CACHIP *Built - in 12 Bit ADC / Touch Key / LCD Driver / 1T 8051 Flash MCU*

CA51F2 系列 MCU
中文用户手册

REV1.9

深圳市锦锐科技有限公司
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

目录

1 概述

CA51F2 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 Flash 程序存储器，可多次重复编程的特性，此系列芯片提供 8 / 16 / 32K 三种 Flash 容量供客户按照产品需求选择，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 ADC、LCD/LED 驱动、Touch Key、PWM、 UART、RTC、无刷直流电机驱动、乘除法器、低电压检测(LVD)等功能模块。支持 IDLE、STOP 和低速运行三种省电模式以适应不同功耗要求的应用。强大的功能及优越的抗干扰性能使其可广泛应用于各种车载音响、家用音响、小家电、蓝牙音箱、汽车电子、数码电机、运动器材、马达控制、医疗保健、仪器仪表、安防、电源控制、工业控制及门铃产品中。

2 基本特性

- **内核**
  - CPU：1T 8051, 最高速度比传统 8051 快 10 倍
  - 兼容 8051 指令集, 双 DPTR 工作模式
  - CPU 频率：最高可支持 24MHz
- **存储器**
  - Flash : 8 / 16 / 32K 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，2K 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**
  - 外部高速振荡器：1 - 24MHz
  - 外部 RTC 振荡器：32.768KHz
  - 内置低速 RC 振荡器：131KHz
  - 内置 PLL：倍频倍数为 2 - 10 倍，参考时钟为 2 - 4MHz 内置 RC 振荡器
  - 内置高速 RC 振荡器：2 - 4MHz，精度可达 1% (出厂初始频率为 3.6864MHz@3.3V/25℃)
  - 内嵌外部时钟监控模块，可有效监控外部各个时钟工作状态，避免因外部时钟停振而造成死机
- **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，每个外部中断都可配置任意信号引脚作为中断输入脚

- **定时器**
  - 3 个 16 位通用定时器：定时器 0，定时器 1，定时器 2
- **通用输入输出口（GPIO）**
  - 最多支持 62 个 GPIO 口，支持推挽、开漏、上拉、下拉、高阻模式
  - 推挽输出时，单个 GPIO 推电流支持 20mA，灌电流支持 40mA
- **触摸按键（Touch Key）**
  - 内置触摸感应控制器
  - 最大支持 24 触摸通道
  - 高抗干扰性能，符合 EMC(CS)标准
  - 可在 STOP 模式下正常工作，支持 STOP 模式触摸唤醒
- **模/数转换器（ADC）**
  - 支持 8 通道 12 位 SAR ADC，内置运放和比较功能
  - 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
  - 支持检测信号缩小和放大功能，缩放倍数可选
- **PWM**
  - 支持 8 通道 PWM，在 16 位范围内可任意配置周期和占空比
  - 支持互补模式和死区控制，可用于驱动直流无刷电机
  - 支持可设置边沿对齐和中心对齐模式
  - 支持可直接输出内部时钟功能
  - 支持 PWM 中断
- **LCD 驱动**
  - 最大可支持 8com x 32seg、7com x 33seg、6com x 34seg、5com x 35seg、4com x 36seg
  - 可配置占空比：1/2、1/3、1/4、1/5、1/6、1/7、1/8 Duty
  - 可配置偏压：1/2、1/3、1/4 Bias
  - 支持 8 级对比度调整
  - 支持 3 种等级驱动电流，用户可根据不同的 LCD 屏进行调整
- **LED 驱动**
  - 最大可支持 8com x 32seg
  - 支持 8 级亮度调节
- **低电压检测（LVD）**
  - 可配置电压检测范围 1.8 - 4.8V
  - 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

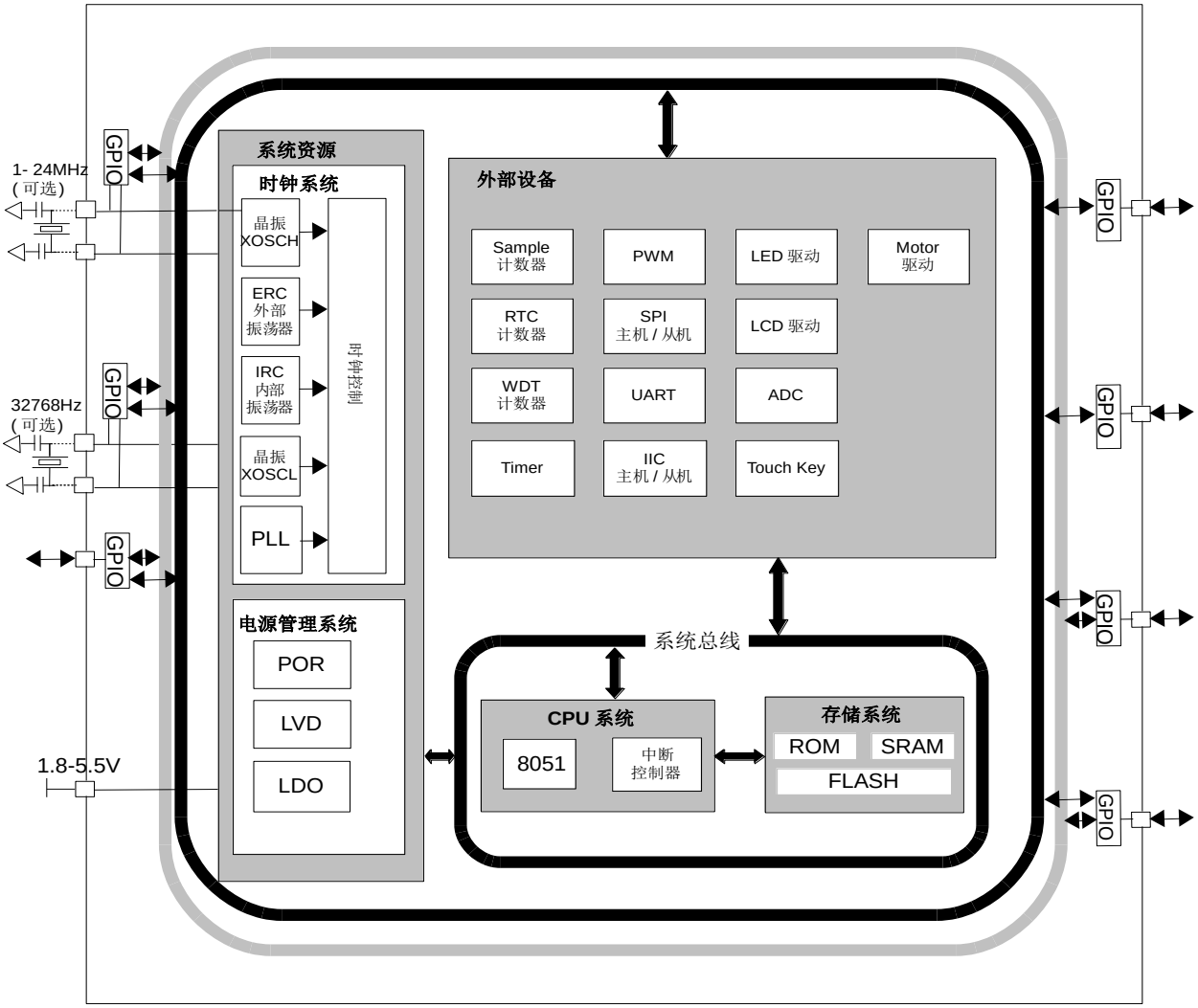
- **遥控接收功能**
  - 内置采样计数器模块（SAMPLE），可通过硬件模块采样任意长度的脉宽，减少软件代码
- **通用串行接口（UART）**
  - 最多支持 3 个 UART 接口
  - 支持 1 字节接收缓存
- **SPI 接口**
  - 内置 1 个 4 线 SPI 接口，支持主从模式
- **I^2C 接口**
  - 内置 1 路 I^2C 接口，支持主从模式，支持标准/快速/高速模式
- **运放和模拟比较器**
  - 支持 4 路模拟比较器、两个运算放大器和一个捕获计数器
  - 模拟比较器参考电压可选择内部基准或外部输入基准
  - 模拟比较器内置 15 位数字滤波器，支持比较器中断
  - 运放可以和 ADC 和模拟比较器结合使用，扩展检测信号电压范围
  - 捕获计数器和模拟比较器结合使用，可用于电机测速及堵转检测
- **无刷直流电机驱动**
  - 内置 60°霍尔和 120°霍尔译码模块
  - 支持自动模式和手动模式，支持刹车功能
  - 支持多种异常检测
  - 结合模拟比较器可实现无霍直流电机驱动
- **乘除法器（MDU）**
  - 支持 1 个时钟周期 16 位 × 16 位乘法
  - 支持 8 个时钟周期 32 位 ÷ 32 位除法
  - 支持 1 个时钟周期 32 位数据左右移位操作
- **程序下载和仿真**
  - 支持 ISP 和 IAP
  - 支持在线仿真功能
- **低功耗**
  - STOP 模式，电流<5uA
  - IDLE 模式，电流<10uA
  - 低速运行模式，电流<15uA
- **封装类型：**LQFP64 (7 x7 mm)
  LQFP48 (7 x7 mm)

3 芯片型号功能介绍

表 3-1 CA51F2 系列具体型号功能特点

芯片型号	Flash 容量[BYTE]	外部 Ram[BYTE]	外部高速晶体振荡器	外部低速晶振[32.768KHz]	GPIO 数量	UART 数量	I²C	SPI	16 bit PWM 通道数量	12 bit ADC 通道数量	SAMPLE 功能	触摸按键数量	LCD 驱动[com x seg]	LED 驱动[com x seg]	直流无刷电机驱动	片上仿真功能	封装形式
CA51F251L2	8K	2K	--	√	46	3	√	√	5	6	--	15	4X25 5X24	5X24	--	√	LQFP48
CA51F252L2	16K	2K	--	√	46	3	√	√	5	6	--	15	4X25 5X24	5X24	--	√	LQFP48
CA51F253L2	32K	2K	--	√	46	3	√	√	5	6	--	15	4X25 5X24	5X24	--	√	LQFP48
CA51F251L3	8K	2K	√	√	62	3	√	√	8	8	√	24	8X32 7X33 6X34 5X35 4X36	8X32	√	√	LQFP64
CA51F252L3	16K	2K	√	√	62	3	√	√	8	8	√	24	8X32 7X33 6X34 5X35 4X36	8X32	√	√	LQFP64
CA51F253L3	32K	2K	√	√	62	3	√	√	8	8	√	24	8X32 7X33 6X34 5X35 4X36	8X32	√	√	LQFP64

4 系统框图


1- 24MHz
(可选)
32768Hz
(可选)
1.8-5.5V
GPIO
系统资源
时钟系统
晶振
XOSCH
ERC
外部
振荡器
IRC
内部
振荡器
晶振
XOSCL
PLL
时钟控制
电源管理系统
POR
LVD
LDO
外部设备
Sample
计数器
PWM
LED 驱动
Motor
驱动
RTC
计数器
SPI
主机 / 从机
LCD 驱动
WDT
计数器
UART
ADC
Timer
IIC
主机 / 从机
Touch Key
系统总线
CPU 系统
8051
中断
控制器
存储系统
ROM
SRAM
FLASH

5 引脚封装及其描述

5.1 封装定义

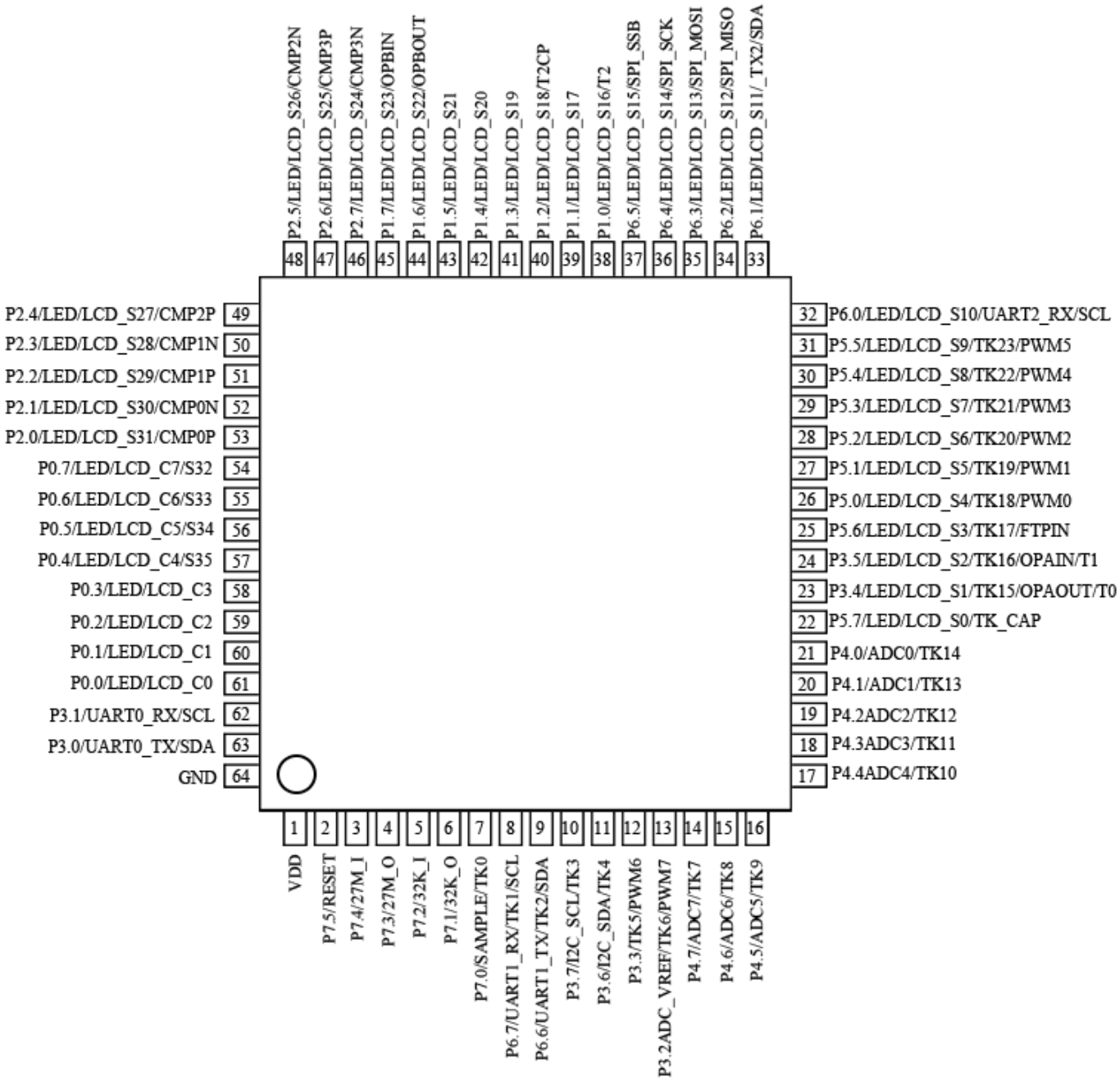


图 5-1-1　LQFP64 封装图

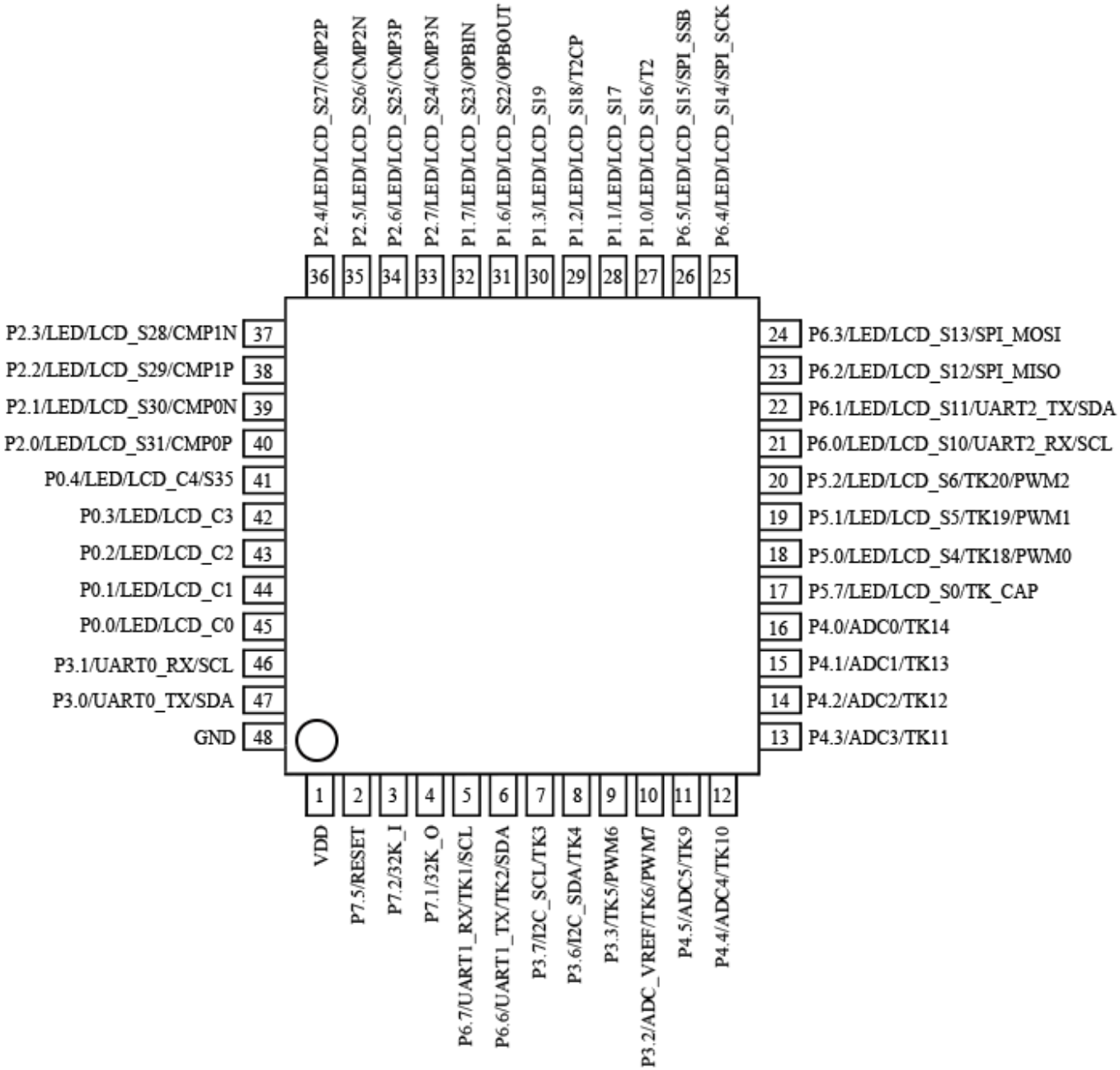


图 5-1-2LQFP48 封装图

5.2 引脚描述

表 5-2-1 引脚描述

引脚序号			管脚名称	管脚功能	默认功能
LQFP64	LQFP48	SSOP28			
1	1		VDD	芯片供电管脚	芯片供电管脚
2	2		P7.5/RESET	通用双向 I/O 口 硬件复位脚	硬件复位脚
3	-		P7.4/XTAL_IN_24M	通用双向 I/O 口 外部高速晶振输入	通用双向 I/O 口
4	-		P7.3/XTAL_OUT_24M	通用双向 I/O 口 外部高速晶振输出	通用双向 I/O 口
5	3		P7.2/XTAL_IN_32K	通用双向 IO 口 32K 外部晶振输入	32K 外部晶振输入
6	4		P7.1/XTAL_OUT_32K	通用双向 IO 口 32K 外部晶振输出	32K 外部晶振输出
7	-		P7.0/SAMPLE/TK[0]	通用双向 IO 口 采样信号数字输入 触摸按键模拟通道输入	通用双向 IO 口
8	5		P6.7/UART[1]_RX/TK[1]/SCL	通用双向 IO 口 串口 1 数据接收端口 I2C 时钟传输端口 触摸按键模拟通道输入	I2C 时钟传输端口
9	6		P6.6/UART[1]_TX/TK[2]/SDA	通用双向 IO 口 串口 1 数据发送端口 I2C 数据传输端口 触摸按键模拟通道输入	I2C 数据传输端口
10	7		P3.7/I2C_SCL/TK[3]	通用双向 IO 口 I2C 时钟传输端口 触摸按键模拟通道输入	I2C 时钟传输端口
11	8		P3.6/I2C_SDA/TK[4]	通用双向 IO 口 I2C 数据传输端口 触摸按键模拟通道输入	I2C 数据传输端口
12	9		P3.3/TK[5]/PWM[6]	通用双向 IO 口 触摸按键模拟通道输入 PWM 数字输出	通用双向 IO 口
13	10		P3.2/TK[6]/PWM[7]/ADC_REF	通用双向 IO 口 触摸按键模拟通道输入 PWM 数字输出 ADC 参考电压输入	通用双向 IO 口
14	-		P4.7/ADC_CH[7]/TK[7]	通用双向 IO 口 ADC 模拟通道输入 触摸按键模拟通道输入	通用双向 IO 口

15	-		P4.6/ADC_CH[6]/TK[8]	通用双向 IO 口 ADC 模拟通道输入 触摸按键模拟通道输入	通用双向 IO 口
16	11		P4.5/ADC_CH[5]/TK[9]	通用双向 IO 口 ADC 模拟通道输入 触摸按键模拟通道输入	通用双向 IO 口
17	12		P4.4/ADC_CH[4]/TK[10]	通用双向 IO 口 ADC 模拟通道输入 触摸按键模拟通道输入	通用双向 IO 口
18	13		P4.3/ADC_CH[3]/TK[11]	通用双向 IO 口 ADC 模拟通道输入 触摸按键模拟通道输入	通用双向 IO 口
19	14		P4.2/ADC_CH[2]/TK[12]	通用双向 IO 口 ADC 模拟通道输入 触摸按键模拟通道输入	通用双向 IO 口
20	15		P4.1/ADC_CH[1]/TK[13]	通用双向 IO 口 ADC 模拟通道输入 触摸按键模拟通道输入	通用双向 IO 口
21	16		P4.0/ADC_CH[0]/TK[14]	通用双向 IO 口 ADC 模拟通道输入 触摸按键模拟通道输入	通用双向 IO 口
22	17		P5.7/LED_SEG[0]/LCD_SEG[0]/TK_CAP	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出 触摸按键模拟通道输入 触摸按键外接电容	通用双向 IO 口
23	-		P3.4/T0/LED_SEG[1]/SEG[1]/TK[15]/OPAOUT	通用双向 IO 定时器 T0 数字输入 LED SEG 数字输出 LCD SEG 模拟输出 触摸按键模拟通道输入 运放 A 模拟输出	通用双向 IO 口
24	-		P3.5/T1/LED_SEG[2]/SEG[2]/TK[16]/OPAIN	通用双向 IO 定时器 T1 数字输入 LED SEG 数字输出 LCD SEG 模拟输出 触摸按键模拟通道输入 运放 A 模拟输入	通用双向 IO 口
25	-		P5.6/LED_SEG[3]/SEG[2]/TK[17]/FTPIN	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出 触摸按键模拟通道输入 电机错误检测数字输入	通用双向 IO 口
26	18		P5.0/LED_SEG[4]/SEG[4]/TK[18]/PWM[0]	通用双向 IO	通用双向 IO 口

				LED SEG 数字输出 LCD SEG 模拟输出 触摸按键模拟通道输入 PWM 数字输出	
27	19		P5.1/LED_SEG[5]/SEG[5]/TK[19]/PWM[1]	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出 触摸按键模拟通道输入 PWM 数字输出	通用双向 IO 口
28	20		P5.2/LED_SEG[6]/SEG[6]/TK[20]/PWM[2]	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出 触摸按键模拟通道输入 PWM 数字输出	通用双向 IO 口
29	-		P5.3/LED_SEG[7]/SEG[7]/TK[21]/PWM[3]	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出 触摸按键模拟通道输入 PWM 数字输出	通用双向 IO 口
30	-		P5.4/LED_SEG[8]/SEG[8]/TK[22]/PWM[4]	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出 触摸按键模拟通道输入 PWM 数字输出	通用双向 IO 口
31	-		P5.5/LED_SEG[9]/SEG[9]/TK[23]/PWM[5]	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出 触摸按键模拟通道输入 PWM 数字输出	通用双向 IO 口
32	21		P6.0/LED_SEG[10]/SEG[10]/UART[2]_RX/SCL	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出 串口[2]RX 端口 IIC_SCL 端口	I2C_SCL 端口
33	22		P6.1/LED_SEG[11]/SEG[11]/UART[2]_TX/SDA	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出 串口[2]TX 端口 IIC_SDA 端口	I2C_SDA 端口
34	23		P6.2/LED_SEG[12]/SEG[12]/SPI_MISO	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出 SPI _MISO 端口	通用双向 IO 口
35	24		P6.3/LED_SEG[13]/SEG[13]/SPI_MOSI	通用双向 IO	通用双向 IO 口

				LED SEG 数字输出 LCD SEG 模拟输出 SPI _MOSI 端口	
36	25		P6.4/LED_SEG[14]/SEG[14]/SPI_SCK	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出 SPI _SCK 端口	通用双向 IO 口
37	26		P6.5/LED_SEG[15]/SEG[15]/SPI_SSB	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出 SPI _SSB 端口	通用双向 IO 口
38	27		P1.0/T2/LED_SEG[16]/SEG[16]	通用双向 IO 定时器 T2 数字输入 LED SEG 数字输出 LCD SEG 模拟输出	通用双向 IO 口
39	28		P1.1/T2EX/LED_SEG[17]/SEG[17]	通用双向 IO 定时器 T2EX 数字输入 LED SEG 数字输出 LCD SEG 模拟输出	通用双向 IO 口
40	29		P1.2/LED_SEG[18]/SEG[18]/T2CP	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出 定时器 T2CP 数字输入	通用双向 IO 口
41	30		P1.3/LED_SEG[19]/SEG[19]	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出	通用双向 IO 口
42	-		P1.4/LED_SEG[20]/SEG[20]	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出	通用双向 IO 口
43	-		P1.5/LED_SEG[21]/SEG[21]	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出	通用双向 IO 口
44	31		P1.6/LED_SEG[22]/SEG[22]/OPBOUT	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出 运放 B 模拟输出	通用双向 IO 口
45	32		P1.7/LED_SEG[23]/SEG[23]/OPBIN	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出 运放 B 模拟输入	通用双向 IO 口
46	33		P2.7/LED_SEG[24]/SEG[24]/CMP3N	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出 比较器 3 负极模拟输入	通用双向 IO 口

47	34		P2.6/LED_SEG[25]/SEG[25]/CMP3P	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出 比较器 3 正极模拟输入	通用双向 IO 口
48	35		P2.5/LED_SEG[26]/SEG[26]/CMP2N	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出 比较器 2 负极模拟输入	通用双向 IO 口
49	36		P2.4/LED_SEG[27]/SEG[27]/CMP2P	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出 比较器 2 正极模拟输入	通用双向 IO 口
50	37		P2.3/LED_SEG[28]/SEG[28]/CMP1N	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出 比较器 1 负极模拟输入	通用双向 IO 口
51	38		P2.2/LED_SEG[29]/SEG[29]/CMP1P	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出 比较器 1 正极模拟输入	通用双向 IO 口
52	39		P2.1/LED_SEG[30]/SEG[30]/CMP0N	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出 比较器 0 负极模拟输入	通用双向 IO 口
53	40		P2.0/LED_SEG[31]/SEG[31]/CMP0P	通用双向 IO LED SEG 数字输出 LCD SEG 模拟输出 比较器 0 正极模拟输入	通用双向 IO 口
54	-		P0.7/LED_COM[7]/COM[7]/SEG[32]	通用双向 IO LED COM 数字输出 LCD COM 模拟输出 LCD SEG 模拟输出	通用双向 IO 口
55	-		P0.6/LED_COM[6]/COM[6]/SEG[33]	通用双向 IO LED COM 数字输出 LCD COM 模拟输出 LCD SEG 模拟输出	通用双向 IO 口
56	-		P0.5/LED_COM[5]/COM[5]/SEG[34]	通用双向 IO LED COM 数字输出 LCD COM 模拟输出 LCD SEG 模拟输出	通用双向 IO 口
57	41		P0.4/LED_COM[4]/COM[4]/SEG[35]	通用双向 IO LED COM 数字输出 LCD COM 模拟输出 LCD SEG 模拟输出	通用双向 IO 口

58	42		P0.3/LED_COM[3]/COM[3]	通用双向 IO LED COM 数字输出 LCD COM 模拟输出	通用双向 IO 口
59	43		P0.2/LED_COM[2]/COM[2]	通用双向 IO LED COM 数字输出 LCD COM 模拟输出	通用双向 IO 口
60	44		P0.1/LED_COM[1]/COM[1]	通用双向 IO LED COM 数字输出 LCD COM 模拟输出	通用双向 IO 口
61	45		P0.0/LED_COM[0]/COM[0]	通用双向 IO LED COM 数字输出 LCD COM 模拟输出	通用双向 IO 口
62	46		P3.1/UART[0]_RX/SCL	通用双向 IO 串口 0 数据接收端口 I2C 时钟传输端口	I2C 时钟传输端口
63	47		P3.0/UART[0]_TX/SDA	通用双向 IO 串口 0 数据发送端口 I2C 数据传输端口	I2C 数据传输端口
64	48		GND	电源地引脚	电源地引脚

备注：信号引脚复用功能设置方法详见表 15-2-9 和表 15-2-10。

6 中央处理器（CPU）

6.1 CPU 简介

CA51F2 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）

CA51F2 系列芯片提供了 256 字节内部 RAM 和 2K 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 2K 字节外部 RAM（地址：0000H ~ 07FFH）可通过 MOVX 指令间接寻址。

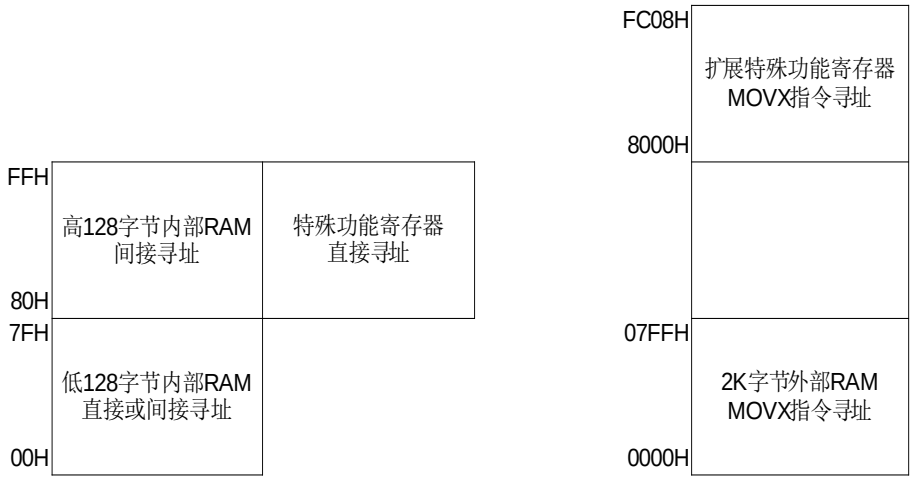


图 7-1-1　RAM 组织结构图

7.2 特殊功能寄存器（SFR）

CA51F2 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	LXCON	LXCFG	LXDAT	LXDIVL	LXDIVH	MDUCON	MDUDAT
D8H	P5	P7	PWMEN	PWMUPD	PWMCMX	PWMCON	PWMCFG	PWMDIVL
D0H	PSW	PWMDIVH	PWMDUTL	PWMDUTH	PWMAIF	PWMBIF	PWMCIF	PWMDIF
C8H	T2CON	T2MOD	T2CL	T2CH	TL2	TH2	TKMSL	TKMSH
C0H	P4	TKCON	TKCFG	TKMTS	TKCHS	ATKCL	ATKCH	TKIF

B8H	IP	ADCON	ADCFGL	ADCFGH	ADCDL	ADCDH	CKMON	CKMIF
B0H	P3	I2CCON	I2CADR	I2CADM	I2CCCR	I2CDAT	I2CSTA	I2CFLG
A8H	IE	P6	WDCON	WDFLG	WDVTHL	WDVTHH	PLLCON	HVTH
A0H	P2	S2CON	S2BUF	S2RELL	S2RELH	SPCON	SPDAT	SPSTA
98H	S0CON	S0BUF	S1CON	S1BUF	S1RELL	S1RELH	RCMSHL	RCMSHH
90H	P1	RCCON	VCKDL	VCKDH	RCTAGL	RCTAGH	RCMSLL	RCMSLH
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，CA51F2 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	P56F	P57F
8030H	P60F	P61F	P62F	P63F	P64F	P65F	P66F	P67F
8038H	P70F	P71F	P72F	P73F	P74F	P75F		
8040H	OPACON	OPBCON	-	-	-	-	-	-
8048H	CP0CON	CP1CON	CP2CON	CP3CON	CPCKS	CPSTA	CPVTC	
8050H	FT0SL	FT0SH	FT1SL	FT1SH	FT2SL	FT2SH	FT3SL	FT3SH
8058H	CTMCON	CTMVTHL	CTMVTHH	CTMCNTL	CTMCNTH	-	-	-
8060H	MOTCON	MOTCFG	MTGCON	MHLCON	MFPCON	MOTCMD	MTGDL	MOTIF
8068H	HDCT0	HDCT1	HDCT2	HDCT3	HDCT4	HDCT5	HDCT6	HDCT7
8070H	HDCT8	HDCT9	HDCT10	HDCT11	MOTPLC	-	-	-
8078H	SMCON	SMSTA	SMDIV	SMDATL	SMDATH	SMVTHL	SMVTHH	-
8080H	CKCON	CKSEL	CKDIV	IHCFGL	IHCFGH	ILCFGL	ILCFGH	TFCFG
8088H	ADCALL	ADCALH	ACPDLL	ACPDLH	ACPDHL	ACPDHH	-	-
8090H	TKMAXF	TLMINF	ATKNL	ATKNH	-	-	-	-
8100H	SPMAX	I2CIOS	-	-	-	-	-	-
FC00H	MECON	FSCMD	FSDAT	LOCK	PADRD	PTSL	PTSH	REPSET

7.3 Flash 存储器

7.3.1 功能简介

Flash 存储器提供 8 / 16/ 32K 字节 Flash 存储器（不同型号容量有所不同），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 256 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 7-3-2　8K Flash 存储器结构

图 7-3-3　16K Flash 存储器结构

图 7-3-4　32K 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	REMAP	DPSTB	-	-		-	BOOT[1:0]	
R/W	R/W	R/W					R/W	
初始值	0	0	-	-		-	0	0

位编号	位符号	说明
7	REMAP	XRAM 地址映射控制位 0：地址映射关闭 1：地址映射使能，2K XRAM 映射到程序空间地址为 8000H~87FFH。
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。*
5~2	-	-
1~0	BOOT	设置软复位后程序启动空间选择位域 01：软复位后程序从 XRAM 启动运行 10：软复位后程序从 FLASH 启动运行 其它值：无效

表 7-3-3-2 寄存器 FSCMD

FC01H	7	6	5	4	3	2	1	0
FSCMD	-	-	-	-	CMD[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	CMD	命令寄存器 0000：无操作 0100：Flash 整片擦除 0001：读 Flash 数据区 0010：写 Flash 数据区 0011：扇区擦 Flash 数据区 1011：块擦 Flash 数据区(每块为 512 字节) 0101：读 Flash 程序区 0110：写 Flash 程序区 0111：扇区擦 Flash 程序区

		1111：块擦 Flash 程序区(每块为 512 字节) 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					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	-							
6	-	-						
5	-							
4	-							
3	FLKF	可编程区解锁标志，1 表示已解锁						
2	PLKF	程序区解锁标志，1 表示已解锁						
1	DLKF	数据区解锁标志，1 表示已解锁						
0	-	-						

表 7-3-3-5 寄存器 PADRD

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

位编号	位符号	说明
7~0	PADRD	程序区和数据区划分配置寄存器 程序区和数据区以 256 字节为单位进行划分，当 PADRD>0 时， 程序区的地址空间为: 0 ~ (PADRD×256 - 1), 数据区的地址空间为: (PADRD×256) ~ 1FFFH/3FFFH/7FFFH. 备注： 1. 当 PADRD=0 时，整个 Flash 空间都是数据空间。 2. 1FFFH/3FFFH/7FFFH 分别表示 8K/16K/32K Flash 的最大地址。 3. 当 Flash 容量为 8K/16K/32K 时，PADRD 的最大值分别为 20H/40H/80H。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	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，32K 的 Flash 空间划分最后 1024 字节为数据空间，其余为程序空间，程序如下：

```
---------------------------------------------------------------------------------
PADRD = 124; //程序区空间地址为：0~0x7BFF,数据区空间地址为：0x7C00~0x7FFF
```

备注：以上设置数据区在 FLASH 中的物理地址是 0x7C00~0x7FFF，但是逻辑地址是 0x0000~0x03FF，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
}
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
}
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。

◆ **程序空间扇区擦除**

例如，需要擦除程序空间扇区 n，程序如下：

```
-------------------------------------------------------------------------------
FSCMD = 0;          //设置 CMD 为 0
LOCK = 0x29;        //程序空间解锁
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 = 7;   //设置擦除命令
LOCK = 0xAA;        //FLASH 加锁
-------------------------------------------------------------------------------
```

备注：扇区序号 n=0、1、2……。

◆ **程序空间写入数据**

例如，往程序空间地址为 n~(n+100)写入数据 0xAA，程序如下：

```
-------------------------------------------------------------------------------
unsigned char i;
FSCMD = 0;          //设置 CMD 为 0
LOCK = 0x29;        //程序空间解锁
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 = 6;   //设置写命令
for(i=0;i<100;i++)
{
    FSDAT = 0xAA;     //连续写入数据
}
LOCK = 0xAA;        //FLASH 加锁
-------------------------------------------------------------------------------
```

备注：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。

◆ **程序空间读出数据**

例如，从程序空间地址为 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 = 0x29;        //程序空间解锁
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 = 5;   //设置读命令
for(i=0;i<100;i++)
{
      *pBuf++ = FSDAT ;//连续写入数据
}
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

外部 2K RAM 可以映射为程序空间使用,当 REMAP=0（详见寄存器 MECON）时，映射地址为 0000H~07FFH（执行软复位从 XRAM 开始运行才有效）；当 REMAP=1 时，映射地址为 8000H~87FFH，映射图如图 7-5-1 所示。用户可以下载程序到外部 RAM 空间，当程序运行时设置 REMP=1 后，再执行跳转指令跳到映射程序区执行。同样效果，也可把 BOOT[1:0]（详见寄存器 MECON）的值设置为 01，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~07FFH）。映射程序区用来实现 IAP 等功能特别方便。

