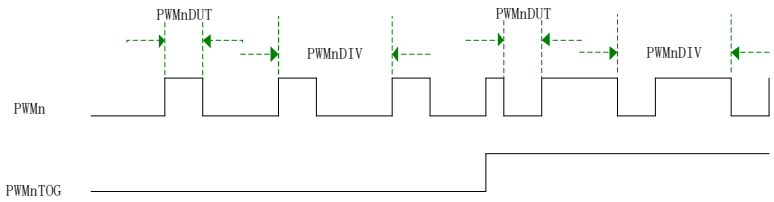


图 18-2-2 PWM 输出波形

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

- **PWM 中断**

PWM 中断通过寄存器 PWMCON 的 PWMIE 位使能，当 PWM 计数器计数到顶点（即等于 PWMDIV)时产生的中断。寄存器 PWMIF 包含 3 个通道的中断标志位。

18.3 PWM 寄存器描述

表 18-3-1 寄存器 PWMEN

DAH	7	6	5	4	3	2	1	0
PWMEN	-	-	-	-	-	PWM2EN	PWM1EN	PWM0EN
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	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

DDH	7	6	5	4	3	2	1	0
PWMCON	PWM0IE	PWMTOG	-	-	-	PWMCKS[2:0]		
R/W	R/W	R/W	-	-	-	R/W		
初始值	0	0	-	-	-	-	0	0

备注：PWMCON 为索引寄存器，INDEX=0/1/2 分别将配置 PWM0/1/2。

位编号	位符号	说明
7	PWMIE	PWM 计数器溢出中断使能控制位，1 有效
6	PWMTOG	PWM 输出取反使能控制位，1 有效
5~3	-	-
2~0	PWMCKS	PWM 工作时钟选择位 001：IRCH

		010：IRCL 100：XOSCL 其他：系统时钟

表 18-3-3 寄存器 PWMCKD

DEH	7	6	5	4	3	2	1	0
PWMCKD	PWMCKD[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
备注：PWMCKD 为索引寄存器，INDEX=0/1/2 分别将配置 PWM0/1/2。								
位编号	位符号	说明						
7~0	PWMCKD	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

DFH	7	6	5	4	3	2	1	0
PWMDIVL	PWMDIV[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
D1H	7	6	5	4	3	2	1	0
PWMDIVH	PWMDIV[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
备注：PWMDIV 为索引寄存器，INDEX=0/1/2 分别将配置 PWM0/1/2。								
位编号	位符号	说明						
15~0	PWMDIV	PWM 周期配置寄存器						

表 18-3-5 寄存器 PWMDUTL、PWMDUTH

D2H	7	6	5	4	3	2	1	0
PWMDUTL	PWMDUT[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

D3H	7	6	5	4	3	2	1	0
PWMDUTH	PWMDUT[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
备注：PWMDUT 为索引寄存器，INDEX=0/1/2 分别将配置 PWM0/1/2。								
位编号	位符号	说明						
15~0	PWMDUT	PWMn 占空比配置寄存器						

表 18-3-6 寄存器 PWMIF

D4H	7	6	5	4	3	2	1	0
PWMIF	-	-	-	-	-	PWMIF2	PWMIF1	PWMIF0
R/W	-	-	-	-	-	R	R	R
初始值	-	-	-	-	-	0	0	0
位编号	位符号	说明						
7~3	-	-						
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 PWMCMAX

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

表 18-3-8 寄存器 PWMHS

8099H	7	6	5	4	3	2	1	0
PWMHS	-	-	-	-	-	PWMHS2	PWMHS1	PWMHS0
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	PWMHS2	PWM2 保持使能位，1 有效
1	PWMHS1	PWM1 保持使能位，1 有效
0	PWMHS0	PWM0 保持使能位，1 有效

表 18-3-9 寄存器 PWMSBC

809CH	7	6	5	4	3	2	1	0
PWMSBC	-	-	-	-	-	PWMSBE2	PWMSBE1	PWMSBE0
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	PWMSBE2	PWM2 软件刹车使能，1 刹车
1	PWMSBE1	PWM1 软件刹车使能，1 刹车
0	PWMSBE0	PWM0 软件刹车使能，1 刹车

表 18-3-10 寄存器 PWMBD

809DH	7	6	5	4	3	2	1	0
PWMBD	-	-	-	-	-	PWMBD2	PWMBD1	PWMBD0
R/W	-	-	-	-	-	R/W	R/W	R/W
初始值	-	-	-	-	-	0	1	0

备注：

软件和硬件刹车，电平都受本寄存器控制。

位编号	位符号	说明
7~3	-	-
2	PWMBD2	PWM2 刹车电平选择 0：刹车时保持低电平 1：刹车时保持高电平
1	PWMBD1	PWM1 刹车电平选择 0：刹车时保持低电平 1：刹车时保持高电平
0	PWMFBD0	PWM0 刹车电平选择 0：刹车时保持低电平 1：刹车时保持高电平

18.4 PWM 功能控制例程

◆ **PWM 单路输出例程**

例如，PWM0 输出频率为 30KHZ、占空比 50%的 PWM 信号，并产生 PWM 中断，程序如下：

```
---------------------------------------------------------------------------------------
#define PIE(N)          (N<<7)
#define TOG(N)          (N<<6)
#define CKS_IH          (1<<0)
#define IHCKE           (1<<6)
#define PWMIF0          (1<<0)
#define PWM_CH0         0
#define PWMDIV_V        (16000000/30000)  //当 PWM 时钟为其他时钟频率时，需相应修改参数
#define PWMDUT_V        (PWMDIV_V/2)      //占空比为 50%
void PWM_init(void)
{
   CKCON |= IHCKE;                        //打开 IRCH 时钟
   INDEX = PWM_CH0;                       //设置 INDEX 值对应 PWM0
   PWMCON = PIE(1)|TOG(0)|CKS_IH;         //设置 PWM 时钟源为 IRCH
   PWMCFG = TOG(0) | 0;
   P54F = 4;
   PWMDIVH    = (unsigned char)(PWMDIV_V>>8);        //设置 PWMDIV、PWMDUT
   PWMDIVL    = (unsigned char)(PWMDIV_V);
   PWMDUTH    =(unsigned char)(PWMDUT_V>>8);
   PWMDUTL    = (unsigned char)(PWMDUT_V);
   PWMEN = (1<<PWM_CH0);                                  //PWM0 使能
   PWMCMX = 0;
   INT9EN = 1;
}
void INT9_ISR(void) interrupt 14
{
   if(PWMIF & PWMIF0)
   {
         PWMIF = PWMIF0;

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

◆ **PWM 输出时钟例程**

例如，PWM0 输出 IRCH，程序如下：

```
---------------------------------------------------------------------------------------
void PWM_init(void)
{
      CKCON |= IHCKE;                        //打开 IRCH 时钟
```