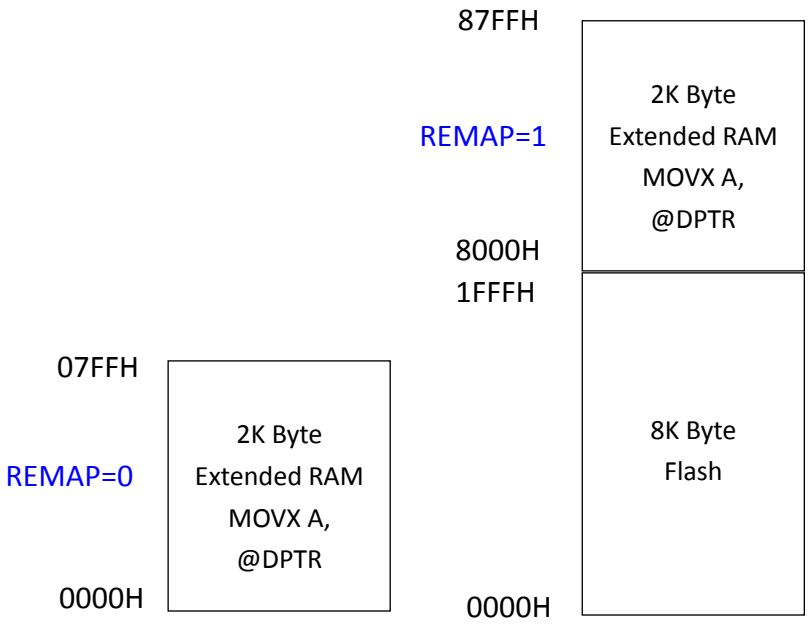


图 7-5-1 XRAM 地址映射图

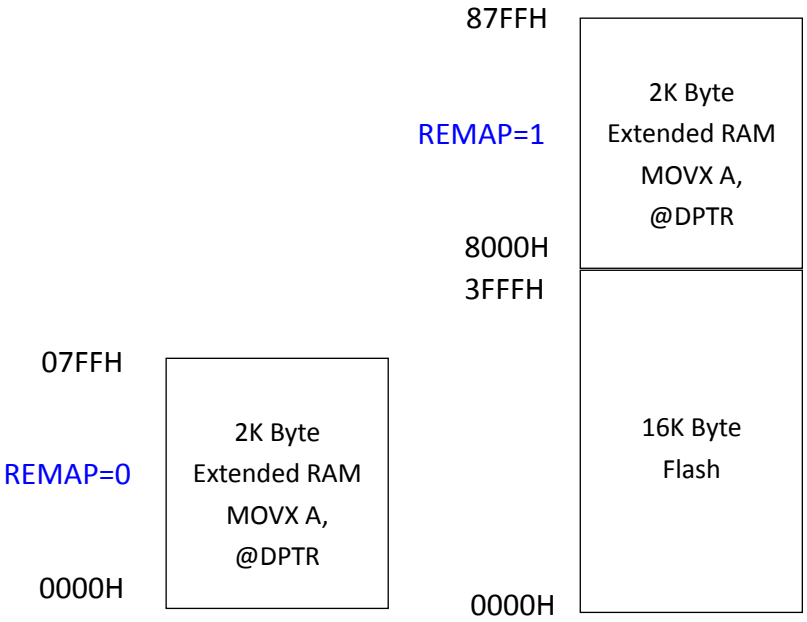


图 7-5-2 XRAM 地址映射图

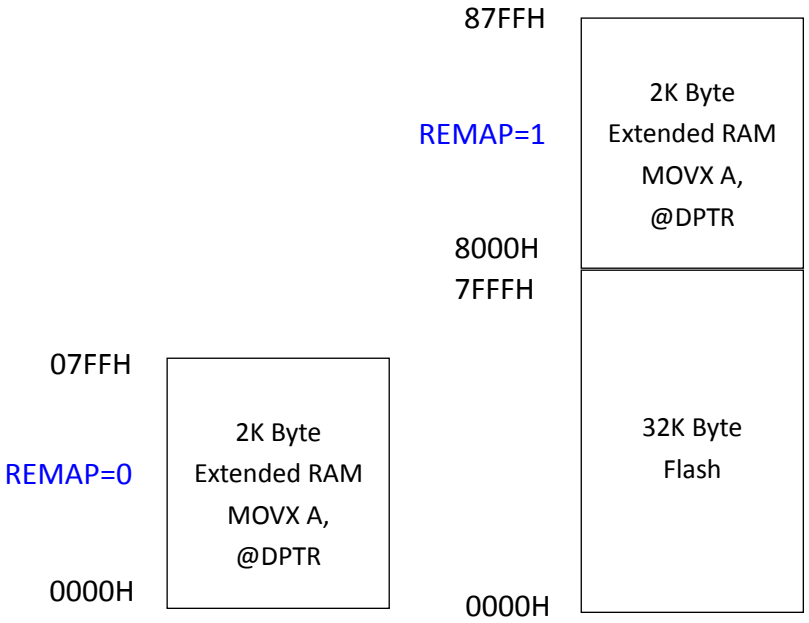


图 7-5-3 XRAM 地址映射图

8 中断系统

8.1 功能简介

CA51F2 系列芯片有一个增强的中断控制系统，共有 15 个中断入口，每个中断入口有若干中断源，每个中断源有 2 级中断优先级。每个中断源都有独立的中断向量、优先级设置位、中断使能位、中断标志。CPU 在响应中断后，进入该中断对应的中断服务程序，接到 RETI 指令后将返回中断前状态。如果同时有多个有效中断产生中断请求，CPU 将根据设置的中断优先级依次响应；如果优先级相同，则根据它们的自然优先级（中断入口地址从低到高）依次响应。

8.2 中断逻辑

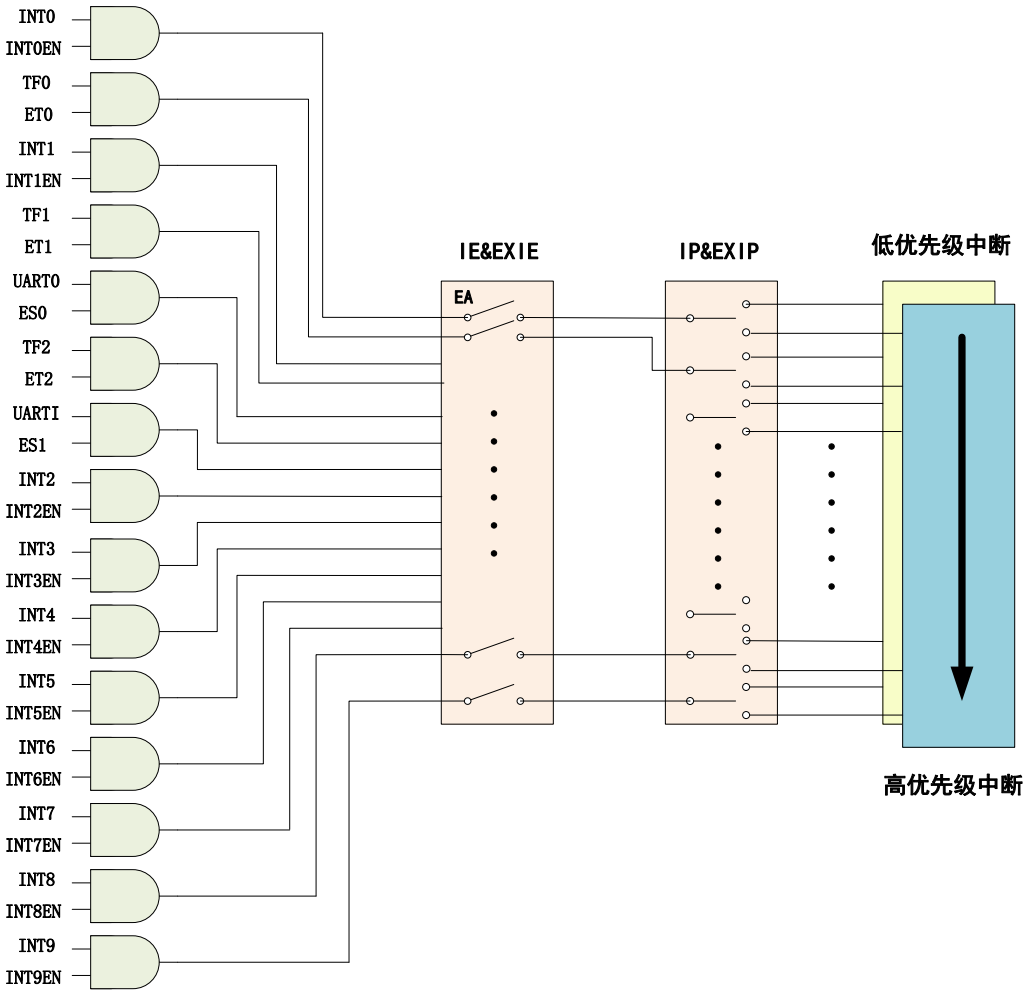


表 8-2-1 中断逻辑图

8.3 中断向量表

中断	中断源	向量	默认优先级
INT0	INT0	03H	0
TF0	定时器 0	0BH	1
INT1	INT1	13H	2
TF1	定时器 1	1BH	3
UART0	UART0	23H	4
TF2	定时器 2	2BH	5
UART1	UART1	33H	6
INT2	ADC/外部中断 2	3BH	7
INT3	UART2/TK/外部中断 3	43H	8
INT4	LVD/外部中断 4	4BH	9
INT5	SPI/时钟监控中断/外部中断 5	53H	10
INT6	I2C/模拟比较器/外部中断 6	5BH	11
INT7	WDT/MOTOR/外部中断 7	63H	12
INT8	RTC/捕获计数器/外部中断 8	6BH	13
INT9	SAMPLE/PWM/外部中断 9	73H	14

表 8-3-1 中断向量表

8.4 中断控制寄存器

表 8-4-1 寄存器 IE

A8H	7	6	5	4	3	2	1	0
IE	EA	ES1	ET2	ES0	ET1	INT1EN	ET0	INT0EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7	EA	全局中断使能控制位 0：全局中断关闭 1：全局中断打开
6	ES1	UART1 中断使能控制位 0：UART1 中断关闭 1：UART1 中断打开
5	ET2	定时器 2 中断使能控制位 0：定时器 2 中断关闭 1：定时器 2 中断打开
4	ES0	UART0 中断使能控制位 0：UART0 中断关闭

		1：UART0 中断打开
3	ET1	定时器 1 中断使能控制位 0：定时器 1 中断关闭 1：定时器 1 中断打开
2	EX1	外部中断 1 使能控制位 0：外部中断 1 中断关闭 1：外部中断 1 中断打开
1	ET0	定时器 0 中断使能控制位
0	EX0	外部中断 0 使能控制位 0：外部中断 0 中断关闭 1：外部中断 0 中断打开

表 8-4-2 寄存器 EXIE

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

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

		1：打开

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

表 8-4-3 寄存器 IP

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

位编号	位符号	说明
7	-	-
6	PS1	UART1 优先级控制位 0：低优先级 1：高优先级
5	PT2	定时器 2 优先级控制位 0：低优先级 1：高优先级
4	PS0	UART0 优先级控制位 0：低优先级 1：高优先级
3	PT1	定时器 1 优先级控制位 0：低优先级 1：高优先级
2	PX1	外部中断 1 优先级控制位 0：低优先级 1：高优先级
1	PT0	定时器 0 优先级控制位 0：低优先级 1：高优先级
0	PX0	外部中断 0 优先级控制位 0：低优先级 1：高优先级

表 8-4-4 寄存器 EXIP

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

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

8.5 外部中断

8.5.1 外部中断介绍

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

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

8.5.2 外部中断寄存器

表 8-5-2-1 寄存器 IT0CON

8FH	7	6	5	4	3	2	1	0
IT0CON	-	-	IT0PS[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	IT0PS[5:0]	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[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	IT1PS[5:0]	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[5:0]					
R/W	R/W	-	R/W					
初始值	0	-	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	-	-
5~0	EPPS[5:0]	中断引脚选择位域 编号和引脚对应表参考表 8-4-7

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	P40	32
P01	1	P41	33
P02	2	P42	34
P03	3	P43	35
P04	4	P44	36
P05	5	P45	37
P06	6	P46	38
P07	7	P47	39
P10	8	P50	40
P11	9	P51	41
P12	10	P52	42
P13	11	P53	43
P14	12	P54	44
P15	13	P55	45
P16	14	P56	46
P17	15	P57	47
P20	16	P60	48
P21	17	P61	49
P22	18	P62	50
P23	19	P63	51
P24	20	P64	52
P25	21	P65	53
P26	22	P66	54
P27	23	P67	55
P30	24	P70	56
P31	25	P71	57
P32	26	P72	58
P33	27	P73	59
P34	28	P74	60
P35	29	P75	61
P36	30		
P37	31		

8.5.3 外部中断控制方法及例程

◆ **外部中断 0/1 控制例程**

例如，设置 P20 为外部中断 0 中断输入引脚并开启外部中断 0，程序如下:

```
---------------------------------------------------------------------------------
void INT0_init(void)
{
    P20F = 1;       //设置 P20 为输入引脚
    IT0CON = 16; //选择 P20 作为中断 0 引脚，16 为 P20 对应的索引编号
    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，设置 P20 为外部中断 1 中断输入引脚并开启外部中断 1，程序如下:

```
---------------------------------------------------------------------------------
void INT1_init(void)
{
    P20F = 1;       //设置 P20 为输入引脚
    IT1CON = 16; //选择 P20 作为中断 1 引脚，16 为 P20 对应的索引编号
    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 P20 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)
{
    P20F = 1;       //设置 P20 为输入引脚
    INDEX = 0;          //设置 EPCON 索引号，0~7 分别对应外部中断 2~9
    EPCON = (0<<7) | 16;    //设置为上升沿触发并设置中断引脚索引编号，16 对应 P20，如果设置为下
                            //降沿触发则设置为 EPCON = (1<<7) | 16;
    INT2EN = 1;   //INT2 中断使能
    EPIE |= 0x01;  //外部中断 2 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

时钟系统负责整个系统时钟的生成、分频以及分配工作。CA51F2 系列芯片共支持以下时钟源：

- 内置 2 - 4MHz RC 振荡器
- 内置 131KHz RC 振荡器
- 内置 4MHz RC 振荡器
- 支持外部 1 - 24MHz 晶体振荡器
- 支持外部 1 - 24MHz RC 振荡器
- 支持外部 32.768KHz 晶体振荡器
- 倍频倍数为 2 - 10 倍的 PLL

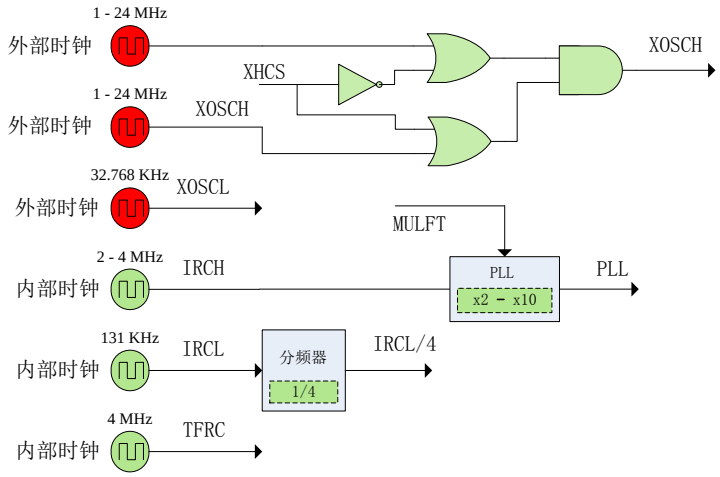


图 9-1-1 时钟源示意图

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	名称缩写	描述
IRCH	内置 2~4MHz RC 振荡器	ERC	外部 RC 振荡器
IRCL	内置 131KHz RC 振荡器	TFRC	内置 4MHz RC 振荡器
XOSCL	外部 32.768KHz 晶体振荡器	PLL	倍数为 2~10 倍的 PLL
XOSCH	外部 1~24MHz 晶体振荡器		

9.1.2 内置 2 - 4MHz 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。IRCH 的频率范围是 2~4MHz，可使用寄存器 IHCFGH、IHCFGL 设置不同频率，也可用内置 RC 校正模块以其他时钟源为参考时钟进行自动校正，校正精度可达 1%，出厂频率校正为 3.6864MHz@3.3V/25℃。

备注：
1.由于芯片与芯片之间制造工艺的差异，IHCFGH、IHCFGL 设置相同的值每个芯片对应的 IRCH 频率不一定相同，IRCL 和 TFRC 也是同样情况。
2.IRCH 在不设置为系统时钟时，IRCH 的频率会在原基础上快约 3%，由于 PLL 的参考时钟为 IRCH，在设置 PLL 为系统时钟时，PLL 的时钟频率也会同步变快，所以设置 PLL 为系统时钟前，须对 IRCH 进行校正，具体可参照本章节例程。

9.1.3 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。

备注：*硬件设计时晶振负载电容地尽量靠近芯片 GND 引脚。*

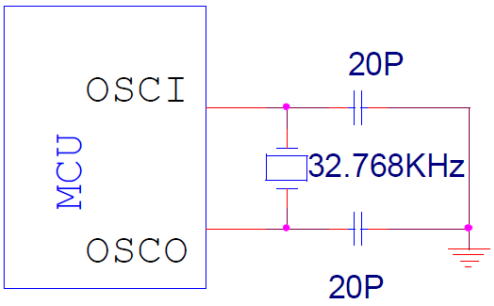


图 9-1-3-1 XOSCL 典型电路图

9.1.4 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。和 XOSCL 类似，IRCL 设为系统时钟也可实现系统低功耗。在没有外挂 32.768KHz 晶振的情况下，IRCL 也可设置为 RTC 模块时钟源，应用于对计时精度不高的场合。可使用寄存器 ILCFGH、ILCFGL 设置不同频率，也可用内置 RC 校正模块以其他时钟源为参考时钟进行自动校正，出厂频率校正为 131KHz@3.3V/25℃。

9.1.5 内置 4MHz RC 振荡器（TFRC）

TFRC 可通过寄存器 CKCON 的 TFCKE 位打开或关闭。TFRC 主要是作为 Flash 工作时钟和触摸模块充放电时钟，可使用寄存器 TFCFG 设置不同频率，也可用内置 RC 校正模块以其他时钟源为参考时钟进行自动校正，

出厂频率校正为 4MHz@3.3V/25℃。

9.1.6 PLL

内置倍频倍数为 2~10 倍的 PLL 以 IRCH 为参考时钟进行倍频，在不需外挂高速晶振的情况下利用 PLL 也可实现芯片的高速运行。PLL 通过寄存器 PLLCON 可以设置 PLL 的开关及倍频倍数，要注意的是，PLL 打开后必须等待 PLL 时钟稳定才可以将其设置为系统时钟或其他外设时钟，PLL 时钟是否稳定可通过寄存器 PLLCON 的 PLSTA 位判断。PLL 打开后，大约需要约 50us 才能达到稳定状态。

备注：由于CPU 的最高频率为24MHz，当 PLL 频率超过 24MHz 时不可设置为CPU 时钟，但是可以设为其他外设的时钟。

9.1.7 外部高速晶体谐振器（XOSCH）和外部高速 RC 振荡器（ERC）

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

备注：硬件设计时晶振负载电容地尽量靠近芯片 GND 引脚。

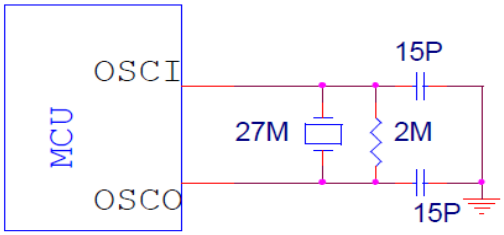


图 9-1-7-1 外部高速晶振典型电路图

9.2 时钟控制寄存器描述

表 9-2-1 寄存器 CKCON

8080H	7	6	5	4	3	2	1	0
CKCON	ILCKE	IHCKE	TFCKE	XHCS	XLCKE	XLSTA	XHCKE	XHSTA
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	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	TFCKE	TFRC 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，如果其他模块选择了该时钟源，该时钟仍然会被打开。*
4	XHCS	XOSCH 时钟源选择位 0：XOSCH 时钟源为外挂高速晶振 1：XOSCH 时钟源为外挂 RC 振荡器
3	XLCKE	XOSCL 使能控制位 1：打开 0：关闭 *备注：* *1 该位为 1 时，时钟模块打开，但是该位为 0 时，如果其他模块选择了该时钟源，该时钟仍然会被打开。* *2 由于 XOSCL 为外部时钟，所以启动 XOSCL 还需要把对应的引脚功能配置为 XOSCL 的功能。*
2	XLSTA	XOSCL 时钟稳定信号标志，1 有效
1	XHCKE	XOSCH 使能控制位 1：打开 0：关闭 *备注：* *1 位有效时，时钟模块开启，但是该位为 0 时，如果系统或者其他模块选择了该时钟源，该时钟仍然会被打开。* *2 由于 XOSCH 为外部时钟，所以若想所以启动 XOSCH 还需要把对应的引脚功能配置为 XOSCH 的功能。*
0	XHSTA	XOSCH 时钟稳定信号标志，1 有效

表 9-2-2 寄存器 PLLCON

AEH	7	6	5	4	3	2	1	0
PLLCON	PLLON	MULFT[3:0]				-	-	PLSTA
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	PLLON	PLL 使能控制位 0：关闭 1：打开
6~3	MULFT	PLL 倍频倍数设置位 0000:2 倍 0001:3 倍 0010:4 倍 0011:5 倍 0100:6 倍 0101:7 倍 0110:8 倍 0111:9 倍 1000:10 倍 其他值：无效
2~1	-	-
0	PLSTA	PLL 时钟稳定标志位，1 表示已稳定

表 9-2-3 寄存器 IHCFGL、IHCFGH

8083H	7	6	5	4	3	2	1	0
IHCFGL	IHCFG[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
8084H	7	6	5	4	3	2	1	0
IHCFGH	IHCFG[15:8]							
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						
15~0	IHCFG	IRCH 频率校正寄存器						

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	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
8086H	7	6	5	4	3	2	1	0
ILCFGH	-	-	-	-	-	-	-	ILCFG[8]
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
8~0	ILCFG	IRCL 频率校正寄存器

表 9-2-5 寄存器 TFCFG

8087H	7	6	5	4	3	2	1	0
TFCFG	TFCFG[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	TFCFG	TFRC 频率校正寄存器						

9.3 系统时钟

CA51F2 系列芯片的所有时钟源都可以设置为系统时钟。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。

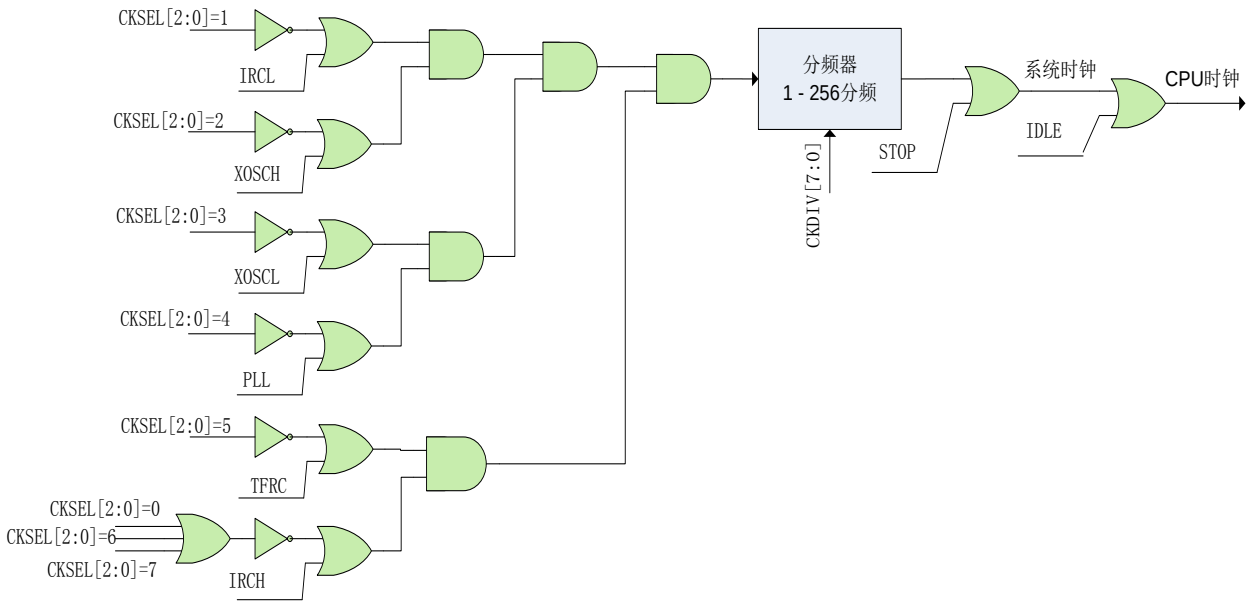


图 9-3-1 系统时钟结构图

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：IRCL 010：XOSCH/ERC 011：XOSCL 100：PLL 101：TFRC 其他：IRCH

表 9-3-2-2 寄存器 CKDIV

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

9.3.3 系统时钟控制方法及例程

◆ **设置系统时钟为 IRCH**

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;        //设置时钟为 IRCH
}
---------------------------------------------------------------------------------
```

◆ **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 3
void Sys_Clk_Set_XOSCL(void)
{
    CKCON |= XLCKE;             //打开 XOSCL 时钟
    while(!(CKCON & XLSTA)); //等待 XOSCL 时钟稳定
    CKSEL = (CKSEL&0xF8) | CKSEL_XOSCL;//设置时钟为 XOSCL
}
---------------------------------------------------------------------------------
```

◆ **设置系统时钟为 IRCL**

设置系统时钟为 IRCL，程序如下：

```
---------------------------------------------------------------------------------
#define ILCKE       (1<<7)
#define CKSEL_IRCL    1
void Sys_Clk_Set_IRCL(void)
{
    CKCON |= ILCKE;       //打开 IRCL 时钟
    CKSEL = (CKSEL&0xF8) | CKSEL_IRCL;   //设置时钟为 IRCL
}
---------------------------------------------------------------------------------
```

◆ **设置系统时钟为 PLL**

设置系统时钟为 PLL，程序如下：

```
--------------------------------------------------------------------------------
#define ILCKE           (1<<7)
#define IHCKE           (1<<6)
#define TFCKE           (1<<5)

//寄存器 RCCON 定义
#define MODE(N)         (N<<6)      //N=0~3
#define MSEX(N)             (N<<5)    //N=0~1
#define HMSK            (1<<4)
#define CKSS(N)             N         //N=0~11

#define PLLON(N)        (N<<7)          //N=0~1
#define MULFT(N)        (N<<3)          //N=0~8
#define PLSTA           (1<<0)
#define CKSEL_PLL       4
#define CKSEL_TFRC      5
void Sys_Clk_Set_PLL(unsigned char Multiple)     //Multiple: 倍频倍数
{
    static bit NeedTrim = 1;
    unsigned char ck_bak;

    if(Multiple < 2 || Multiple > 8) return;
    if(NeedTrim)
    {
        ck_bak = CKCON;
        CKCON |= ILCKE|IHCKE|TFCKE;
        VCKDH = 0xAF;
        VCKDL = 0xC7;
        RCCON = MODE(2) | MSEX(0) | CKSS(10);
        while(RCCON&0xC0);

        RCTAGH = RCMSLH;
        RCTAGL = RCMSLL;
        CKSEL = (CKSEL&0xF8) | CKSEL_IRCL;
        IHCFGH=0x07;
        IHCFGL=0xFF;
        RCCON = MODE(3) | MSEX(0) | CKSS(10);
        while(RCCON&0xC0);
    }

    PLLCON = PLLON(1) | MULFT(Multiple-2);
    while(!(PLLCON & PLSTA));
    CKSEL = (CKSEL&0xF8) | CKSEL_PLL;
    if(NeedTrim)
    {
```

```
            CKCON = ck_bak | IHCKE;
            NeedTrim = 0;
        }
    }
```

◆ **设置系统时钟为 XOSCH**

设置系统时钟为 XOSCH，程序如下：

```
#define XHCKE           (1<<1)
#define XLSTA           (1<<2)
#define CKSEL_XOSCH     2
void Sys_Clk_Set_XOSCH(void)
{
    P74F = 3; //设置 P74 为 XOSCH 引脚
    P73F = 3; //设置 P73 为 XOSCH 引脚
    CKCON |= XHCKE;     //打开 XOSCH 时钟
    while(!(CKCON & XHSTA));/等待 XOSCH 时钟稳定
    CKSEL = (CKSEL&0xF8) | CKSEL_XOSCH;     //设置时钟为 XOSCH
}
```

◆ **设置系统时钟为 TFRC**

设置系统时钟为 TFRC，程序如下：

```
#define TFCKE           (1<<5)
#define CKSEL_TFRC  5
void Sys_Clk_Set_TFRC(void)
{
    CKCON |= TFCKE;
    CKSEL = (CKSEL&0xF8) | CKSEL_TFRC;
}
```

9.4 内部 RC 振荡器校正

9.4.1 校正模块介绍

由于工艺的偏差，每个芯片的 RC 振荡器的频率出厂时都不一定相同，所以芯片出厂后有必要对 RC 振荡器进行校正。CA51F2 系列芯片内置自动校正模块，可以对内部 RC 振荡器进行校正。校正模块共有 5 个参考时钟源选择：IRCH、IRCL、XOSCL、XOSCH、TFRC，有 3 个目标时钟选择：IRCH、IRCL、TFRC。选择参考时钟源时必须保证参考时钟源的频率是准确的，否则校正后的目标时钟也相应的不准确。图 9-4-1 为校正模块电路示意图，其中 VCLK 为参考时钟，TCLK 为目标时钟。

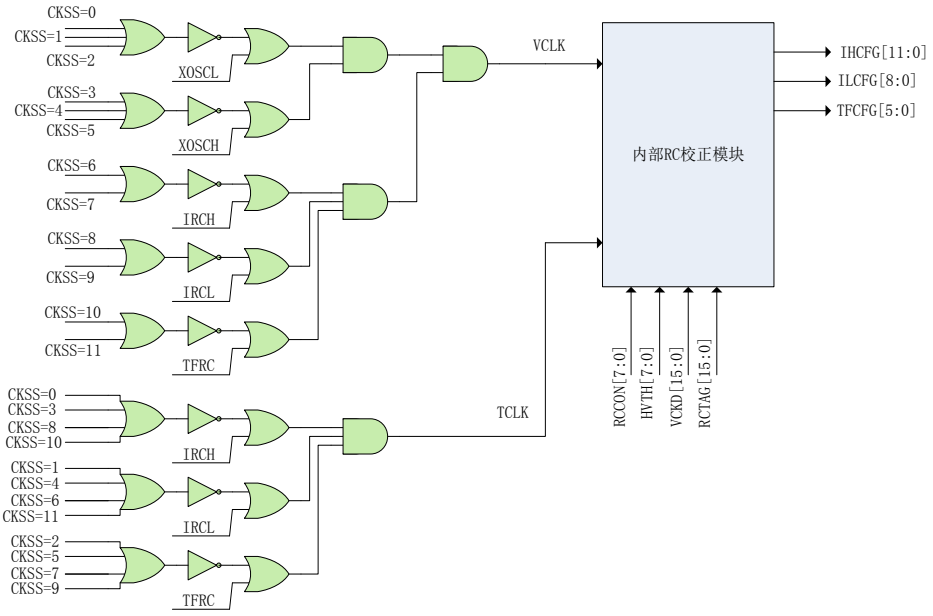


图 9-4-1 校正电路示意图

RC 校正模块共有 3 种工作模式：

- **计数模式**

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

- **测量模式**

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

TCLK 的周期＝ (VCLK 的周期 x VCKD) ÷ RCMS

- **校正模式**

校正模式通过二分法，把若干个 VCLK 时钟周期内 TCLK 的计数值和设定值（对应目标频率）不断进行比对，比对达到用户设置的差值的阈值 HVTH 时产生 HMSK 标志。用户可以通过设置 MSE 位来选择 HMSK 位置 1 时即停止校正或二分法拆分到最后才停止。用户设置寄存器 HVTH 的原则是根据用户对 TCLK 时钟频率误差范围接受程度来设置，HVTH 越小，校正精度越高，但校正时间可能会更长。设置 MODE 为 3 启动校正，校正结束后，MODE 自动清 0。和测量模式一样，单个测量周期应尽量延长，以提高校正精度，但是校正时间也会相应延长。VCLK 和 TCLK 之间的关系是：

VCLK 的周期 x VCKD ＝ TCLK 的目标周期 x RCTAG

9.4.2 校正模块控制寄存器

表 9-4-2-1 寄存器 RCCON

91H	7	6	5	4	3	2	1	0
RCCON	MODE[1:0]		MSE	HMSK	CKSS[3:0]			
R/W	R/W		R	R	R/W			
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~6	MODE	工作模式选择位 01：计数模式，设置 MODE 为 0 则退出计数模式 10：测量模式，完成后 MODE 自动清 0 11：校正模式，完成后 MODE 自动清 0
5	MSE	校正模式下退出方式选择位 0：即使检测到 HMSK 被置 1，也等到校正次数足够才退出 1：检测到 HMSK 被置 1 后直接退出
4	HMSK	在校正模式下，校正值与目标值差值小于等于 HVTH，HMSK 将被置 1，否则清 0
3~0	CKSS	时钟配对选择位 0000：目标时钟 IRCH，参考时钟 XOSCL 0001：目标时钟 IRCL，参考时钟 XOSCL 0010：目标时钟 TFRC，参考时钟 XOSCL 0011：目标时钟 IRCH，参考时钟 XOSCH 0100：目标时钟 IRCL，参考时钟 XOSCH 0101：目标时钟 TFRC，参考时钟 XOSCH 0110：目标时钟 IRCL，参考时钟 IRCH 0111：目标时钟 TFRC，参考时钟 IRCH 1000：目标时钟 IRCH，参考时钟 IRCL 1001：目标时钟 TFRC，参考时钟 IRCL 1010：目标时钟 IRCH，参考时钟 TFRC 1011：目标时钟 IRCL，参考时钟 TFRC 其他值：无效

表 9-4-2-2 寄存器 HVTH

AFH	7	6	5	4	3	2	1	0
HVTH	HVTH[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	HVTH	校正模式下，校正值与目标值差值阈值寄存器						

表 9-4-2-3 寄存器 VCKDL、VCKDH

92H	7	6	5	4	3	2	1	0
VCKDL	VCKD[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
93H	7	6	5	4	3	2	1	0
VCKDH	VCKD[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
15~0	VCKD	测量和校正模式下，参考时钟分频倍数，为 VCKD 倍分频(VCKD>1)						

表 9-4-2-4 寄存器 RCTAGL、RCTAGH

94H	7	6	5	4	3	2	1	0
RCTAGL	RCTAGL[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
95H	7	6	5	4	3	2	1	0
RCTAGH	RCTAGH[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
15~0	RCTAG	校正模式下，目标时钟分频倍数，为 RCTAG 倍分频(RCTAG>=1)						

表 9-4-2-5 寄存器 RCMSLL、RCMSLH、RCMSHL、RCMSHH

96H	7	6	5	4	3	2	1	0
RCMSLL	RCMS[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
97H	7	6	5	4	3	2	1	0
RCMSLH	RCMS[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
9EH	7	6	5	4	3	2	1	0
RCMSHL	RCMS[23:16]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
9FH	7	6	5	4	3	2	1	0
RCMSHH	RCMS[31:24]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
31~0	RCMS	计数模式完成后，存放计数结果 测量模式完成后，存放测量结果 校正模式下，RCMS 的值无意义

9.4.3 校正模块控制例程

◆ **校正 IRCH 例程**

参考时钟设为 XOSCL，目标时钟设为 IRCH，校正目标频率为 3.6864MHz,程序如下：

```
--------------------------------------------------------------------------------
#define IHCKE           (1<<6)
#define XLCKE           (1<<3)
#define XLSTA           (1<<2)

#define MODE(N)         (N<<6)      //N=0~3
#define MSEX(N)         (N<<5)      //N=0~1
#define CKSS(N)         N           //N=0~11

CKCON |= IHCKE;       //打开 IRCH 时钟
CKCON |= XLCKE;       //打开 XOSCL 时钟
while(!(CKCON & XLSTA)); //等待 XOSCL 时钟稳定
RCTAGH =    ((3686400*400)/32768)/256;  //设置目标频率
RCTAGL =    ((3686400*400)/32768)%256;
VCKDH = 400/256;                        //设置参考时钟分频，分频后的频率为 81.92HZ
VCKDL = 400%256;
RCCON = MODE(3) | MSEX(0) | CKSS(0);  //设置目标时钟为 IRCH，参考时钟为 XOSCL，设置校正方式，
                                       //启动校正模式
while(RCCON&0xC0);  //等待校正完成
--------------------------------------------------------------------------------
```

◆ **校正 IRCL 例程**

参考时钟设为 XOSCL，目标时钟设为 IRCL，校正目标频率为 131KHz,程序如下：

```
--------------------------------------------------------------------------------
#define ILCKE       (1<<7)
#define XLCKE           (1<<3)
#define XLSTA           (1<<2)

#define MODE(N)         (N<<6)      //N=0~3
#define MSEX(N)         (N<<5)      //N=0~1
#define CKSS(N)         N           //N=0~11

CKCON |= ILCKE;                 //打开 IRCL 时钟
CKCON |= XLCKE;                 //打开 XOSCL 时钟
while(!(CKCON & XLSTA)); //等待 XOSCL 时钟稳定
RCTAGH =    ((131000*400)/32768)/256;   //设置目标频率
RCTAGL =    ((131000*400)/32768)%256;
VCKDH = 400/256;                        //设置参考时钟分频，分频后的频率为 81.92HZ
VCKDL = 400%256;
```

```
RCCON = MODE(3) | MSEX(0) | CKSS(1);   //设置目标时钟为 IRCL，参考时钟为 XOSCL，设置校正方式，
                                       //启动校正模式
while(RCCON&0xC0);   //等待校正完成
--------------------------------------------------------------------------------
```

◆ **校正 TFRC 例程**

参考时钟设为 XOSCL，目标时钟设为 TFRC，校正目标频率为 4MHz,程序如下：

```
--------------------------------------------------------------------------------
#define TFCKE           (1<<5)
#define XLCKE           (1<<3)
#define XLSTA           (1<<2)

#define MODE(N)         (N<<6)          //N=0~3
#define MSEX(N)         (N<<5)          //N=0~1
#define CKSS(N)         N               //N=0~11

CKCON |= TFCKE;                 //打开 TFRC 时钟
CKCON |= XLCKE;                 //打开 XOSCL 时钟
while(!(CKCON & XLSTA)); //等待 XOSCL 时钟稳定
RCTAGH =     ((4000000*400)/32768)/256;  //设置目标频率
RCTAGL =     ((4000000*400)/32768)%256;
VCKDH = 400/256;                        //设置参考时钟分频，分频后的频率为 81.92HZ
VCKDL = 400%256;
RCCON = MODE(3) | MSEX(0) | CKSS(2);   //设置目标时钟为 TFRC，参考时钟为 XOSCL，设置校正方式，
                                       //启动校正模式
while(RCCON&0xC0);   //等待校正完成
--------------------------------------------------------------------------------
```

9.5 外部时钟监控

9.5.1 功能描述

时钟监控模块用于监控外部时钟的异常，并且作出处理，增强了系统的可靠性。开启时钟监控模块后，当外部时钟作为系统时钟时，如果外部时钟停振，系统时钟将切换到 IRCL。当外部时钟作为 RTC 时钟时，如果外部时钟停振，RTC 时钟将切换到 IRCL 四分频时钟。当外部时钟作为 WDT 时钟时，如果外部时钟停振，WDT 时钟将切换到 IRCL 四分频时钟。

外部高速时钟（XOSCH）监控通过 MHE 位使能，而中断使能通过 IHE 位设置。当 XOSCH 时钟出现异常后，时钟异常标志位 XHFD 置 1，如果设置 ATH=1，当 XOSCH 时钟恢复正常后，时钟源会自动恢复回 XOSCH；当 ATH=0，清除中断标志位 XHFD 就会恢复时钟源为 XOSCH。

外部低速时钟（XOSCL）监控通过 MLE 位使能，而中断使能通过 ILE 位设置。当 XOSCL 时钟出现异常后，时钟异常标志位 XLFD 置 1，如果设置 ATL=1，当 XOSCL 时钟恢复正常后，时钟源会自动恢复回 XOSCL；当 ATH=0，清除中断标志位 XLFD 就会恢复时钟源为 XOSCL。

标志位 HSP 和 LSP 分别指示 XOSCH 和 XOSCL 的当前状态，当 HSP=1 或 LSP=1，分别表示 XOSCH 和 XOSCL 出现异常，时钟已切换到内部 RC 时钟。

系统在 STOP 或 IDLE 模式也可通过时钟监控中断唤醒。

9.5.2 外部时钟监控控制寄存器

表 9-5-2-1 寄存器 CKMON

BEH	7	6	5	4	3	2	1	0
CKMON	MHE	IHE	ATH	HSW	MLE	ILE	ATL	LSW
R/W	R/W	R/W	R/W	W	R/W	R/W	R/W	W
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7	MHE	时钟监控模块监控 XOSCH 时钟控制位，1 有效
6	IHE	时钟监控 XOSCH 中断使能位，1 有效
5	ATH	时钟监控模块自动恢复为 XOSCH 使能位，1 有效 备注： 模块检测到 XOSCH 异常后，将会把对应电路的工作时钟切换为 IRCL，如果 ATH 位为 1，那么在 XOSCH 恢复正常之后，模块将会自动把对应电路的工作时钟切换回 XOSCH。如果 ATH 位为 0，则必须用户对 HSW 写 1 才能切换回 XOSCH。
4	HSW	只写寄存器，对其写 1 将清除 HSP（CKMIF[7]）
3	MLE	时钟监控模块监控 XOSCL 时钟控制位，1 有效
2	ILE	时钟监控 XOSCL 中断使能位，1 有效
1	ATL	时钟监控模块自动恢复为 XOSCL 使能位，1 有效 备注：

		模块检测到 XOSCL 异常后，将会把对应电路的工作时钟切换为 IRCL，如果 ATL 位为 1，那么在 XOSCL 恢复正常之后，模块将会自动把对应电路的工作时钟切换回 XOSCL。如果 ATL 位为 0，则必须用户对 LSW 写 1 才能切换回 XOSCL。
0	LSW	只写寄存器，对其写 1 将清除 LSP（CKMIF[6]）

表 9-5-2-2 寄存器 CKMIF

BFH	7	6	5	4	3	2	1	0
CKMIF	HSP	LSP	-	-	-	-	XHFD	XLFD
R/W	R	R	-	-	-	-	R	R
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7	HSP	XOSCH 异常，时钟切换到内部时钟的状态标志位，1 表示工作于内部时钟
6	LSP	XOSCL 异常，时钟切换到内部时钟的状态标志位，1 表示工作于内部时钟
5~2	-	-
1	XHFD	监控 XOSCH 异常中断标志，1 有效，写 1 清 0
0	XLFD	监控 XOSCL 异常中断标志，1 有效，写 1 清 0

10 供电和复位系统

10.1 供电系统

在 CA51F2 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。

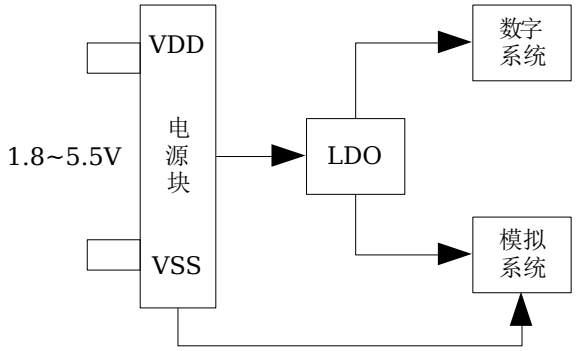


图 10-1-1 供电系统示意图

图 10-1-2 为芯片供电典型电路图。

备注：以下电路中，滤波电容 47uF 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。

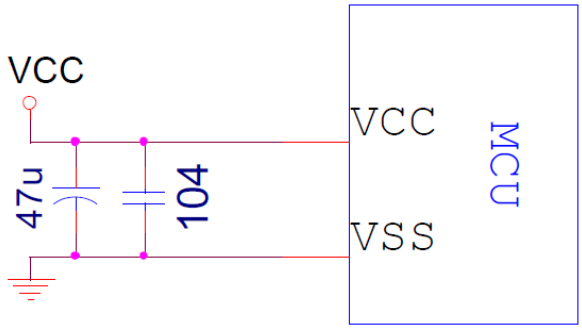


图 10-1-2 芯片供电典型电路图

10.1.1 LDO 功能简介

CA51F2 系列芯片有一个内置的低压差线性稳压器（LDO）。LDO 模块为芯片提供核心电压。LDO 的输出电压通过 VLEVEL 位（PWCON[2:0]）设置，VLEVEL 默认值为 3，对应的输出电压为 1.58V。当 VDD/VSS 小于 VLEVEL 位设定的输出电压时，LDO 直接输出 VDD；当 VDD/VSS 大于设定电压时，LDO 输出设定的电压。

LDO 设置高电压有助于时钟模块快速启动，而设置为较低电压时有助于降低芯片功耗。LDO 有两种不同的工作模式：高功率模式和低功率模式，通过 VHL 位（PWCON[3]）设置。两种模式下，LDO 的负载能力不相同。在高功率模式，LDO 能提供的电流更大，但自身功耗也更大，低功率模式则反之。当系统处于正常工作状态时，LDO 一般都设置为高功率模式，而低功率模式一般应用于省电模式，例如 STOP、IDLE、低速运行模式等。

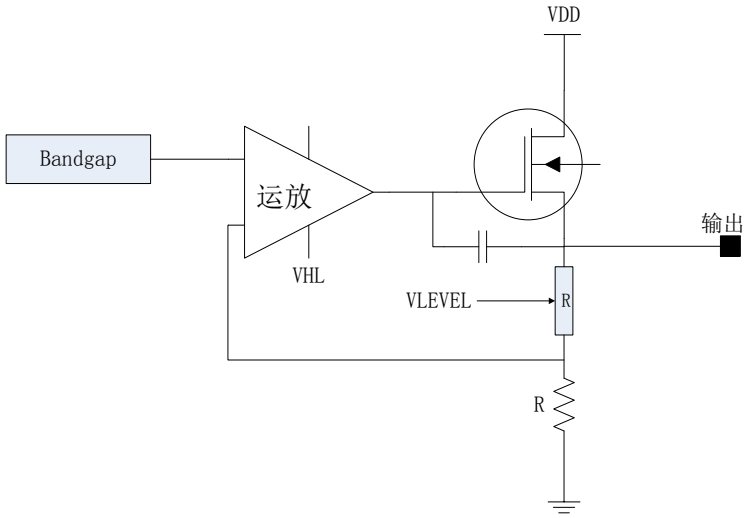


图 10-1-1 LDO 模块示意图

10.1.2 LDO 控制寄存器

表 10-1-2-1 寄存器 PWCON

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

位编号	位符号	说明
7	-	-
6~4	FLEVEL	内部基准电压（Bandgap）输出调整位域
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 100：1.65V 011：1.58V（默认） 010：1.50V 001：1.42V 000：1.35V *备注：LDO 输出电压不建议设置 1.58V 以下，否则有可能导致芯片工作异常。*

10.2 复位系统

CA51F2 系列芯片有多个内部和外部复位源，如图 10-2-1 所示。

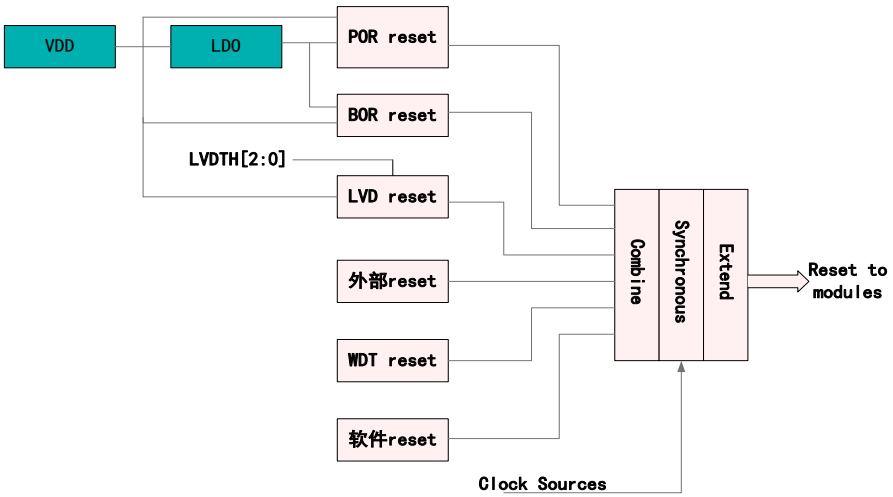
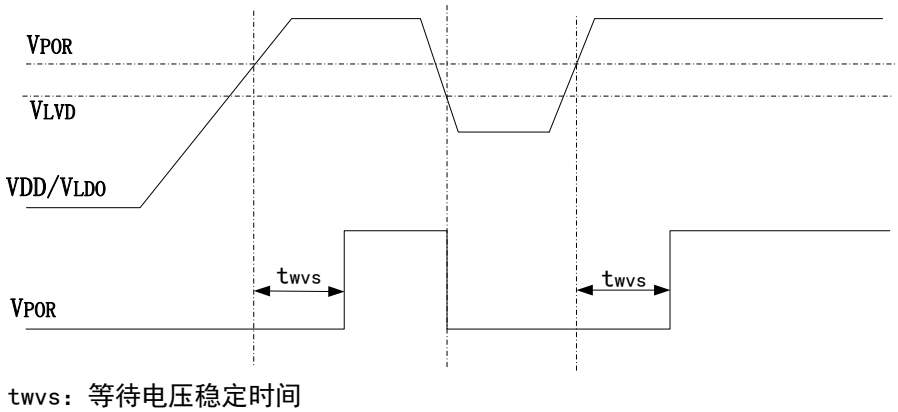


图 10-2-1 复位系统结构图

- **上电复位（POR）**

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

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



twvs：等待电压稳定时间

图 10-2-2 上电复位电路示例及上电过程

- **掉电复位（BOR）**

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

- **低电压复位**

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

过 20us 就可以产生复位信号（前提是 LVD 设置为复位模式）。

- **外部复位**

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

- **看门狗复位**

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

- **软复位**

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

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

11 功耗管理

CA51F2 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 5uA，低速运行时功耗小于 15uA。

11.1 IDLE 模式

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

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

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

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

11.2 STOP 模式

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

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

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

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

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

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

11.3 低速运行模式

由于芯片的功耗与运行速度直接相关，所以把主时钟切换到低速时钟运行也可以显著降低功耗。系统支持两个低速时钟源：IRCL、XOSCL。系统时钟设为 XOSCL 时的电流小于 9uA，而设为 IRCL 时的电流小于 22uA.

关于 IRCL、XOSCL 更多介绍请参考时钟系统章节。

11.4 低功耗相关寄存器描述

表 14-4-1 寄存器 PCON

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

位编号	位符号	说明
7	SMOD	UART0 波特率倍频控制位 在 UART0 工作于模式 1,2,3 时，设置 SMOD=1 会使波特率倍频，与标准 8051 相同。
6	-	-
5	SWRST	软复位控制位，1 有效 设置 SWRST=1 产生软复位，复位产生后自动清 0。
4	-	-
3	TSME	调试模式控制位 0：禁止进入调试模式 1：允许进入调试模式 备注：调试模式用于芯片在线仿真
2	TSMODE	调试模式标志位，为 1 表示芯片正工作于调试模式
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 14-4-2 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	SMINT/PWMINT/EPIF[7]	IDLE 模式时，SAMPLE/PWM/外部中断 9 的中断状态
13	RTCINT/CTMINT/EPIF[6]	IDLE 模式时，RTC/比较器计数器/外部中断 8 的中断状态
12	MOTINT/WDFLG[1]/EPIF[5]	IDLE 模式时，MOTOR/WDT/外部中断 7 的中断状态
11	I2CINT/CPINT/EPIF[4]	IDLE 模式时，I2C/模拟比较器/外部中断 6 的中断状态
10	SPINT/CKMINT/EPIF[3]	IDLE 模式时，SPI/时钟监控/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/U2INT/EPIF[1]	IDLE 模式时，TK/UART2/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	U0INT	IDLE 模式时，UART0 的中断状态
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 14-4-2 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	STOP 模式时，I2C 的中断状态
12	CKMWKF	STOP 模式时，时钟监控的中断状态
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)
{
    I2CCON = 0;  //关闭 I2C 功能，因为 I2C 默认是使能的，如果 I2C 不关闭将无法关闭 IRCH 时钟
    CKCON = 0;  //关闭所有时钟
    PWCON &=0xF7; //设置 LDO 进入低功率模式
    MECON |= (1<<6); //设置 FLASH 进入深度睡眠状态
    while(STPSTH|STPSTL);//如果有中断未响应,等待中断被响应
    PCON |= 0x02;       //进入 STOP 模式
    _nop_();
    _nop_();
    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)
{
    I2CCON = 0;  //关闭 I2C 模块，因为 I2C 默认是使能的，如果 I2C 不关闭将无法关闭 IRCH 时钟
    CKCON = 0;  //关闭除主时钟外的所有时钟

    Sys_Clk_Set_XOSCL(); //切换主时钟到 XOSCL，见备注说明
    //Sys_Clk_Set_IRCL();    //切换主时钟到 IRCL，见备注说明

    PWCON &=0xF7; //设置 LDO 进入低功率模式
    MECON |= (1<<6); //设置 FLASH 进入深度睡眠状态
    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)
{
    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
    Sys_Clk_Set_XOSCL(); //切换主时钟到 XOSCL，见备注说明
    //Sys_Clk_Set_IRCL();  //切换主时钟到 IRCL，见备注说明
    CKCON = 0;  //关闭除主时钟外的所有时钟
    PWCON &=0xF7; //设置 LDO 进入低功率模式
}
```

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

---

12 通用定时器（定时器 0, 定时器 1, 定时器 2）

12.1 定时器 0

12.1.1 定时器 0 介绍

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

- **模式 0**

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

- **模式 1**

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

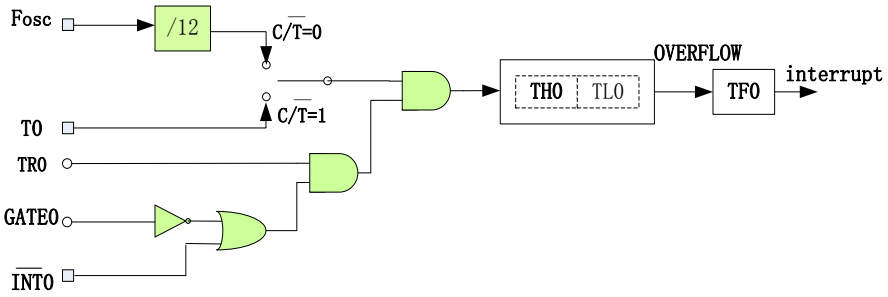


图 12-1-1-1 定时器 0 的模式 0 与 1

- **模式 2**

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

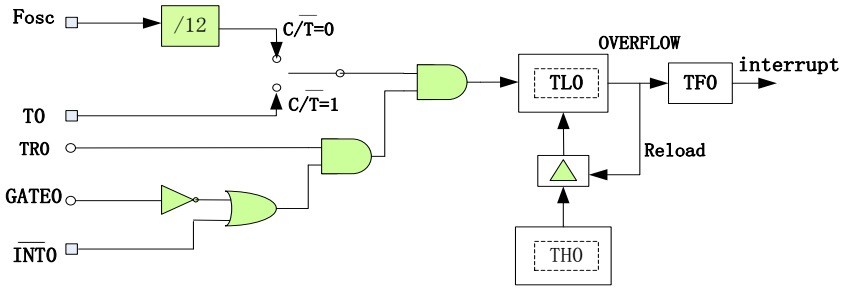


图 12-1-1-2 定时器 0 的模式 2

- **模式 3**

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

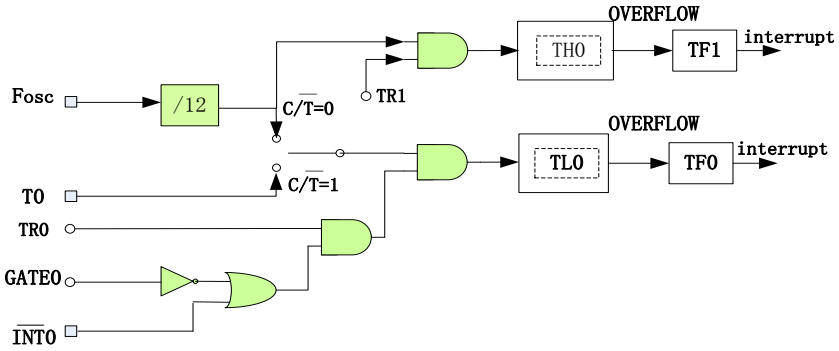


图 12-1-1-3 定时器 0 的模式 3

12.1.2 定时器 0 寄存器描述

表 12-1-2-1 寄存器 TCON

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

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

表 12-1-2-2 寄存器 TMOD

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

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

表 12-1-2-3 寄存器 TL0

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

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

表 12-1-2-4 寄存器 TH0

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

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

12.2 定时器 1

12.2.1 定时器 1 介绍

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

- **模式 0**

  此模式保留，不能使用。

- **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。

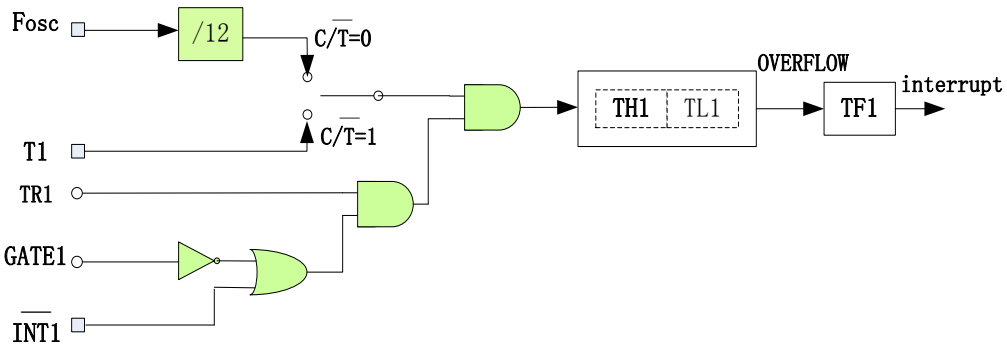


图 12-2-1 定时器 1 的模式 1

- **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。

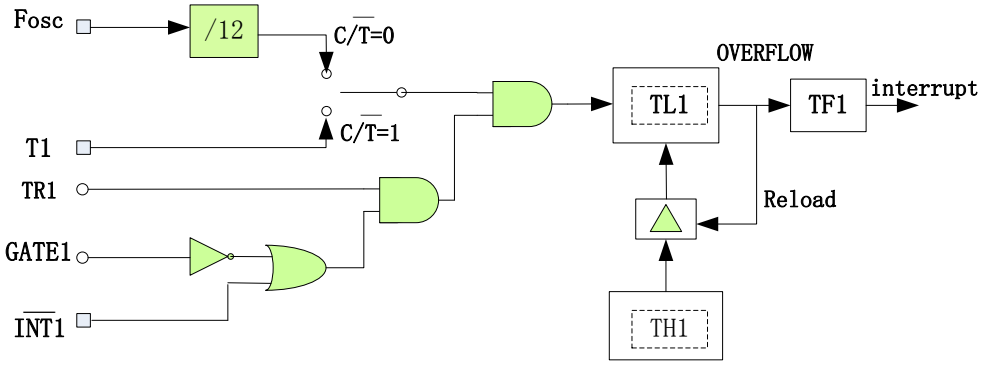


图 12-2-2 定时器 1 的模式 2

- **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。

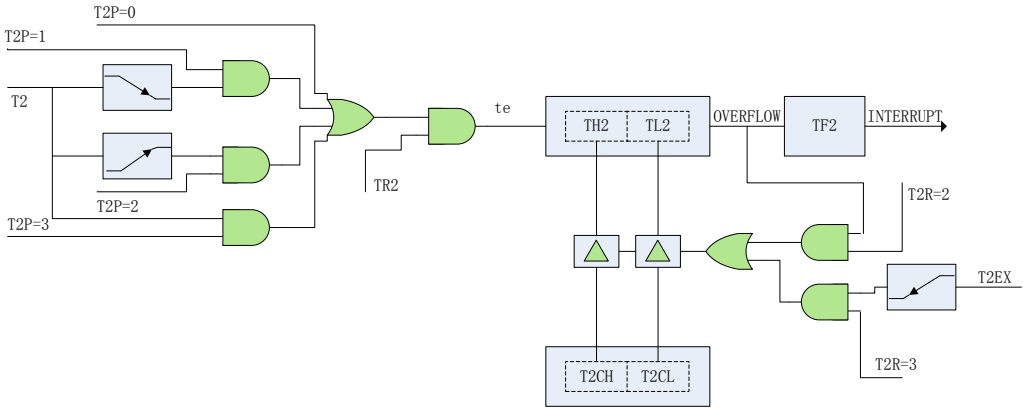


图 12-3-1-1 定时器 2 的重载模式

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。

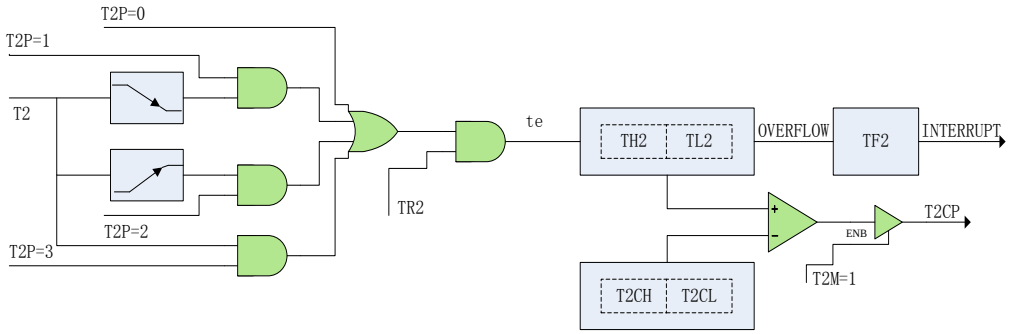


图 12-3-1-2 定时器 2 的比较模式

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。

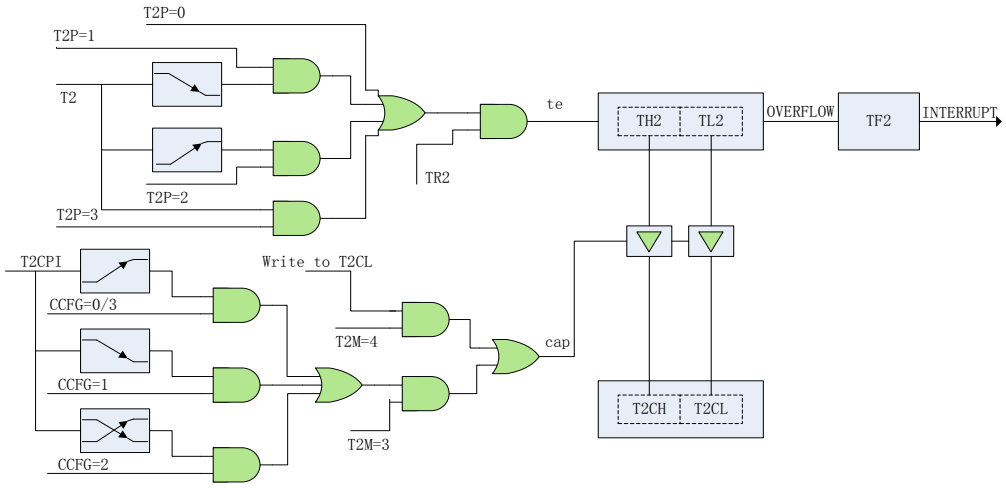


图 12-3-1-2 定时器 2 的抓取模式

12.3.2 定时器 2 寄存器描述

表 12-3-2-1 寄存器 T2CON

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

位编号	位符号	说明
7	-	-
6	TR2	定时器 2 运行控制位，1 有效
5	T2R1	[T2R1,T2R0]是定时器 2 重载模式选择位 10：模式 0 11：模式 1 其他：重载功能关闭
4	T2R0	
3	T2IE	定时器 2 中断使能位，1 有效
2	UCKS	UART0 时钟选择位 0：UART0 使用定时器 1 溢出脉冲 1：UART0 使用定时器 2 溢出脉冲
1	T2P1	[T2P1,T2P0]是定时器 2 引脚 T2 功能选择位 00：定时器 2 使用内部系统时钟计数，没有使用 T2 01：定时器 2 检测 T2 下降沿计数 10：定时器 2 检测 T2 上升沿计数
0	T2P0	

		11：定时器 2 使用内部系统时钟计数，通过 T2 门控

表 12-3-2-2 寄存器 T2MOD

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

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

表 12-3-2-3 寄存器 T2CL

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

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

表 12-3-2-4 寄存器 T2CH

CBH	7	6	5	4	3	2	1	0
T2CH	T2CH							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~0	T2CH	在重载模式，T2CH 是重载值的高字节 在比较模式，T2CH 是比较值的高字节 在捕获模式，T2CH 保存捕获值的高字节

表 12-3-2-5 寄存器 TL2

CCH	7	6	5	4	3	2	1	0
TL2	TL2							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~0	TL2	定时器 2 计数值的低字节

表 12-3-2-6 寄存器 TH2

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

位编号	位符号	说明
7~0	TH2	定时器 2 计数值的高字节

13 看门狗定时器（WDT）

13.1 看门狗定时器(WDT)功能简介

看门狗定时器是一个可选时钟源的 27 位减法计数器，时钟为 3.6864MHz 下计数时间范围为 0.56ms - 36.4s，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。

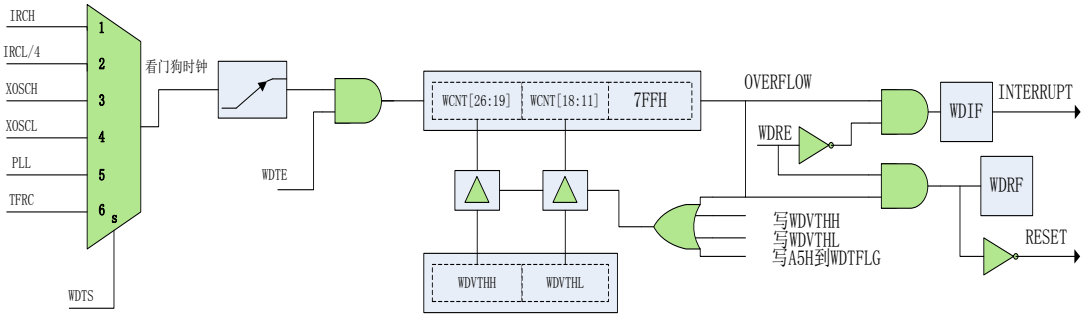


图 13-1-1 看门狗模块结构图

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[2:0]			-	-	-	-	WDRE
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
7~5	WDTS	WDT 时钟选择位 001：选择 IRCH 010：选择 IRCL 四分频 011：选择 XOSCH 100：选择 XOSCL 101：选择 PLL 110：选择 TFRC 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
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 当看门狗时钟为 3.6864M 时，覆盖范围为 0.56ms ~ 36s						

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 3.6864MHz，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 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 = 0x07;                    //设置看门狗时间为 1 秒
    WDVTHL = 0x08;
    WDFLG = 0xA5;                     //刷新看门狗
}
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 3.6864MHz，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 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 = 0x07;                    //设置看门狗时间为 1 秒
    WDVTHL = 0x08;
    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。

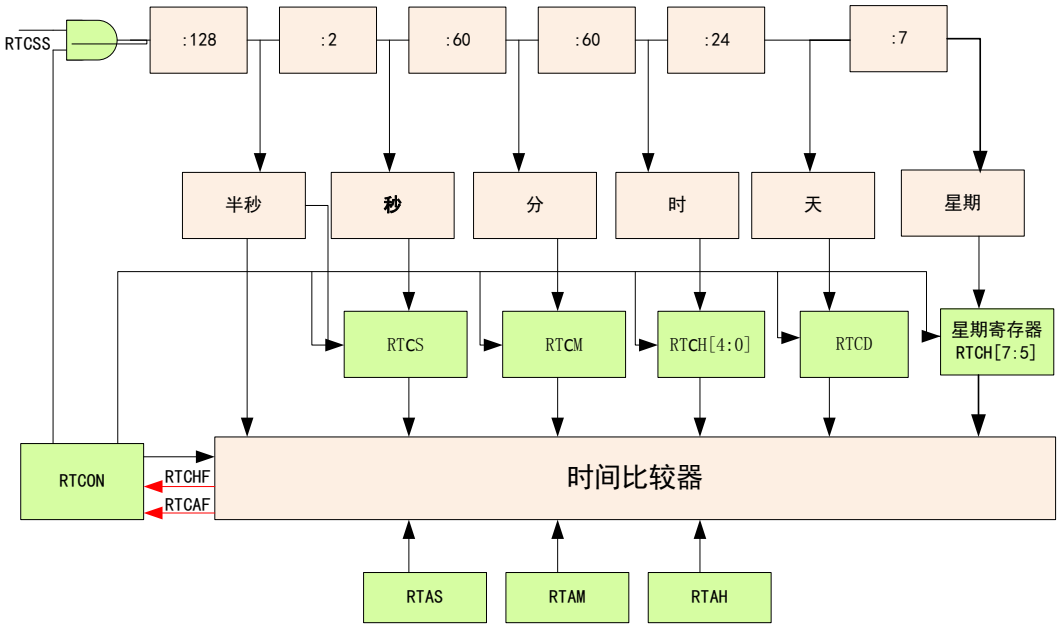


图 14-1-1　RTC 结构图

- **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：

```
-------------------------------------------------------------------------------
#define RTCE        (1<<7)
void RTC_WriteHour(unsigned char hour)     //hour=0~23
{
    RTCON |= RTCWE; //写入时间使能
    RTCH = hour;    //写入小时值
    Delay_50us();   //必须延时 50 微秒
    RTCON &= ~RTCWE;//写入时间禁能
}
void RTC_WriteMinute(unsigned char minute)//minute=0~59
{
     RTCON |= RTCWE;//写入时间使能
     RTCM = minute;//写入分值
     Delay_50us();//必须延时 50 微秒
     RTCON &= ~RTCWE;//写入时间禁能
}
void RTC_WriteSecond(unsigned char second)//second=0~59
{
     RTCON |= RTCWE;//写入时间使能
     RTCS = second;//写入秒值
     Delay_50us();//必须延时 50 微秒
     RTCON &= ~RTCWE;//写入时间禁能
}
-------------------------------------------------------------------------------
```

◆ **RTC 设置闹钟时间**

例如，设置闹钟时间为 11:30:0，时、分、秒比较全使能，程序如下：

```
-------------------------------------------------------------------------------
#define SCE(N)          (N<<4)    //N=0~1
#define MCE(N)          (N<<3)    //N=0~1
#define HCE(N)          (N<<2)    //N=0~1
#define RTC_AF          (1<<0)
Void RTM_init(void)
{
     RTAH = 11;    //设置闹钟小时
     RTAM = 30;    //设置闹钟分
     RTAS = 0;     //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：

```
#define XLCKE           (1<<3)
#define XLSTA           (1<<2)
void RTC_init(void)
{
    CKCON |= XLCKE;             //开启 XOSCL 时钟
    while(!(CKCON & XLSTA));//等待 XOSCL 时钟稳定
    RTCON = RTCE(1) | MSE(1) | HSE(1);  //RTC 使能，毫秒中断使能，半秒中断使能
    RTC_WriteHour(10);      //写入小时
    RTC_WriteMinute(30);//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

通用输入/输出口用于芯片和外部进行数据传输，CA51F2 系列芯片最大封装有 62 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=0~7，代表 P0~P7，x=0~7，代表 Pn.0~Pn.7），用户可通过寄存器 PnxF 配置引脚的主功能和其他选项。详见寄存器部分介绍。

GPIO 的主要特性如下：

- 可配置为高阻模式
- I/O 结构可独立设置上拉下拉电阻
- 输出模式可选开漏输出或推挽输出
- 数据输出锁存支持读-修改-写
- 支持 1.8~5.5V 宽电压范围
- 设为推挽输出时，推电流大于 20mA，灌电流大于 40mA。

GPIO 推挽模式结构图如图 15-1-1 所示。

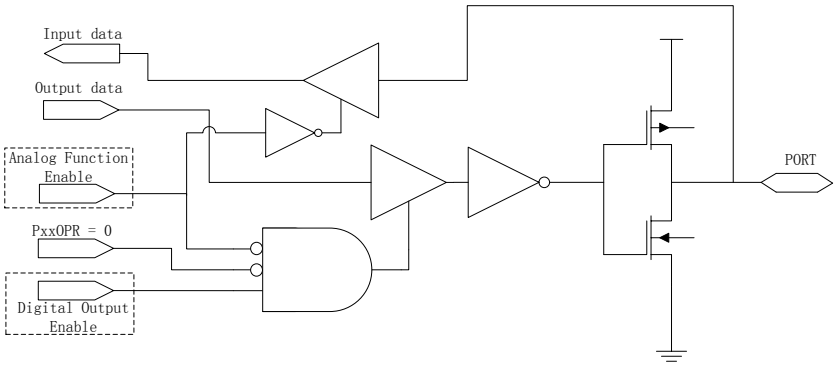


图 15-1-1 I/O 推挽模式结构示意图

GPIO 开漏模式结构图如图 15-1-2 所示。

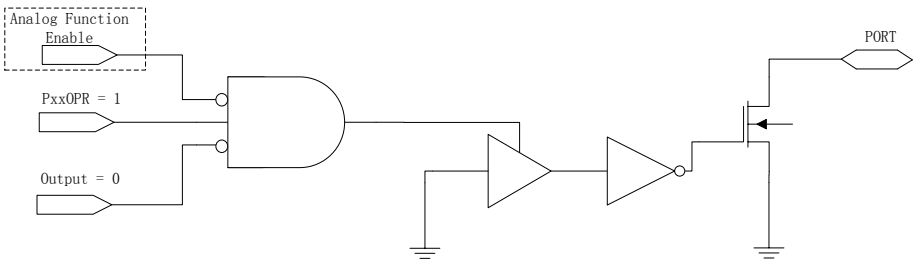


图 15-1-2 I/O 开漏模式结构示意图

GPIO 下拉结构图如图 15-1-3 所示。

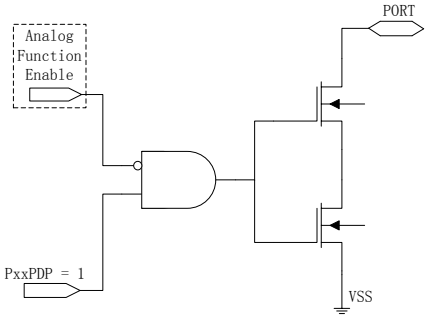


图 15-1-3 I/O 下拉模式结构示意图

GPIO 上拉结构图如图 15-1-4 所示。

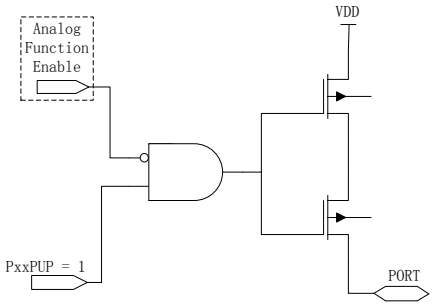


图 15-1-4 I/O 上拉模式结构示意图

15.2 引脚寄存器描述

表 15-2-1 寄存器 P0

80H	7	6	5	4	3	2	1	0
P0	P07	P06	P05	P04	P03	P02	P01	P00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
7~0	P0x	引脚 P0x 的数据寄存器，管脚功能设置为 GPIO 时有效 0：设为输入时 P0x 电平为低，设为输出时 P0x 输出低电平 1：设为输入时 P0x 电平为高，设为输出时 P0x 输出高电平						

表 15-2-2 寄存器 P1

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

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

表 15-2-3 寄存器 P2

A0H	7	6	5	4	3	2	1	0
P2	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
7~0	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
7~0	P3x	引脚 P3x 的数据寄存器，管脚功能设置为 GPIO 时有效 0：设为输入时 P3x 电平为低，设为输出时 P3x 输出低电平 1：设为输入时 P3x 电平为高，设为输出时 P3x 输出高电平

表 15-2-5 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	P57	P56	P55	P54	P53	P52	P51	P50
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	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 寄存器 P6

A9H	7	6	5	4	3	2	1	0
P6	P67	P66	P65	P64	P63	P62	P61	P60
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	P6x	引脚 P6x 的数据寄存器，管脚功能设置为 GPIO 时有效 0：设为输入时 P6x 电平为低，设为输出时 P6x 输出低电平 1：设为输入时 P6x 电平为高，设为输出时 P6x 输出高电平						

表 15-2-8 寄存器 P7

D9H	7	6	5	4	3	2	1	0
P7	-	-	P75	P74	P73	P72	P71	P70
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						
5~0	P7x	引脚 P7x 的数据寄存器，管脚功能设置为 GPIO 时有效 0：设为输入时 P7x 电平为低，设为输出时 P7x 输出低电平 1：设为输入时 P7x 电平为高，设为输出时 P7x 输出高电平						

表 15-2-9 引脚功能控制寄存器

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

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

8011H	7	6	5	4	3	2	1	0
P21F	P21PUP	P21PDP	P21OPR	-	-	P21S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0

8012H	7	6	5	4	3	2	1	0
P22F	P22PUP	P22PDP	P22OPR	-	-	P22S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0

8013H	7	6	5	4	3	2	1	0
P23F	P23PUP	P23PDP	P23OPR	-	-	P23S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0

8014H	7	6	5	4	3	2	1	0
P24F	P24PUP	P24PDP	P24OPR	-	-	P24S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0

8015H	7	6	5	4	3	2	1	0
P25F	P25PUP	P25PDP	P25OPR	-	-	P25S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0

8016H	7	6	5	4	3	2	1	0
P26F	P26PUP	P26PDP	P26OPR	-	-	P26S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0

8017H	7	6	5	4	3	2	1	0
P27F	P27PUP	P27PDP	P27OPR	-	-	P27S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0

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

8019H	7	6	5	4	3	2	1	0
P31F	P31PUP	P31PDP	P31OPR	-	-	P31S		

R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
801AH	7	6	5	4	3	2	1	0
P32F	P32PUP	P32PDP	P32OPR	-	-	P32S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
801BH	7	6	5	4	3	2	1	0
P33F	P33PUP	P33PDP	P33OPR	-	-	P33S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
801CH	7	6	5	4	3	2	1	0
P34F	P34PUP	P34PDP	P34OPR	-	-	P34S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
801DH	7	6	5	4	3	2	1	0
P35F	P35PUP	P35PDP	P35OPR	-	-	P35S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
801EH	7	6	5	4	3	2	1	0
P36F	P36PUP	P36PDP	P36OPR	-	-	P36S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
801FH	7	6	5	4	3	2	1	0
P37F	P37PUP	P37PDP	P37OPR	-	-	P37S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
8020H	7	6	5	4	3	2	1	0
P40F	P40PUP	P40PDP	P40OPR	-	-	P40S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
8021H	7	6	5	4	3	2	1	0
P41F	P41PUP	P41PDP	P41OPR	-	-	P41S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0

8022H	7	6	5	4	3	2	1	0
P42F	P42PUP	P42PDP	P42OPR	-	-	P42S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
8023H	7	6	5	4	3	2	1	0
P43F	P43PUP	P43PDP	P43OPR	-	-	P43S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
8024H	7	6	5	4	3	2	1	0
P44F	P44PUP	P44PDP	P44OPR	-	-	P44S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
8025H	7	6	5	4	3	2	1	0
P45F	P45PUP	P45PDP	P45OPR	-	-	P45S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
8026H	7	6	5	4	3	2	1	0
P46F	P46PUP	P46PDP	P46OPR	-	-	P46S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
8027H	7	6	5	4	3	2	1	0
P47F	P47PUP	P47PDP	P47OPR	-	-	P47S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
8028H	7	6	5	4	3	2	1	0
P50F	P50PUP	P50PDP	P50OPR	-	-	P50S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
8029H	7	6	5	4	3	2	1	0
P51F	P51PUP	P51PDP	P51OPR	-	-	P51S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
802AH	7	6	5	4	3	2	1	0
P52F	P52PUP	P52PDP	P52OPR	-	-	P52S		
R/W	R/W	R/W	R/W	-	-	R/W		

初始值	0	0	0	-	-	0	0	0
802BH	7	6	5	4	3	2	1	0
P53F	P53PUP	P53PDP	P53OPR	-	-	P53S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
802CH	7	6	5	4	3	2	1	0
P54F	P54PUP	P54PDP	P54OPR	-	-	P54S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
802DH	7	6	5	4	3	2	1	0
P55F	P55PUP	P55PDP	P55OPR	-	-	P55S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
802EH	7	6	5	4	3	2	1	0
P56F	P56PUP	P56PDP	P56OPR	-	-	P56S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
802FH	7	6	5	4	3	2	1	0
P57F	P57PUP	P57PDP	P57OPR	-	-	P57S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
8030H	7	6	5	4	3	2	1	0
P60F	P60PUP	P60PDP	P60OPR	-	-	P60S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
8031H	7	6	5	4	3	2	1	0
P61F	P61PUP	P61PDP	P61OPR	-	-	P61S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
8032H	7	6	5	4	3	2	1	0
P62F	P62PUP	P62PDP	P62OPR	-	-	P62S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
8033H	7	6	5	4	3	2	1	0

P63F	P63PUP	P63PDP	P63OPR	-	-	P63S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
8034H	7	6	5	4	3	2	1	0
P64F	P64PUP	P64PDP	P64OPR	-	-	P64S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
8035H	7	6	5	4	3	2	1	0
P65F	P65PUP	P65PDP	P65OPR	-	-	P65S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
8036H	7	6	5	4	3	2	1	0
P66F	P66PUP	P66PDP	P66OPR	-	-	P66S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
8037H	7	6	5	4	3	2	1	0
P67F	P67PUP	P67PDP	P67OPR	-	-	P67S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
8038H	7	6	5	4	3	2	1	0
P70F	P70PUP	P70PDP	P70OPR	-	-	P70S		
R/W	R/W	R/W	R/W	-	-	R/W		
初始值	0	0	0	-	-	0	0	0
8039H	7	6	5	4	3	2	1	0
P71F	P71PUP	P71PDP	P71OPR	-	-	-	P71S	
R/W	R/W	R/W	R/W	-	-	-	R/W	
初始值	0	0	0	-	-	-	0	0
803AH	7	6	5	4	3	2	1	0
P72F	P72PUP	P72PDP	P72OPR	-	-	-	P72S	
R/W	R/W	R/W	R/W	-	-	-	R/W	
初始值	0	0	0	-	-	-	0	0
803BH	7	6	5	4	3	2	1	0
P73F	P73PUP	P73PDP	P73OPR	-	-	-	P73S	
R/W	R/W	R/W	R/W	-	-	-	R/W	
初始值	0	0	0	-	-	-	0	0

803CH	7	6	5	4	3	2	1	0
P74F	P74PUP	P74PDP	P74OPR	-	-	-	P74S	
R/W	R/W	R/W	R/W	-	-	-	R/W	
初始值	0	0	0	-	-	-	0	0
803DH	7	6	5	4	3	2	1	0
P75F	P75PUP	P75PDP	P75OPR	-	-	-	P75S	
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	PnxPUP	上拉电阻使能控制位 0: 上拉电阻关闭 1: 上拉电阻打开
6	PnxPDP	下拉电阻使能控制位 0: 下拉电阻关闭 1: 下拉电阻打开
5	PnxOPR	开漏使能控制位，引脚设为数字输出时才有效 0: 开漏关闭 1: 开漏打开

备注：Pnx → n=0~7，代表 P0~P7
x=0~7，代表 Pn.0~Pn.7

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	COM[0]/LED_COM[0]	-	-	-	-
P01S	高阻	数字输入	数字输出	COM[1]/LED_COM[1]	-	-	-	-
P02S	高阻	数字输入	数字输出	COM[2]/LED_COM[2]	-	-	-	-
P03S	高阻	数字输入	数字输出	COM[3]/LED_COM[3]	-	-	-	-
P04S	高阻	数字输入	数字输出	COM[4]/LED_COM[4]	SEG[35]	高阻	高阻	高阻
P05S	高阻	数字输入	数字输出	COM[5]/LED_COM[5]	SEG[34]	高阻	高阻	高阻
P06S	高阻	数字输入	数字输出	COM[6]/LED_COM[6]	SEG[33]	高阻	高阻	高阻
P07S	高阻	数字输入	数字输出	COM[7]/LED_COM[7]	SEG[32]	高阻	高阻	高阻
P10S	高阻	数字输入	数字输出	SEG[16]/LED_SEG[16]	-	-	-	-
P11S	高阻	数字输入	数字输出	SEG[17]/LED_SEG[17]	-	-	-	-
P12S	高阻	数字输入	数字输出	SEG[18]/LED_SEG[18]	T2CP	高阻	高阻	高阻

P13S	高阻	数字输入	数字输出	SEG[19]/LED_SEG[19]	-	-	-	-
P14S	高阻	数字输入	数字输出	SEG[20]/LED_SEG[20]	-	-	-	-
P15S	高阻	数字输入	数字输出	SEG[21]/LED_SEG[21]	-	-	-	-
P16S	高阻	数字输入	数字输出	SEG[22]/LED_SEG[22]	-	-	-	-
P17S	高阻	数字输入	数字输出	SEG[23]/LED_SEG[23]	-	-	-	-
P20S	高阻 /CMP0P	数字输入	数字输出	SEG[31]/LED_SEG[31]	-	-	-	-
P21S	高阻 /CMP0N	数字输入	数字输出	SEG[30]/LED_SEG[30]	-	-	-	-
P22S	高阻 /CMP1P	数字输入	数字输出	SEG[29]/LED_SEG[29]	-	-	-	-
P23S	高阻 /CMP1N	数字输入	数字输出	SEG[28]/LED_SEG[28]	-	-	-	-
P24S	高阻 /CMP2P	数字输入	数字输出	SEG[27]/LED_SEG[27]	-	-	-	-
P25S	高阻 /CMP2N	数字输入	数字输出	SEG[26]/LED_SEG[26]	-	-	-	-
P26S	高阻 /CMP3P	数字输入	数字输出	SEG[25]/LED_SEG[25]	-	-	-	-
P27S	高阻 /CMP3N	数字输入	数字输出	SEG[24]/LED_SEG[24]	-	-	-	-
P30S	高阻	数字输入	数字输出	UART0_TX	IIC_SDA	高阻	高阻	高阻
P31S	高阻	数字输入	数字输出	UART0_RX	IIC_SCL	高阻	高阻	高阻
P32S	高阻 /ADCVRF	数字输入 /INT0	数字输出	高阻	TK[6]	PWM[7]	高阻	高阻
P33S	高阻	数字输入 /INT1	数字输出	高阻	TK[5]	PWM[6]	高阻	高阻
P34S	高阻 /OPOUT	数字输入/T0	数字输出	SEG[1]/LED_SEG[1]	TK[15]	高阻	高阻	高阻
P35S	高阻 /OPIN	数字输入/T1	数字输出	SEG[2]/LED_SEG[2]	TK[16]	高阻	高阻	高阻
P36S	高阻	数字输入	数字输出	IIC_SDA	TK[4]	高阻	高阻	高阻
P37S	高阻	数字输入	数字输出	IIC_SCL	TK[3]	高阻	高阻	高阻
P40S	高阻	数字输入	数字输出	ADC_CH[0]	TK[14]	高阻	高阻	高阻
P41S	高阻	数字输入	数字输出	ADC_CH[1]	TK[13]	高阻	高阻	高阻
P42S	高阻	数字输入	数字输出	ADC_CH[2]	TK[12]	高阻	高阻	高阻
P43S	高阻	数字输入	数字输出	ADC_CH[3]	TK[11]	高阻	高阻	高阻
P44S	高阻	数字输入	数字输出	ADC_CH[4]	TK[10]	高阻	高阻	高阻
P45S	高阻	数字输入	数字输出	ADC_CH[5]	TK[9]	高阻	高阻	高阻
P46S	高阻	数字输入	数字输出	ADC_CH[6]	TK[8]	高阻	高阻	高阻
P47S	高阻	数字输入	数字输出	ADC_CH[7]	TK[7]	高阻	高阻	高阻
P50S	高阻	数字输入	数字输出	SEG[4]/LED_SEG[4]	TK[18]	PWM[0]	高阻	高阻

P51S	高阻	数字输入	数字输出	SEG[5]/LED_SEG[5]	TK[19]	PWM[1]	高阻	高阻
P52S	高阻	数字输入	数字输出	SEG[6]/LED_SEG[6]	TK[20]	PWM[2]	高阻	高阻
P53S	高阻	数字输入	数字输出	SEG[7]/LED_SEG[7]	TK[21]	PWM[3]	高阻	高阻
P54S	高阻	数字输入	数字输出	SEG[8]/LED_SEG[8]	TK[22]	PWM[4]	高阻	高阻
P55S	高阻	数字输入	数字输出	SEG[9]/LED_SEG[9]	TK[23]	PWM[5]	高阻	高阻
P56S	高阻	数字输入	数字输出	SEG[3]/LED_SEG[3]	-	-	-	-
P57S	高阻 /TKCAP	数字输入	数字输出	SEG[0]/LED_SEG[0]	-	-	-	-
P60S	高阻	数字输入	数字输出	SEG[10]/LED_SEG[10]	UART2_RX	IIC_SCL	高阻	高阻
P61S	高阻	数字输入	数字输出	SEG[11]/LED_SEG[11]	UART2_TX	IIC_SDA	高阻	高阻
P62S	高阻	数字输入	数字输出	SEG[12]/LED_SEG[12]	SPI_MISO	高阻	高阻	高阻
P63S	高阻	数字输入	数字输出	SEG[13]/LED_SEG[13]	SPI_MOSI	高阻	高阻	高阻
P64S	高阻	数字输入	数字输出	SEG[14]/LED_SEG[14]	SPI_SCK	高阻	高阻	高阻
P65S	高阻	数字输入	数字输出	SEG[15]/LED_SEG[15]	SPI_SSB	高阻	高阻	高阻
P66S	高阻	数字输入	数字输出	UART1_TX	TK[2]	IIC_SDA	高阻	高阻
P67S	高阻	数字输入	数字输出	UART1_RX	TK[1]	IIC_SCL	高阻	高阻
P70S	高阻	数字输入	数字输出	SAMPLE	TK[0]	高阻	高阻	高阻
P71S	高阻	数字输入	数字输出	XTAL_OUT_32K	-	-	-	-
P72S	高阻	数字输入	数字输出	XTAL_IN_32K	-	-	-	-
P73S	高阻	数字输入	数字输出	XTAL_OUT_27M	-	-	-	-
P74S	高阻	数字输入	数字输出	XTAL_IN_27M	-	-	-	-
P75S	高阻	数字输入	数字输出	RESET	-	-	-	-

15.3 引脚控制例程

◆ **引脚功能设置**

例如，P20 设置为推挽输出，程序如下：