```
    INDEX = PWM_CH0;                    //设置 INDEX 值对应 PWM0
    PWMCON = PIE(0)|TOG(0)|CKS_IH;      //设置 PWM 时钟源为 IRCH
    PWMCFG = TOG(0) | 0;
    P54F = 4;
    PWMDIVH     = 0;                    //PWMDIV、PWMDUT 都设置为 0 可直接输出时钟
    PWMDIVL     = 0;
    PWMDUTH     = 0;
    PWMDUTL     = 0;
    PWMEN = (1<<PWM_CH0);               //PWM0 使能
}
-----------------------------------------------------------------------------------------
```

19 模/数字转换器（ADC）

19.1 功能简介

模拟/数字转换器是 12 位逐次逼近寄存器型（SAR）ADC，最多提供多达 8 个输入通道。ADC 时钟源是系统时钟，可设置时钟预分频。ADC 有多种参考电压源可选，其中选择内部电压为参考电压时可用于检测芯片供电电压。ADC 选择内部电压为参考电压时有自动校正功能，避免芯片一致性问题。ADC 和运放结合一起使用，可以把检测信号缩小后再进行转换。

19.2 主要特性

- 12 位的分辨率
- 最多提供多达 8 个输入通道
- 支持 ADC 中断
- 可设置 ADC 时钟预分频
- 多种参考电压可选：内部参考电压、VDD、外部参考电压。
- 支持 VDD 和参考地电压的测量
- 选择内部参考电压时，支持自动数据校正功能
- 内置运放，支持检测信号缩小，缩小倍数可选
- 输入电压范围：VSS<=VIN<=VDD。

19.3 结构框图

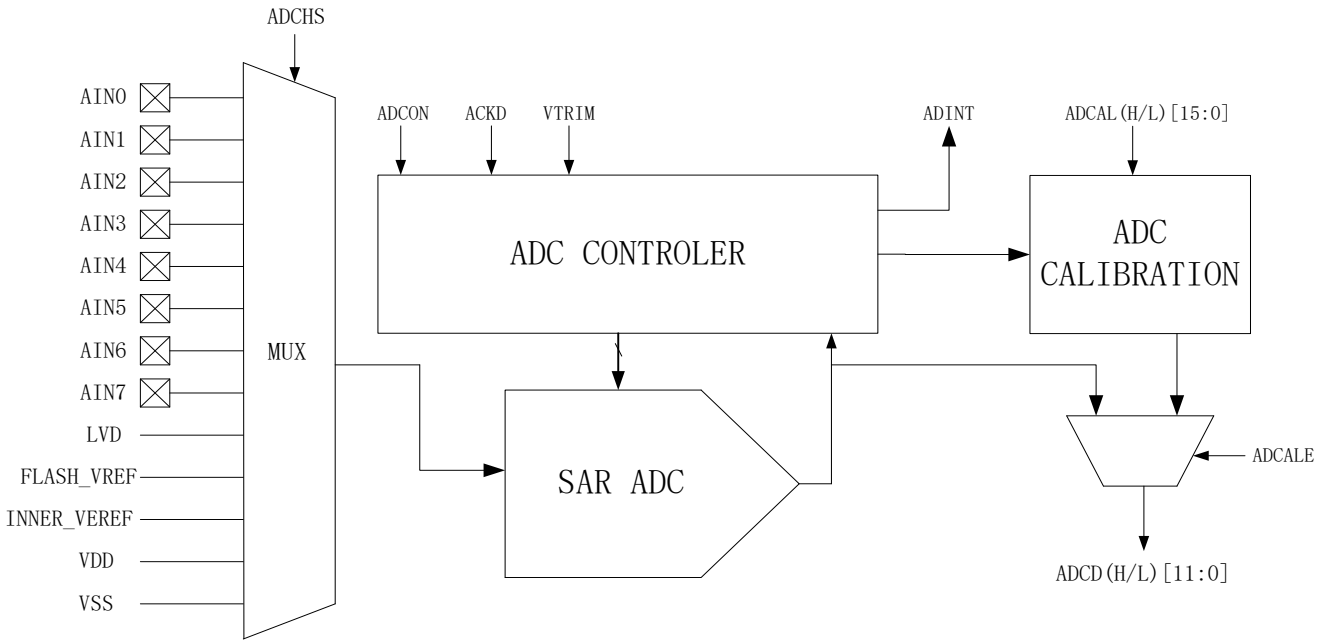


图 19-3-1 ADC 结构示意图

19.4 功能描述

ADC 的启动通过 AST 位使能，设置 AST=1 后，ADC 模块对 ADCHS 选择的输入电压源进行模/数转换。ADC 可通过 ACKD 设置时钟预分频，由系统时钟预分频后的时钟作为 ADC 转换时钟。在 ADC 时钟不变的条件下，ADC 的单次转换时间是由 HTME 设置的，转换时间为(13+2^HTME)个 ADC 时钟周期。当转换结束后，12 位的 A/D 值会被加载到寄存器 ADCDH、ADCDL，转换完后的 2.5 个时钟周期，AST 位自动清 0，同时中断标志 ADIF 位会置 1，如果 ADC 中断使能，会产生 ADC 中断。图 19-4-1 为 ADC 的转换时序图。

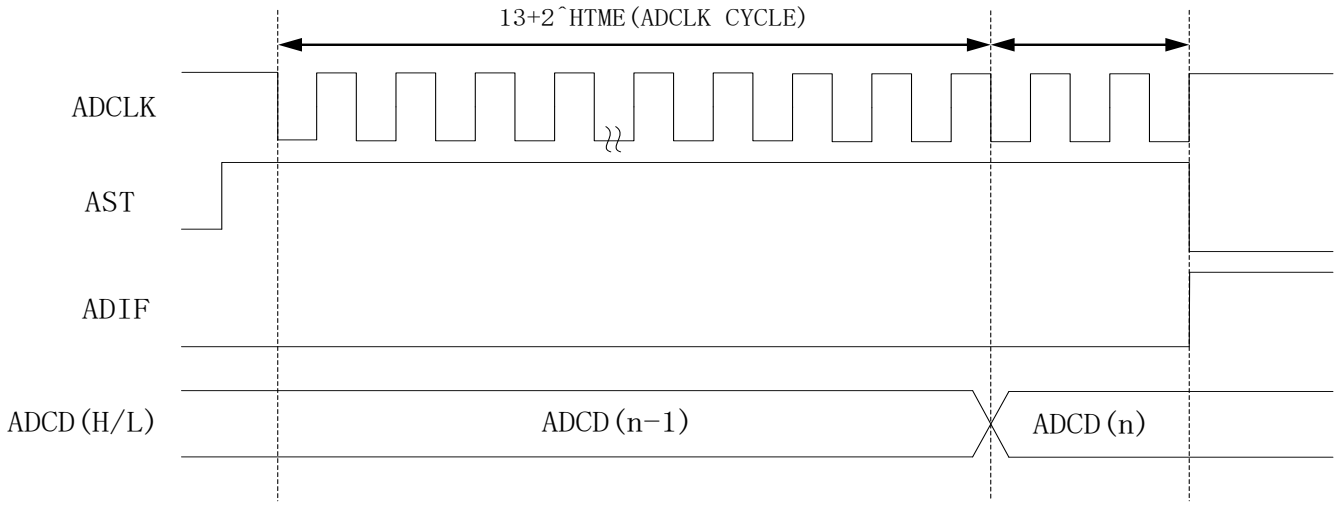


图 19-4-1 ADC 时序示意图

- **ADC 数据校正**

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

- **运放功能**

运放功能支持对 ADC 检测信号进行缩小，可有效扩展检测信号范围。缩小倍数由 AOPS 设置，详细请参考寄存器部分描述。

19.5 寄存器描述

表 19-5-1 寄存器 ADCON

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

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

表 19-5-2 寄存器 ADCFGL

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

位编号	位符号	说明
7~5	ACKD	ADC 时钟分频设置 000：不分频 001：2 分频 010：4 分频 … 111：14 分频
4	ADCALE	ADC 校准使能位，1 有效 此位只有选择参考电压为内部 1.5V 时才有效，当 ADCALE=1，ADC 的转换结果将根据 ADCAL 寄存器的数值进行校准。具体参考寄存器 ADCAL 说明。
3~0	ADCHS	ADC 通道使能选择位域 0000：通道关闭 0001：通道 AD_CH[0](P27)使能

		0010：通道 AD_CH[1](P31)使能 0011：通道 AD_CH[2](P32)使能 0100：通道 AD_CH[3](P33)使能 0101：通道 AD_CH[4](P34)使能 0110：通道 AD_CH[5](P35)使能 0111：通道 AD_CH[6](P36)使能 1000：通道 AD_CH[7](P37)使能 1001：检测 VDD 的 1/4 使能 1010：检测 VSS 使能 其他：通道关闭

表 19-5-3 寄存器 ADCFGH

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

位编号	位符号	说明
7~6	AOPS	运放放大倍数选择位域 00：无缩放 01：1/4 倍 10：1/3 倍 11：1/2 倍
5~0	VTRIM	内部 1.5V 参考电压校正寄存器，校正精度±1mV

表 19-5-4 寄存器 ADCAL

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

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

表 19-5-5 寄存器 ADCD

BCH	7	6	5	4	3	2	1	0
ADCDL	ADCDL[3:0]				-			
R/W	R				-			
初始值	0	0	0	0	-	-	-	-
BDH	7	6	5	4	3	2	1	0
ADCDH	ADCDH[11:4]							
R/W	R							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
11~0	ADCD	ADC 转换值						

表 19-5-6 寄存器 ADOPC

808FH	7	6	5	4	3	2	1	0
ADOPC	-	-	-	-	-	-	GAIN[1:0]	
R/W	-	-	-	-	-	-	R/W	
初始值	-	-	-	-	-	-	0	0
位编号	位符号	说明						
7~2	-	-						
1~0	GAIN	运放缩放倍数选择位 00:无缩放 01:缩小 4 倍 10:缩小 3 倍 11:缩小 2 倍						

19.6 ADC 控制例程

例如，设置 ADC 参考电压为外部 VDD，采集 ADC 通道 0，ADC 中断开启，程序如下：

```
-----------------------------------------------------------------------------------------------
//ADCON 定义
#define AST(N)      (N<<7)      //ADC 启动 ，  AST=0 结束
#define ADIE(N)     (N<<6)      //中断使能
#define ADIF        (1<<5)      //中断标志
#define HTME(N)     (N<<2)      //N=0-7          //采样时间设置，时间为 2 的 HTME 次方的时钟周期
#define VSEL(N)     (N)         //N=0-3          //选择参考电压 0-内部 1-VDD 2-外部
//ADCFGL 定义
```

```
#define ACKD(N)        (N<<5)   //N=0-7            //ADC 时钟分频     分频倍数=（ACKD+1)
#define ADCALE(N)      (N<<4)   //ADC 校正，选择内部参考电压才有效
#define ADCHS(N)       (N)      //N=0-15           //ADC 通道选择 ，  1-13 对应通道 0-12
void ADC_init(void)
{
    P27F = 3;                                          //设置 P27 为 ADC 引脚功能
    ADCON = AST(0) | ADIE(1) | HTME(7) | VSEL(1);      //设置 ADC 参考电压为 VDD
    ADCFGL = ACKD(7) | ADCALE(1) | ADCHS(1);           //选择 ADC0 通道
    ADCON |= AST(1);                                   //启动 AD 转换
    INT2EN = 1;                                        //INT2 中断使能
}
void ADC_ISR (void) interrupt 7
{
    unsigned int   AD_Value;
    if(ADCON & ADIF)
    {
        ADCON |= ADIF;                         //清中断标志
        AD_Value = ADCDH*256 + ADCDL;          //读取 AD 值
        AD_Value >>= 4;
        ADCON |= AST(1);                       //启动下一次 AD 转换
    }
}
-----------------------------------------------------------------------------------------------
```

20 触摸按键（TOUCH KEY）

20.1 功能简介

CA51F4 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 16 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 16 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

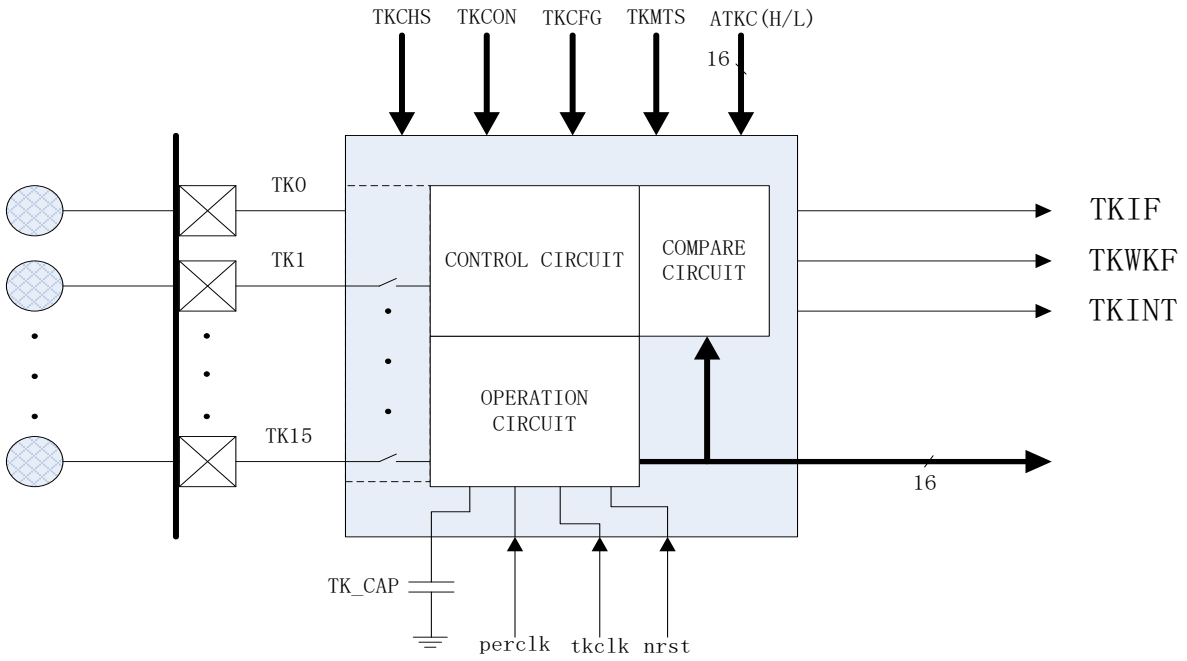


图 20-3-1 触摸模块结构图

20.4 功能描述

20.4.1 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，通过索引寄存器 INDEX 及寄存器 TKCHS 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，此时可通过设置索引寄存器 INDEX 后从寄存器 TKMS 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，定时器的时钟源可以是 IRCL 或 XOSCL，由寄存器 CKSEL 的 RTCKS 位选择；定时器定时时间由寄存器 TKMTS 设置。

20.4.2 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 的四分频，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。

20.4.3 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 TFRC 和低速时钟（IRCL 或 XOSCL）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一组通道的触发阈值，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。

20.4.4 触摸按键共用 LED 驱动功能描述

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=0~19,对应 TK0~TK19)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=0~19，对应 LED0~LED19)可独立控制每路 LED 灯亮灭。COM 引脚为 P06(P06F 设置为 6)。

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。*

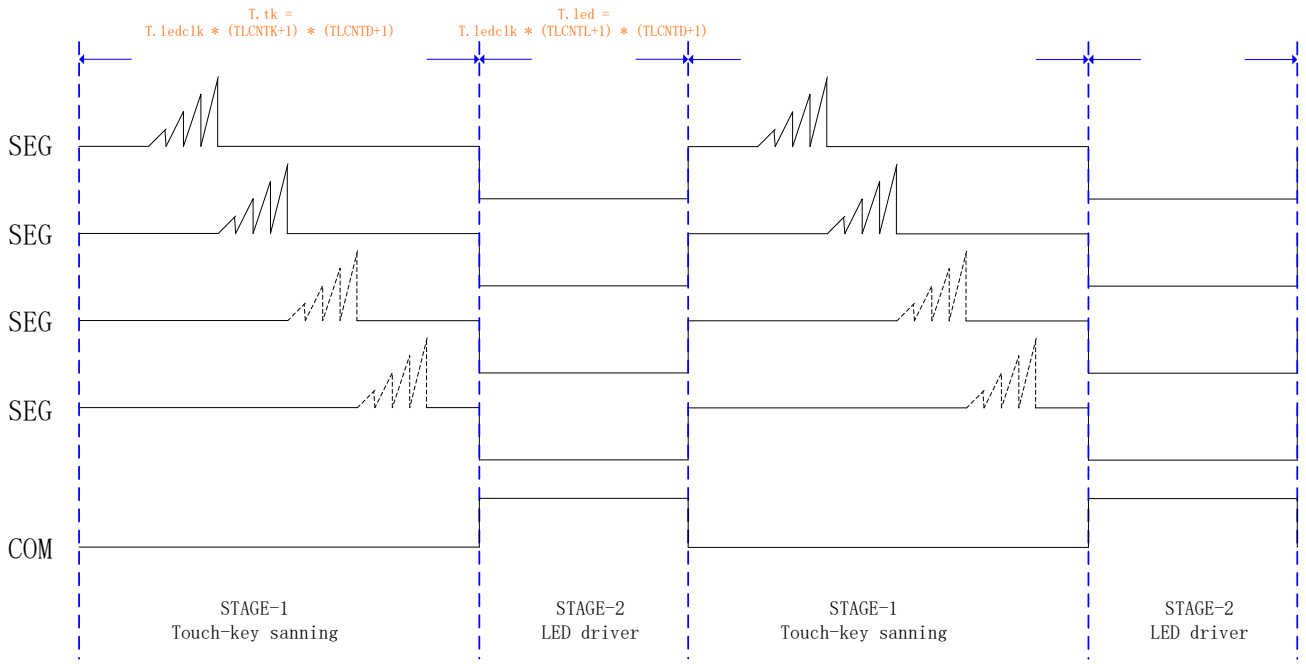


图 20-5-1 触摸共用 LED 驱动示意图

20.4.5 触摸内部基准和内部运放

触摸模块在内部集成了运放，可通过 TKPWS(TKPWC[1])选择内部运放作为触摸键的充电电源，充电电压通过 VDS(TKPWC[5:4])来选择。另外，触摸也可通过 TKCVS(TKPWC[0])选择内部基准作为触摸内部比较器的阈值电压，内部基准电压通过 VIRS(TKPWC[3:2])来选择。

20.4.6 触摸防水补偿机制

触摸设计了防水补偿机制，可通过 TKPC(TKPWC[7:6])为 2 开启。开启此功能后，非选中的触摸引脚会同步输出与充电频率相同的补偿波形，可有效降低触摸按键之间寄生电容的影响，实现防水的效果。注意：在开启防水补偿时，触摸充电电源最好选择为外部电源。

20.5 寄存器描述

表 20-5-1 寄存器 TKCON

C1H	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 TKCFG

C2H	7	6	5	4	3	2	1	0
TKCFG	TKDIV			TKTMS				
R/W	R/W			R/W				
初始值	0	0	0	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 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 20-5-3 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：不输出低也不输出补偿 01：输出低 其他：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 00：1.2V 01：1.6V 10：2.0V 11：2.4V
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 20-5-4 寄存器 TKMTS

C3H	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 32 x 低速时钟周期，如果低速时钟频率为 32.768K，时间范围是 0.977ms - 250ms。

表 20-5-5 寄存器 TKCHS

C4H	7	6	5	4	3	2	1	0
TKCHS	POL	NPOL	-	TKPS				
R/W	R/W	R/W	-	R/W				
初始值	0	0	-	0	0	0	0	0

备注：TKCHS 是带索引的寄存器，设置 INDEX=0~5 分别对应 TKCHS0~TKCH5

位编号	位符号	说明
7	POL	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	NPOL	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
4~0	TKPS	通道选择位域 00000：TK0~TK23 关闭 00001：选择 TK0 00010：选择 TK1 00011：选择 TK2 …… 10100：选择 TK19 10101：选择内部参考电容

表 20-5-6 寄存器 ATKS

C5H	7	6	5	4	3	2	1	0
ATKSL	ATKSL[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
C6H	7	6	5	4	3	2	1	0
ATKSH	ATKSH[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

备注：ATKC 是带索引的寄存器，设置 INDEX=0~5 分别对应 ATKC0~ATKC5

位编号	位符号	说明
15~0	ATKC	比较阈值设置寄存器，当 TWKE=1 时，ATKC0~ATKC5 自动与 TKMS0~TKMS5 比较

表 20-5-7 寄存器 ATKN

8092H	7	6	5	4	3	2	1	0

ATKNL	ATKN[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
8093H	7	6	5	4	3	2	1	0
ATKNH	ATKN[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
备注：ATKC 是带索引的寄存器，设置 INDEX=0~5 分别对应 ATKC0~ATKC5								
位编号	位符号	说明						
15~0	ATKN	比较阈值设置寄存器，当 TWKE=1 时，ATK0N~ATK5N 自动与 TK0MS~TK5MS 比较						

表 20-5-8 寄存器 TKMS

CEH	7	6	5	4	3	2	1	0
TKMSL	TKMS[7:0]							
R/W	R							
初始值	0	0	0	0	0	0	0	0
CFH	7	6	5	4	3	2	1	0
TKMSH	TKMS[15:8]							
R/W	R							
初始值	0	0	0	0	0	0	0	0
备注：TKMS 是带索引的寄存器，设置 TKMS=0~5 分别对应 TKMS0~TKMS5								
位编号	位符号	说明						
15~0	TKMS	触摸采样数据寄存器						

表 20-5-10 寄存器 TKIF

C7H	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	-	-						
5~0	TKIFx(x=5~0)	触摸采集中断标志位，按顺序分别对应 6 个选定通道，当 TWKE=1 时，TKIFx 表示触摸数据超出阈值范围						

表 20-5-11 寄存器 TKMAXF

8090H	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	-	-
5~0	TKMXFx(x=5~0)	1 表示 TKxMS 超出 ATKxC 的阈值范围，0 表示 TKxMS 不超出 ATKxC 的阈值，极性由 POLx 决定；如果 TWKE=1，那么置位 TKMXFx 的同时会置位 TKIFx；软件无法对其操作

表 20-5-12 寄存器 TKMINF

8091H	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	-	-
5~0	TKMNFx(x=5~0)	1 表示 TKxMS 超出 ATKxN 的阈值范围，0 表示 TKxMS 不超出 ATKxN 的阈值，极性由 NPOLx 决定；如果 TWKE=1，那么置位 TKMNFx 的同时会置位 TKIFx；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
TLEN (INDEX=0)	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
TLEN (INDEX=1)	TLEN15	TLEN14	TLEN13	TLEN12	TLEN11	TLEN10	TLEN9	TLEN8
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

备注：

1. LEDx 要使能，除了 TLENx=1 (x=0,1,2,...,15)，还必须对应的 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
7~0 (INDEX=1)	TLEN15~TLEN8	LED15~LED8 使能，1 有效
7~0 (INDEX=0)	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
TLDAT (INDEX=0)	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
TLDAT (INDEX=1)	TLDAT15	TLDAT14	TLDAT13	TLDAT12	TLDAT11	TLDAT10	TLDAT9	TLDAT8
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 (INDEX=1)	TLDAT15~TLDAT8	LED15~LED8 数据，1 表示驱动有效电平
7~0 (INDEX=0)	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：IRCH 010：IRCL 011：XOSCL 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) ~~x N~~ ***备注：*** *1. T.led，表示 LED 驱动阶段的时间。*

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）

21 低电压检测（LVD）

21.1 功能简介

低电压检测（LVD）用于监控芯片自身的供电 VDD，共有四档电压可选：2.0V、2.7V、3.7V、4.4V。当 VDD 小于所设定的电压值时，可设置触发中断或复位。

备注：*由于生产工艺的影响，芯片之间 LVD 触发电压存在一定的差异。*

LVD 结构图如图 21-1-1 所示。

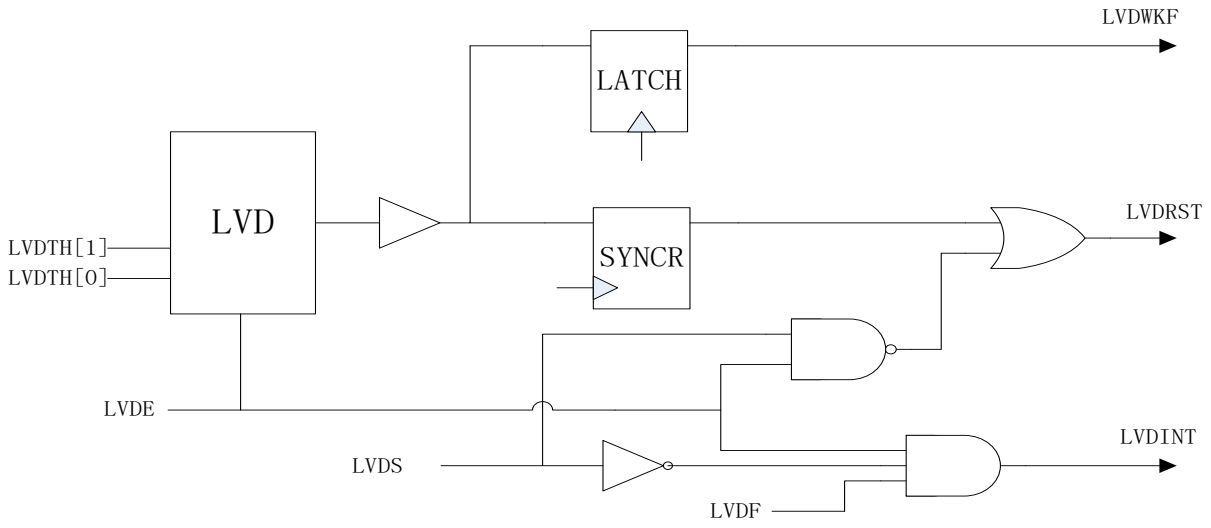


图 22-1-1 LVD 模块示意图

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

21.3 寄存器描述

表 21-3-1 寄存器 LVDCON

EFH	7	6	5	4	3	2	1	0
LVDCON	LVDE	LVDS	LVDF	-	-	-	LVDTH[1:0]	
R/W	R/W	R/W	R/W	-	-	-	R/W	
初始值	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
4~2	-	-
1~0	LVDTH	LVD 触发电平选择位域 00：2.0V 01：2.7V 10：3.7V 11：4.4V

21.4 LVD 控制例程

LVD 中断例程

例如，设置 LVD 为中断模式，检测电压为 3.7 V，程序如下：