```
-------------------------------------------------------------------------------------
P20F = 2;
-------------------------------------------------------------------------------------
```

P20 设置为开漏输出，程序如下：

```
-------------------------------------------------------------------------------------
P20F = (1<<5)|2;
-------------------------------------------------------------------------------------
```

P20 设置为开漏输出，并且打开上拉，程序如下：

```
-------------------------------------------------------------------------------------
P20F = (1<<7) | (1<<5) | 2;
-------------------------------------------------------------------------------------
```

P20 设置为输入功能，并且打开上拉，程序如下：

```
-------------------------------------------------------------------------------------------
P20F = (1<<7)｜1;
-------------------------------------------------------------------------------------------
```

P20 设置为 LCD/LED SEG31，程序如下：

```
-------------------------------------------------------------------------------------------
P20F = 3;
-------------------------------------------------------------------------------------------
```

16 采样计数器（SAMPLE）

16.1 功能简介

采样计数器通过引脚 SAMPLE 采样输入脉冲宽度，理论上可采样任意长度的脉冲。输入脉冲必须维持采样时钟的 5 个周期以上，否则会检测不到。采样计数器可设置上升沿、下降沿或双沿触发。采样计数器有多种时钟源可选择。图 16-1-1 为采样计数器结构图。

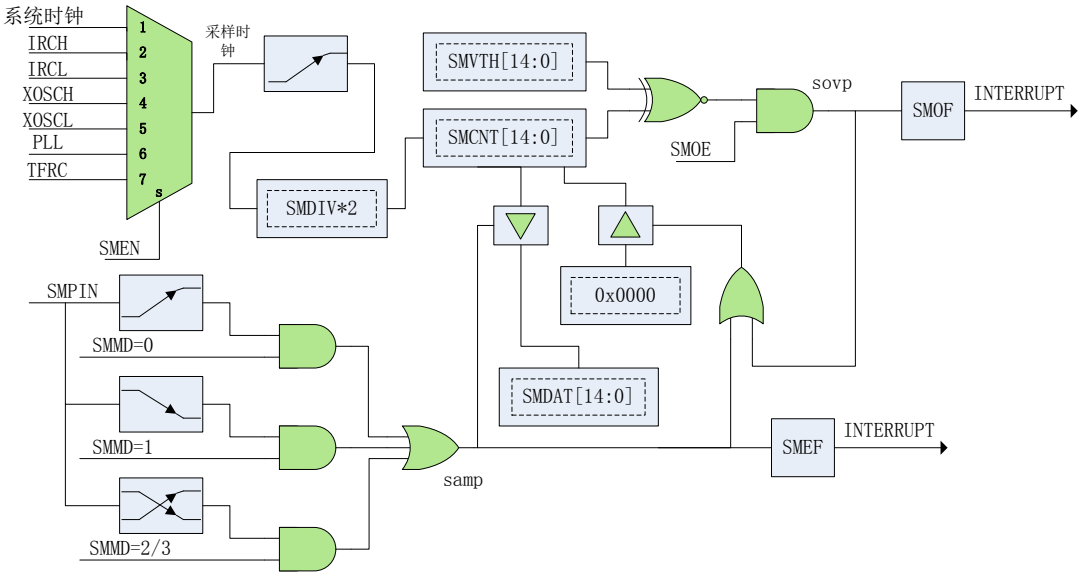


图 16-1-1 采样计数器结构图

用户可以通过寄存器 SMDIV 设置采样时钟分频，分频数越大，单次计数可采样的脉宽越长，但采样精度越低。采样计数器可通过寄存器 SMVTHL、SMVTHH 设置采样脉宽阈值，当计数达到所设的阈值时，会被认为计数溢出，计数器重新从 0 开始计数，同时产生溢出中断标志 SMOF。这样设计有两个作用，一是当采样的脉宽比较长，超过计数器单次计数范围时，软件可通过累加溢出的次数来计算总脉宽的长度；二是可通过设置阈值来识别有效的脉冲，用户可设置有效脉宽的最大值作为阈值，当溢出事件发生时，表示当前计数已超出范围，软件应重置程序状态以识别下次有效脉冲。

采样计数器开启后，每次检测到有效沿，都会重置计数器为 0 并开始计数。采样计数器开启后检测到的第一个有效沿并不会产生中断，而且检测到有效沿之前也不会产生溢出中断。当检测到第二个或以上有效沿时，计数器的值会被存入寄存器 SMDATL、SMDATH，并产生中断，中断标志为 SMEF，软件可根据此计数值计算脉冲宽度。

此模块特点：采样计数器用于实现红外遥控接收等功能时，节省软件代码，方便开发。

16.2 SAMPLE 功能寄存器描述

表 16-2-1 寄存器 SMCON

8078H	7	6	5	4	3	2	1	0
SMCON	SMEN[2:0]			SMIE	SMOE	-	SMMD[1:0]	
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
7~5	SMEN	SAMPLE 时钟选择位 000：关闭 001：系统时钟 010：IRCH 011：IRCL 100：XOSCH 101：XOSCL 110：PLL 111：TFRC
4	SMIE	SAMPLE 中断使能位，1 有效
3	SMOE	SAMPLE 溢出阈值使能位 0：阈值不起作用，计满为 7FFFH，溢出时不会产生溢出中断标志 1：阈值起作用，溢出将产生溢出中断标志，发中断使能，会产生中断
2	-	-
1~0	SMMD	采样边沿选择位 00：上升沿采样 01：下降沿采样 其他：双沿采样

表 16-2-2 寄存器 SMSTA

8079H	7	6	5	4	3	2	1	0
SMSTA	-	-	-	-	-	-	SMEF	SMOF
R/W	-	-	-	-	-	-	R	R
初始值	-	-	-	-	-	-	0	0

位编号	位符号	说明
7~2	-	-
1	SMEF	SAMPLE 边沿中断标志位，1 有效，写 1 清 0
0	SMOF	SAMPLE 溢出中断标志位，1 有效，写 1 清 0

表 16-2-3 寄存器 SMDIV

807AH	7	6	5	4	3	2	1	0
SMDIV	SMDIV[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	SMIDV	采样时钟分频器，分频系数为 SMDIV * 2						

表 16-2-4 寄存器 SMDAT

807BH	7	6	5	4	3	2	1	0
SMDATL	SMDAT[7:0]							
R/W	R							
初始值	0	0	0	0	0	0	0	0
807CH	7	6	5	4	3	2	1	0
SMDATH	LVBIT	SMDAT[14:8]						
R/W	R	R						
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
15（SMDATH.7）	LVBIT	当前触发沿标志位 0：下降沿 1：上升沿						
14~0	SMDAT	SAMPLE 计数寄存器，保存计数数值						

表 16-2-5 寄存器 SMVTHL、SMVTHH

807DH	7	6	5	4	3	2	1	0
SMVTHL	SMVTH[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
807EH	7	6	5	4	3	2	1	0
SMVTHH	-	SMVTH[14:8]						
R/W	-	R/W						
初始值	-	0	0	0	0	0	0	0
位编号	位符号	说明						
15	-	-						
14~0	SMVTH	计数器溢出阈值设置寄存器						

16.3 SAMPLE 控制例程

用 SAMPLE 功能实现红外遥控，程序如下：

```
--------------------------------------------------------------------------------------------------------------------------------
#define     SMEN_SYS_CLK    (1<<5)
#define     SMEF    (1<<1)
#define     SMOF    (1<<0)

#define SYN_WIDTH1          0x103b          //同步信号脉冲宽度
#define DAT_1_WIDTH         0x02b1          //位 1 的脉冲宽度
#define DAT_0_WIDTH         0x015c          //位 0 的脉冲宽度
#define WIDTH               (DAT_0_WIDTH/6)

unsigned char OverFlowCount;            //计数溢出计数器
unsigned char IR_BitCount;              //遥控码值位指示
unsigned char IR_Code[4];               //遥控码值
bit IR_SyncFlag;                        //同步信号标志，为 1 表示已接收到同步信号
bit IR_RxEndFlag;

void Sample_init(void)
{
P70F = 3;                           //P70 设置为 SAMPLE 功能引脚
    SMCON = SMEN_SYS_CLK | SMOE(1) | SMMD(1);       //初始化 SAMPLE 功能
    SMDIV = 6;                                      //设置 SAMPLE 时钟分频
    SMVTHL = (SYN_WIDTH1*2)%256;                    //设置溢出时钟宽度
    SMVTHH = (SYN_WIDTH1*2)/256;
    SMCON |= SMIE(1);                               //使能 SAMPLE 中断
    INT9EN = 1;                                     //使能 INT9
}
void INT9_ISR (void) interrupt 14
{
    unsigned int PulseWidth;
    if(SMSTA & SMEF)
    {
        SMSTA |= SMEF;
        if(OverFlowCount == 0)
        {
            PulseWidth = (SMDATH&0x7F)*256 + SMDATL;    //获取脉冲宽度
            if(!IR_SyncFlag)
            {
                if((PulseWidth > (SYN_WIDTH1-WIDTH*6)) && (PulseWidth < (SYN_WIDTH1+WIDTH*6))) //如果未同步，判断当前脉冲是否同步信号
                {
                    IR_SyncFlag = 1;
```

```
                    IR_BitCount = 0;
                }
            }
            else
            {
                if((PulseWidth > (DAT_1_WIDTH-WIDTH*2)) && (PulseWidth < (DAT_1_WIDTH+WIDTH*2)))
//判断当前脉冲是否位 1
                {
                    IR_Code[IR_BitCount/8] |= (1<<(7-(IR_BitCount%8)));
                    IR_BitCount++;
                }
                else if((PulseWidth > (DAT_0_WIDTH-WIDTH)) && (PulseWidth < (DAT_0_WIDTH+WIDTH)))
//判断当前脉冲是否位 0
                {
                    IR_Code[IR_BitCount/8] &= ~(1<<(7-(IR_BitCount%8)));
                    IR_BitCount++;
                }
                else
                {
                    IR_SyncFlag = 0;            //如果不是位 1 或 0 的宽度，则等待接收下一个同步信号
                }
                if((IR_BitCount == 32) && IR_SyncFlag)      //32 位遥控码接收完成
                {
    //在此处可读取遥控码
                    IR_SyncFlag = 0;
                    IR_RxEndFlag = 1;
                }
            }
        }
        OverFlowCount=0;                    //复位溢出计数器
    }
    if(SMSTA & SMOF)
    {
        SMSTA |= SMOF;
        if(OverFlowCount < 0xFF)
        {
            OverFlowCount++;            //计数溢出累加
        }
    }
}
void main(void)
{
    Sample_init();
IR_SyncFlag = 0;
    IR_RxEndFlag = 0;
```

```
        OverFlowCount=0;
        EA = 1;
        while(1)
        {
            if(IR_RxEndFlag)
            {
                IR_RxEndFlag = 0;
            }
        }
    }
```

---

17 通用串行接口（UART）

17.1 UART0

17.1.1 功能简介

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

UART0 有 4 种工作模式，如表 17-1-1-1 所示。

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

表 17-1-1-1　UART0 通信工作模式

备注：由于定时器 2 的时钟是直接来自系统时钟，没有经过分频，所以选择定时器 2 作为 UART0 时钟发生器会有更高的波特率，在系统时钟为 3.6864MHz 时，最高波特率可达 115200。

- **模式 0**

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

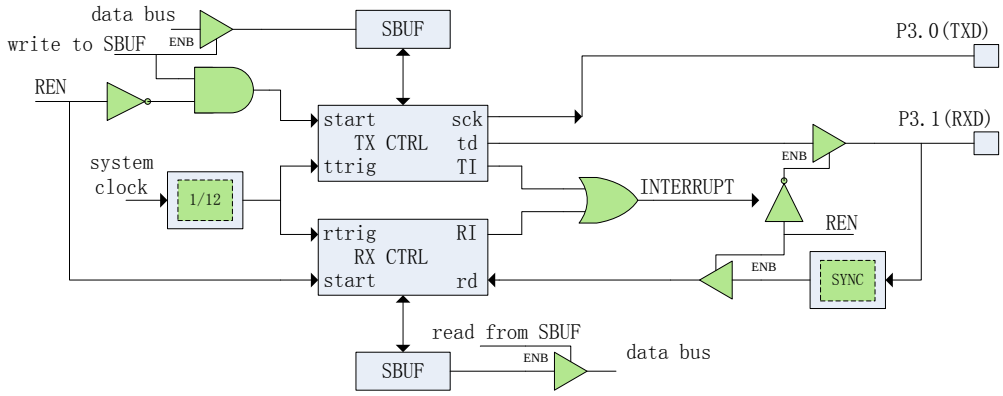


图 17-1-1-1　UART0 模式 0 示意图

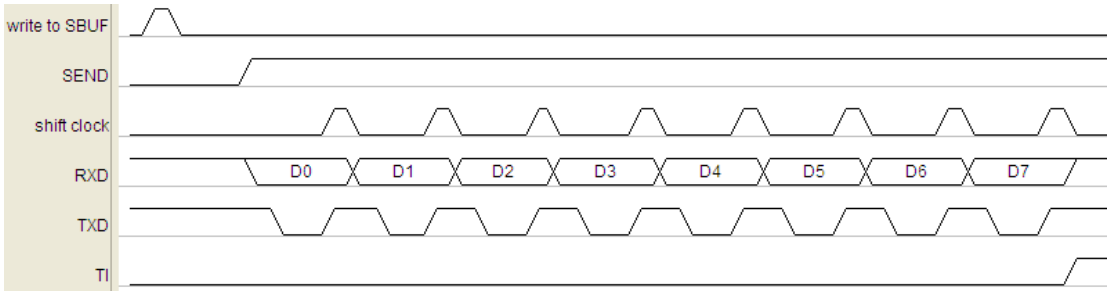


图 17-1-1-2 UART0 模式 0 发送数据波形

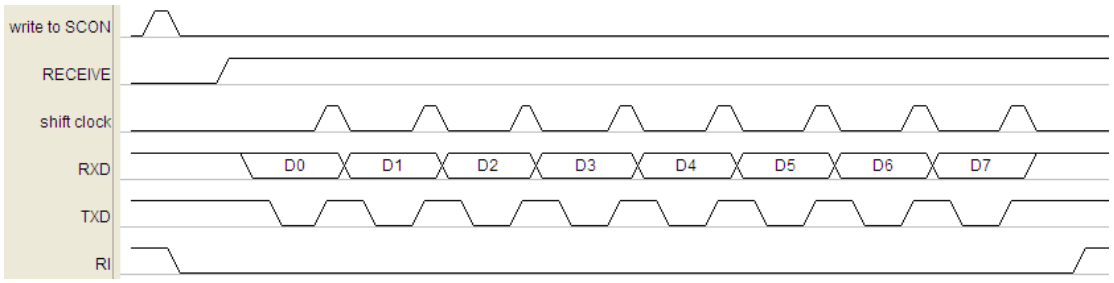


图 17-1-1-3 串口 0 模式 0 接收数据波形

- **模式 1**

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

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

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

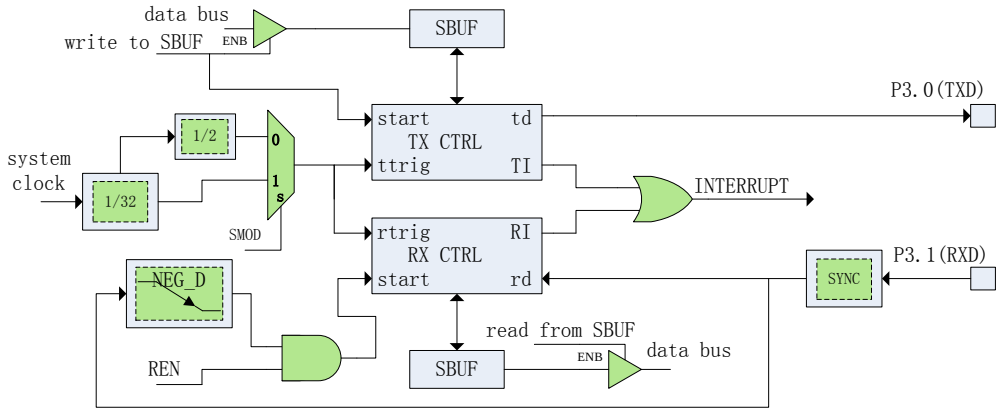


图 17-1-1-4　UART0 模式 1 示意图

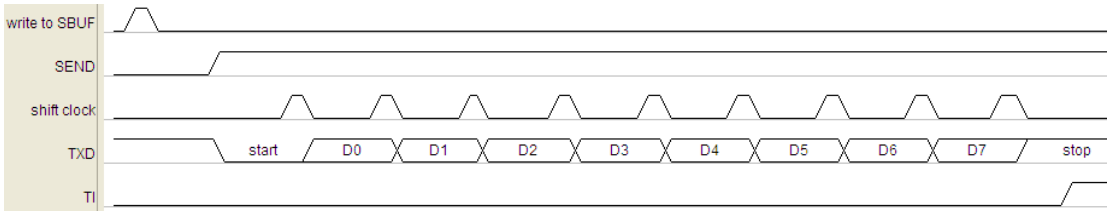


图 17-1-1-5　UART0 模式 1 发送数据波形

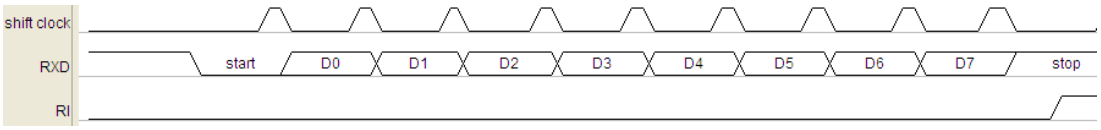


图 17-1-1-6　UART0 模式 1 接收数据波形

- **模式 2**

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

写入数据到寄存器 S0BUF 会启动 UART0 发送。第一个传送的位是开始位（为 0），然后是 9 位数据（低位先发），第 9 位数据是寄存器 S0CON 的 TB80 位，最后传送的是停止位（为 1）。

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

在接收状态，UART0 通过检测引脚 RX 的下降沿来同步。传送过程完成后，低 8 位数据存放在寄存器 S0BUF，第 9 位数据存放在 RB80 位（S0CON[2]）。

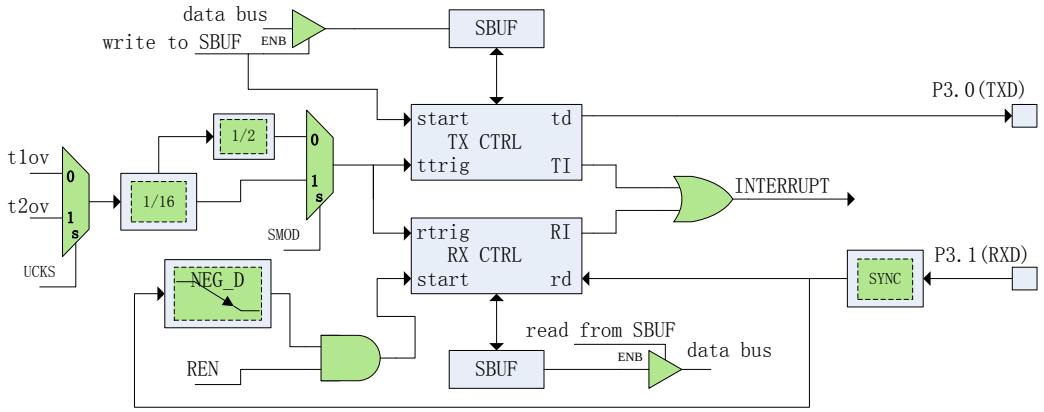


图 17-1-1-7 UART0 模式 2 示意图

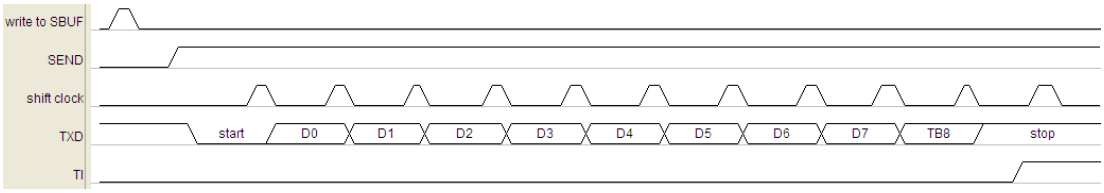


图 17-1-1-7 UART0 模式 2 发送数据波形

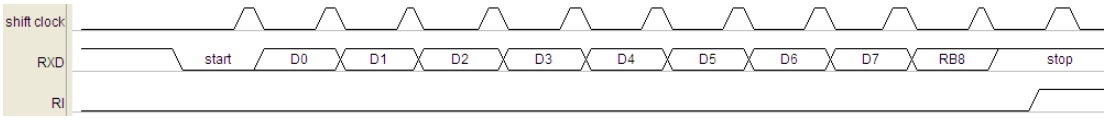


图 17-1-1-7 UART0 模式 2 接收数据波形

- **模式 3**

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

- **UART0 多机通信**

在 UART0 模式 2 和 3 中有一个专门适用于多机通信的机制。当寄存器 S0CON 的 SM20 位置 1，只有接收到第 9 位数据为 1（RB80=1）的从机才会产生接收中断，利用这个功能可进行多机通信，从机将它们的 SM20 位都置为 1，主机传送从机的地址时将第 9 位数据设为 1，这样所有的从机都会产生接收中断；从机的软件用它们自己的地址和接收的地址进行比较，如果一致，被寻址的从机设置 SM20=0，然后主机继续传送后面的数据时设置第 9 位为 0，因为其他的从机 SM20 仍然设为 1，这样就只有被寻址的从机才会产生接收中断。

17.1.2 寄存器描述

表 17-1-2-1 寄存器 S0CON

98H	7	6	5	4	3	2	1	0
S0CON	SM00	SM10	SM20	REN0	TB80	RB80	TI0	RI0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0

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

表 17-1-2-2 寄存器 S0BUF

99H	7	6	5	4	3	2	1	0
S0BUF	S0BUF[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7~0	S0BUF	发送接收缓冲器 写 S0BUF 将启动发送所写的数据 读 S0BUF 将读取已经接收的数据

17.2 UART1 和 UART2

17.2.1 介绍

UART1 和 UART2 是设计完全相同的两个全双工异步串行数据收发器，和 UART0 一样的是，UARTx（x=1、2,代指 UART1、UART2）也有一字节的接收缓存。UARTx 有两种不同的工作模式，如表 17-2-1-1 所示。

SMx	模式	描述	波特率
0	A	9 位异步模式，与 UART0 的模式 2 和 3 相同	CPUCLK/(32*(1024-SxREL))
1	B	8 位异步模式，与 UART0 的模式 1 相同	CPUCLK/(32*(1024-SxREL))

表 17-2-1-1 UARTx 工作模式

UARTx 的工作原理与 UART0 的异步模式（模式 1/2/3）相同，只是波特率的配置方法有所区别，无论是模式 A，还是模式 B，都可以参考 UART0 的工作波形。和 UART0 不同的是，UARTx 设计了专门的波特率发生器，波特率通过寄存器 SxRELL、SxRELH 来配置。

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

图 17-2-1-1 是 UARTx 的原理示意图。

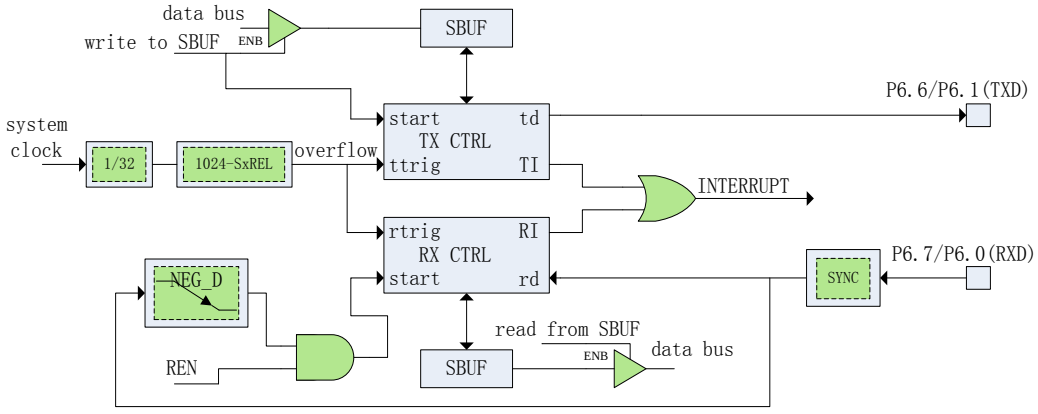


图 17-2-1-1 UARTx 工作原理示意图

- **模式 A**

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

- **模式 B**

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

- **UARTx 多机通信**

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

17.2.2 UARTx 寄存器描述

表 17-2-2-1 寄存器 S1CON

9AH	7	6	5	4	3	2	1	0
S1CON	SM1	U1IE	SM21	REN1	TB81	RB81	TI1	RI1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7	SM1	UART1 模式选择位，详见表 17-2-1-1
6	U1IE	UART1 中断使能位，1 有效
5	SM21	多机通信使能位，1 有效
4	REN1	串行接收使能位，1 有效
3	TB81	发送数据的第 9 位 在模式 A，这个位用于 UART1 传送数据，对应传送数据的第 9 位 （例如奇偶校验或多主机通信），由软件控制
2	RB81	接收数据的第 9 位 在模式 A，这个位用于 UART1 接收数据，对应接收数据的第 9 位； 在模式 B，这个位是接收到的停止位
1	TI1	传送中断标志位，1 有效，写 1 清 0
0	RI1	接收中断标志位，1 有效，写 1 清 0

表 17-2-2-2 寄存器 S1BUF

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

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

表 17-2-2-3 寄存器 S1RELL、S1RELH

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

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

表 17-2-2-4 寄存器 S2CON

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

位编号	位符号	说明
7	SM2	UART2 模式选择位，详见表 17-2-1-1
6	U2IE	串口 2 中断使能位，1 有效
5	SM22	多机通信使能位，1 有效
4	REN2	串行接收使能位，1 有效
3	TB82	发送数据的第 9 位 在模式 A，这个位用于串口 1 传送数据，对应传送数据的第 9 位 （例如奇偶校验或多主机通信），由软件控制
2	RB82	接收数据的第 9 位 在模式 A，这个位用于 UART2 接收数据，对应接收数据的第 9 位 在模式 B，这个位是接收到的停止位
1	TI2	传送中断标志位，1 有效，写 1 清 0
0	RI2	接收中断标志，1 有效，写 1 清 0

表 17-2-2-5 寄存器 S2BUF

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

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

表 17-2-2-6 寄存器 S2REL

A3H	7	6	5	4	3	2	1	0
S2RELL	S2RELL[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
A4H	7	6	5	4	3	2	1	0
S2RELH	-	-	-	-	-	-	S2REL[9:8]	
R/W	-	-	-	-	-	-	R/W	
初始值	-	-	-	-	-	-	0	0
位编号	位符号	说明						
9~0	S2REL	波特率配置寄存器 波特率为 CPUCLK/(32 * (1024 - S2REL))						

18 SPI 接口

18.1 功能简介

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

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

图 18-1-1 和图 18-1-2 分别是 SPI 主机模式和从机模式的原理示意图。

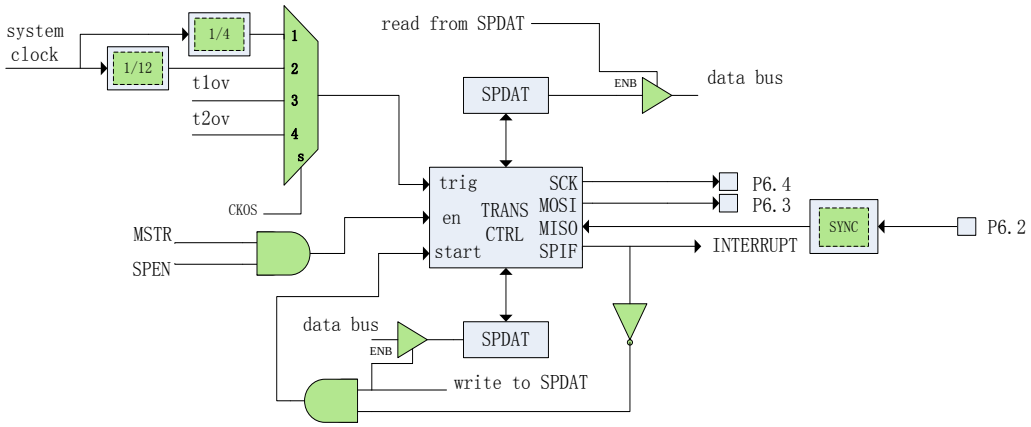


图 18-1-1 SPI 主机模式示意图

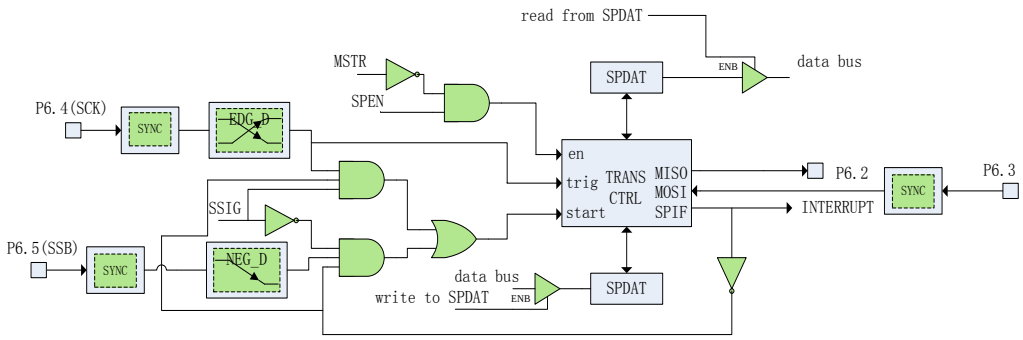


图 18-1-2 SPI 从机模式示意图

SPI 工作模式如下表所示。

表 18-1-1 SPI 工作模式

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

表 18-1-2 SPI 接口引脚描述

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

表 18-1-3 SPI 相位与极性

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

结合表 18-1-3，实际传输时的波形如图 18-1-3 和图 18-1-4 所示。

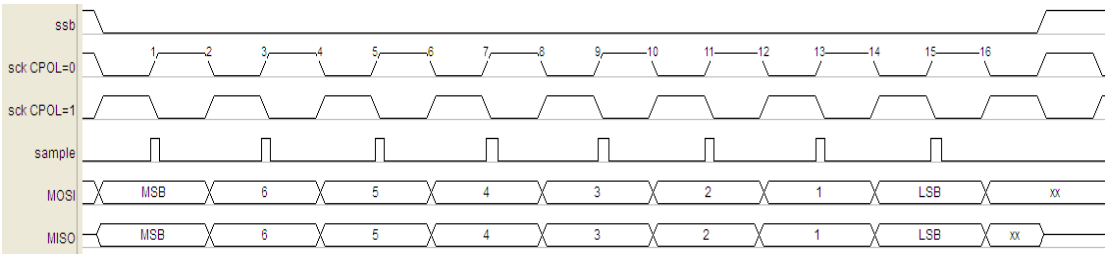


图 18-1-3 CPHA=0 时 SPI 时序图

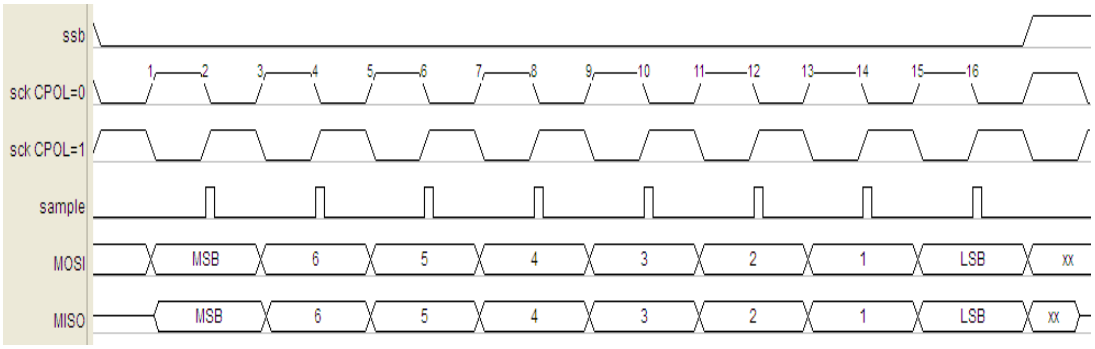


图 18-1-4 CPHA=1 时 SPI 时序图

18.2 寄存器描述

表 18-2-1 寄存器 SPCON

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

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

表 18-2-2 寄存器 SPDAT

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

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

表 18-2-3 寄存器 SPSTA

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

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

18.3 SPI 控制例程

◆ **SPI 主机例程**

SPI 作为主机，向从机发送 10 个字节数据，程序如下：