```
-----------------------------------------------------------------------------------------
#define LVDE(N)          (N<<7)    //N=0~1
#define LVDS_int         (0<<6)
#define LVDF             (1<<5)
#define LVDTH_3p7V       2
void LVD_init(void)
{
    LVDCON = LVDE(1) | LVDS_int |   LVDTH_3p7V;//设置 LVD 使能，LVD 为中断模式，检测电压为 3.7 V
    INT4EN = 1;   //INT4 中断使能
    EA = 1;       //开启总中断
}
void INT4_ISR (void) interrupt 9    //LVD 中断服务程序
{
    if(LVDCON & LVDF)
    {
        LVDCON |= LVDF;       //清除 LVD 中断标志

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

LVD 复位例程

例如，设置 LVD 为复位模式，检测电压为 3.7 V，程序如下：

```
-----------------------------------------------------------------------------------------
#define LVDE(N)          (N<<7)    //N=0~1
#define LVDS_reset       (1<<6)
#define LVDTH_3p7V       2
void LVD_init(void)
{
    LVDCON = LVDE(1) | LVDS_reset| LVDTH_3p7V;//设置 LVD 使能，LVD 为复位模式，检测电压为 3.7 V
}
-----------------------------------------------------------------------------------------
```

22 LCD 驱动

22.1 LCD 驱动

22.1.1 功能简介

内置 LCD 驱动最大可支持 5com x 31seg、4com x 32seg，共 36 个输出引脚。可编程占空比为：1/2、1/3、1/4、1/5。LCD 驱动有三种模式：内建电压电荷泵模式、电荷泵分压模式和电阻分压模式。对于内建电压电荷泵模式和电荷泵分压模式，偏压比固定为 1/3；对于电阻分压模式，可编程偏压比为：1/2、1/3、1/4。

图 22-1-1-1 是 LCD 的原理示意图。

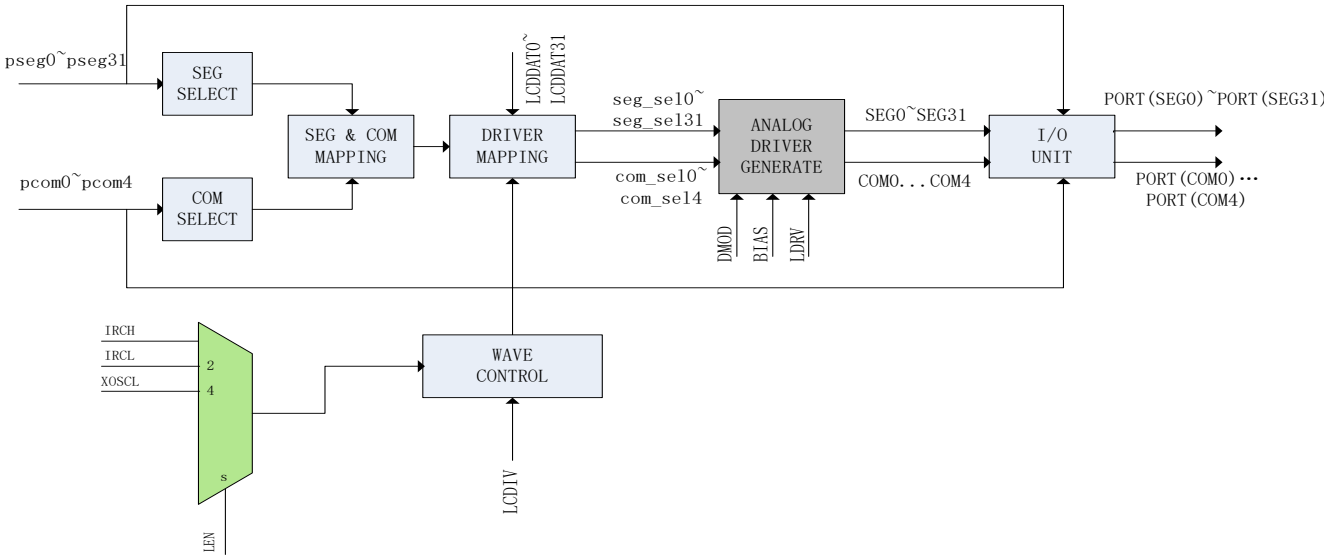


图 22-1-1-1 LCD 原理示意图

22.1.2 LCD 偏压

LCD 可编程偏压比为：1/2、1/3、1/4，以下分别是其对应的信号图。

- **LCD 偏压比 1/2**

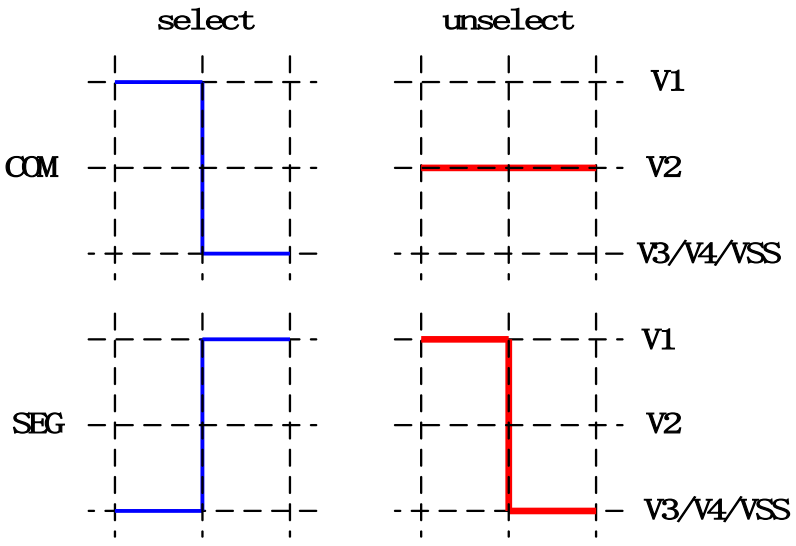


图 22-1-1-2　LCD 偏压比 1/2 信号

- LCD 偏压比 1/3

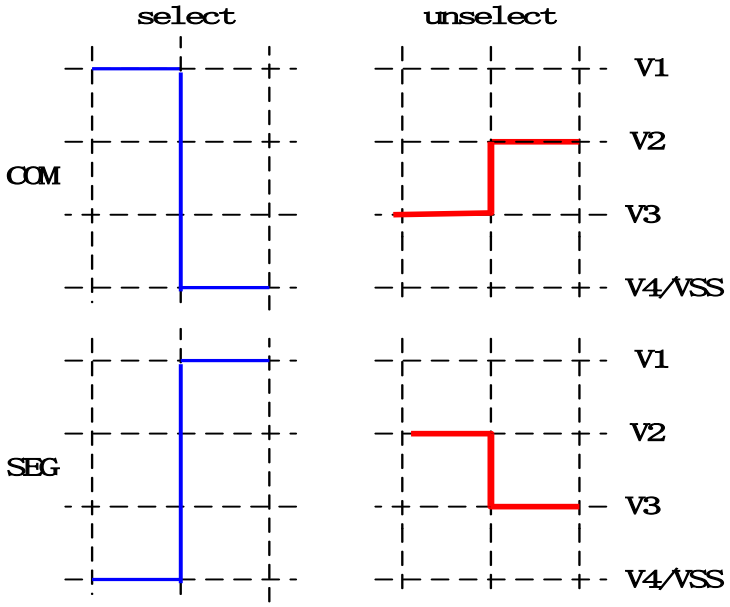


图 22-1-1-3　LCD 偏压比 1/3 信号

- LCD 偏压比 1/4

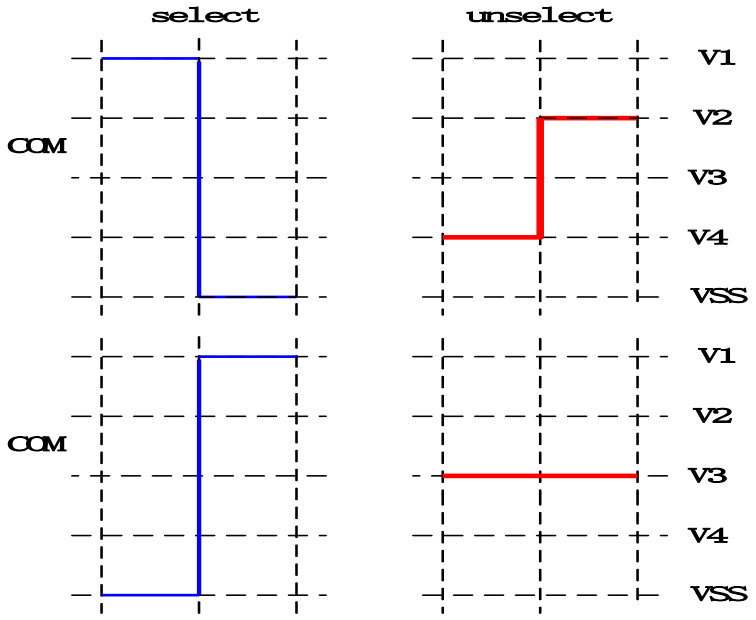


图 22-1-1-4　LCD 偏压比 1/4 信号

22.1.3 LCD 功能描述

LCD 模式通过设置 LMOD=1 来选择。LCD 驱动可以通过 LEN 选择时钟源，当时钟源被选择后，LCD 驱动同时被使能；要注意的是，在设置时钟源时必须要确定该时钟源是被打开并且正常工作。寄存器 LXDIV 是 LCD 时钟分频器，针对不同的时钟源可设置不同的分频系数，LCD 扫描的帧频率典型值为 64Hz。

LCD 可编程的占空比为：1/2、1/3、1/4、1/5，占空比是由使能的 COM 数量决定的，例如使能了 3 个 COM，那占空比就为 1/3，使能了 5 个 COM，占空比就为 1/5。而 COM 的使能不需要按 COM 引脚编号的顺序进行，可以任意的组合，但使能的 COM 引脚按序号从小到大对应实际的 COM0、COM1、COM2...，例如使能了引脚 COM1、COM3、COM5 为 COM 口，那 COM1 对应实际 COM0,COM3 对应实际 COM1，COM5 对应实际 COM2，占空比为 1/3。SEG 引脚也可任意使能，使能的 SEG 引脚按序号从小到到大对应实际的 SEG0、SEG1、SEG2...，例如使能引脚 SEG3、SEG5、SEG7...,那 SEG3 对应实际 SEG0,SEG5 对应实际 SEG1，SEG7 对应实际 SEG2...。没有被使能的 COM 和 SEG 引脚可以设置为其他功能使用，和 LCD 驱动没有冲突。

LCD 驱动有 32 字节的 LCD 显示缓存。LCD 显示缓存是与实际的 COM,SEG 对应的，32 字节显示缓存按顺序分别对应 SEG0~SEG31，而 COM0~COM4 对应每个字节的 0~4 位。显示缓存是通过索引寄存器 INDEX 和数据寄存器 LCDAT 来访问的，设置 INDEX 为 0~31 按顺序对应 SEG0~SEG31 的显示缓存。

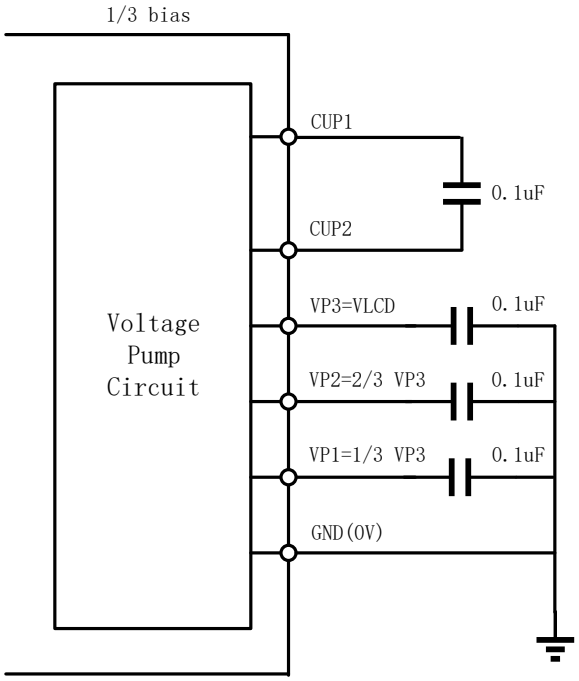
- **电阻分压模式**

电阻分压模式下，LCD 驱动有 8 级驱动强度可调，通过 LDRV 来设置，不同的驱动强度输出电压不同，在应用时可应针对不同的 LCD 显示器调整此数值。为满足不同的功耗要求，LCD 驱动有 4 级驱动电流选择，通过 DMOD 来设置，当驱动电流越小时，驱动本身功耗也越小，但 LCD 引脚上出现的杂波也越大。

- **电荷泵分压模式**

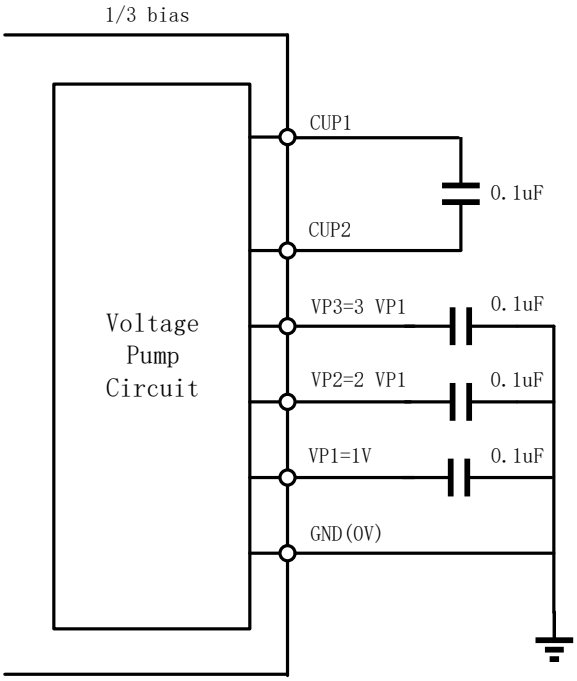
电荷泵分压模式下，LCD 驱动电压为 VDD。驱动强度和偏压比不可调，偏压比恒定为 1/3。管脚 P00、P01、P02、P03 和 P04 需设置为 LCD 对应的 VP3、VP2、VP1、CUP1 和 CUP2，接上 0.1uF 的电容，连接方式如下图。使用电容偏压模式时，需先设置 LEN 选择好时钟,TYPE 设置为电容偏压型，置 PMPE 为 1 使能电荷泵，

等待至少 25ms 后再打开 LCD，即 LCE 位置 1，点亮 LCD 面板。



- **内建电压电荷泵模式**

内建电压电荷泵模式下，当 VDD 介于 1.8V 及 5V 之间，LCD 驱动电压可恒定输出，输出电压由 LDRV 设置，范围为 2.4V~4.5V。在此模式下，偏压比恒定为 1/3，驱动强度不可设置。管脚 P00、P01、P02、P03 和 P04 需设置为 LCD 对应的 VP3、VP2、VP1、CUP1 和 CUP2，接上 0.1uF 的电容，连接方式如下图。使用内建电压电荷泵模式时，需先设置 LEN 选择好时钟,TYPE 设置为内建电压电荷泵模式，置 PMPE 为 1 使能电荷泵，等待至少 25ms 后再打开 LCD，即 LCE 位置 1，点亮 LCD 面板。