```
--------------------------------------------------------------------------------------------------------------------------
#define SPEN(N)          (N<<7)
#define LSBF(N)          (N<<6)
#define SSIG(N)          (N<<5)
#define MSTR(N)          (N<<4)
#define CPOL(N)          (N<<3)
#define CPHA(N)          (N<<2)
#define CPOS(N)          (N)        //3:系统时钟选择为定时器 2 溢出
                                    //2:系统时钟选择为定时器 1 溢出
                                    //1:系统时钟选择为 1/8 系统时钟
                                    //0:系统时钟选择为 1/4 系统时钟

//SPSTA 位定义
#define SPIE        (1<<7)    //SPI 中断使能
#define WCOL        (1<<2)    //写入冲突标志
#define MODF        (1<<1)    //故障模式
#define SPIF        (1<<0)    //传输完成
unsigned char txIndex;
unsigned char txLength;
unsigned char txBuf[10]={0,1,2,3,4,5,6,7,8,9};
void SPI_init(void)
{
    P64F = 4;//设置 P64 为 SPI SCK
    P62F = 4;//设置 P62 为 SPI MISO
    P63F = 4;//设置 P63 为 SPI MOSI
    P65F = 2;//设置 P65 为输出功能，作为 SPI 片选引脚

    P6 |= 1<<5;    //设置片选引脚为高
    SPCON = SPEN(1) | LSBF(0) | SSIG(1) | MSTR(1) | CPOL(0) | CPHA(0) |CPOS(1);          //高位先发，SSB 有效，主机, 1/4 系统时钟
    SPSTA |= SPIE;          //使能 SPI 中断
    INT5EN = 1;         //使能 INT5 中断
}
void INT5_ISR (void) interrupt 10
{
    if(SPSTA & SPIF)                    //SPI 中断
    {
        SPSTA |= SPIF;      //清除 SPI 中断标志
```

```
            txIndex++;
            If(txIndex >= txLength)
            {
                P6 |= 1<<5;     //数据发送完毕，设置片选引脚为高
            }
            else
            {
                SPDAT = txBuf[txIndex];       //发送下一字节数据
            }
        }
        if(SPSTA&MODF)
        {
          SPSTA|=MODF;
        }
}
void main(void)
{
        SPI_init();
        EA = 1;
        txIndex = 0;
        txLength = 0;
        SPDAT = txBuf[txIndex];       //写 SPDAT 启动数据发送
        while(1)
        {
        }
}
```

◆ **SPI 从机例程**

SPI 作为从机，接收主机发送的数据，程序如下：

```
unsigned char rxIndex;
unsigned char txLength;
unsigned char rxBuf[10];
void SPI_init(void)
{
    P64F = 4;//设置 P64 为 SPI SCK
    P62F = 4;//设置 P62 为 SPI MISO
    P63F = 4;//设置 P63 为 SPI MOSI
    P65F = 4;//设置 P65 为 SPI SSB

     P6 |= 1<<5;     //设置片选引脚为高
     SPCON = SPEN(1) | LSBF(0) | SSIG(1) | MSTR(0) | CPOL(0) | CPHA(0) |CPOS(1);            //高位先发，SSB 有效，从机, 1/4 系统时钟
```

```
    SPSTA |= SPIE;              //使能 SPI 中断
    INT5EN = 1;          //使能 INT5 中断
}
void INT5_ISR (void) interrupt 10
{
    if(SPSTA & SPIF)                          //SPI 中断
    {
        SPSTA |= SPIF;      //清除 SPI 中断标志
        rxBuf[rxIndex++] = SPDAT;  //接收一字节数据
    }
    if(SPSTA&MODF)
    {
      SPSTA|=MODF;
    }
}
void main(void)
{
    SPI_init();
    EA = 1;
    rxIndex= 0;
    while(1)
    {
    }
}
```

---

19 I²C 接口

19.1 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。

19.2 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

19.3 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。

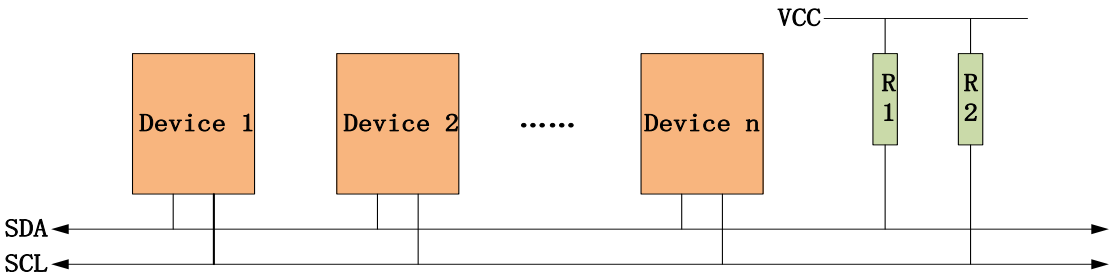


图 19-3-1 I2C 总线互连图

I²C 模块原理示意图如图图 19-3-2 所示。

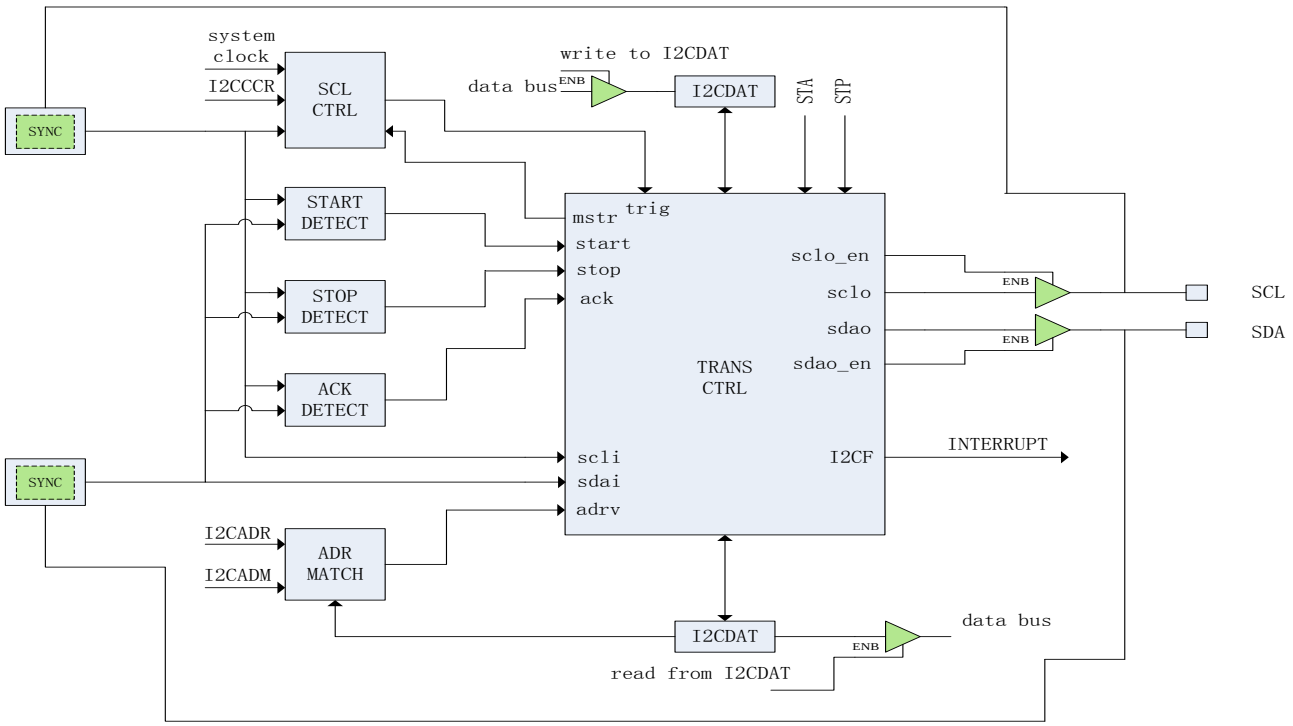


图 19-3-2　I²C 模块原理示意图

- **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。

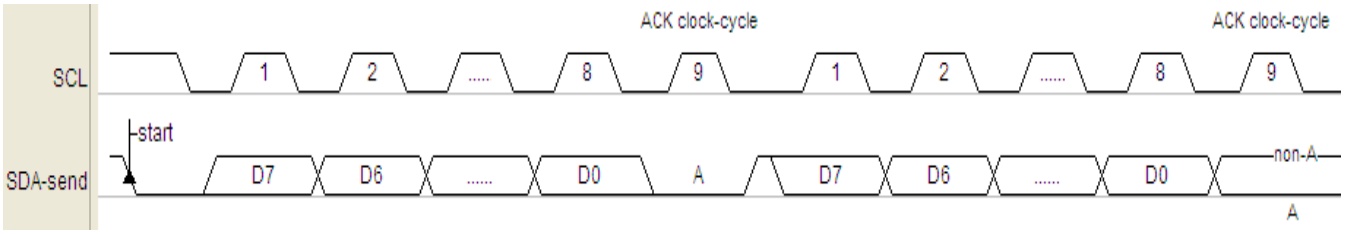


图 19-1-3　I2C 总线数据传输格式

- **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。当中断标志 I2CF 产生时，如果 SHD=1，在没有清除 I2CF 之前，SCL 会被从机拉低，主机检测到 SCL 被释放后才会进行下一步操作；如果 SHD=0，从机不会拉低 SCL，这样设计是为了兼容主机是软件模拟 I2C 的应用，此时，主机的软件必须等待足够长的时间让从机响应每字节数据传输的处理。

当 I^2C 接口作为从机时，SCL 的时钟由主机输入，和从机的时钟配置无关。作为从机时，需要保证 SCL 为低电平的宽度最少为 6.5 个系统时钟，而高电平最少为 2.5 个系统时钟。所以，外部主机发送的 SCL 频率最高为系统时钟频率的 1/9。

19.4 I^2C 通信引脚的映射

为了方便硬件设计，I^2C 通信引脚可以有不同的映射，由寄存器 I2CIOS 设置不同的值来选择。详细见寄存器 I2CIOS 的描述。

19.5 寄存器描述

表 19-5-1 寄存器 I2CCON

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

位编号	位符号	说明
7	I2CE	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。同时需要注意，因为 CBUS 总线的地址都是 7 位的，所以还要配置 HCE 位为 0。
0	STFE	为 1 时，I^2C 模块检测到 START 信号时将置位 I2CF

表 19-5-2 寄存器 I2CADR

B2H	7	6	5	4	3	2	1	0
I2CADR	GCE	I2CADR[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	I2CADR	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 19-5-3 寄存器 I2CADM

B3H	7	6	5	4	3	2	1	0
I2CADM	SPFE	I2CADM[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	SPFE 位为 1 时，I^2C 模块检测到 STOP 信号时将置位 I2CF
6~0	I2CADM	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 19-5-4 寄存器 I2CCCR

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

位编号	位符号	说明
7~0	I2CCCR	I^2C 时钟配置寄存器 SCL 时钟频率为 Fscl＝Fsys/(CCR+8)。 备注： CCR 的最大值为 F7H，设置 F8H~FFH 等效于设置 00H。

表 19-5-5 寄存器 I2CDAT

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

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

表 19-5-6 寄存器 I2CSTA

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

位编号	位符号	说明
7~0	I2CSTA	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 19-5-7 寄存器 I2CFLG

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

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

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

表 19-5-8 寄存器 I2CIOS

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

位编号	位符号	说明
7~2	-	-
1~0	I2CS	I²C 引脚选择位 00：SCL 在引脚 P37，SDA 在引脚 P36 01：SCL 在引脚 P31，SDA 在引脚 P30 10：SCL 在引脚 P67，SDA 在引脚 P66 11：SCL 在引脚 P60，SDA 在引脚 P61

19.6 I²C 控制例程

◆ **I²C 作为主机例程**

例如，主机循环向从机写入 20 字节数据，程序如下：

```
--------------------------------------------------------------------------------------------------------------
------
//I2CCON 定义
#define I2CE(N)     (N<<7)
#define I2CIE(N)    (N<<6)
#define STA(N)          (N<<5)
#define STP(N)          (N<<4)
#define CKHD(N)         (N<<3)
#define AAK(N)          (N<<2)
#define CBSE(N)         (N<<1)
```

```
#define STFE(N)          (N<<0)
//I2CADR 定义
#define    GCE           (1<<7)
//I2CFLG 定义
#define I2CF             (1<<0)
#define I2C_ADDR              0xA0

unsigned char txIndex;
unsigned char regAddr;
unsigned char txLength;
unsigned char txBuf[20]={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19};
void I2C_init(void)
{
    I2CIOS = 0;          //选择 P36,P37 作为 I2C 通信引脚
    P36F = 3|(1<<7); //设置 P36 作为 I2C SDA，并打开上位
    P37F = 3|(1<<7); //设置 P36 作为 I2C SCL，并打开上位
    I2CCON = I2CE(1) | I2CIE(1) | STA(0) | STP(0)| CKHD(1) | AAK(1)| CBSE(0) | STFE(1);          //使能 I2C，开启
I2C 中断，设置 I2CF 置 1 后拉低 SCL，设置 START 信号产生中断
    I2CCCR = 5;  //设置 I2C 时钟分频
    INT6EN = 1;  //使能 INT6 中断
}
void INT6_ISR(void) interrupt 11
{
    unsigned char Sta_Temp;
    if(I2CFLG & I2CF)                    //I2C 中断
    {
        Sta_Temp = I2CSTA;                    //读取 I2C 状态
        if(Sta_Temp == 0x08)          //START 信号已产生
        {
            I2CDAT = I2C_ADDR;  //发送 I2C 从机地址+写位
        }
        else if(Sta_Temp == 0x18)     //地址+写位已发送，收到从机应答
        {
            I2CDAT = regAddr;       //发送数据首地址
            txIndex = 0;
        }
        else if(Sta_Temp == 0x28)     //数据已发送，已收到从机应答
        {
            if(txIndex >= txLength)   //判断是否所有数据已发送完
            {
                I2CCON |= STP(1); //如果已发送完所有数据则发送 STOP 信号
                iicEnd = 1;      //置位结束标志
            }
            else
            {
```

```
                I2CDAT = txBuf[txIndex+regAddr];      //如未发送则发送下一字节数据
                txIndex++;
            }
        }
        I2CFLG   |= I2CF;  //清除 I2C 中断标志
    }
}
void main(void)
{
    I2C_init();
    EA = 1; //开启全局中断
    while(1)
    {
        regAddr = 0;   //复位数据地址
        iicEnd = 0;      //结束标志清 0
        txLength = 10; //设置发送数据长度
        I2CCON |= STA(1); //发送 START 信号，启动数据发送
        While(!iicEnd);            //等待数据发送完
    }
}
```

例如，主机循环从从机读取 20 字节数据，程序如下：

```
bit rdSession;
bit iicEnd;
unsigned char rxIndex;
unsigned char rxLength;
unsigned char rxBuf[20];
void I2C_init(void)
{
    I2CIOS = 0;          //选择 P36,P37 作为 I2C 通信引脚
    P36F = 3|(1<<7); //设置 P36 作为 I2C SDA，并打开上位
    P37F = 3|(1<<7); //设置 P36 作为 I2C SCL，并打开上位
    I2CCON = I2CE(1) | I2CIE(1) | STA(0) | STP(0)| CKHD(1) | AAK(1)| CBSE(0) | STFE(1);          //使能 I2C，开启 I2C 中断，设置 I2CF 置 1 后拉低 SCL，设置 START 信号产生中断
    I2CCCR = 5;  //设置 I2C 时钟分频
    INT6EN = 1;  //使能 INT6 中断
}

void INT6_ISR(void) interrupt 11
{
    unsigned char Sta_Temp;
    if(I2CFLG & I2CF) //I2C 中断
    {
```

```
Sta_Temp = I2CSTA;            //读取 I2C 状态
if(Sta_Temp == 0x08)     //START 信号已产生
{
    if(rdSession)
    {
        I2CDAT = I2C_ADDR|1;            //读数据阶段，发送从机地址+读位
    }
    else
    {
        I2CDAT = I2C_ADDR;  //写数据阶段，发送从机地址+写位
    }
}
else if(Sta_Temp == 0x18)     //地址+写位已发送，收到从机应答
{
    I2CDAT  = regAddr;      //发送数据首地址
}
else if(Sta_Temp == 0x28)     //已发送或接收到一字节数据，收到从机应答
{
    if(rdSession) //判断是写数据阶段还是读数据阶段
    {
        if(rxIndex >= rxLength)  //如果是读数据阶段，判断是否已读取完数据
        {
            I2CCON |= STP(1); //如果已读取完，发送 STOP 信号
            iicEnd = 1;             //置位结束标志
        }
        else
        {
            rxBuf[rxIndex+regAddr] = I2CDAT;       //读取一字节数据
            rxIndex++;
            if(rxIndex <= (rxLength-1))
            {
                I2CCON |= AAK(1);       //如果不是最后一字节数据，发送应答信号
            }
            else
            {
                I2CCON &= ~AAK(1);//最后一字节数据发送不应答信号
            }
        }
    }
    else
    {
        I2CCON |= STA(1);       //如果是写数据阶段，重新发送 START 信号
        rdSession = 1;
    }
}
```

```
            else if(Sta_Temp == 0x30)      //主机发送/接收一字节数据，没有收到应答信号
            {
                if(rdSession)
                {
                    if(rxIndex >= rxLength)  //判断如果是最后一字节数据，则发送 STOP 信号
                    {
                        I2CCON |= STP(1);
                    }
                }
            }
            else if(Sta_Temp == 0x40)      //从机地址+读位已发送，已收到应答信号
            {
                I2CCON |= AAK(1);              //先设置好应答位
                rxIndex=0;
            }
            I2CFLG    |= I2CF; //清除 I2C 中断标志
        }
    }
    void main(void)
    {
        I2C_init();
        EA = 1; //开启全局中断
        while(1)
        {
            regAddr = 0;    //复位数据地址
            iicEnd = 0;       //结束标志清 0
            rxLength= 20; //设置发送数据长度
            I2CCON |= STA(1); //发送 START 信号，启动 I2C 通信
            While(!iicEnd);              //等待数据接收完
            //在此处可读取 I2C 接收缓存 rxBuf
        }
    }
```

◆ **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：

```
unsigned char rxIndex;
unsigned char regAddr;
bit iicReadMode;
bit iicEnd;
unsigned char xdata rxBuf[20];
unsigned char txBuf[20]={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19};

void I2C_init(void)
```

```
{
    I2CIOS = 0;          //选择 P36,P37 作为 I2C 通信引脚
    P36F = 3|(1<<7); //设置 P36 作为 I2C SDA，并打开上位
    P37F = 3|(1<<7); //设置 P36 作为 I2C SCL，并打开上位
    I2CCON = I2CE(1) | I2CIE(1) | STA(0) | STP(0)| CKHD(1) | AAK(1)| CBSE(0) | STFE(0);          //使能 I2C，开启 I2C 中断，设置 I2CF 置 1 后拉低 SCL，设置 START 信号不产生中断
    I2CADR = (I2C_ADDR>>1); //设置从机 I2C 地址
    INT6EN = 1;  //使能 INT6 中断
}
void INT6_ISR(void) interrupt 11 // using 1
{
    unsigned char Sta_Temp;
    if(I2CFLG & I2CF)                    //I2C 中断
    {
        Sta_Temp = I2CSTA;                    //读取 I2C 状态
        if(Sta_Temp == 0x60)                  //接收到从机地址+写位
        {
            rxIndex = 0xff;
            iicReadMode = 0;   //设置标志为写状态
            I2CCON |= AAK(1);          //预设应答位
        }
        else if(Sta_Temp == 0x80)
        {
            if(iicReadMode)          //判断当前为读状态还是写状态
            {
                rxIndex++;
                I2CDAT = rxBuf[rxIndex+regAddr];     //发送一字节数据
            }
            else                          //接收到一字节数据
            {
                if(rxIndex == 0xff)  //第一个数据为数据首地址
                {
                    regAddr=I2CDAT;
                    rxIndex=0;
                    I2CCON |= AAK(1);
                }
                else                          //数据
                {
                    rxBuf[rxIndex+regAddr] = I2CDAT;     //接收一个字节数据
                    rxIndex++;
                    I2CCON |= AAK(1);
                }
            }
        }
```

```
            else if(Sta_Temp==0xA8)                    //接收到从机地址+读位，应答信号已发送
            {
                I2CDAT = rxBuf[rxIndex+regAddr];        //预设待发送的一字节数据
                iicReadMode = 1;                //设置当前为读状态
            }
            I2CFLG   |= I2CF;  //清除 I2C 中断标志
        }
}
void main(void)
{
        I2C_init();
        EA = 1; //开启全局中断
        while(1)
        {
        }
}
```

---

20 LCD/LED 驱动

20.1 LCD 驱动

20.1.1 功能简介

内置 LCD 驱动最大可支持 8com x 32seg、7com x 33seg、6com x 34seg、5com x 35seg、4com x 36seg，共 40 个输出引脚。可编程占空比为：1/2、1/3、1/4、1/5、1/6、1/7、1/8。可编程偏压比为：1/2、1/3、1/4。可编程 LCD 驱动强度为 8 个等级。图 20-1-1-1 是 LCD 的原理示意图。

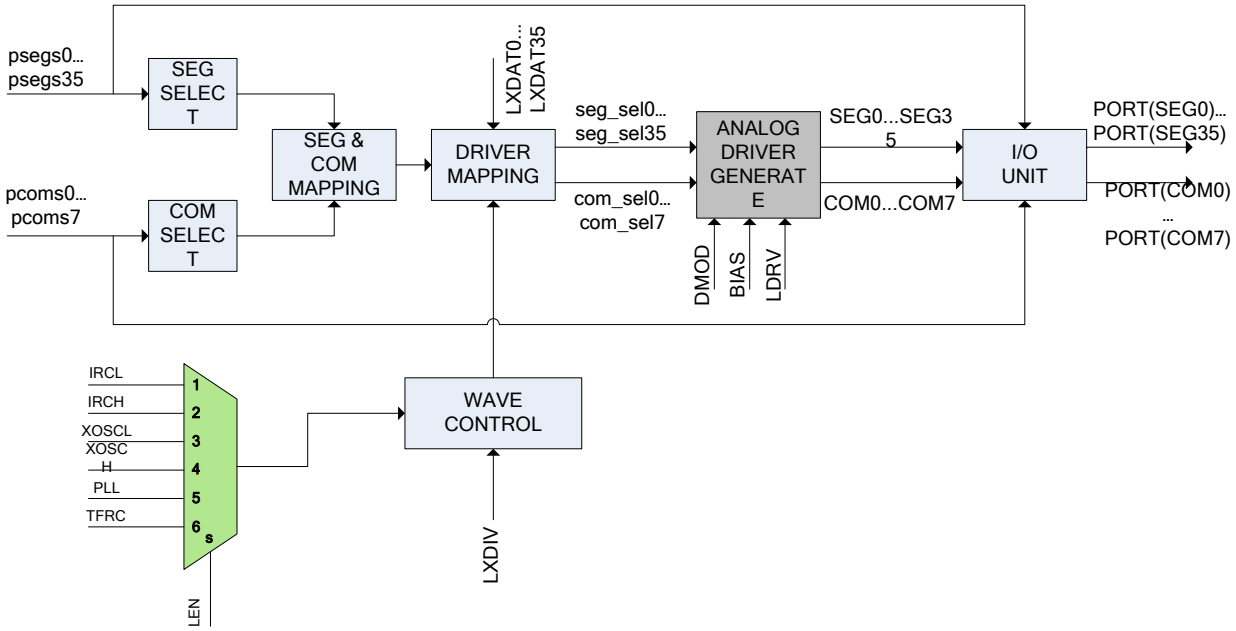


图 20-1-1-1 LCD 原理示意图

20.1.2 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**

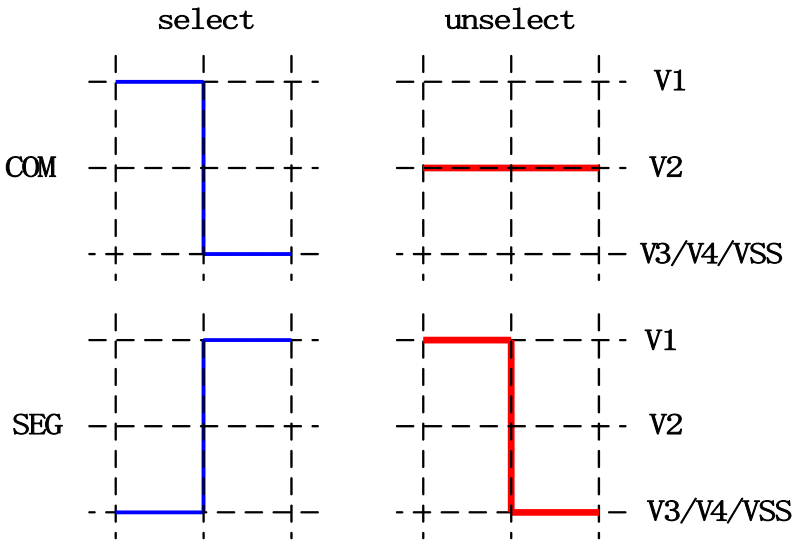


图 20-1-1-2　LCD 偏压比 1/2 信号

- **LCD 偏压比 1/3**

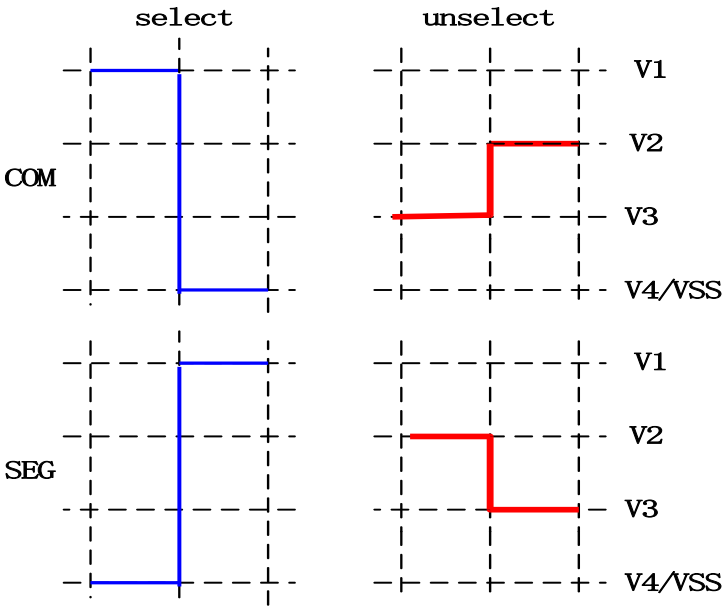


图 20-1-1-3　LCD 偏压比 1/3 信号

- **LCD 偏压比 1/4**

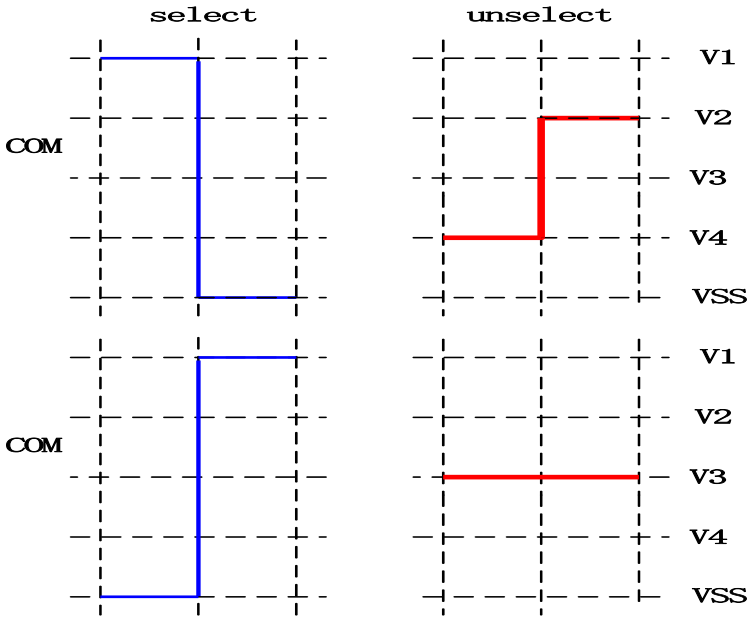


图 20-1-1-4　LCD 偏压比 1/4 信号

20.1.3 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 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。

LCD 可编程的占空比为：1/2、1/3、1/4、1/5、1/6、1/7、1/8，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 8 个 COM，占空比就为 1/8。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 COM3、COM5、COM7 为 COM 口，那 COM3 对应实际 COM0,COM5 对应实际 COM1，COM7 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 36 字节的 LCD 显示缓存。LCD 显示缓存是与实际的 COM,SEG 对应的，36 字节显示缓存按顺序分别对应 SEG0~SEG35，而 COM0~COM7 对应每个字节的 0~7 位。显示缓存是通过索引寄存器 INDEX 和数据寄存器 LXDAT 来访问的，设置 INDEX 为 0~35 按顺序对应 SEG0~SEG35 的显示缓存。

20.2 LED 驱动

20.2.1 功能简介

内置 LED 驱动最大可支持 8com x 32seg，与 LCD 驱动共用显示缓存和驱动引脚。LED 驱动有 8 级亮度可调。内置了全局闪烁功能，在扫描频率为 256Hz 时以 1Hz 的频率闪烁。图 20-2-1-1 是 LED 的原理示意图。

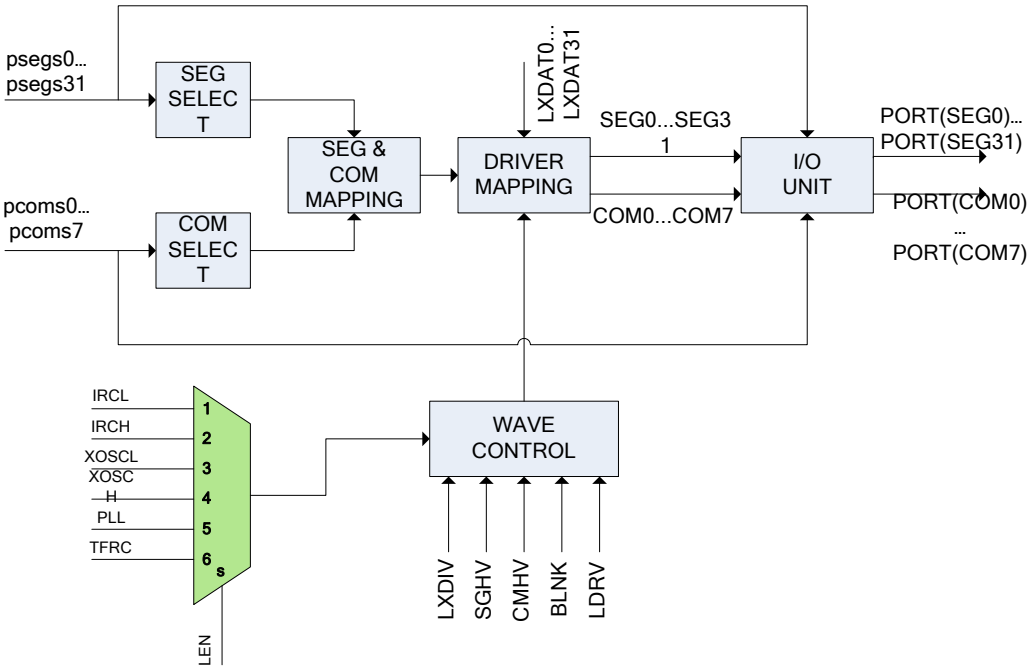


图 20-2-1-1 LED 原理示意图

图 20-2-1-1 是 LED 的驱动波形图。

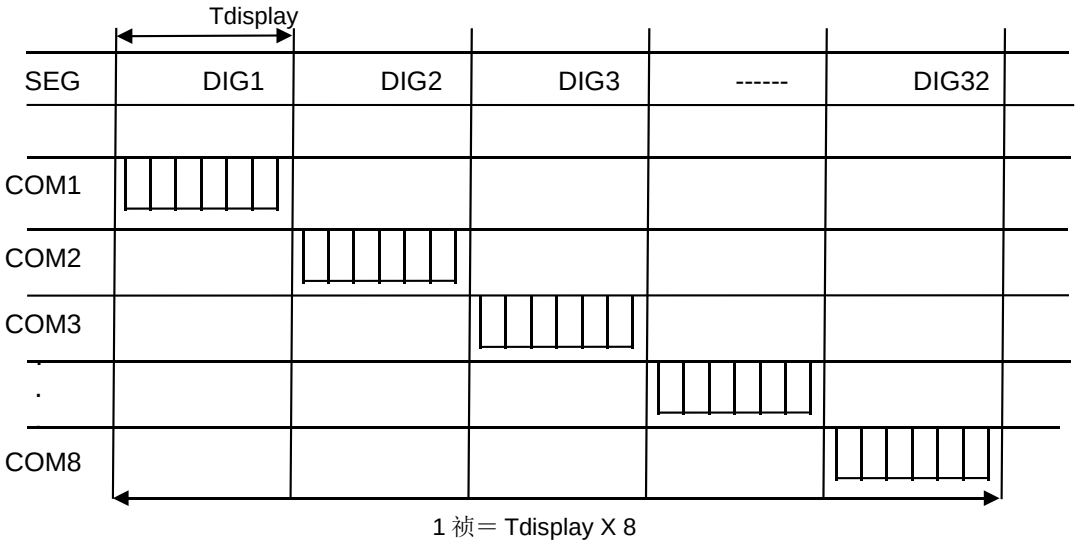


图 20-2-1-2 LED 驱动波形

20.2.2 LED 功能描述

LED 模式通过设置 LMOD=0 来选择。LED 驱动可以通过 LEN 选择时钟源，当时钟源被选择后，LED 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 LED 时钟分频器，针对不同的时钟源可设置不同的分频系数，LED 扫描的帧频率典型值为 256Hz。LED 驱动有 8 级亮度可调，通过 LDRV 来设置，不同的数值对应不同的占空比。

和 LCD 驱动相同的是，LED 的基本占空比也是由使能的 COM 数量决定的，COM 引脚和 SEG 引脚也可以任意组合，可以参考 LCD 部分的相关描述。

LED 驱动和 LCD 驱动共同显示缓存，显示缓存和 COM、SEG 口的对应关系和显示缓存的访问方式都和 LCD 驱动一样，也可参考 LCD 驱动部分的相关描述。

20.3 LCD/LED 寄存器描述

表 20-3-1 寄存器 LXCON

E1H	7	6	5	4	3	2	1	0
LXCON	LEN[2:0]			LMOD	-	-	-	-
R/W	R/W			R/W	-	-	-	-
初始值	0	0	0	0	-	-	-	-

位编号	位符号	说明
7~5	LEN	LCD/LED 时钟选择位 001：IRCL 010：IRCH 011：XOSCL 100：XOSCH 101：PLL 110：TFRC 其他:模块关闭
4	LMOD	模式选择位 0：LCD 模式 1：LED 模式
3~0	-	-

表 20-3-2 寄存器 LXCFG

E2H	7	6	5	4	3	2	1	0
LCD 模式								
LXCFG	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

LED 模式								
LXCFG	-	-	COMHV	SEGHV	BLNK	LDRV[2:0]		
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
LCD 模式		
7~6	DMOD	LCD 驱动电流大小选择位 00：5uA 01：40uA 10：80uA 11：130uA
5~4	BIAS	LCD 偏压选择位 01：1/2 Bias 10：1/3 Bias 其他：1/4 Bias
3	-	-
2~0	LDRV	LCD 驱动强度控制位 000：Level 1（最小） 001：Level 2 … 111：Level 8（最大）
LED 模式		
7~6	-	-
5	COMHV	当为 0/1 时，表示 COM 输出 0/1 有效
4	SEGHV	当为 0/1 时，表示 SEG 输出 0/1 有效
3	BLNK	LED 全局闪烁控制位，1 有效 备注： 闪烁时间为 128 帧 LED 的输出。例如，LED 工作时钟选择 32.768KHz，同时 LXDIV=0，此时 LED 的输出频率为 256Hz，那么闪烁的频率是 1Hz。
2~0	LDRV	LED 亮度调节位 000：Level 1（最暗） 001：Level 2 010：Level 3 011：Level 4 100：Level 5 101：Level 6 110：Level 7 111：Level 8（最亮）

表 20-3-3 寄存器 LXDIV

E4H	7	6	5	4	3	2	1	0
LXDIVL	LXDIV[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
LXDIVH	-	-	-	-	LXDIV[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	LXDIV	LXD 时钟分频器 LCD 扫描帧频率=LXD 时钟频率 ÷((LXDIV+1) x 512) LED 选择高速时钟扫描帧频率=LXD 时钟频率 ÷((LXDIV+1) x 1024) LED 选择低速时钟扫描帧频率=LXD 时钟频率 ÷((LXDIV+1) x 128) 备注： 1. LED 选择高速时钟表示，LED 工作时钟选择 IRCH/XOSCH/PLL/TFRC；LED 选择低速时钟表示，LED 工作时钟选择 IRCL/XOSCL。 2. 当 LCD/LED 时钟选择为 IRCL 时，时钟频率为 IRCL 的 1/4。 3. LCD 典型扫描帧频率为 64Hz，LED 典型扫描帧频率为 256Hz。

表 20-3-4 寄存器 LXDAT

E3H	7	6	5	4	3	2	1	0
LXDAT	LXDAT[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
备注：LXDAT 是带索引的寄存器，设置 INDEX=0~35 分别对应 LXDAT0~LXDAT35								

位编号	位符号	说明
7~0	LXDAT	显示缓存读写寄存器 *备注：对于 LED 驱动，LXDAT32~LXDAT35 无作用。*

表 20-3-5 LCD/LED 显示缓存

INDEX	SEG	COM0	COM1	COM2	COM3	COM4	COM5	COM6	COM7
0	0	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
1	1	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
2	2	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
3	3	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
4	4	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
5	5	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7

6	6	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
7	7	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
8	8	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
9	9	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
10	10	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
11	11	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
12	12	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
13	13	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
14	14	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
15	15	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
16	16	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
17	17	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
18	18	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
19	19	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
20	20	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
21	21	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
22	22	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
23	23	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
24	24	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
25	25	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
26	26	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
27	27	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
28	28	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
29	29	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
30	30	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
31	31	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
32	32	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	-
33	33	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	-	-
34	34	BIT0	BIT1	BIT2	BIT3	BIT4	-	-	-
35	35	BIT0	BIT1	BIT2	BIT3	-	-	-	-

20.4 LCD 驱动控制例程

例如，要驱动 LCD 为 8com×20seg，1/4bias，程序如下：

```
--------------------------------------------------------------------------------
#define XLCKE           (1<<3)
#define XLSTA           (1<<2)