备注：由于SOP28 封装没有引出VP3、VP2、VP1、CUP1 和CUP2 引脚，所以只能应用电阻分压模式。

22.2 LCD 寄存器描述

表 22-2-1 寄存器 LCCON

E1H	7	6	5	4	3	2	1	0
LCCON	LEN[2:0]			-	-	TYPE[1:0]		LCE
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
7~5	LEN	LCD 时钟选择位 001：IRCL 010：IRCH 011：XOSCL 其他:模块关闭
4~3	-	-
2~1	TYPE	模式选择位 00：电阻分压模式 01：电荷泵分压模式 10：内建电压电荷泵模式 11：电阻分压模式
0	LCE	LCD 模块使能，1 使能

表 22-2-2 寄存器 LCCFG

E2H	7	6	5	4	3	2	1	0
LCCFG	DMOD[1:0]		BIAS[1:0]		-	LDRV[2:0]		
R/W	R/W		R/W		-	R/W		
初始值	0	0	0	0	-	0	0	0

位编号	位符号	说明
7~6	DMOD	LCD 电阻型的驱动电流大小选择位 00：5uA 01：40uA 10：80uA 11：130uA *备注：此项设置仅在电阻分压模式下有效。*
5~4	BIAS	LCD 电阻型的偏压选择位 01：1/2 Bias

		10：1/3 Bias 其他：1/4 Bias *备注：此项设置仅在电阻分压模式下有效。*
3	-	-
2~0	LDRV	**电阻分压模式：** LCD 驱动强度控制位 000：Level 1（最小） 001：Level 2 … 111：Level 8（最大） **内建电压电荷泵模式：** LCD 输出电压控制位 000：2.4V 001：2.7V 010：3.0V 011：3.3V 100：3.6V 101：3.9V 110：4.2V 111：4.5V **电荷泵分压模式：** 此项设置无效。

表 22-2-3 寄存器 LCDIV

E4H	7	6	5	4	3	2	1	0
LCDIVL	LCDIV[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
E5H	7	6	5	4	3	2	1	0
LCDIVH	-	-	-	-	LCDIV[11:8]			
R/W	-	-	-	-	R/W			
初始值	-	-	-	-	0	0	0	0

位编号	位符号	说明
15~12	-	-
11~0	LCDIV	LCD 时钟分频器 LCD 扫描帧频率=LCD 时钟频率 ÷((LCDIV+1) x 512) *备注：* *2. 当 LCD 时钟选择为 IRCL 时，时钟频率为 IRCL 的 1/4。*

表 22-2-4 寄存器 LCDAT

E3H	7	6	5	4	3	2	1	0
LCDAT	LCDAT[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
备注：LCDAT 是带索引的寄存器，设置 INDEX=0~31 分别对应 LCDAT0~LCDAT31								
位编号	位符号		说明					
7~0	LCDAT		显示缓存读写寄存器					

表 2-5-5 LCD 显示缓存

INDEX	SEG	COM0	COM1	COM2	COM3	COM4			
0	0	BIT0	BIT1	BIT2	BIT3	BIT4			
1	1	BIT0	BIT1	BIT2	BIT3	BIT4			
2	2	BIT0	BIT1	BIT2	BIT3	BIT4			
3	3	BIT0	BIT1	BIT2	BIT3	BIT4			
4	4	BIT0	BIT1	BIT2	BIT3	BIT4			
5	5	BIT0	BIT1	BIT2	BIT3	BIT4			
6	6	BIT0	BIT1	BIT2	BIT3	BIT4			
7	7	BIT0	BIT1	BIT2	BIT3	BIT4			
8	8	BIT0	BIT1	BIT2	BIT3	BIT4			
9	9	BIT0	BIT1	BIT2	BIT3	BIT4			
10	10	BIT0	BIT1	BIT2	BIT3	BIT4			
11	11	BIT0	BIT1	BIT2	BIT3	BIT4			
12	12	BIT0	BIT1	BIT2	BIT3	BIT4			
13	13	BIT0	BIT1	BIT2	BIT3	BIT4			
14	14	BIT0	BIT1	BIT2	BIT3	BIT4			
15	15	BIT0	BIT1	BIT2	BIT3	BIT4			
16	16	BIT0	BIT1	BIT2	BIT3	BIT4			
17	17	BIT0	BIT1	BIT2	BIT3	BIT4			
18	18	BIT0	BIT1	BIT2	BIT3	BIT4			
19	19	BIT0	BIT1	BIT2	BIT3	BIT4			
20	20	BIT0	BIT1	BIT2	BIT3	BIT4			
21	21	BIT0	BIT1	BIT2	BIT3	BIT4			
22	22	BIT0	BIT1	BIT2	BIT3	BIT4			
23	23	BIT0	BIT1	BIT2	BIT3	BIT4			
24	24	BIT0	BIT1	BIT2	BIT3	BIT4			
25	25	BIT0	BIT1	BIT2	BIT3	BIT4			
26	26	BIT0	BIT1	BIT2	BIT3	BIT4			
27	27	BIT0	BIT1	BIT2	BIT3	BIT4			

28	28	BIT0	BIT1	BIT2	BIT3	BIT4			
29	29	BIT0	BIT1	BIT2	BIT3	BIT4			
30	30	BIT0	BIT1	BIT2	BIT3	BIT4			
31	31	BIT0	BIT1	BIT2	BIT3	-			

表 22-2-6 寄存器 LCCAD

8117H	7	6	5	4	3	2	1	0
LCCAD	-	-	-	-	CAD_MOD[1:0]		CAD_LTH[1:0]	
R/W	-	-	-	-	R/W		R/W	
初始值	-	-	-	-	0	1	0	1

位编号	位符号	说明
7~4	-	-
3~2	CAD_MOD	LCD_CAD 模式选择 00：关闭 LCD_CAD 01：LCD_CAD 长度由数字信号长度决定 其他：LCD_CAD 长度由模拟信号控制
1~0	CAD_LTH	模拟信号控制 LCD_CAD 长度选择，仅在 CAD_MOD=2/3 时有效 00：4us 01：8us 10：12us 11：16us

表 22-2-7 寄存器 LCPMP

8116H	7	6	5	4	3	2	1	0
LCPMP	-	-	-	-	PCKD			PMPE
R/W	-	-	-	-	R/W			R/W
初始值	-	-	-	-	0	0	0	0

位编号	位符号	说明
7~4	-	-
3~1	PCKD	PUMP 三相时钟分频配置寄存器，PMPE=1 时才有效 000：1 分频 001：2 分频 010：4 分频 011：8 分频 100：16 分频 101：32 分频 110：64 分频 111：128 分频

		备注： *1. 三相时钟源为 LCD PUMP 的工作时钟，当 LCD 工作时钟选择 16MHz(IRCH)时，要作为三相时钟的源时钟之前会先分频为 500KHz。* *2. 最后出来的三相时钟的频率为分频之后的值再除以 3。*
0	PMPE	电容型时，打开 PUMP，1 有效

22.3 LCD 驱动控制例程

电阻分压模式

例如，电阻分压模式下 4com*10seg、1/3bias LCD 显示功能，程序如下：

```
-----------------------------------------------------------------------------------------
//LCCON 定义
#define LEN_IRCH          (1<<5)
#define LCD_TYPE(N)       (N<<1)
#define LCE(N)            (N<<0)
//LCCFG 定义
#define DMOD_130ua        (3<<6)
#define BIAS_1_3          (2<<4)
#define LCDRV_LEV(N)      (N) //N=0-7
//LCCAD 定义
#define CAD_MOD(N)        (N<<2)   //N=0-3
#define CAD_LTH(N)        (N<<0)   //N=0-3
//LCPMP 定义
#define PCKD(N)           (N<<1)   //N=0-7
#define PMPE(N)           (N<<0)   //N=0-1
void LCD_init(void)
{
    unsigned char i;
    //初始化 COM 引脚
    P51F = 6;
    P50F = 6;
    P47F = 6;
    P46F = 6;
    //初始化 SEG 引脚
    P34F = 6;
    P35F = 6;
    P36F = 6;
    P37F = 6;
    P40F = 6;
    P41F = 6;
    P42F = 6;
    P43F = 6;
    P44F = 6;
```

```
    P45F = 5;
    //初始化相关寄存器
    LCDIVH = 0x01;                                  //设置 LCD 时钟分频，目标帧频率为 64HZ
    LCDIVL = 0xdd;
    LCCFG =  DMOD_130ua | BIAS_1_3 | LCDRV_LEV(7);  //设置 LCD 驱动电流、偏压(bias)、驱动强度
    LCCON =   LEN_IRCH | LCD_TYPE(0) | LCE(1);      //设置 LCD 时钟源、选择 LCD 模式、使能 LCD
    //LCD RAM 清零
    for(i = 0; i < 10; i++)
    {
        INDEX = i;
        LCDAT = 0;
    }
}
```

电荷泵分压模式

例如，电荷泵分压模式下 4com*10seg、1/3bias LCD 显示功能，程序如下：

```
void LCD_init(void)
{
    unsigned char i;
    //初始化 COM 引脚
    P51F = 6;
    P50F = 6;
    P47F = 6;
    P46F = 6;
    //初始化 SEG 引脚
    P34F = 6;
    P35F = 6;
    P36F = 6;
    P37F = 6;
    P40F = 6;
    P41F = 6;
    P42F = 6;
    P43F = 6;
    P44F = 6;
    P45F = 5;
    //初始化相关电容引脚
    P00F = 6;
    P01F = 6;
    P02F = 6;
    P03F = 6;
    P04F = 6;
    //初始化相关寄存器
```

```
    LCDIVH = 0x01;                                  //设置 LCD 时钟分频，目标帧频率为 64HZ
    LCDIVL = 0xdd;
    LCPMP = PCKD(2) | PMPE(1);                      //设置三相时钟分频、使能
    LCCFG =  DMOD_130ua | BIAS_1_3 | LCDRV_LEV(7);  //设置 LCD 驱动电流、偏压(bias)、驱动强度
    LCCON =   LEN_IRCH | LCD_TYPE(1) | LCE(0);      //设置 LCD 时钟源、选择 LCD 模式、使能 LCD
    Delay_ms(30);
    LCCON |= LCE(1);
    //LCD RAM 清零
    for(i = 0; i < 10; i++)
    {
        INDEX = i;
        LCDAT = 0;
    }
}
```