#define LEN_XOSCL           (3<<5)
#define LMOD_lcd        (0<<4)
#define DMOD_5ua        (0<<6)
#define BIAS_1_4        (0<<4)
#define LCDRV_LEV(N)        (N) //N=0~7
void LCD_init(void)
{
    unsigned char i;

    P00F = 3; //设置 P00 为 COM0
    P01F = 3; //设置 P01 为 COM1
    P02F = 3; //设置 P02 为 COM2
    P03F = 3; //设置 P03 为 COM3
    P04F = 3; //设置 P04 为 COM4
    P05F = 3; //设置 P05 为 COM5
    P06F = 3; //设置 P06 为 COM6
    P07F = 3; //设置 P07 为 COM7

    P57F = 3; //设置 P57 为 SEG0
    P34F = 3; //设置 P34 为 SEG1
    P35F = 3; //设置 P35 为 SEG2
    P56F = 3; //设置 P56 为 SEG3
    P50F = 3; //设置 P50 为 SEG4
    P51F = 3; //设置 P51 为 SEG5
    P52F = 3; //设置 P52 为 SEG6
    P53F = 3; //设置 P53 为 SEG7
    P54F = 3; //设置 P54 为 SEG8
    P55F = 3; //设置 P55 为 SEG9
    P60F = 3; //设置 P60 为 SEG10
    P61F = 3; //设置 P61 为 SEG11
    P62F = 3; //设置 P62 为 SEG12
    P63F = 3; //设置 P63 为 SEG13
    P64F = 3; //设置 P64 为 SEG14
    P65F = 3; //设置 P65 为 SEG15
    P10F = 3; //设置 P10 为 SEG16
```

```
    P11F = 3; //设置 P11 为 SEG17
    P12F = 3; //设置 P12 为 SEG18
    P13F = 3; //设置 P13 为 SEG19

    CKCON |= XLCKE;//打开 XOSCL 时钟
    while(!(CKCON & XLSTA)); //等待 XOSCL 时钟稳定

    LXDIVH = 0;             //设置时钟分频，分频后 LCD 帧频率为 64Hz
    LXDIVL = 0;
    LXCFG =  DMOD_5ua | BIAS_1_4 | LCDRV_LEV(7); //设置 LCD 驱动电流、bias、驱动强度
    LXCON =   LEN_XOSCL | LMOD_lcd;      //选择 LCD 时钟源为 XOSCL、设置为 LCD 模式

    for(i=0;i<20;i++)    //写 LCD 显示缓存
    {
        INDEX = i;          //设置显示缓存索引
        LXDAT = 0;          //写入缓存，当前写 0 为清屏
    }
}
-------------------------------------------------------------------------------
```

20.5 LED 驱动控制例程

例如，要驱动 LED 为 8com×20seg，共阴接法，程序如下：

```
-------------------------------------------------------------------------------
#define XLCKE           (1<<3)
#define XLSTA           (1<<2)

#define LEN_XOSCL           (3<<5)
#define LMOD_led        (1<<4)
#define CMHV(N)             (N<<5)   //N=0~1
#define SGHV(N)             (N<<4)   //N=0~1
#define BLNK(N)             (N<<3)   //N=0~1
void LED_init(void)
{
    unsigned char i;

    P00F = 3; //设置 P00 为 COM0
    P01F = 3; //设置 P01 为 COM1
    P02F = 3; //设置 P02 为 COM2
    P03F = 3; //设置 P03 为 COM3
    P04F = 3; //设置 P04 为 COM4
```

```
    P05F = 3; //设置 P05 为 COM5
    P06F = 3; //设置 P06 为 COM6
    P07F = 3; //设置 P07 为 COM7

    P57F = 3; //设置 P57 为 SEG0
    P34F = 3; //设置 P34 为 SEG1
    P35F = 3; //设置 P35 为 SEG2
    P56F = 3; //设置 P56 为 SEG3
    P50F = 3; //设置 P50 为 SEG4
    P51F = 3; //设置 P51 为 SEG5
    P52F = 3; //设置 P52 为 SEG6
    P53F = 3; //设置 P53 为 SEG7
    P54F = 3; //设置 P54 为 SEG8
    P55F = 3; //设置 P55 为 SEG9
    P60F = 3; //设置 P60 为 SEG10
    P61F = 3; //设置 P61 为 SEG11
    P62F = 3; //设置 P62 为 SEG12
    P63F = 3; //设置 P63 为 SEG13
    P64F = 3; //设置 P64 为 SEG14
    P65F = 3; //设置 P65 为 SEG15
    P10F = 3; //设置 P10 为 SEG16
    P11F = 3; //设置 P11 为 SEG17
    P12F = 3; //设置 P12 为 SEG18
    P13F = 3; //设置 P13 为 SEG19

    CKCON |= XLCKE;//打开 XOSCL 时钟
    while(!(CKCON & XLSTA)); //等待 XOSCL 时钟稳定

    LXDIVH = 0;             //设置时钟分频，分频后 LED 帧频率为 256Hz
    LXDIVL = 0;
    LXCFG = CMHV(0) | SGHV(1) | BLNK(0) | LEDRV_LEV(7);     //设置 COM 低有效，SEG 高有效，全屏闪烁
    关闭，驱动亮度为最大
    LXCON =   LEN_XOSCL | LMOD_led;       //选择 LED 时钟源为 XOSCL、设置为 LED 模式

    for(i=0;i<20;i++)    //写 LED 显示缓存
    {
        INDEX = i;          //设置显示缓存索引
        LXDAT = 0;          //写入缓存，当前写 0 为清屏
    }
}
```

---

21 PWM

21.1 PWM 功能简介

CA51F2 系列芯片最多有 8 通道 PWM 输出，PWM 周期和占空比可在 16 位范围内任意配置。每路 PWM 都支持边沿对齐模式和中心对称模式。此外，PWM 还支持死区控制及互补输出，设置为互补模式时，8 路 PWM 组成 4 对互补通道。PWM 的这种特性是专门针对无刷直流电机驱动而设计的。

21.2 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 模块有一个 PWM 数据更新寄存器 PWMUPD，当改写寄存器 PWMDIV、PWMDUT 和 PWMCKD 时，寄存器 PWMUPD 必须置位相应位才会更新数据，当数据刷新之后 PWMUPD 相应位自动清 0。PWM 可通过 PWMTOG 位设置 PWM 引脚输出反相。PWM 有多种时钟源可以选择，时钟源是以两路 PWM 为单位进行设置的，分别是：PWM0 和 PWM1、PWM2 和 PWM3、PWM4 和 PWM5、PWM6 和 PWM7，也就是说，每组 PWM 的时钟源是共同设置的，时钟源通过 PWM0、PWM2、PWM4、PWM6 对应的控制寄存器 PWMCON 的 PWMCKS 来选择。另外，每路 PWM 的时钟分频可通过 PWMCKD 独立设置。

- **边沿对齐模式和中心对齐模式**

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

当边沿对齐模式和中心对齐模式单路 PWM 输出波形见图 21-2-1 和图 21-2-2（注：以下所有 PWM 波形满足条件 PWMDIV>PWMDUT>0）。如图所示，设置相同的 PWMDIV 和 PWMDUT 值，中心对齐模式一个 PWM 周期是边沿对齐模式的两倍。

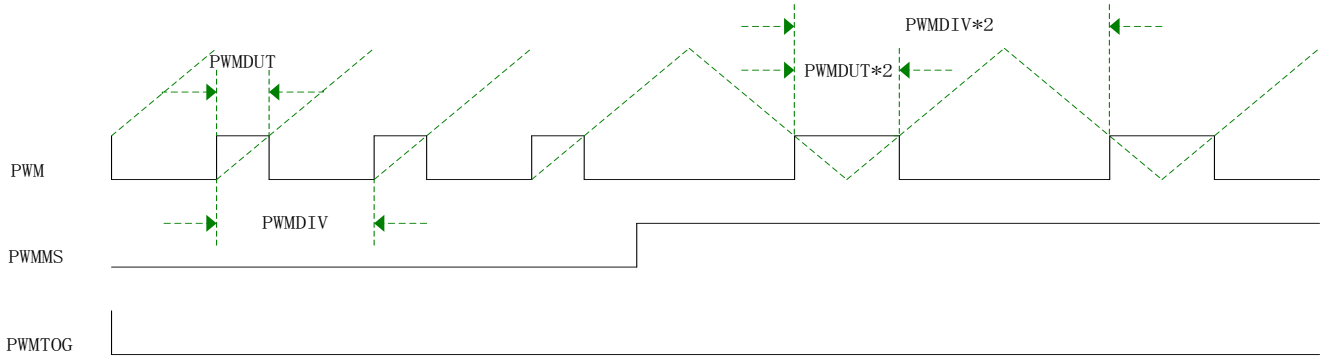


图 21-2-1　PWMTOG＝0 时 PWM 输出波形

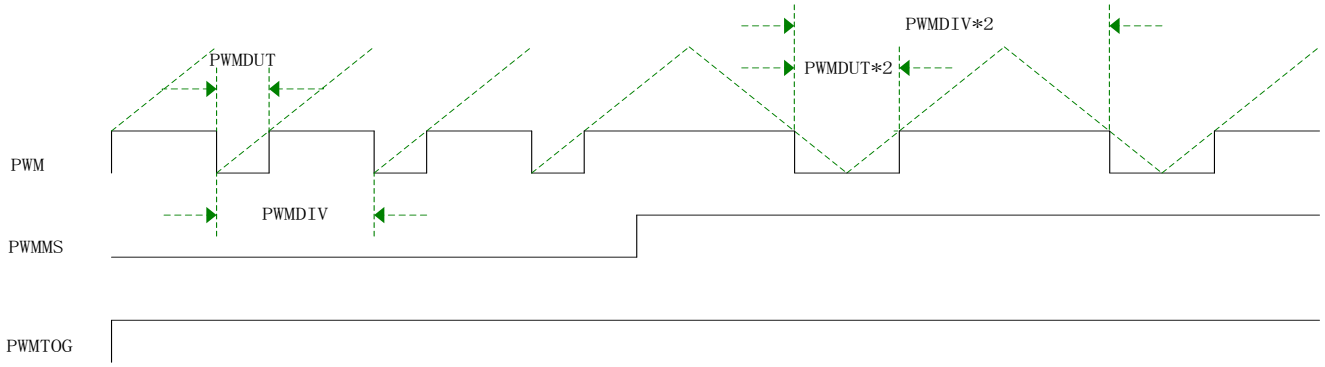


图 21-1-2　PWMTOG＝1 时 PWM 输出波形

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

- **互补模式**

在互补模式下，8 路 PWM 可组成 4 对互补通道：PWM0 和 PWM1、PWM2 和 PWM3、PWM4 和 PWM5、PWM6 和 PWM7。PWM 的互补模式是通过 PWM1、PWM3、PWM5、PWM7 的控制寄存器 PWMCON 的 PWMMOD 位设置的。在互补模式，PWM 对齐方式、周期、占空比、预分频时钟都是由 PWM0、PWM2、PWM4、PWM6 对应的寄存器设置，只有 PWMTOG 仍是由各通道对应寄存器独立控制。PWM 互补模式原理图如图 21-1-3 所示。

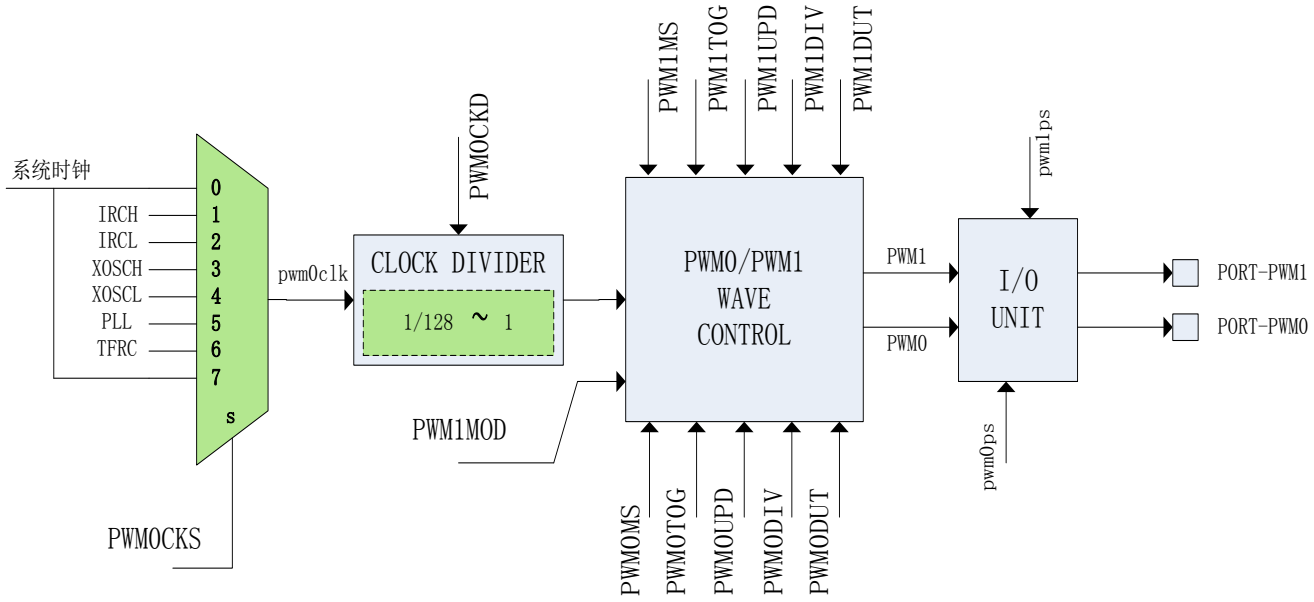


图 21-1-3　PWM0、PWM1 原理示意图

每组 PWM 输出的波形在相位上互补，如图 21-1-4 和图 21-1-5 所示（以 PWM0、PWM1 为例）。

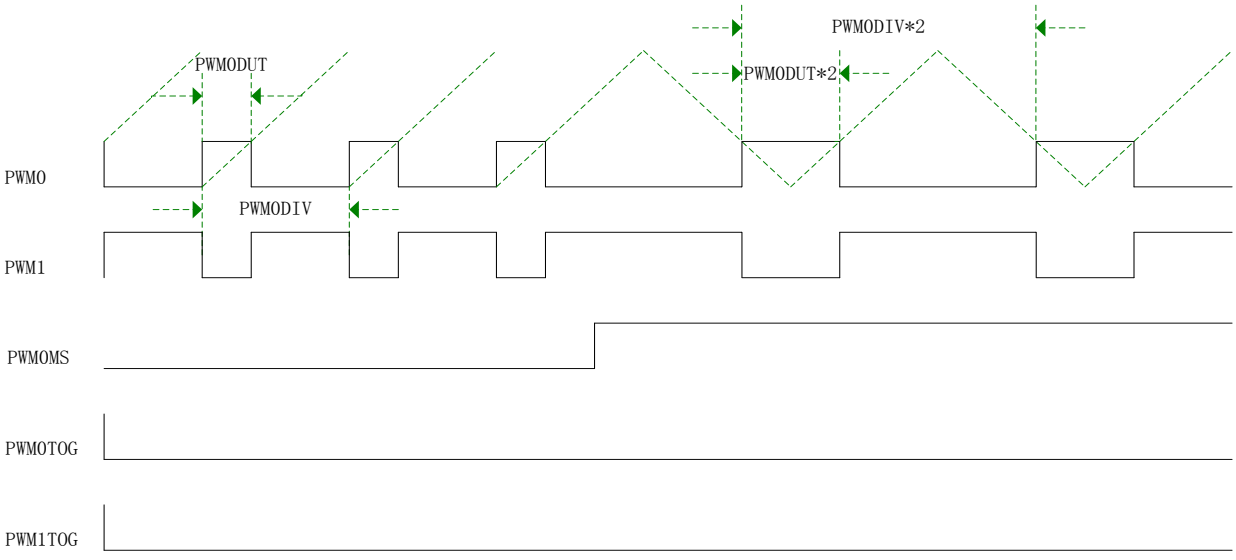


图 21-1-4　PWMTOG＝0 时 PWM0、PWM1 输出互补波形

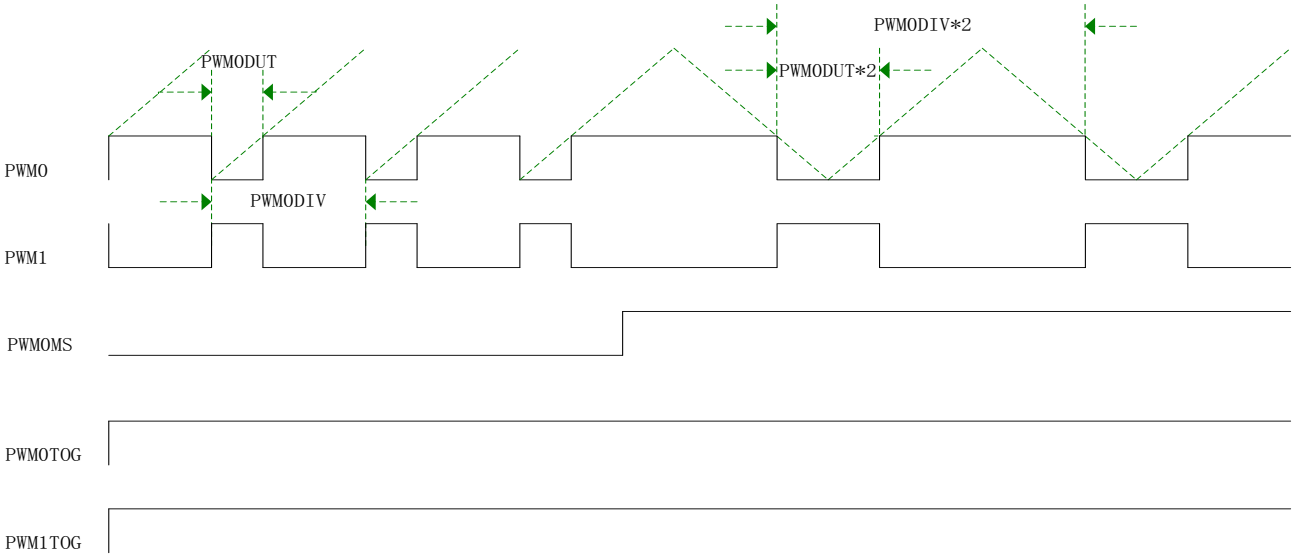


图 21-1-5　PWMTOG＝1 时 PWM0、PWM1 输出互补波形

- **死区控制**

在桥形驱动电路中，为防止上下半桥同时导通，需要在 PWM 互补信号中插入死区控制。死区时间由 PWM1、PWM3、PWM5、PWM7 对应的寄存器 PWMDIV 和 PWMDUT 设置，PWMDIV 设置的是左边的死区时间，而 PWMDUT 设置的是右边的死区时间。设置死区时间需要满足以下条件（以 PWM0、PWM1 为例）：

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

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

死区控制输出波形如图 21-1-6 所示（以 PWM0、PWM1 为例）。

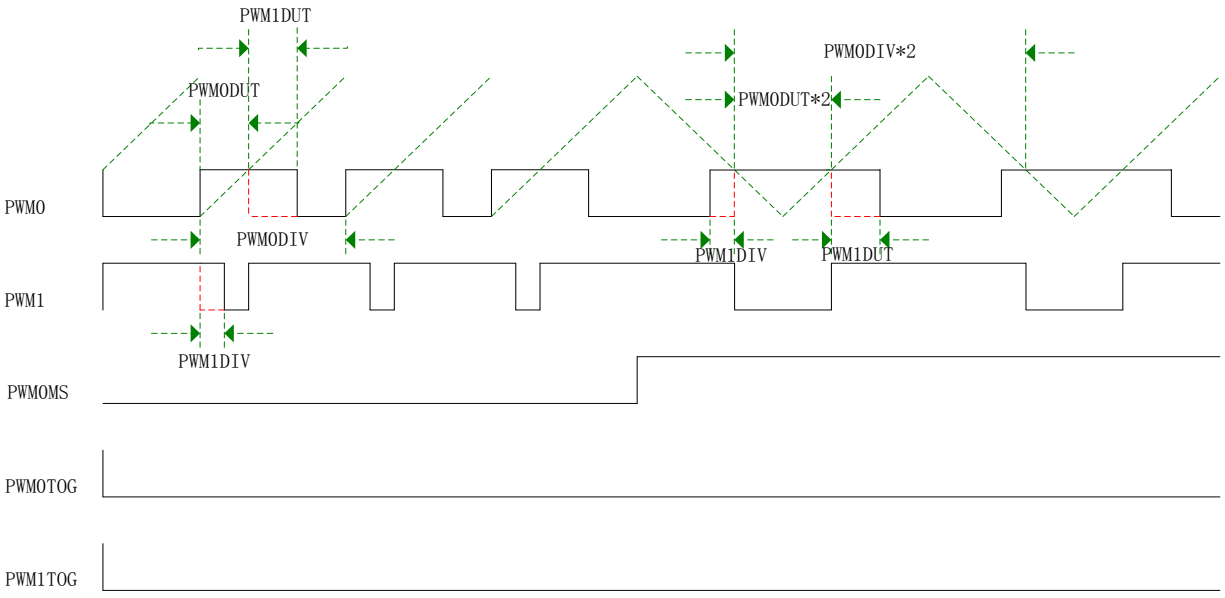


图 21-1-6　PWMTOG＝0 时 PWM0、PWM1 死区控制波形

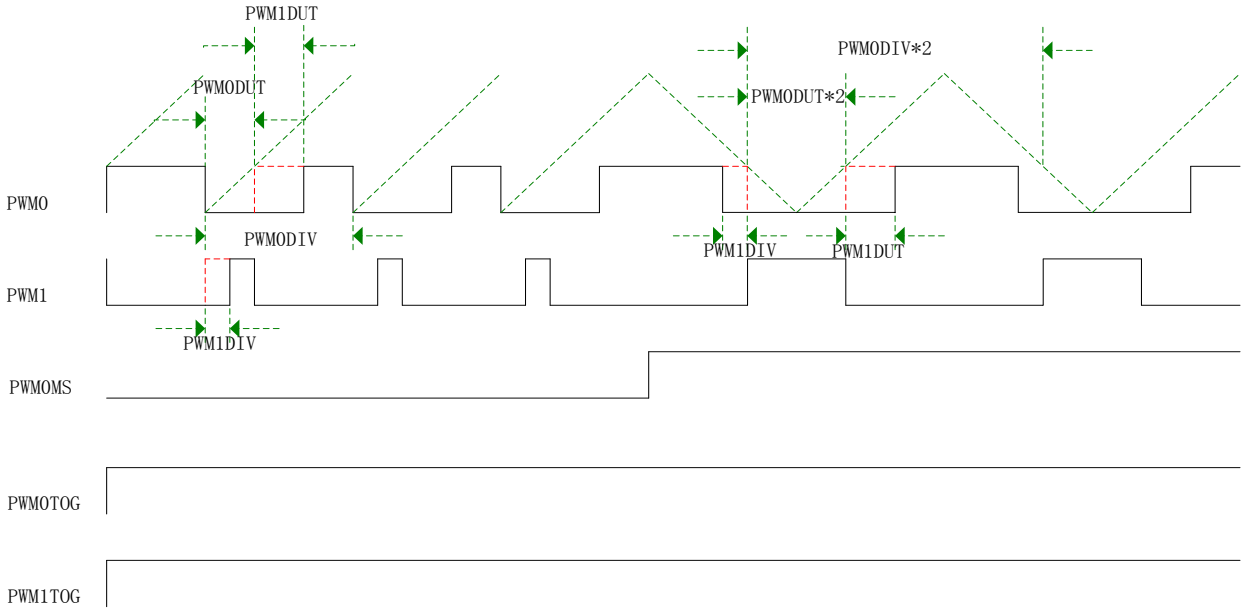


图 21-1-6　PWMTOG＝1 时 PWM0、PWM1 死区控制波形

- **PWM 中断**

PWM 中断通过寄存器 PWMCON 的 PWMTIE、PWMZIE、PWMPIE、PWMNIE 位使能，PWMTIE 位对应的是 PWM 计数器计数到顶点（即等于 PWMDIV)产生的中断，PWMZIE 位对应的是 PWM 计数器计数到最低点（即等于 0）时产生的中断，PWMNIE 对应的是输出引脚下降沿产生的中断，PWMPIE 对应的是输出引脚上升沿产生的中断。其中，在边沿对齐模式，没有 PWMTIE 位和 PWMZIE 位对应的中断。寄存器 PWMAIF、PWMBIF、PWMCIF、PWMDIF 是 8 个通道的中断状态寄存器，其中，PWMxTIF、PWMxZIF、PWMxNIF、PWMxPIF 分别对应使能位 PWMTIE、PWMZIE、PWMNIE、PWMPIE。

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

21.3　PWM 寄存器描述

表 21-1-1 寄存器 PWMEN

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

表 21-1-2 寄存器 PWMUPD

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

表 21-1-3 寄存器 PWMCMX

DCH	7	6	5	4	3	2	1	0
PWMCMX	PWMCMX[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
备注：PWMCMX 是带索引的寄存器，设置 INDEX=0~7 分别对应 PWMCMX0~PWMCMX7								
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 21-1-4 寄存器 PWMCON

DDH	7	6	5	4	3	2	1	0
PWMCON①	PWMTIE	PWMZIE	PWMPIE	PWMNIE	PWMMS	PWMCKS[2:0]		
R/W	R/W	R/W	R/W	R/W	R/W	R/W		
初始值	0	0	0	0	0	0	0	0
PWMCON②	PWMTIE	PWMZIE	PWMPIE	PWMNIE	PWMMS	-	-	PWMMOD
R/W	R/W	R/W	R/W	R/W	R/W	-	-	R/W
初始值	0	0	0	0	0	-	-	0
位编号	位符号	说明						
备注： *PWMCON①是 PWM0/PWM2/PWM4/PWM6 通道的控制寄存器* *PWMCON②是 PWM1/PWM3/PWM5/PWM7 通道的控制寄存器* *PWMCON 是带索引的寄存器，设置 INDEX=0~7 分别对应 PWMCON0~PWMCON7*								
7	PWMTIE	PWM 计数器顶点中断使能控制位，1 有效						

6	PWMZIE	PWM 计数器最低点中断使能控制位，1 有效
5	PWMPIE	PWM 上升沿中断使能控制位，1 有效
4	PWMNIE	PWM 下降沿中断使能控制位，1 有效
3	PWMMS	PWM 模式选择位 0：边沿对齐模式 1：中心对齐模式
2~0	PWMCKS	PWM 工作时钟选择位 001：IRCH 010：IRCL 011：XOSCH 100：XOSCL 101：PLL 110：TFRC 其他：系统时钟 *备注：* *PWM0/PWM1，都由 PWMCKS0 来配置；* *PWM2/PWM3，都由 PWMCKS2 来配置；* *PWM4/PWM5，都由 PWMCKS4 来配置；* *PWM6/PWM7，都由 PWMCKS6 来配置。*
0	PWMMOD	互补模式使能寄存器，1 有效 *备注：* *设置 PWMMOD1=1，PWM0、PWM1 进入互补模式* *设置 PWMMOD3=1，PWM2、PWM3 进入互补模式* *设置 PWMMOD5=1，PWM4、PWM5 进入互补模式* *设置 PWMMOD7=1，PWM6、PWM7 进入互补模式*

表 21-1-5 寄存器 PWMCFG

DEH	7	6	5	4	3	2	1	0
PWMCFG	PWMTOG	PWMCKD[6:0]						
R/W	R/W	R/W						
初始值	0	0	0	0	0	0	0	0
备注：PWMCFG 是带索引的寄存器，设置 INDEX=0~7 分别对应 PWMCFG0~PWMCFG7								

位编号	位符号	说明
7	PWMTOG	PWM 输出取反使能寄存器，1 有效
6~0	PWMCKD	PWM 工作时钟预分频配置寄存器 0000000：不分频 0000001：2 分频 0000010：3 分频 …… 1111110：127 分频 1111111：128 分频

表 21-1-6 寄存器 PWMDIVL、PWMDIVH

DFH	7	6	5	4	3	2	1	0
PWMDIVL	PWMDIV[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
D1H	7	6	5	4	3	2	1	0
PWMDIVH	PWMDIV[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
备注：PWMDIV 是带索引的寄存器，设置 INDEX=0~7 分别对应 PWMDIV0~PWMDIV7								

位编号	位符号	说明
15~0	PWMDIV	PWM 周期配置寄存器 在互补模式下，PWMDIV1/PWMDIV3/PWMDIV5/PWMDIV7 有不同的含义，参考寄存器 PWMDUT 相关描述

表 21-1-7 寄存器 PWMDUTL、PWMDUTH

D2H	7	6	5	4	3	2	1	0
PWMDUTL	PWMDUT[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
D3H	7	6	5	4	3	2	1	0
PWMDUTH	PWMDUT[15:8]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
备注：PWMDUT 是带索引的寄存器，设置 INDEX=0~7 分别对应 PWMDUT0~PWMDUT7								

<table>
<tr><th>位编号</th><th>位符号</th><th>说明</th></tr>
<tr><td>15~0</td><td>PWMDUT</td><td>PWM 占空比配置寄存器
在互补模式下，PWMDUT1/PWMDUT3/PWMDUT5/PWMDUT7 有不同的含义，如下表：
<table>
<tr><td>PWMDIV1</td><td>控制 PWM0/PWM1 的左边的死区的宽度</td></tr>
<tr><td>PWMDUT1</td><td>控制 PWM0/PWM1 的右边的死区的宽度</td></tr>
<tr><td>PWMDIV3</td><td>控制 PWM2/PWM3 的左边的死区的宽度</td></tr>
<tr><td>PWMDUT3</td><td>控制 PWM2/PWM3 的右边的死区的宽度</td></tr>
<tr><td>PWMDIV5</td><td>控制 PWM4/PWM5 的左边的死区的宽度</td></tr>
<tr><td>PWMDUT5</td><td>控制 PWM4/PWM5 的右边的死区的宽度</td></tr>
<tr><td>PWMDIV7</td><td>控制 PWM6/PWM7 的左边的死区的宽度</td></tr>
<tr><td>PWMDUT7</td><td>控制 PWM6/PWM7 的右边的死区的宽度</td></tr>
</table></td></tr>
</table>

表 21-1-8 寄存器 PWMAIF

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

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

表 21-1-9 寄存器 PWMBIF

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

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

表 21-1-10 寄存器 PWMCIF

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

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

表 21-1-11 寄存器 PWMDIF

D7H	7	6	5	4	3	2	1	0
PWMDIF	PWM7TIF	PWM7ZIF	PWM7PIF	PWM7NIF	PWM6TIF	PWM6ZIF	PWM6PIF	PWM6NIF
R/W	R	R	R	R	R	R	R	R
初始值	0	0	0	0	0	0	0	0

位编号	位符号	说明
7	PWM7TIF	PWM7 计数器顶点中断标志位，写 1 清 0
6	PWM7ZIF	PWM7 计数器最低点中断标志位，写 1 清 0
5	PWM7PIF	PWM7 上升沿中断标志位，写 1 清 0
4	PWM7NIF	PWM7 下降沿中断标志位，写 1 清 0
3	PWM6TIF	PWM6 计数器顶点中断标志位，写 1 清 0
2	PWM6ZIF	PWM6 计数器最低点中断标志位，写 1 清 0
1	PWM6PIF	PWM6 上升沿中断标志位，写 1 清 0
0	PWM6NIF	PWM6 下降沿中断标志位，写 1 清 0

21.4 PWM 功能控制例程

◆ **PWM 单路输出例程**

例如，PWM0 输出频率为 30K 的时钟，占空比为 30%，程序如下：

```
--------------------------------------------------------------------------------
#define PWM_CH0        0

#define TIE(N)              (N<<7)    //N=0~1
#define ZIE(N)              (N<<6)    //N=0~1
#define PIE(N)              (N<<5)    //N=0~1
#define NIE(N)                   (N<<4)    //N=0~1
#define MS(N)               (N<<3)    //N=0~1

#define CKS_IH                   (1<<0)
```

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

void PWM_init(void)
{
    P50F = 5;              //设置 P50 为 PWM 引脚
    INDEX = PWM_CH0;    //设置 INDEX 指向 PWM0
    PWMCON = TIE(0) | ZIE(0) | PIE(0) | NIE(0) | MS(0) | CKS_IH ;//设置 PWM0 中断关闭，设置 PWM0 为 边沿对齐模式，设置 PWM0 时钟源为 IRCH
    PWMCFG = TOG(0) | 0; //设置反相关闭，时钟不分频

    PWMDIVH    = 0;
    PWMDIVL    = 123;    //3686400/30000=123
    PWMDUTH    = 0;
    PWMDUTL    = 37;//123*0.3=37

    PWMUPD  |= (1<<PWM_CH0);  //PWM 设置更新
    while(PWMUPD);  //等待 PWM 设置更新完成，在使能 PWM 之前必须要执行此操作

    PWMEN  |= (1<<PWM_CH0);    //PWM0 使能
}
```

例如，PWM0 要直接输出 IRCH 的时钟，程序如下：

```
void PWM_init(void)
{
    P50F = 5;              //设置 P50 为 PWM 引脚
    INDEX = PWM_CH0;    //设置 INDEX 指向 PWM0
    PWMCON = TIE(0) | ZIE(0) | PIE(0) | NIE(0) | MS(0) | CKS_IH ;//设置 PWM0 中断关闭，设置 PWM0 为 边沿对齐模式，设置 PWM0 时钟源为 IRCH
    PWMCFG = TOG(0) | 0; //设置反相关闭，时钟不分频

    PWMDIVH    = 0;  //PWMDIV 和 PWMDUT 都设为 0 则直接输出时钟源时钟
    PWMDIVL    = 0;
    PWMDUTH    = 0;
    PWMDUTL    = 0;

    PWMUPD  |= (1<<PWM_CH0);  //PWM 设置更新
    while(PWMUPD);  //等待 PWM 设置更新完成，在使能 PWM 之前必须要执行此操作

    PWMEN  |= (1<<PWM_CH0);    //PWM0 使能
}
```

◆ **PWM 互补输出和死区控制例程**

以 PWM0,PWM1 为例，此两路 PWM 输出互补的频率为 30K 的时钟，占空比为 50%，同时插入 2 个时钟周期的死　　区时间，程序如下：

```
---------------------------------------------------------------------------------------
#define PWM_CH0        0
#define PWM_CH1        1

#define TIE(N)         (N<<7)    //N=0~1
#define ZIE(N)         (N<<6)    //N=0~1
#define PIE(N)         (N<<5)    //N=0~1
#define NIE(N)              (N<<4)    //N=0~1
#define MS(N)          (N<<3)    //N=0~1

#define CKS_IH              (1<<0)

#define TOG(N)              (N<<7)    //N=0~1
#define MOD(N)              (N<<0)    //N=0~1

void PWM_init(void)
{
    P50F = 5;          //设置 P50 为 PWM0 引脚
    P51F = 5;          //设置 P51 为 PWM1 引脚

    INDEX = PWM_CH0;    //设置 INDEX 指向 PWM0
    PWMCON = TIE(0) | ZIE(0) | PIE(0) | NIE(0) | MS(0) | CKS_IH ;//设置 PWM0 中断关闭，设置 PWM0　为　边沿对齐模式，设置 PWM0 时钟源为 IRCH
    PWMCFG = TOG(0) | 0; //设置反相关闭，时钟不分频

    PWMDIVH    = 0;
    PWMDIVL    = 123;     //3686400/30000=123
    PWMDUTH    = 0;
    PWMDUTL    = 61;//123*0.5=61

    INDEX = PWM_CH1;    //设置 INDEX 指向 PWM1
    PWMCON = TIE(0) | ZIE(0) | PIE(0) | NIE(0) | MS(0) | MOD(1) ;//设置 PWM0、PWM1 为互补模式
    PWMCFG = TOG(0) | 0; //设置反相关闭，时钟不分频

    PWMDIVH    = 0;
    PWMDIVL    = 2; //左边插入 2 个时钟周期死区，如不需插入死区时间，则设置为 0
    PWMDUTH    = 0;
    PWMDUTL    = 2; //右边插入 2 个时钟周期死区，如不需插入死区时间，则设置为 0
```

```
    PWMUPD  |= (1<<PWM_CH0)|(1<<PWM_CH1);  //PWM 设置更新
    while(PWMUPD);  //等待 PWM 设置更新完成，在使能 PWM 之前必须要执行此操作

    PWMEN   |= (1<<PWM_CH0)|(1<<PWM_CH1);    //PWM0、PWM1 使能
}
-------------------------------------------------------------------------------------
```

◆ **PWM 中断例程**

例如，PWM0 设为中心对齐模式，开启顶点、底点、上升沿和下降沿中断，程序如下：

```
-------------------------------------------------------------------------------------
#define PWM_CH0     0

#define TIE(N)          (N<<7)   //N=0~1
#define ZIE(N)          (N<<6)   //N=0~1
#define PIE(N)          (N<<5)   //N=0~1
#define NIE(N)              (N<<4)   //N=0~1
#define MS(N)           (N<<3)   //N=0~1

#define CKS_IH              (1<<0)

#define TOG(N)              (N<<7)   //N=0~1

void PWM_init(void)
{
    P50F = 5;           //设置 P50 为 PWM 引脚
    INDEX = PWM_CH0;    //设置 INDEX 指向 PWM0
    PWMCON = TIE(1) | ZIE(1) | PIE(1) | NIE(1) | MS(1) | CKS_IH ;//设置 PWM0 中断关闭，设置 PWM0 为 中心对齐模式，设置 PWM0 时钟源为 IRCH
    PWMCFG = TOG(0) | 0; //设置反相关闭，时钟不分频

     PWMDIVH    = 0;
     PWMDIVL    = 123;    //3686400/30000=123
     PWMDUTH    = 0;
     PWMDUTL    = 37;     //123*0.3=37

    PWMUPD  |= (1<<PWM_CH0); //PWM 设置更新
    while(PWMUPD);  //等待 PWM 设置更新完成，在使能 PWM 之前必须要执行此操作

    PWMEN   |= (1<<PWM_CH0);    //PWM0 使能

    PWMCMAX = 0;    //设置每个 PWM 周期都产生中断
    INT9EN = 1;         //使能 INT9 中断
}
void INT9_ISR(void) interrupt 14
{
```

```
    if(PWMAIF & TIF0)              //边沿对齐模式无效
    {
       PWMAIF = TIF0;
        //顶点中断服务程序
        ...
    }
    if(PWMAIF & ZIF0)              //边沿对齐模式无效
    {
        PWMAIF = ZIF0;
        //底点中断服务程序
        ...
    }
    if(PWMAIF & PIF0)
    {
        PWMAIF = PIF0;
        //上升沿中断服务程序
        ...
    }
    if(PWMAIF & NIF0)
    {
       PWMAIF = NIF0;
        //下降沿中断服务程序
        ...
    }
        ......
}
```

---

22 模/数字转换器（ADC）

22.1 功能简介

模拟/数字转换器是 12 位逐次逼近型（SAR）ADC，最多提供多达 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。内置比较器模式，可设定比较器阈值，超出阈值可产生中断。ADC 选择内部电压为参考电压时有自动校正功能，避免芯片一致性问题。ADC 和运放结合一起使用，可以把检测信号进行放大或缩小后再进行转换。

22.2 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。
- 选择内部参考电压时，支持自动数据校正功能
- 支持可设置的比较器模式
- 支持检测信号经过运放进行放大或缩小再进行转换
- 输入电压范围：VSS<=VIN<=VDD。