内建电压电荷泵模式

例如，内建电压电荷泵模式下 4com*10seg、1/3bias LCD 显示功能，程序如下：

```
void LCD_init(void)
{
    unsigned char i;
    //初始化 COM 引脚
    P51F = 6;
    P50F = 6;
    P47F = 6;
    P46F = 6;
    //初始化 SEG 引脚
    P34F = 6;
    P35F = 6;
    P36F = 6;
    P37F = 6;
    P40F = 6;
    P41F = 6;
    P42F = 6;
    P43F = 6;
    P44F = 6;
    P45F = 5;
    //初始化相关电容引脚
    P00F = 6;
    P01F = 6;
    P02F = 6;
    P03F = 6;
    P04F = 6;
```

```
    //初始化相关寄存器
    LCDIVH = 0x01;                                  //设置 LCD 时钟分频，目标帧频率为 64HZ
    LCDIVL = 0xdd;
    LCPMP = PCKD(2) | PMPE(1);                      //设置三相时钟分频、使能
    LCCFG =  DMOD_130ua | BIAS_1_3 | LCDRV_LEV(7);  //设置 LCD 驱动电流、偏压(bias)、驱动强度
    LCCON =   LEN_IRCH | LCD_TYPE(2) | LCE(0);      //设置 LCD 时钟源、选择 LCD 模式、使能 LCD
    Delay_ms(30);
    LCCON |= LCE(1);
    //LCD RAM 清零
    for(i = 0; i < 10; i++)
    {
        INDEX = i;
        LCDAT = 0;
    }
}
-----------------------------------------------------------------------------------------
```

23 程序下载和仿真

23.1 程序下载

CA51F4 系列芯片主要采用 ISP 方式下载程序，有两种端口可以选择：
两线通信端口：
通过 I2C 端口进行下载，需要连接 4 根线：VDD、I2C_SCL(P05)、I2C_SDA(P06)、GND。
单线通信端口：
通过 SWIM 端口进行下载，需要连接 3 根线：VDD、SWIM(P07)、GND。

注意：对于 LQFP48 封装芯片而言，以上两种端口都可使用，而 SOP28 封装由于没有引出 SWIM 和 RESET 引脚，只能使用两线通信端口进行。另外，要注意的是，以上两种仿真端口分别对应不同的下载器。
更多关于程序下载步骤的细节请参考“CA51F4 系列开发下载工具使用说明”。

23.2 在线仿真

CA51F4 系列芯片支持在线仿真，与程序下载相似，同样有两种端口可以选择：
两线通信端口：
通过 I2C 端口进行在线仿真，需要连接 3 根线：I2C_SCL(P05)、I2C_SDA(P06)、GND。
单线通信端口：
通过 SWIM 端口进行在线仿真，需要连接 3 根线：RESET、SWIM(P07)、GND。要注意的是，进入仿真之前，芯片内的程序不能设置 RESET 引脚为 GPIO 功能，否则无法进入单线仿真模式。

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。

对于 LQFP48 封装芯片而言，以上两种端口都可使用，而 SOP28 封装由于没有引出 SWIM 和 RESET 引脚，只能使用两线通信端口进行在线仿真。另外，要注意的是，以上两种仿真端口分别对应不同的仿真器。更多关于仿真功能的细节可参考仿真器的相关文档介绍。

24 电气特性

24.1 极限参数

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

24.2 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		1.87		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		1.87			
		VDD=5V		1.88			
	Iop2	VDD=1.8V		19.7		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		20.0			
		VDD=5V		20.2			
	Iop3	VDD=1.8V		7.6		uA	系统时钟为 XOSCL(32.768kHZ)，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		7.8			
		VDD=5V		7.9			
	Iop4	VDD=1.8V		9.5		uA	系统时钟为 XOSCL(32.768kHZ)，其他时钟关闭，LDO 设置为低功率模式，输出电压为 1.61V，打开 LCD 驱动（不外接 LCD 面板），LCD 模式设置为内建电压电荷泵模式，LCD 输出电压为 3V，占空比为 1/4duty、LCD 时钟为 XOSCL，LCD_CAD 关闭（CAD_MOD=0），所有 LCD 引脚打开，其他所有输出引脚无负载，所有数字输入引脚不浮动，其他外设关闭
		VDD=3.3V		9.6			
		VDD=5V		9.8			

	Iop5	VDD=1.8V		12.0		uA	系统时钟为 XOSCL(32.768kHZ)，其他时钟关闭，LDO 设置为低功率模式，输出电压为 1.61V，打开 LCD 驱动（不外接 LCD 面板），LCD 模式设置为电阻分压模式，偏压比 1/3，占空比为 1/4duty、LCD 电流设为最小，驱动强度设为最高，LCD 时钟为 XOSCL，LCD_CAD 关闭（CAD_MOD=0），所有 LCD 引脚打开，其他所有输出引脚无负载，所有数字输入引脚不浮动，其他外设关闭
		VDD=3.3V		13.9			
		VDD=5V		16.1			
	Iop6	VDD=1.8V		24.0		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，打开 LCD 驱动（不外接 LCD 面板），LCD 模式设置为内建电压电荷泵模式，LCD 输出电压为 3V，占空比为 1/4duty、LCD 时钟为 IRCL，LCD_CAD 关闭（CAD_MOD=0），所有 LCD 引脚打开，其他所有输出引脚无负载，所有数字输入引脚不浮动，其他外设关闭
		VDD=3.3V		24.2			
		VDD=5V		24.4			
	Iop7	VDD=1.8V		23.1		uA	系统时钟为 IRCL(161kHZ)，其他时钟关闭，LDO 设置为低功率模式，输出电压为 1.61V，打开 LCD 驱动（不外接 LCD 面板），LCD 模式设置为电阻分压模式，偏压比为 1/3，占空比为 1/4duty、LCD 电流设为最小，LCD 时钟为 XOSCL，LCD_CAD 关闭（CAD_MOD=0），所有 LCD 引脚打开，其他所有输出引脚无负载，所有数字输入引脚不浮动，其他外设关闭
		VDD=3.3V		25.0			
		VDD=5V		27.3			
STOP 模式电流	I_{stp1}	VDD=1.8V		2.2		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		2.3			
		VDD=5V		2.4			
	I_{stp2}	VDD=1.8V		5.5		uA	除了 XOSCL(32.768kHZ)外，其他时钟关闭，LDO 设置为低功率模式，输出电压为 1.61V，打开 LCD 驱动（不外接 LCD 面板），LCD 模式设置为内建电压电荷泵模式，LCD 输出电压为 3V，占空比为 1/4duty、LCD 时钟为 XOSCL，LCD_CAD 关闭（CAD_MOD=0），所有 LCD 引脚打开，其他所有输出引脚无负载，所有数字输入引脚不浮动，其他外设关闭，Flash 进入睡眠模式，CPU 进入 STOP 模式。
		VDD=3.3V		5.6			
		VDD=5V		5.7			
	I_{stp3}	VDD=1.8V		7.8		uA	除了 XOSCL(32.768kHZ)外，其他时钟关闭，LDO 设置为低功率模式，输出电压为 1.61V，打开 LCD 驱动（不外接 LCD 面板），LCD 模式设置为电阻分压模式，偏压比为 1/3，占空比为 1/4duty、LCD

		VDD=3.3V		9.7			电流设为最小，LCD 时钟为 XOSCL，LCD_CAD 关闭（CAD_MOD=0），所有 LCD 引脚打开，其他所有输出引脚无负载，所有数字输入引脚不浮动，其他外设关闭，Flash 进入睡眠模式，CPU 进入 STOP 模式。
		VDD=5V		11.9			
	I_{stp4}	VDD=1.8V		5.3		uA	除了 IRCL(131kHZ)外，其他时钟关闭，LDO 设置为低功率模式，输出电压为 1.61V，打开 LCD 驱动（不外接 LCD 面板），LCD 模式设置为内建电压电荷泵模式，LCD 输出电压为 3V，占空比为 1/4duty、LCD 时钟为 XOSCL，LCD_CAD 关闭（CAD_MOD=0），所有 LCD 引脚打开，其他所有输出引脚无负载，所有数字输入引脚不浮动，其他外设关闭，Flash 进入睡眠模式，CPU 进入 STOP 模式。
		VDD=3.3V		5.4			
		VDD=5V		5.5			
	I_{stp5}	VDD=1.8V		7.6		uA	除了 IRCL(131kHZ)外，其他时钟关闭，LDO 设置为低功率模式，输出电压为 1.61V，打开 LCD 驱动（不外接 LCD 面板），LCD 模式设置为电阻分压模式，偏压比为 1/3，占空比为 1/4duty、LCD 电流设为最小，LCD 时钟为 XOSCL，LCD_CAD 关闭（CAD_MOD=0），所有 LCD 引脚打开，其他所有输出引脚无负载，所有数字输入引脚不浮动，其他外设关闭，Flash 进入睡眠模式，CPU 进入 STOP 模式。
		VDD=3.3V		9.3			
		VDD=5V		11.6			
IDLE 模式电流	I_{idl1}	VDD=1.8V		0.816		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		0.817			
		VDD=5V		0.821			
	I_{idl2}	VDD=1.8V		9.4		uA	系统时钟设为 IRCL（131KHz），其他时钟关闭，所有输出引脚无负载，所有数字输入引脚不浮动，所有外设关闭，LDO 设置为低功率模式，CPU 进入 IDLE 模式。
		VDD=3.3V		9.5			
		VDD=5V		9.6			
	I_{idl3}	VDD=1.8V		6.6		uA	系统时钟设为 XOSCL（32.768KHz），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		6.6			
		VDD=5V		6.7			
	I_{idl4}	VDD=1.8V		6.6		uA	除了 XOSCL(32.768kHZ)外，其他时钟关闭，LDO 设置为低功率模式，输出电压为 1.61V，打开 LCD 驱动（不外接 LCD 面板），LCD 模式设置为内建电压电荷泵模式，LCD 输出电压为 3V，占空比为 1/4duty、LCD 时钟为 XOSCL，LCD_CAD 关闭（CAD_MOD=0），所有 LCD 引脚打开，其他所有输出引脚无负载，所有数字输入引脚不浮动，
		VDD=3.3V		6.7			
		VDD=5V		6.8			