22.3 结构框图

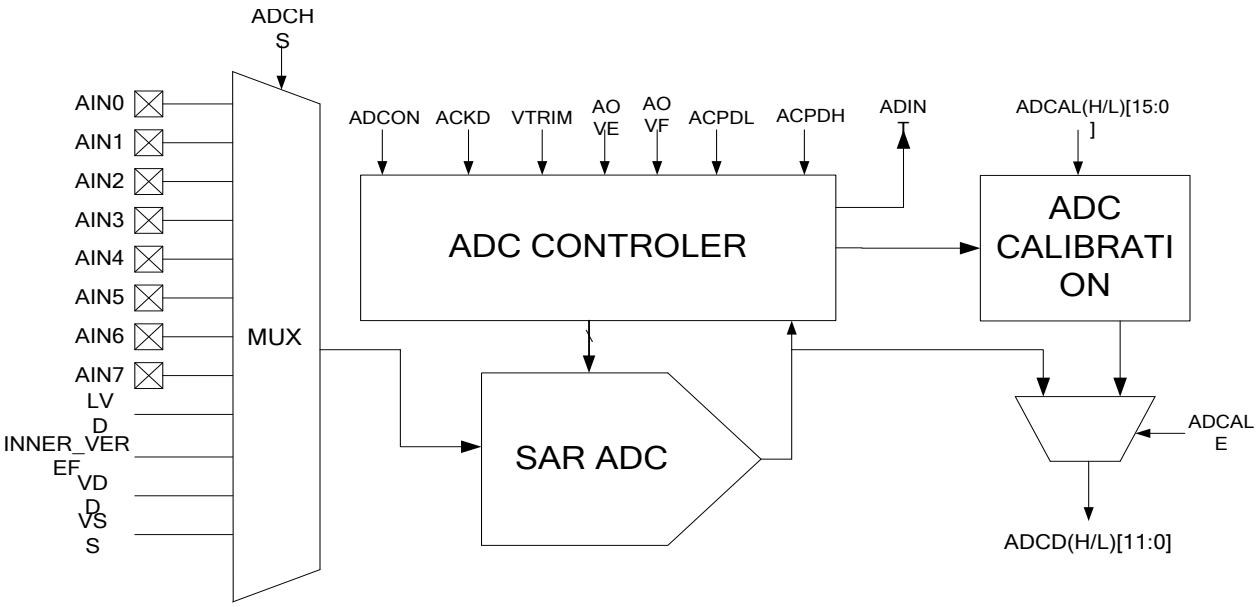


图 22-3-1 ADC 结构示意图

22.4 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。ADC 最小的转换时间为 0.5us。图 22-4-1 为 ADC 的转换时序图。

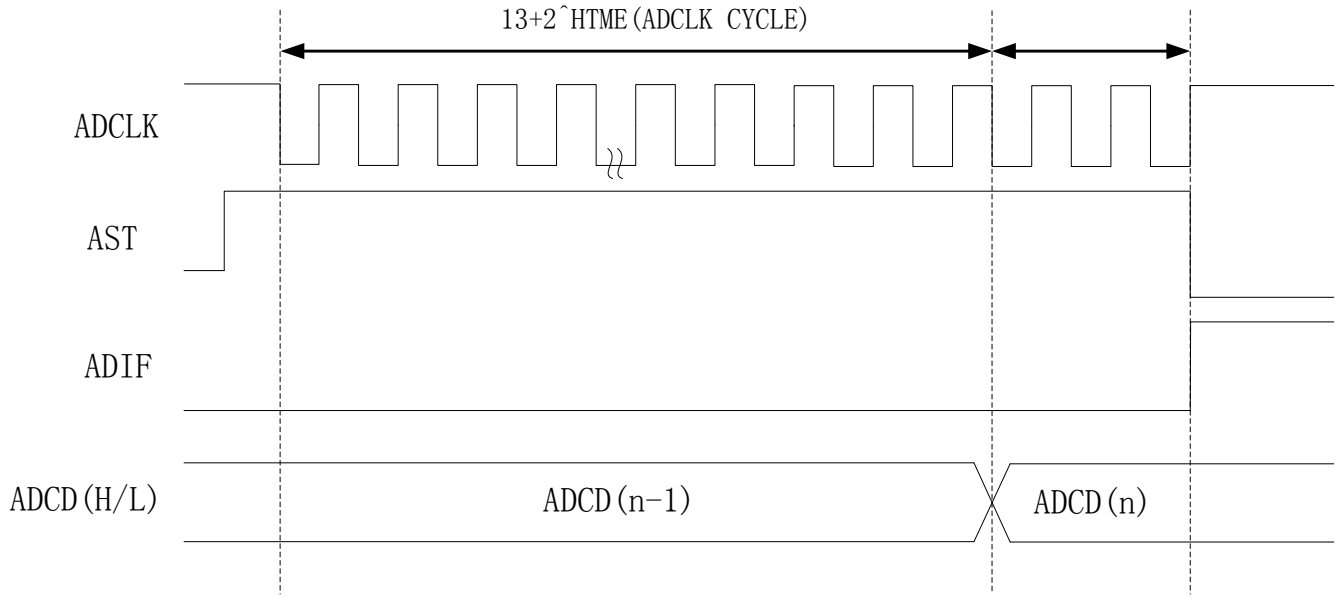


图 22-4-1 ADC 时序示意图

- **比较模式**

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

- **ADC 数据校正**

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

- **ADC 和运放结合使用**

ADC 的检测信号可通过运放 A 进行放大或缩小，详见运放部分描述。

22.5 寄存器描述

表 22-5-1 寄存器 ADCON

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

位编号	位符号	说明
7	AST	ADC 转换开始控制位，写 1 启动转换
6	ADIE	ADC 中断使能位，1 有效
5	ADCIF	ADC 转换完成标志，当 ADCIF 为 1 时，可读取 ADC 转换结果；同时 ADCIF 也是 ADC 中断标志，当 ADIE 为 1 时，ADCIF 置 1 将会产生 ADC 中断。此位通过写 1 清 0。 *备注：无论 ADIE 是否为 1，当 ADC 转换完成后 ADCIF 都会置 1，置 1 后必须写 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 22-5-2 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](P40)使能 0010：通道 AD_CH[1](P41)使能 0011：通道 AD_CH[2](P42)使能 0100：通道 AD_CH[3](P43)使能 0101：通道 AD_CH[4](P44)使能 0110：通道 AD_CH[5](P45)使能 0111：通道 AD_CH[6](P46)使能 1000：通道 AD_CH[7](P47)使能 1001：检测 VDD 的 1/4 使能 1011：检测 INNER_VREF 使能 1100：检测 LDO 电压使能 1101：检测 VSS 使能 其他：通道关闭

表 22-5-3 寄存器 ADCFGH

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

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

表 22-5-4 寄存器 ADCAL

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

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

表 22-5-5 寄存器 ADCPDL

808AH	7	6	5	4	3	2	1	0
ADCPDLL	ADCPDL[3:0]				-	-	-	-
R/W	R/W				-	-	-	-
初始值	0	0	0	0	0	0	0	0
808BH	7	6	5	4	3	2	1	0
ADCPDLH	ADCPDL[11:4]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
15~0	ADCPDL	比较模式阈值下限值设定寄存器						

表 22-5-6 寄存器 ADCPDH

808CH	7	6	5	4	3	2	1	0
ADCPDHL	ADCPDH[3:0]				-	-	-	-
R/W	R/W				-	-	-	-
初始值	0	0	0	0	0	0	0	0
808DH	7	6	5	4	3	2	1	0
ADCPDHH	ADCPDH[11:4]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
位编号	位符号	说明						
15~0	ADCPDH	比较模式阈值上限值设定寄存器						

表 22-5-7 寄存器 ADCD

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

11~0	ADCD	ADC 转换值

22.6 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：

```
---------------------------------------------------------------------------------------------
#define AST(N)          (N<<7)    //N=0~1
#define ADIE(N)         (N<<6)    //N=0~1
#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

#define ACKD(N)             (N<<5)   //N=0~7
#define ADCALE(N)           (N<<4)   //N=0~1
#define ADCHS(N)      (N)  //N=0~15       //ADC 通道选择， 1~13 对应通道 0~12
void ADC_init(void)
{
    P40F = 3; //设置 P40 为 ADC 引脚
    ADCON = AST(0) | ADIE(1) | HTME(7) | VSEL(1);//使能 ADC 中断，设置采样周期，选择 VDD 为参考电压
    ADCFGL = ACKD(1) | ADCALE(0) | ADCHS(1);//设置 ADC 时钟分频，选择 ADC 通道为 ADC0
    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;   //清除 ADC 中断标志
        AD_Value = ADCDH*256 + ADCDL;   //读取 ADC 值
        AD_Value >>= 4;
        ADCON |= AST(1); //启动下一次 AD 转换
    }
    .....
}
---------------------------------------------------------------------------------------------
```

23 模拟比较器和运放（OPCMP）

23.1 功能简介

OPCMP 模块包含 4 路比较器、两个运算放大器和一个捕获计数器。比较器可选择外部输入电压或内部 DA 为参考电压，输入信号可设置为 IO 输入或模拟输入（比较器 3 只能选择模拟输入），当为 IO 输入时，检测的实际是 IO 口的逻辑电平。每个比较器都有一个数字滤波器，滤波时间可在 15 位范围内任意设置。经过滤波器后的信号翻转可产生中断信号。模块集成了两个运放：运放 A、运放 B。运放 A、B 可对 OPIN 引脚的输入信号进行放大或缩小后从 OPOUT 引脚输出。运放 A 可以作为 ADC 的输入，实现对 ADC 检测信号的放大或缩小，扩展了 ADC 的检测电压范围。运放 B 可设置为比较器的输入，实现对小信号的检测。捕获计数器用于捕获比较器翻转的时间间隔，在电机应用中主要用于计算电机转速。

在 CA51F2 系列芯片中，比较器主要是针对无刷直流电机驱动而设计的。比较器的以上特性可实现有霍电机的霍尔检测、无霍电机的过零点检测，结合运放功能也可用于过流、过压等检测。

23.2 结构图

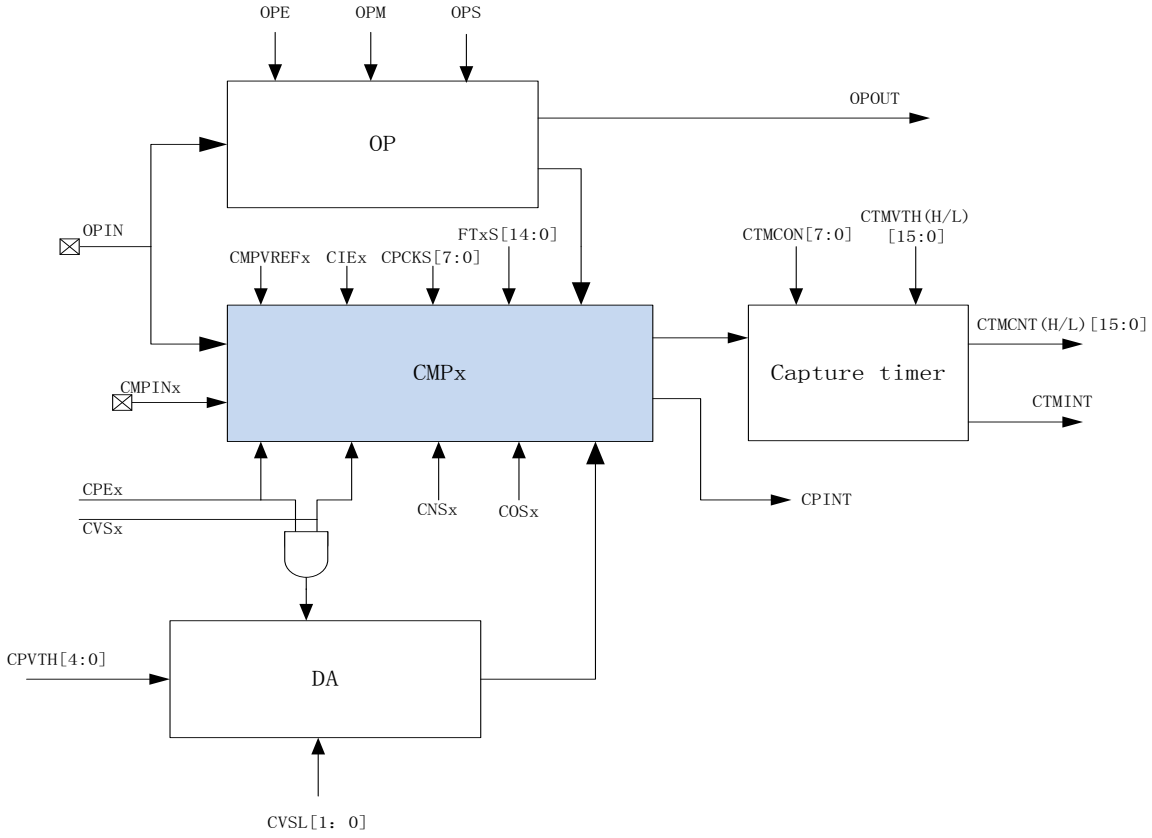


图 23-1-1 OPCMP 模块示意图

23.3 功能描述

23.3.1 运放

运放 A 通过寄存器 OPACON 进行控制，当设置 OPAM=1 时，运放选择 OPAIN（P35）引脚为运放输入，OPAOUT（P34）引脚为运放输出，运放的增益通过 OPAS 位域进行设置。当设置 OPAM=2 时，运放的输入为 ADC 选择的检测通道，运放的输出作为 ADC 转换器的输入，这样的话，ADC 的检测信号以 OPAS 设置的增益进行放大或缩小，显著地扩展了 ADC 的电压检测范围。

运放 B 通过寄存器 OPBCON 进行控制，当设置 OPBM=1 时，运放选择 OPBIN（P17）引脚为运放输入，OPBOUT（P16）引脚为运放输出，运放的增益通过 OPBS 位域进行设置。当设置 OPBM=2、3、4、5 时，运放的输入分别为比较器正端输入引脚 CMP0P、CMP1P、CMP2P、CMP3P，运放的输出分别作为比较器 0、1、2、3 的正端输入，这种设计可实现比较器对小信号的快速检测。

23.3.2 比较器

内置的比较器 0、1、2 设计完全相同，比较器 3 的不同之处在于不能选择为 IO 输入及不能作为捕获计数器的触发源，其他功能和比较器 0、1、2 也完全相同。每个比较器都有两个输入引脚：CMPxN 和 CMPxP(x=0、1、2、3），其中 CMPxN 作为比较器 x 的参考电压输入，CMPxP 是比较器的正端输入。比较器的参考电压可选择为 CMPxN 输入电压或 DA，通过 CVSx 位进行选择。当 CMPxP 的电压大于参考电压时，比较器输出逻辑 1，反之则输出逻辑 0。当选择 DA 为参考电压时，DA 的电压源可以通过 VSEL 选择为 BANDGAP、LDO 输出或 VDD，而 CPVTH 设置的是所选电压源的分压。

每个比较器都有一个 15 位的数字滤波器，滤波器的阈值通过寄存器 FTxS 设置。当比较器的输出翻转时，数字滤波器开始计数，计数的时钟通过寄存器 CPCKS 进行选择和设置预分频。当计数到滤波器阈值时，比较器的输出逻辑才会在比较器状态寄存器 CPSTA 的 CPxD 位更新，同时置位中断标志 CPxIF。当计数还没到阈值比较器输出又翻转了，此时计数重新从 0 开始计数。也就是说，比较器的输出状态要维持阈值设置的时间才算有效。当比较器的输出经过滤波后，信号的翻转可以触发比较器中断。比较器中断的触发沿可选择为上升沿、下降沿或双沿触发，通过 COSx 位域设置，CP0IF 位是比较器的中断标志。

23.3.3 捕获计数器

捕获计数器是 16 位的计数器，计数器时钟和比较器数字滤波器时钟同源，而捕获的触发事件是比较器的输出，可通过 CTMS 位域选择比较器 0、1、2 的上升沿或三个比较器的上升沿作为触发源。捕获计数器通过 CTME 位使能，当计数值到达设定阈值时产生阈值中断，阈值中断标志 CTMIVF 位置 1，计数器继续计数，当计数到最大值（FFFFH）时将产生溢出中断，溢出中断标志 CTMOVF 位置 1 并会重置计数器为 0。当设定的比较器有效沿发生时，当前计数值会被装载到寄存器 CTMCNT，并且会重置计数器为 0。

捕获计数器是专门为无刷电机驱动而设计的，捕获的计数值是对应于霍尔状态的间隔时间，通过对此数值的简单换算可以得出电机的转速。计数器阈值用于检测电机堵转，可设定合适的数值，当阈值中断发生时可认为是电机堵转。

23.4 寄存器描述

表 23-2-1 寄存器 OPACON

8040H	7	6	5	4	3	2	1	0
OPACON	OPAM		-			OPAS		
R/W	R/W		-			R/W		
初始值	0	0	-	-	-	0	0	0

位编号	位符号	说明
7~6	OPAM	运放 A 使能控制位域 00：运放 A 关闭 01：运放 A 使能,选择 OPAIN（P35）作为运放 A 输入，OPAOUT(P34）作为运放 A 输出 10：运放 A 使能，选择 ADC 检测通道作为运放 A 输入，运放 A 输出作为 ADC 输入 11：运放关闭
5~3	-	-
2~0	OPAS	ADC 增益选择位域 000： 1/4 倍 001： 1/3 倍 010： 1/2 倍 011： 5 倍 100： 10 倍 101： 15 倍 110： 20 倍 111： 30 倍

表 23-2-2 寄存器 OPBCON

8041H	7	6	5	4	3	2	1	0
OPBCON	OPBM			-		OPBS		
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	OPBM	运放 B 使能控制位域 000：运放 B 关闭 001：运放 B 使能，选择 OPBIN（P17）作为输入，OPBOUT（P16）作为输出 010：运放 B 使能，选择 CMP0P（P20）作为输入，运放 B 输出作为比较器 0 正端输入 011：运放 B 使能，选择 CMP1P（P22）作为输入，运放 B 输出作为比较

		器 1 正端输入 100：运放 B 使能，选择 CMP2P（P24）作为输入，运放 B 输出作为比较器 2 正端输入 101：运放 B 使能，选择 CMP3P（P26）作为输入，运放 B 输出作为比较器 3 正端输入 其他：运放 B 关闭
4~3	-	-
2~0	OPBS	ADC 增益选择位域 000： 1/4 倍 001： 1/3 倍 010： 1/2 倍 011： 5 倍 100： 10 倍 101： 15 倍 110： 20 倍 111： 30 倍

表 23-2-3 寄存器 CP0CON

8048H	7	6	5	4	3	2	1	0
CP0CON	CPE0	CIE0	CVS0	CZS0	-	-	COS0[1:0]	
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	CPE0	比较器 0 使能位，1 有效
6	CIE0	比较器 0 中断使能位，1 有效
5	CVS0	比较器 0 参考电压选择位 0：外部 1：DA
4	CZS0	比较器 0 迟滞电压选择位 0: 迟滞功能关闭 1: 迟滞功能打开
3~2	-	-
1~0	COS0	比较器中断触发边沿选择位域 00：上升沿触发 01：下降沿触发 其他：双沿触发

表 23-2-4 寄存器 CP1CON

8049H	7	6	5	4	3	2	1	0
CP1CON	CPE1	CIE1	CVS1	CZS1	-	-	COS1[1:0]	
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	CPE1	比较器 1 使能位，1 有效
6	CIE1	比较器 1 中断使能位，1 有效
5	CVS1	比较器 1 参考电压选择位 0：外部 1：DA
4	CZS1	比较器 1 迟滞电压选择位 0:迟滞功能关闭 1:迟滞功能打开
3~2	-	-
1~0	COS1	比较器 1 使能位，1 有效

表 23-2-5 寄存器 CP2CON

804AH	7	6	5	4	3	2	1	0
CP2CON	CPE2	CIE2	CVS2	CZS2	-	-	COS2[1:0]	
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	CPE2	比较器 2 使能位，1 有效
6	CIE2	比较器 2 中断使能位，1 有效
5	CVS2	比较器 2 参考电压选择位 0：外部 1：DA
4	CZS2	比较器 2 迟滞电压选择位 0: 迟滞功能关闭 1: 迟滞功能打开
3~2	-	-
1~0	COS2	比较器 2 使能位，1 有效

表 23-2-6 寄存器 CP3CON

804BH	7	6	5	4	3	2	1	0
CP3CON	CPE3	CIE3	CVS3	CZS3	-	-	COS3[1:0]	
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	CPE3	比较器 3 使能位，1 有效
6	CIE3	比较器 3 中断使能位，1 有效
5	CVS3	比较器 3 参考电压选择位 0：外部 1：DA
4	CZS3	比较器 3 迟滞电压选择位 0: 迟滞功能关闭 1: 迟滞功能打开
3~2	-	-
1~0	COS3	比较器 3 使能位，1 有效

表 23-2-7 寄存器 CPCKS

804CH	7	6	5	4	3	2	1	0
CPCKS	CPDIV					CPCKSEL		
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
7~3	CPDIV	比较器时钟分频设置位域 00000：不分频 00001：2 分频 00010：4 分频 00011：6 分频 … 11111：62 分频
2~0	CPCKSEL	比较器时钟选择位域 000：选择系统时钟 001：IRCH 010：IRCL 011：XOSCH/ERC 100：XOSCL 101：TFRC 110：PLL

		111：选择系统时钟

表 23-2-8 寄存器 CPSTA

804DH	7	6	5	4	3	2	1	0
CPSTA	CP3D	CP2D	CP1D	CP0D	CP3IF	CP2IF	CP1IF	CP0IF
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	CP3D	模拟比较器 3 的输出
6	CP2D	比较器 2 的输出或者外部引脚 P2.4 的输入
5	CP1D	比较器 1 的输出或者外部引脚 P2.2 的输入
4	CP0D	比较器 0 的输出或者外部引脚 P2.0 的输入
3	CP3IF	比较器 3 中断标志
2	CP2IF	比较器 2 中断标志
1	CP1IF	比较器 1 中断标志
0	CP0IF	比较器 0 中断标志

表 23-2-9 寄存器 CPVTC

804EH	7	6	5	4	3	2	1	0
CPVTC	VSEL[1:0]		-	CPVTH[4:0]				
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	VSEL	DA 模块基准电压选择位域 00：内部基准电压 01：LDO 输出 10：VDD 11：保留
5	-	-
4~0	CPVTH	DA 模块输出电压设置位域 输出电压 = 基准电压值÷(2^5)×(CPVTH+1)

表 23-2-10 寄存器 FT0S

8050H	7	6	5	4	3	2	1	0
FT0SL	FT0S[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
8051H	7	6	5	4	3	2	1	0

FT0SH	-	FT0S[14:8]						
R/W	-	R/W						
初始值	-	0	0	0	0	0	0	0
位编号	位符号	说明						
14~0	FT0S	比较器 0 数字滤波阈值						

表 23-2-11 寄存器 FT1S

8052H	7	6	5	4	3	2	1	0
FT1SL	FT1S[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
8053H	7	6	5	4	3	2	1	0
FT1SH	-	FT1S[14:8]						
R/W	-	R/W						
初始值	-	0	0	0	0	0	0	0
位编号	位符号	说明						
14~0	FT1S	比较器 1 数字滤波阈值						

表 23-2-12 寄存器 FT2S

8054H	7	6	5	4	3	2	1	0
FT2SL	FT2S[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
8055H	7	6	5	4	3	2	1	0
FT2SH	-	FT2S[14:8]						
R/W	-	R/W						
初始值	-	0	0	0	0	0	0	0
位编号	位符号	说明						
14~0	FT2S	比较器 2 数字滤波阈值						

表 23-2-13 寄存器 FT3S

8056H	7	6	5	4	3	2	1	0
FT3SL	FT3S[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0

8057H	7	6	5	4	3	2	1	0
FT3SH	-	FT3S[14:8]						
R/W	-	R/W						
初始值	-	0	0	0	0	0	0	0
位编号	位符号	说明						
14~0	FT3S	比较器 3 数字滤波阈值						

表 23-2-14 寄存器 CTMCON

8058H	7	6	5	4	3	2	1	0
CTMCON	CTME	CTMIE	CTMOE	CTMVE	CTMS		CTMIVF	CTMOVF
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	CTME	捕获计数器使能位，1 有效
6	CTMIE	捕获计数器中断使能位，1 有效
5	CTMOE	捕获计数器溢出使能位，1 有效 *备注：如果 CTMOE=1 并且 CTMIE=1，计数器溢出将产生溢出中断，对应中断标志 CTMOVF 位。*
4	CTMVE	捕获计数器阈值触发使能位，1 有效 备注：如果 CTMVE=1 并且 CTMIE=1，计数器达到阈值将产生阈值触发中断，对应中断标志 CTMIVF 位。
3~2	CTMS	捕获计数器捕获触发源选择位域 00：CP0D 的上升沿 01：CP1D 的上升沿 10：CP2D 的上升沿 11：CP0D 或 CP1D 或 CP2D 的上升沿
1	CTMIVF	捕获计数器阈值触发中断标志位
0	CTMOVF	捕获计数器溢出中断标志位

表 23-2-15 寄存器 CTMVTH

8059H	7	6	5	4	3	2	1	0
CTMVTHL	CTMVTH[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
805AH	7	6	5	4	3	2	1	0
CTMVTHH	CTMVTH[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	CTMVTH	捕获计数器阈值

表 23-2-16 寄存器 CTMCNT

805BH	7	6	5	4	3	2	1	0
CTMCNTL	CTMCNT[7:0]							
R/W	R/W							
初始值	0	0	0	0	0	0	0	0
805CH	7	6	5	4	3	2	1	0
CTMCNTH	CTMCNT[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	CTMCNT	捕获计数器捕获的计数值

24 直流无刷电机驱动（MOTOR）

24.1 功能简介

CA51F2 系列芯片内部集成了一个直流无刷电机控制模，该模块内置霍尔状态自动译码功能，支持 60°霍尔和 120°霍尔。模块还设计了自动模式、刹车模式和手动模式，在自动模式下，可预设每个霍尔状态对应的驱动状态，实现电机正转和反转的 6 步自动换相；刹车模式可选择带驱动刹车和自动停转模式，只需发送刹车命令即可实现刹车功能；在手动模式下，可通过软件写 HDCT 寄存器就可设置电机驱动状态，在实现无霍电机驱动或弦波驱动时可极大减少设置代码时间，响应速度更快。模块有多种电机异常检测，可选择出错异常时电机暂停或停止。模块外围的 PWM、模拟比较器、ADC 等功能模块的特性也是为电机驱动而设计的，令电机驱动更具灵活性。

24.2 结构框图

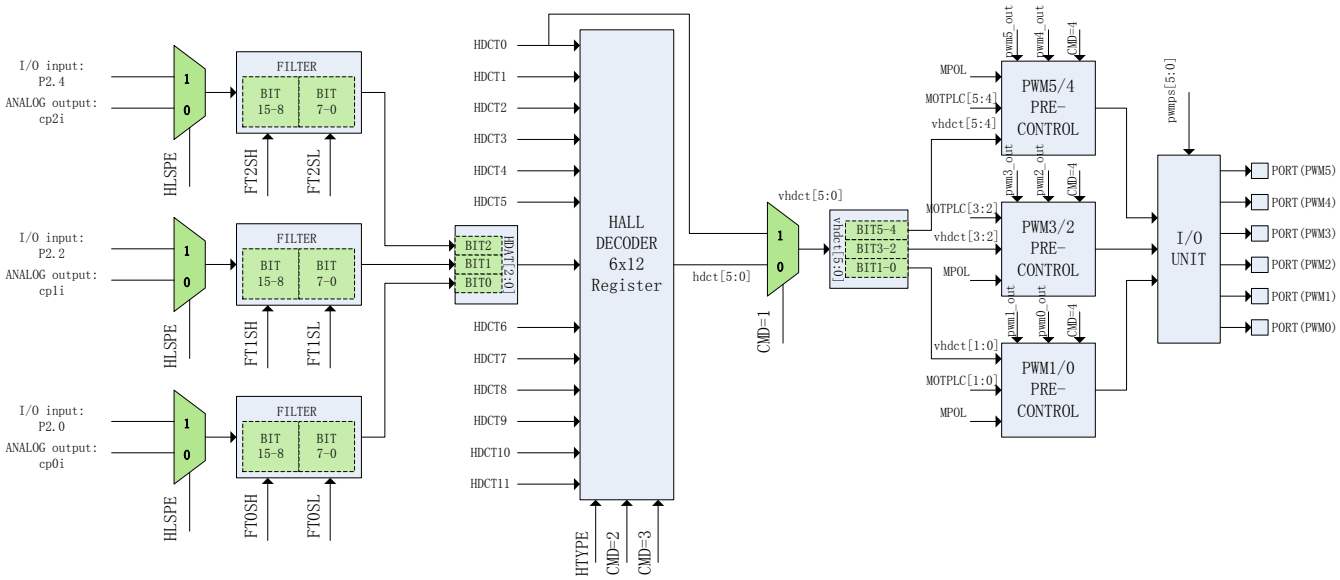


图 24-2-1 直流电机驱动框图

24.3 功能描述

24.3.1 霍尔状态译码功能

当电机控制模块使能并设置为自动模式（MOTCMD=1 或 MOTCMD=2）时，译码功能自动使能。电机的霍尔输出连接模拟比较器 0、1、2 的输入端，模拟比较器 0、1、2 的输出作为译码器的输入 HA、HB、HC。霍尔状态值[HC:HB:HA]在不同状态下选择相应的 HDCT 寄存器控制电机驱动，其中 HDCT0~HDCT5 对应电机向前转的 6 个相位的控制寄存器，HDCT6~HDCT11 对应电机向后转的 6 个相位的控制寄存器，霍尔状态和 HDCT 的对应关系如表 24-3-1-1 所示。

	120°霍尔					60°霍尔			
向前转（MOTCMD=2）	HC	HB	HA	HDCT	向前转（MOTCMD=2）	HC	HB	HA	HDCT
	0	0	1	HDCT0		0	0	1	HDCT0
	0	1	1	HDCT1		0	1	1	HDCT1
	0	1	0	HDCT2		1	1	1	HDCT2
	1	1	0	HDCT3		1	1	0	HDCT3
	1	0	0	HDCT4		1	0	0	HDCT4
	1	0	1	HDCT5		0	0	0	HDCT5
向后转（MOTCMD=3）	0	0	1	HDCT6	向后转（MOTCMD=3）	0	0	1	HDCT6
	0	1	1	HDCT7		0	1	1	HDCT7
	0	1	0	HDCT8		1	1	1	HDCT8
	1	1	0	HDCT9		1	1	0	HDCT9
	1	0	0	HDCT10		1	0	0	HDCT10
	1	0	1	HDCT11		0	0	0	HDCT11

表 24-3-1-1 霍尔状态和 HDCT 对应表

24.3.2 手动模式

当设置 MOTCMD=1 时，电机模块工作于手动模式。在手动模式下，霍尔译码功能不起作用，电机控制状态由 HDCT0 决定。手动模式可应用于无霍电机或弦波驱动应用中，可减少软件代码及更快速响应。

24.3.3 MASK 功能

MASK 功能框图如图 24-3-3-1 所示。

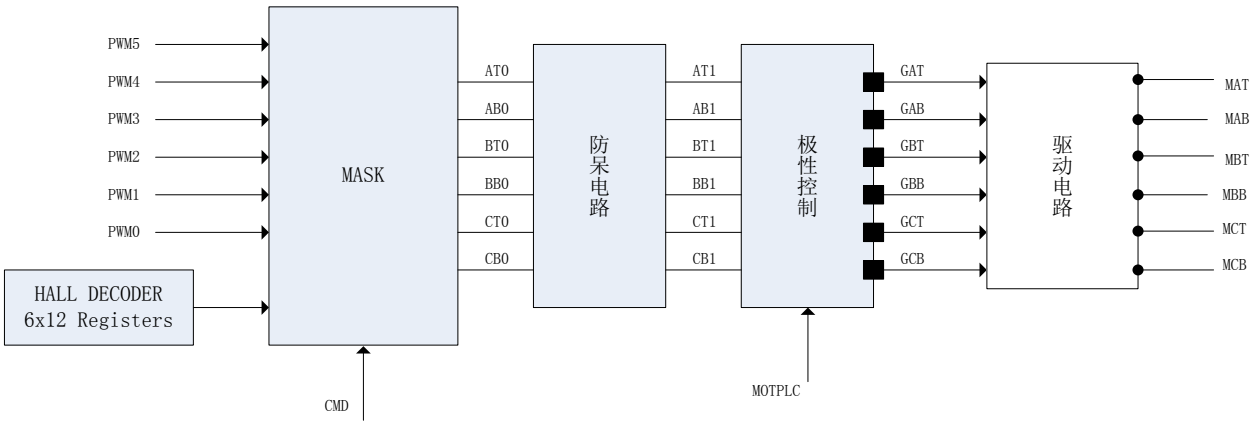


图 24-3-3-1 MASK 功能框图

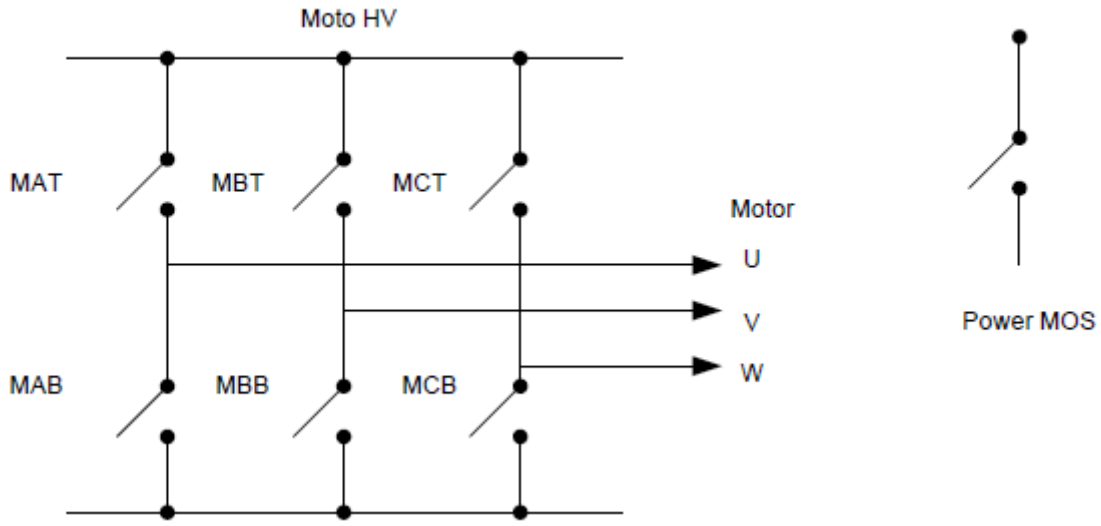


图 24-3-3-2　MASK 转换图

MASK 功能有三种模式：正常模式、刹车模式、无驱动模式。

◆ **正常模式**

在正常模式下，通过设置 MPOL 决定是顶端驱动还是底端驱动，MPOL=1 时，设置为顶端驱动，MPOL=0 时，设置为底端驱动（在顶端驱动模式，PWM 输出在上半桥进行调制，而底端驱动模式 PWM 输出在下半桥进行调制）。

在 MASK 模块中，PWM0~PWM5 作为源驱动信号，定义 AT0（对应 PWM0）、AB0（对应 PWM1）、BT0（对应 PWM2）、BB0（对应 PWM3）、CT0（对应 PWM4）、CB0（对应 PWM5）作为 MASK 电路后级输出，表 22-2-2-1 说明 HDCT 控制位和 AT0/AB0/BT0/BB0/CT0/CB0 的对应关系。

	HAT	HAB	AT0	AB0		HAT	HAB	AT0	AB0
底端驱动 (MPOL=0)	0	0	0	0	顶端驱动（MPOL=1）	0	0	0	0
	0	1	PWM0	PWM1		0	1	1	0
	1	0	0	1		1	0	PWM1	PWM0
	1	1	0	0		1	1	0	0
	HBT	HBB	BT0	BB0		HBT	HBB	BT0	BB0
底端驱动 (MPOL=0)	0	0	0	0	顶端驱动（MPOL=1）	0	0	0	0
	0	1	PWM2	PWM3		0	1	1	0
	1	0	0	1		1	0	PWM3	PWM2
	1	1	0	0		1	1	0	0
	HCT	HCB	CT0	CB0		HCT	HCB	CT0	CB0
底端驱动 (MPOL=0)	0	0	0	0	顶端驱动（MPOL=1）	0	0	0	0
	0	1	PWM4	PWM5		0	1	1	0
	1	0	0	1		1	0	PWM5	PWM4
	1	1	0	0		1	1	0	0

表 24-3-3-1 HDCT 和 AT0/AB0/BT0/BB0/CT0/CB0 对应关系表

◆ **刹车模式**

当 MOTCMD=4，MASK 电路工作于刹车模式。在刹车模式下，驱动电路下半桥全导通，上半桥全截止。如表 22-2-2-3 所示。

刹车模式（MOTCMD=4）	AT0	AB0	BT0	BB0	CT0	CB0
MPOL=0	0	1	0	1	0	1
MPOL=1	1	0	1	0	1	0

表 24-2-3-3 刹车模式 AT0/AB0/BT0/BB0/CT0/CB0 输出表

◆ **无驱动模式**

当 MOTCMD=0，MASK 电路工作于无驱动模式。在无驱动状态下，驱动电路上下半桥全截止。如表 22-2-2-4 所示。

无驱动模式（MOTCMD=0）	AT0	AB0	BT0	BB0	CT0	CB0
	0	0	0	0	0	0

表 22-3-3-4 无驱动模式 AT0/AB0/BT0/BB0/CT0/CB0 输出表

◆ **防呆功能**

若发生软件错误或是外界干扰如 ESD，导致 AT0/AB0、BT0/BB0、CT0/CB0 同时输出为高，防呆电路将强迫 AT1/AB1、BT1/BB1、CT1/CB1 输出低以防止驱动 MOS 上下管同时导通，造成短路。如表 22-2-2-5 所示。

AT0	AB0	AT1	AB1
1	1	0	0
BT0	BB0	BT1	BB1
1	1	0	0

CT0	CB0	CT1	CB1
1	1	0	0

表 24-3-3-5 防呆电路真值表

◆ **极性控制**

寄存器 MOTPLC 可控制 AT1/AB1/BT1/BB1/CT1/CB1 输出反相，寄存器 MOTPLC 的每一位对应一个通道，此功能可使电机模块灵活地匹配不同的驱动电路。

24.3.3 电机异常检测及保护

电机模块有多种机制可检测不同的电机异常，当异常发生时，可以选择 pause 或 fault 模式进行保护。在 pause 模式下，当发生异常时，电机驱动可选择刹车或无驱动，当异常消除时，电机就可恢复正常运转。在 fault 模式下，当异常消除后，电机无法自动恢复正常运转，必须要软件关闭电机模块并重新开启才可正常运转。Pause 位和 fault 位分别是 pause 和 fault 状态指示位。

电机发生异常时，电机刹车或无驱动通过 ZSE 位设置。

电机共有以下几种异常检测源：fault pin 检测、adc 超出设定阈值检测、比较器 3 中断检测、霍尔出错检测、捕获计数器超出阈值检测。

◆ **Fault pin 检测**

Fault pin 连接外部检测电路，可通过 FTPOL 位设置有效电平，通过 FTPFTS 位域设置 Fault pin 滤波时间。当设置 FTPME=1,Fault pin 检测使能，使能后默认为 pause 模式，当 FTPSE=1 时，fault pin 检测为 fault 模式（FTPME=1 且 FTPSE=1 才会使能 fault 模式）。当 fault pin 有效电平产生时，fault pin 检测中断标志产生，电机自动停转。

◆ **Adc 超出阈值检测**

Adc 用于电机过流或过压检测，adc 阈值中断功能在 adc 章节有详细介绍。Adc 与电机结合使用时，可以通过设置 PWMTM=0 和通过 PWMTS 选择 PWM 通道触发使能，另外还可通过寄存器 MTGDL 设置触发 ADC 的延迟时间。当设置 ADCME=1，pause 模式使能。当 ADCME=1 且 ADCSE=1，fault 模式使能。当 ADC 值超过设定阈值时，ADC 的阈值中断标志使电机驱动进入 pause 或 fault 模式，电机自动停转。

◆ **比较器 3 中断检测**

比较器 3 也可用于电机过流或过压检测，作用与 ADC 类似。CPME=1 使能 pause 模式，CPME=1 且 CPSE=1 使能 fault 模式。

◆ **霍尔状态出错检测**

HLME=1 使能 pause 模式，HLME=1 且 HLSE=1 使能 fault 模式。当霍尔出现非正常状态时，电机自动停转，霍尔出错中断标志 HLIF 置位。

◆ **捕获计数器超出阈值检测**

捕获计数器用于检测电机转速，可通过设置捕获计数阈值，阈值对应电机堵转时的转速，当电机堵转时，捕获计数器阈值中断产生。设置 CTME=1 使能 pause 模式，设置 CTME=1 且 CTSE=1 使能 fault 模式。当电机堵转时，电机自动停转。