							其他外设关闭，Flash 进入睡眠模式，CPU 进入 STOP 模式。
	I_{idl5}	VDD=1.8V		8.9			除了 XOSCL(32.768kHZ)外，其他时钟关闭，LDO 设置为低功率模式，输出电压为 1.61V，打开 LCD 驱动（不外接 LCD 面板），LCD 模式设置为电阻分压模式，偏压比为 1/3，占空比为 1/4duty、LCD 电流设为最小，LCD 时钟为 XOSCL，LCD_CAD 关闭（CAD_MOD=0），所有 LCD 引脚打开，其他所有输出引脚无负载，所有数字输入引脚不浮动，其他外设关闭，Flash 进入睡眠模式，CPU 进入 STOP 模式。
		VDD=3.3V		10.7			
		VDD=5V		12.9			
	I_{idl6}	VDD=1.8V		9.5			除了 IRCL(131kHZ)外，其他时钟关闭，LDO 设置为低功率模式，输出电压为 1.61V，打开 LCD 驱动（不外接 LCD 面板），LCD 模式设置为内建电压电荷泵模式，LCD 输出电压为 3V，占空比为 1/4duty、LCD 时钟为 XOSCL，LCD_CAD 关闭（CAD_MOD=0），所有 LCD 引脚打开，其他所有输出引脚无负载，所有数字输入引脚不浮动，其他外设关闭，Flash 进入睡眠模式，CPU 进入 STOP 模式。
		VDD=3.3V		9.6			
		VDD=5V		9.7			
	I_{idl7}	VDD=1.8V		11.8		uA	除了 IRCL(131kHZ)外，其他时钟关闭，LDO 设置为低功率模式，输出电压为 1.61V，打开 LCD 驱动（不外接 LCD 面板），LCD 模式设置为电阻分压模式，偏压比为 1/3，占空比为 1/4duty、LCD 电流设为最小，LCD 时钟为 XOSCL，LCD_CAD 关闭（CAD_MOD=0），所有 LCD 引脚打开，其他所有输出引脚无负载，所有数字输入引脚不浮动，其他外设关闭，Flash 进入睡眠模式，CPU 进入 STOP 模式。
		VDD=3.3V		13.6			
		VDD=5V		15.8			
IO 端口输高电压（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	-	2.4	-	mA	IO 设为推挽输出模式， Vol=VDD-0.3V
		VDD=5V	-	3.3	-		

IO 端口灌电流(不包括 P06,P22,P23)	I_{ol}	VDD=3.3V	-	13.34	-	mA	IO 设为推挽输出模式， Vol=GND+0.3V
		VDD=5V	-	19.05	-		
IO 端口灌电流(P06,P22,P23)	I_{ol}	VDD=3.3V	-	11	-	mA	IO 设为推挽输出模式， Vol=GND+0.3V
		VDD=5V	-	15	-		
REM(P5.4)引脚灌电流	I_{rem}			400		mA	VDD=3.0V, Vol = GND+1.2V，灌电流能力设为最高档
IO 端口上拉电阻	Ru2	VDD=1.8~5.5V		10		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。

24.3 交流电气特性

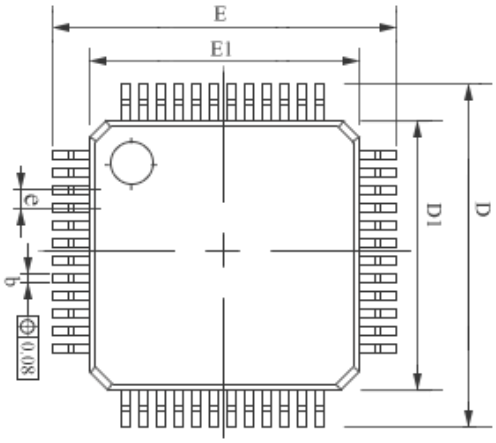
交流电气特性（VDD=1.8-5.5V, TA=25℃，除非其它说明）

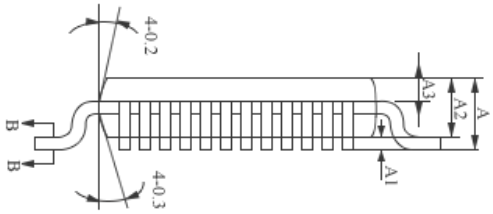
芯片参数	符号	最小值	典型值	最大值	单位	条件
内部低速时钟（IRCL）起振时间	Trc1	-	50	-	us	IRCL 频率为 131K
内部高速时钟（IRCH）起振时间	Trc2	-	10	-	us	IRCH 频率为 16MHz
复位脉冲时间	Trst	-	0.5	-	us	

备注：VDD=3.3V,TA=25 ℃,内部高速时钟出厂频率为16MHz，精度为±1%.

25 封装类型

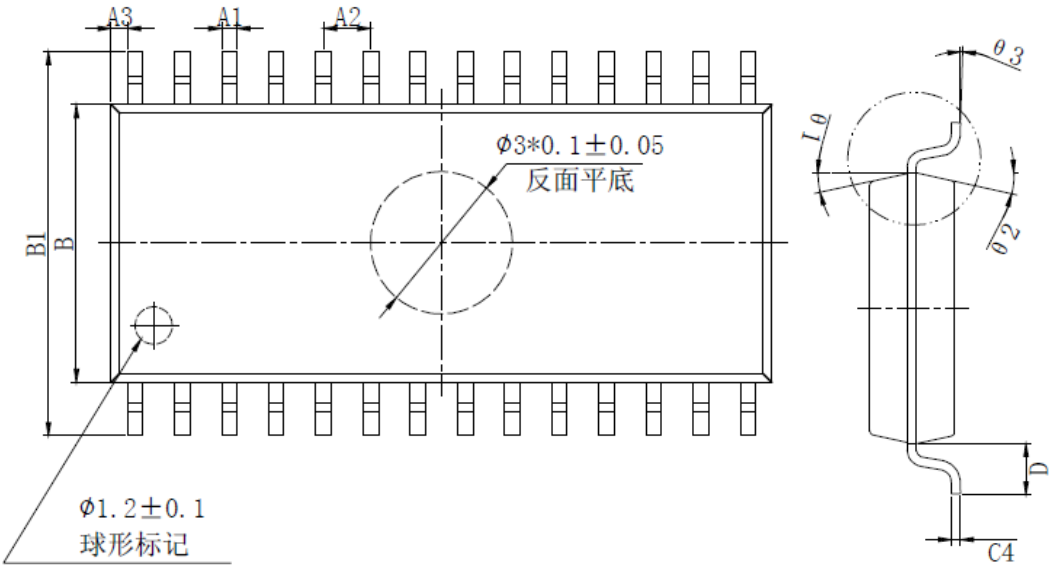
封装形式一 (LQFP48)

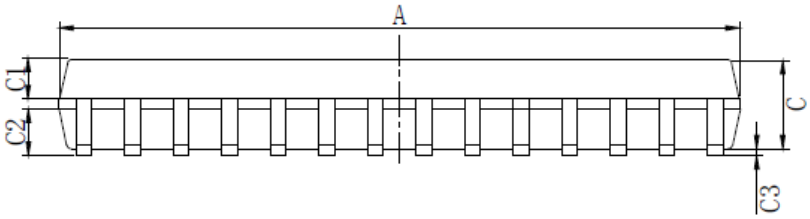




序号	最小值	标准值	最大值
A	————	————	1.60
A1	0.05	————	0.15
A2	1.35	1.40	1.45
A3	0.59	0.54	0.69
b	0.18	————	0.27
D	8.80	9.00	9.20
D1	6.90	7.00	7.10
E	8.80	9.00	9.20
E1	6.90	7.00	7.10
e	0.50		

封装形式二（SOP28）





序号	最小值(mm)	标准值(mm)	最大值(mm)
A	17.90	18.00	18.10
A1	0.356	0.40	0.456
A2	1.24	1.27	1.30
A3	---	0.542 TYP	---
B	7.40	7.50	7.60
B1	10.206	10.30	10.406
C	2.18	2.23	2.28
C1	0.938	1.0	1.038
C2	0.938	1.0	1.038
C3	0.03	0.09	0.17
D	1.353	1.40	1.453
C4	0.244	0.25	0.264

26 附录

附录 1 指令集速查表

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

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

		(B) ← REM	

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)) (SP) ← (SP) - 1	2
RETI	中断返回	(PC15-8) ← ((SP)) (SP) ← (SP) - 1 (PC7-0) ← ((SP)) (SP) ← (SP) - 1	2
AJMP addr11	绝对短转移	(PC) ← (PC) + 2	2

		(PC10-0) ← page address	
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
		(C) ← 1
		ELSE
		(C) ← 0
			2

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	2

		(PC) ← (PC) + relative offset
		IF (Rn) < data
		THEN
		(C) ← 1
		ELSE
		(C) ← 0	

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	2

		(PC) ← (PC) + rel	
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