24.4 电机控制寄存器描述

表 24-4-1 寄存器 MOTCON

8060H	7	6	5	4	3	2	1	0
MOTCON	MOTEN	ZSE	MPOL	FTPME	ADCME	CPME	HLME	CTME
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	MOTEN	MOTOR 模块使能，1 有效
6	ZSE	无驱动使能，为 1 时，如果有错误出现，电机将不对外驱动
5	MPOL	底端/顶端 PWM 输出选择，0 为底端，1 为顶端
4	FTPME	检测 Fault Pin 使能寄存器，为 1 时，电路检测到 Fault Pin 有效的信号将刹车或者无驱动
3	ADCME	检测 ADC 结果使能寄存器，为 1 时，如果 ADC 的转换结果超出用户配置的阈值将刹车或无驱动
2	CPME	检测比较器 3 的中断标志使能寄存器，为 1 时，检测到比较器 3 的中断标志将刹车或无驱动
1	HLME	检测霍尔解码结果使能寄存器，为 1 时，霍尔解码出错时将刹车或无驱动
0	CTME	检测比较计数器中断标志使能寄存器，为 1 时，检测到比较计数器计到阈值后所产生的中断标志将刹车或无驱动

表 24-4-2 寄存器 MOTCFG

8061H	7	6	5	4	3	2	1	0
MOTCFG	FTIE	HLIE	-	FTPSE	ADCSE	CPSE	HLSE	CTSE
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	FTIE	Fault Pin 中断使能寄存器，为 1 时，检测到有效的 Fault Pin 信号将产生中断
6	HLIE	霍尔解码出错中断使能寄存器，为 1 时，检测到霍尔解码出错将产生中断
5	-	-
4	FTPSE	与 FTPME 功能类似，但是此位为 1，检测到有效的 Fault Pin 信号后，将会锁定在刹车或无驱动状态，把 MOTEN 清除后重新置位才能恢复
3	ADCSE	与 ADCME 功能类似，但是此位为 1，检测到 ADC 结果超出阈值后，将会锁定在刹车或无驱动状态，把 MOTEN 清除后重新置位才能恢复
2	CPSE	与 CPME 功能类似，但是此位为 1，检测到比较器 3 的中断标志后，将会锁定在刹车或无驱动状态，把 MOTEN 清除后重新置位才能恢复
1	HLSE	与 HLME 功能类似，但是此位为 1，检测到霍尔解码出错后，将会锁定在刹车或无驱动状态，把 MOTEN 清除后重新置位才能恢复
0	CTSE	与 CTPME 功能类似，但是此位为 1，检测到比较计数器寄到阈值后，将会锁定在刹车或无驱动状态，把 MOTEN 清除后重新置位才能恢复

表 24-4-3 寄存器 MTGCON

8062H	7	6	5	4	3	2	1	0
MTGCON	PWMTS[2:0]			PWMTM	-	-	-	-
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	PWMTS	PWM 中断作为触发源选择： 0：关闭 PWM 此功能 1：PWM0 中断作为触发沿 2：PWM1 中断作为触发沿 3：PWM2 中断作为触发沿 4：PWM3 中断作为触发沿 5：PWM4 中断作为触发沿 6：PWM5 中断作为触发沿
4	PWMTM	PWM 中断触发目标选择： 0：触发 ADC 启动转换 1：触发 TIMER2 启动计数
3~0	-	-

表 24-4-4 寄存器 MHLCON

8063H	7	6	5	4	3	2	1	0
MHLCON	HTYPE	HLSPE	-	-	-	HDAT[2:0]		
R/W	R/W	R/W	-	-	-	R		
初始值	1	0	0	0	0	0	0	0

位编号	位符号	说明
7	HTYPE	霍尔类型选择： 0：60 度 1：120 度
6	HLSPE	比较器的数字滤波电路输入来源选择： 0：比较器 0 的数字滤波电路输入，来自模拟比较器 0 的输出 比较器 1 的数字滤波电路输入，来自模拟比较器 1 的输出 比较器 2 的数字滤波电路输入，来自模拟比较器 2 的输出 1：比较器 0 的数字滤波电路输入，来自外部 I/O，P2.0 比较器 1 的数字滤波电路输入，来自外部 I/O，P2.2 比较器 2 的数字滤波电路输入，来自外部 I/O，P2.4
5~3	-	-
2~0	HDAT	霍尔传感器的解码数据： HDAT[2] <-> HCD HDAT[1] <-> HBD

		HDAT[0] <-> HAD

表 24-4-5 寄存器 MFPCON

8064H	7	6	5	4	3	2	1	0
MFPCON	FTPOL	FTPFTS[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	FTPOL	Fault Pin 有效电平选择，0 为高电平有效，1 为低电平有效
6~0	FTPFTS	Fault Pin 滤波配置寄存器，最大为 128 个系统时钟

表 24-4-6 寄存器 MOTCMD

8065H	7	6	5	4	3	2	1	0
MOTCMD	-	-	-	-	-	CMD[1:0]		
R/W	-	-	-	-	-	R/W		
初始值	-	-	-	-	-	0	0	0

位编号	位符号	说明
7~3	-	-
2~0	CMD	电机驱动控制： 001：手动模式 010：向前转 011：向后转 100：刹车 其他：空闲

表 24-4-7 寄存器 MTGDL

8066H	7	6	5	4	3	2	1	0
MTGDL	MTGDL[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	MTGDL	触发 ADC 启动转换延迟寄存器，一旦有效的 PWM 产生有效的触发信号，延迟计数器将清零并开始计数，知道 MTGDL 的值才启动 ADC 转换，最大 256 个系统时钟

表 24-4-8 寄存器 MOTIF

8067H	7	6	5	4	3	2	1	0
MOTIF	FAULT	PAUSE	-	-	-	-	FTIF	HLIF
R/W	R	R	-	-	-	-	R	R
初始值	0	0	-	-	-	-	0	0

位编号	位符号	说明
7	FAULT	错误标志，1 表示电机处于刹车或无驱动状态，必须清除出错源并且清除 MOTEN 才能恢复
6	PAUSE	错误标志，1 表示电机处于刹车或无驱动状态，清除出错源就能恢复
5~2	-	-
1	FTIF	Fault Pin 中断标志，1 有效，写 1 清 0
0	HLIF	霍尔解码出错中断标志，1 有效，写 1 清 0

表 24-4-9 寄存器 HDCT0

8068H	7	6	5	4	3	2	1	0
HDCT0	-	-	HDCT0[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	HDCT0	BIT5-BIT4：HCT-HCB BIT3-BIT2：HBT-HBB BIT1-BIT0：HAT-HAB

表 24-4-10 寄存器 HDCT1

8069H	7	6	5	4	3	2	1	0
HDCT1	-	-	HDCT1[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	HDCT1	BIT5-BIT4：HCT-HCB BIT3-BIT2：HBT-HBB BIT1-BIT0：HAT-HAB

表 24-4-11 寄存器 HDCT2

806AH	7	6	5	4	3	2	1	0
HDCT2	-	-	HDCT2[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	HDCT2	BIT5-BIT4：HCT-HCB BIT3-BIT2：HBT-HBB BIT1-BIT0：HAT-HAB

表 24-4-12 寄存器 HDCT3

806BH	7	6	5	4	3	2	1	0
HDCT3	-	-	HDCT3[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	HDCT3	BIT5-BIT4：HCT-HCB BIT3-BIT2：HBT-HBB BIT1-BIT0：HAT-HAB

表 24-4-13 寄存器 HDCT4

806CH	7	6	5	4	3	2	1	0
HDCT4	-	-	HDCT4[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	HDCT4	BIT5-BIT4：HCT-HCB BIT3-BIT2：HBT-HBB BIT1-BIT0：HAT-HAB

表 24-4-14 寄存器 HDCT5

806DH	7	6	5	4	3	2	1	0
HDCT5	-	-	HDCT5[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	HDCT5	BIT5-BIT4：HCT-HCB BIT3-BIT2：HBT-HBB BIT1-BIT0：HAT-HAB

表 24-4-15 寄存器 HDCT6

806EH	7	6	5	4	3	2	1	0
HDCT6	-	-	HDCT6[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	HDCT6	BIT5-BIT4：HCT-HCB BIT3-BIT2：HBT-HBB BIT1-BIT0：HAT-HAB

表 24-4-16 寄存器 HDCT7

806FH	7	6	5	4	3	2	1	0
HDCT7	-	-	HDCT7[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	HDCT7	BIT5-BIT4：HCT-HCB BIT3-BIT2：HBT-HBB BIT1-BIT0：HAT-HAB

表 24-4-17 寄存器 HDCT8

8070H	7	6	5	4	3	2	1	0
HDCT8	-	-	HDCT8[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	HDCT8	BIT5-BIT4：HCT-HCB BIT3-BIT2：HBT-HBB BIT1-BIT0：HAT-HAB

表 24-4-18 寄存器 HDCT9

8071H	7	6	5	4	3	2	1	0
HDCT9	-	-	HDCT9[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	HDCT9	BIT5-BIT4：HCT-HCB BIT3-BIT2：HBT-HBB BIT1-BIT0：HAT-HAB

表 24-4-19 寄存器 HDCT10

8072H	7	6	5	4	3	2	1	0
HDCT10	-	-	HDCT10[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	HDCT10	BIT5-BIT4：HCT-HCB BIT3-BIT2：HBT-HBB BIT1-BIT0：HAT-HAB

表 24-4-20 寄存器 HDCT11

8073H	7	6	5	4	3	2	1	0
HDCT11	-	-	HDCT11[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	HDCT11	BIT5-BIT4：HCT-HCB BIT3-BIT2：HBT-HBB BIT1-BIT0：HAT-HAB

表 24-4-21 寄存器 MOTPLC

8074H	7	6	5	4	3	2	1	0
MOTPLC	-	-	MOTPLC[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	MOTPLC	电机电路使能时： BIT5=0/1：PWM5 输出正向/反向 BIT4=0/1：PWM4 输出正向/反向 BIT3=0/1：PWM3 输出正向/反向 BIT2=0/1：PWM2 输出正向/反向 BIT1=0/1：PWM1 输出正向/反向 BIT0=0/1：PWM0 输出正向/反向

25 触摸按键（Touch Key）

25.1 功能简介

CA51F2 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 24 个通道。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。

25.2 主要特性

- 高抗干扰性能，符合 EMC(CS)标准
- 最大支持 24 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
- STOP 模式下可设自动唤醒阈值

25.3 结构图

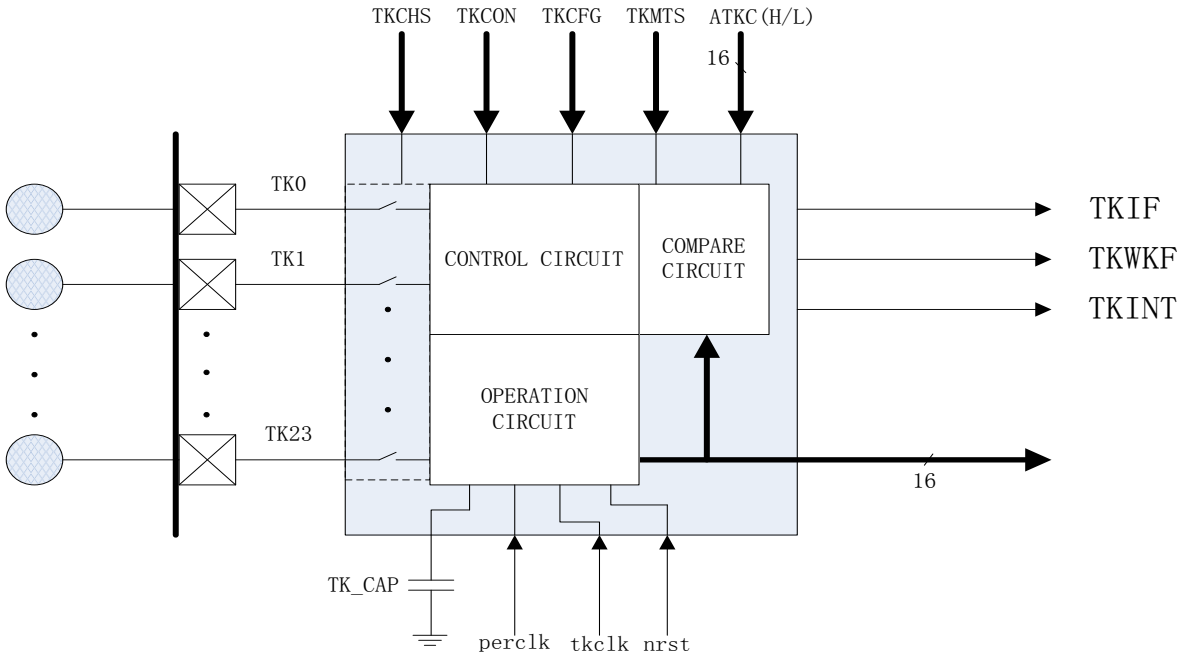


图 25-3-1 触摸模块结构图

25.4 功能描述

25.4.1 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。

25.4.2 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 TFRC，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。

25.4.3 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。

25.5 寄存器描述

表 25-1-1 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 25-1-2 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	1	1	1	1	1	1	1	1

位编号	位符号	说明
7~5	TKDIV	触摸时钟分频选择 000: 不分频 001：2 分频 010：3 分频 … 111：8 分频
4~0	TKTMS	外挂调制电容放电时间设置 放电时间 = TKTMS x 128 x 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 25-1-3 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 25-1-4 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	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						
5~0	TKPS	通道选择位域 000000：TK0~TK23 关闭 000001：选择 TK0 000010：选择 TK1 000011：选择 TK2 …… 011000：选择 TK23 011001：选择内部参考电容						

表 25-1-5 寄存器 ATKC

C5H	7	6	5	4	3	2	1	0
ATKCL	ATKC[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
ATKCH	ATKC[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 25-1-6 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 25-1-7 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 25-1-8 寄存器 TKIF

C7H	7	6	5	4	3	2	1	0
TKIF	-		TKIF5	TKIF4	TKIF3	TKIF2	TKIF1	TKIF0
R/W	-		R/W	R/W	R/W	R/W	R/W	R/W
初始值	-	-	0	0	0	0	0	0
位编号	位符号	说明						
7~6	-	-						
5~0	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 表示触摸数据超出对应的 ATKC 或者 ATKN 的阈值范围						

表 25-5-9 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 25-5-10 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

25.6 触摸控制例程

备注：触摸应用实例请参考本公司标准触摸库软件及相关文档。

26 低电压检测（LVD）

26.1 功能简介

低电压检测（LVD）用于监控芯片自身的供电 VDD，可设置检测电压范围为 1.8V~4.8V。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 26-1-1 所示。

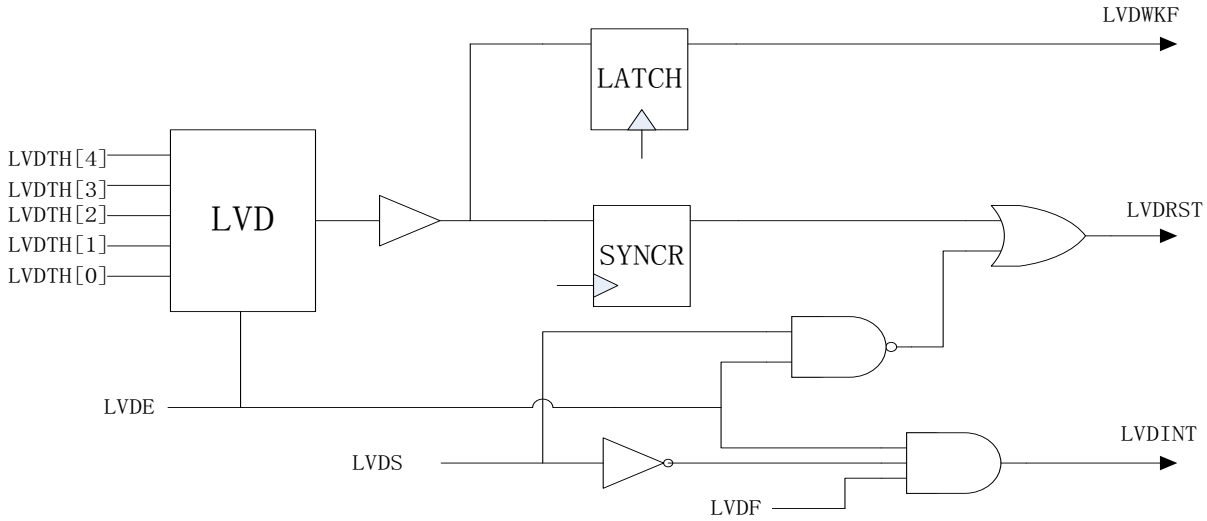


图 26-1-1 LVD 模块示意图

26.2 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。

26.3 寄存器描述

表 26-2-1 寄存器 LVDCON

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

位编号	位符号	说明
7	LVDE	LVD 使能位，1 有效
6	LVDS	LVD 功能选择位 0：中断 1：复位
5	LVDF	LVD 产生标志位，写 1 清 0
4	-	-
3~0	LVDTH	LVD 触发电平选择位域 0000：1.8V 0001：2.0V 0010：2.2V 0011：2.4V 0100：2.6V 0101：2.8V 0110：3.0V 0111：3.2V 1000：3.4V 1001：3.6V 1010：3.8V 1011：4.0V 1100：4.2V 1101：4.4V 1110：4.6V 1111：4.8V

26.4 LVD 控制例程

LVD 中断例程

例如，设置 LVD 为中断模式，检测电压为 3V，程序如下：

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

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

LVD 复位例程

例如，设置 LVD 为复位模式，检测电压为 3V，程序如下：

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

27 乘除法器（MDU）

27.1 功能简介

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

27.2 结构图

图 27-2-1 为乘法电路原理示意图。

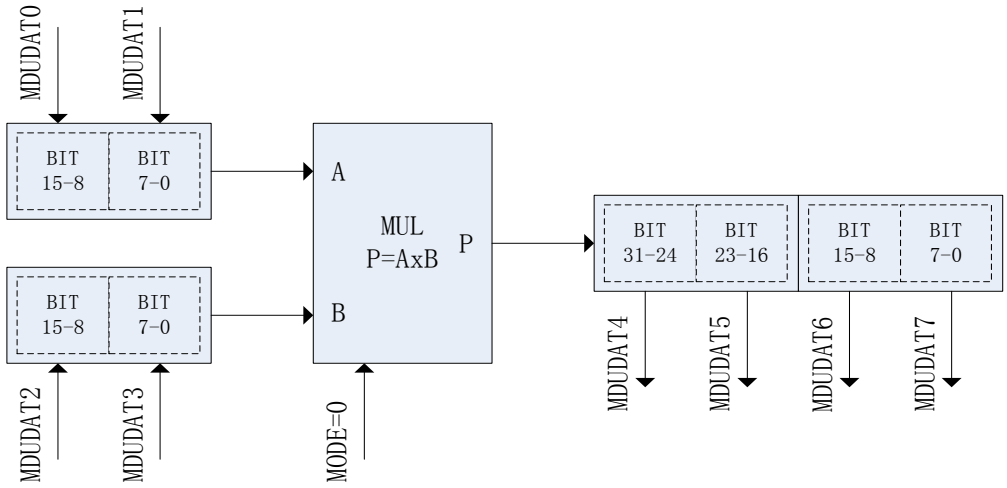


图 27-2-1 乘法电路原理示意图

图 27-2-2 为除法电路原理示意图。

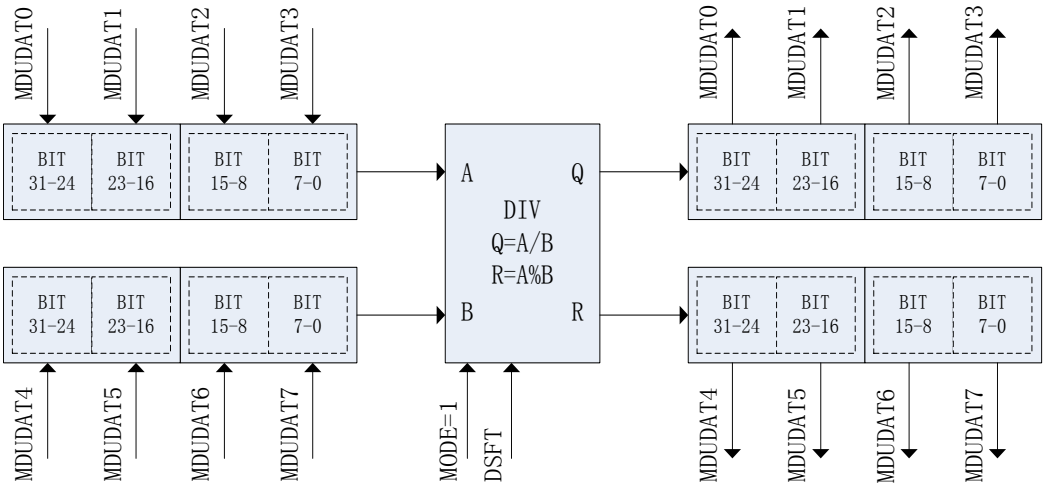


图 27-2-2 除法电路原理示意图

图 27-2-3 为移位电路原理示意图。

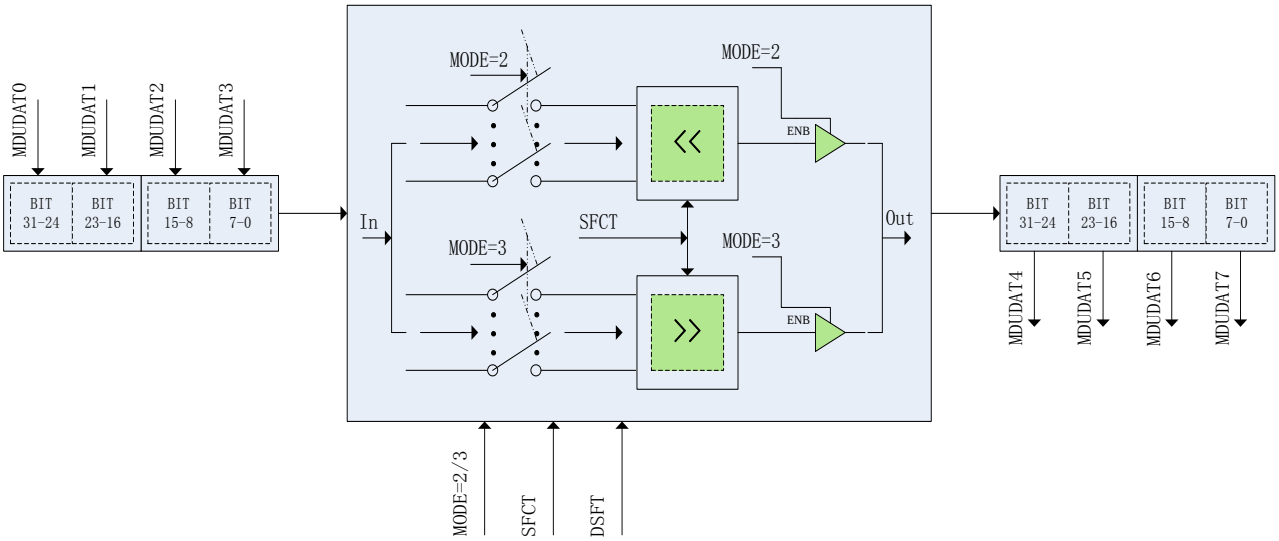


图 27-2-3 移位电路原理示意图

27.3 功能描述

27.3.1 乘法器

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

27.3.2 除法器

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

27.3.3 移位运算

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

27.4 寄存器描述

表 27-4-1 寄存器 MDUCON

E6H	7	6	5	4	3	2	1	0
MDUCON	MODE[1:0]		DSFT	SFCT[4:0]				
R/W	R/W		R/W	R/W				
初始值	0	0	0	-	0	0	0	0

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

表 27-4-2 寄存器 MDUDAT

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

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

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

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

27.5 MDU 控制例程

先定义以下联合体：

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

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

◆ **乘法运算操作例程**

例如，被乘数为 65535，乘数为 1000，程序如下：

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

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

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

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

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

◆ **除法运算操作例程**

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

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

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

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

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

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

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

```
    MDUDAT = Divisor.bVal[2];  //填写除数 8~15 位
    INDEX = 7;
    MDUDAT = Divisor.bVal[3];  //填写除数 0~7 位

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

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

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

◆ **移位运算操作例程**

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

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

    MDUCON = MOD_SHIFT_LEFT|8;     //设置向左移位操作及移位位数
    //MDUCON = MOD_SHIFT_RIGHT|8;  //设置向右移位操作及移位位数

    SourceData.dwVal = 0x88880001;

    INDEX = 0;
```

```
    MDUDAT = SourceData.bVal[0];   //填写源数据 24~31 位
    INDEX = 1;
    MDUDAT = SourceData.bVal[1];   //填写源数据 16~23 位
    INDEX = 2;
    MDUDAT = SourceData.bVal[2];   //填写源数据 8~15 位
    INDEX = 3;
    MDUDAT = SourceData.bVal[3];   //填写源数据 0~7 位

    MDUCON |= DSFT;       //启动移位操作

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

28 程序下载和仿真

28.1 程序下载

CA51F2 系列芯片主要采用 ISP 方式下载程序，芯片通过 UART 接口与下载工具相连接，任意一组 UART 口都可以用于 ISP。

更多关于程序下载步骤的细节请参考“CACHIP 开发下载工具使用说明”。

28.2 在线仿真

CA51F2 系列芯片支持在线仿真，芯片与仿真器之间通过 IIC 接口进行通信，出厂默认的 IIC 接口是 P30(IIC SDA）和 P31(IIC SCL）。要注意的是，由于芯片与仿真器间通过 IIC 通信，所以与仿真器连接的 IIC 接口引脚不能设置为其他功能，并且应用程序里不能使用 IIC 功能，否则将无法进入仿真模式。另外，由于 IIC 的通信速度是由主时钟决定，所以应用程序里不能将主时钟设置为低速时钟，也不能进入省电模式，否则都会影响芯片与仿真器间的通信。

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

更多关于仿真功能的细节可参考仿真器的相关文档介绍。

30 电气特性

30.1 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	-45	125	℃
CPU 工作频率	-	24	MHz

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

30.2 直流电气特性

芯片参数	符号	最小值	典型值	最大值	单位	条件
工作电压		1.80	3.3	5.5	V	
工作电流	Iop1	3.16	3.22	3.53	mA	系统时钟为 XOSCH(24MHz)，其他时钟关闭，所有输出引脚无负载，所有数字输入引脚不浮动，所有外设关闭，CPU 执行 NOP 指令
	Iop2	0.463	0.467	0.498	mA	系统时钟为 IOSC(3.6864MHz)，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，CPU 执行 NOP 指令
	Iop3	2.89	2.91	3.09	mA	系统时钟为 PLL 输出，PLL 设置为 6 倍频，参考时钟 IRCH 频率为 3.6864MHz，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，CPU 执行 NOP 指令
	Iop4	20.4	20.7	22	uA	系统时钟为 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，CPU 执行 NOP 指令
	Iop5	8.6	8.8	9.3	uA	系统时钟为 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，CPU 执行 NOP 指令
	Iop6	12.1	15.5	20.8	uA	系统时钟为 XOSCL(32.768kHZ)，其他时钟关闭，打开 LCD 驱动，LCD 设置为最小电流驱动、1/4bias、1/8duty、LCD 时钟为 XOSCL，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

STOP 模式电流	I_{stp}	2.4	2.5	2.8	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。
IDLE 模式电流	I_{idl1}	2.05	2.12	2.32	mA	系统时钟设为 XOSCH（24MHz），其他时钟关闭，所有输出引脚无负载，所有数字输入引脚不浮动，所有外设关闭，LDO 设置为低功率模式，Flash 进入睡眠模式，CPU 进入 IDLE 模式。
	I_{idl2}	0.282	0.285	0.299	mA	系统时钟设为 IRCH（3.6864MHz），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。
	I_{idl3}	1.77	1.79	1.87	mA	系统时钟为 PLL 输出，PLL 设置为 6 倍频，参考时钟 IRCH 频率为 3.6864MHz，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，CPU 进入 IDLE 模式。
	I_{idl4}	12.6	12.7	13.7	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。
	I_{idl5}	6.4	6.6	6.9	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。
	I_{idl6}	10.1	13.5	18.6	uA	系统时钟为 XOSCL(32.768kHZ)，其他时钟关闭，打开 LCD 驱动，LCD 设置为最小电流驱动、1/4bias、1/8duty、LCD 时钟为 XOSCL，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，CPU 进入 IDLE 模式。
IO 端口输入高电压	Vhi	-	0.7*Vdd	-	V	VDD=1.8~5.5V
IO 端口输入低电压	Vlo	-	0.2*Vdd	-	V	VDD=1.8~5.5V
IO 端口推电流	Ipu	-	20	-	mA	-
IO 端口灌电流	Iol	-	40	-	mA	-
IO 端口下拉电阻	Rd	-	15	-	KΩ	-
IO 端口上拉电阻	Ru	-	10	-	KΩ	-

备注：　最小值的数据测量条件：VDD=1.8V, TA=25℃，除非另有说明。

典型值的数据测量条件：VDD=3.3V, TA=25℃，除非另有说明。

最大值的数据测量条件：VDD=5.5V, TA=25℃，除非另有说明。

30.2 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 3.6864MHz
外部低速时钟（XOSCL）起振时间	Tosc1	-	1	-	s	XOSCL 频率为 32.768KHz
外部高速时钟（XOSCH）起振时间	Tosc2	-	2	-	ms	XOSCH 频率为 24MHz
PLL 稳定时间	Tpll	-	50	-	us	参考时钟 IRCH 频率为 3.6864MHz，PLL 为 6 倍频
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3.6864MHz，误差小于1%.

30.3 ADC 电气特性

模/数转换器（ADC）电气特性(Ta=25℃,参考电压为 VDD)

芯片参数	符号	最小值	典型值	最大值	单位	条件
工作电压	V_{AD}	1.8		5.5	V	
ADC 精度	NR		12		Bit	GND<=Vin<=Vref
ADC 输入电压	Vin	0	-	VDD	V	
ADC 输入电阻	Rin	2	-	-	MΩ	VDD=5V
ADC 转换电流	I_{ADC}	-	180	-	uA	VDD=5V
微分非线性误差	DNL	-	-	±3	LSB	VDD=5V
积分非线性误差	INL	-	-	±3	LSB	VDD=5V
满刻度误差	EF	-	±3	±4	LSB	VDD=5V
偏移量误差	Ez	-	±0.5	±1	LSB	VDD=5V
转换时间	T_{CON}	-	16	-	时钟周期	

备注：（1）ADC 输入电阻是直流条件下 ADC 自身的输入电阻；

（2）测试 ADC 时连接通路的信号源内阻需要小于 10KΩ

30.4 内部高速 RC 温度特性

内部高速 RC 时钟（IRCH）温度特性（条件：V_{DD}=3.3V，Ta=25℃时时钟频率为 3.6841MHz）

测试温度	频率
-40℃	3.6720 MHz
-35℃	3.6720 MHz
-30℃	3.6720 MHz
-25℃	3.6760 MHz
-20℃	3.6710 MHz
-15℃	3.6700 MHz
-10℃	3.6740 MHz
-5℃	3.6745 MHz
0℃	3.6800 MHz
5℃	3.6800 MHz
10℃	3.6820 MHz
15℃	3.6841 MHz
20℃	3.6840 MHz
25℃	**3.6841 MHz**
30℃	3.6844 MHz
35℃	3.6847 MHz
40℃	3.6848 MHz
45℃	3.6853 MHz
50℃	3.6855 MHz
55℃	3.6868 MHz
60℃	3.6868 MHz
65℃	3.6910 MHz
70℃	3.6950 MHz
75℃	3.6960 MHz
80℃	3.6960 MHz
85℃	3.6990 MHz

备注：以上高速 RC 温度特性数据是单一芯片测试值，仅供参考。

30.5 内部低速 RC 温度特性

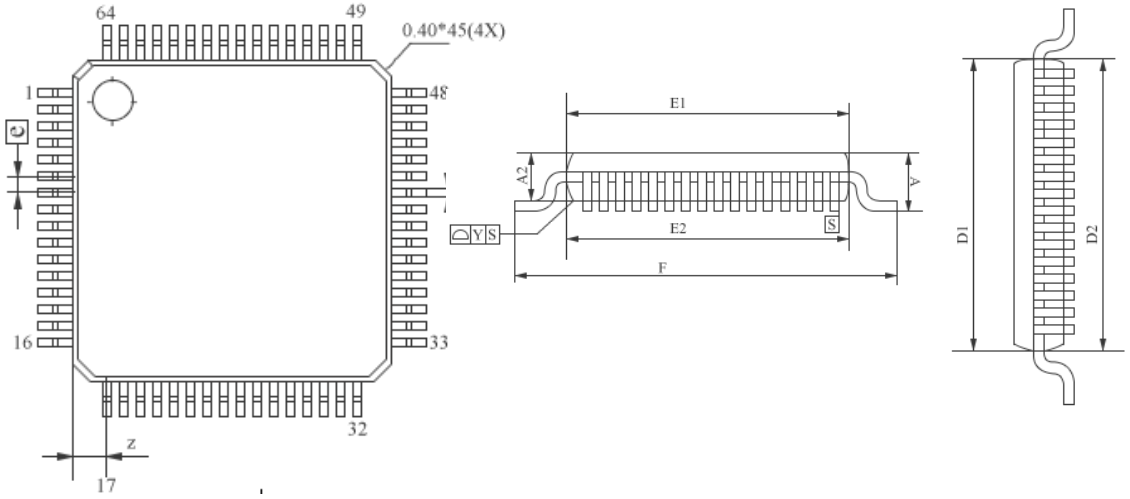
内部低速 RC 时钟（IRCL）温度特性（条件：V_{DD}=3.3V，Ta=25℃时时钟频率为 131.5KHz）

测试温度	频率
-40℃	132.15 KHz
-35℃	131.76 KHz
-30℃	131.65 KHz
-25℃	131.35 KHz
-20℃	131.50 KHz
-15℃	131.20 KHz
-10℃	131.10 KHz
-5℃	131.20 KHz
0℃	131.20 KHz
5℃	131.10 KHz
10℃	131.78 KHz
15℃	131.45 KHz
20℃	131.60 KHz
25℃	**131.50 KHz**
30℃	131.50 KHz
35℃	132.50 KHz
40℃	132.70 KHz
45℃	133.20 KHz
50℃	133.50 KHz
55℃	133.75 KHz
60℃	134.10 KHz
65℃	134.80 KHz
70℃	135.20 KHz
75℃	135.86 KHz
80℃	135.85 KHz
85℃	136.65 KHz

备注：以上低速 RC 温度特性数据是单一芯片测试值，仅供参考。

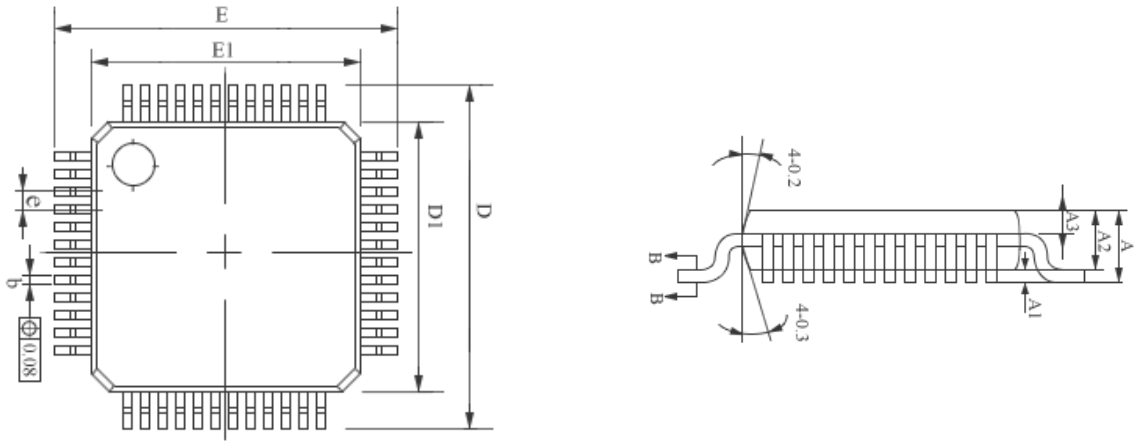
31 封装类型

封装形式（一）(LQFP 64)



序号	最小值	标准值	最大值
A	————	————	1.63
A2	1.30	1.40	1.50
D1	6.85	6.95	7.05
D2	6.90	7.00	7.10
E1	6.85	6.95	7.05
E2	6.90	7.00	7.10
e	————	0.40	————
F	8.80	9.0	9.20
Z	————	0.5	————

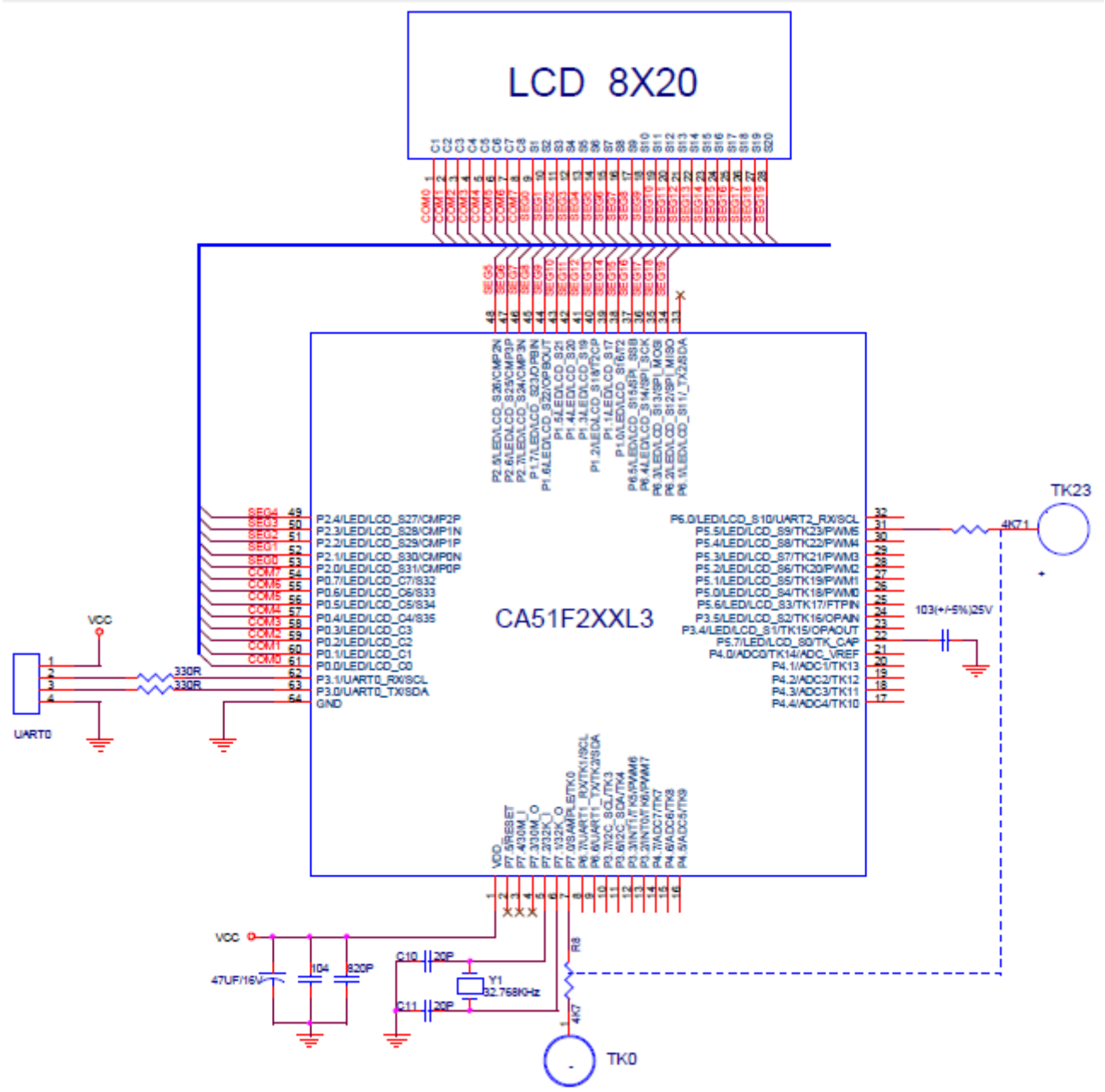
封装形式（二）(LQFP 48)



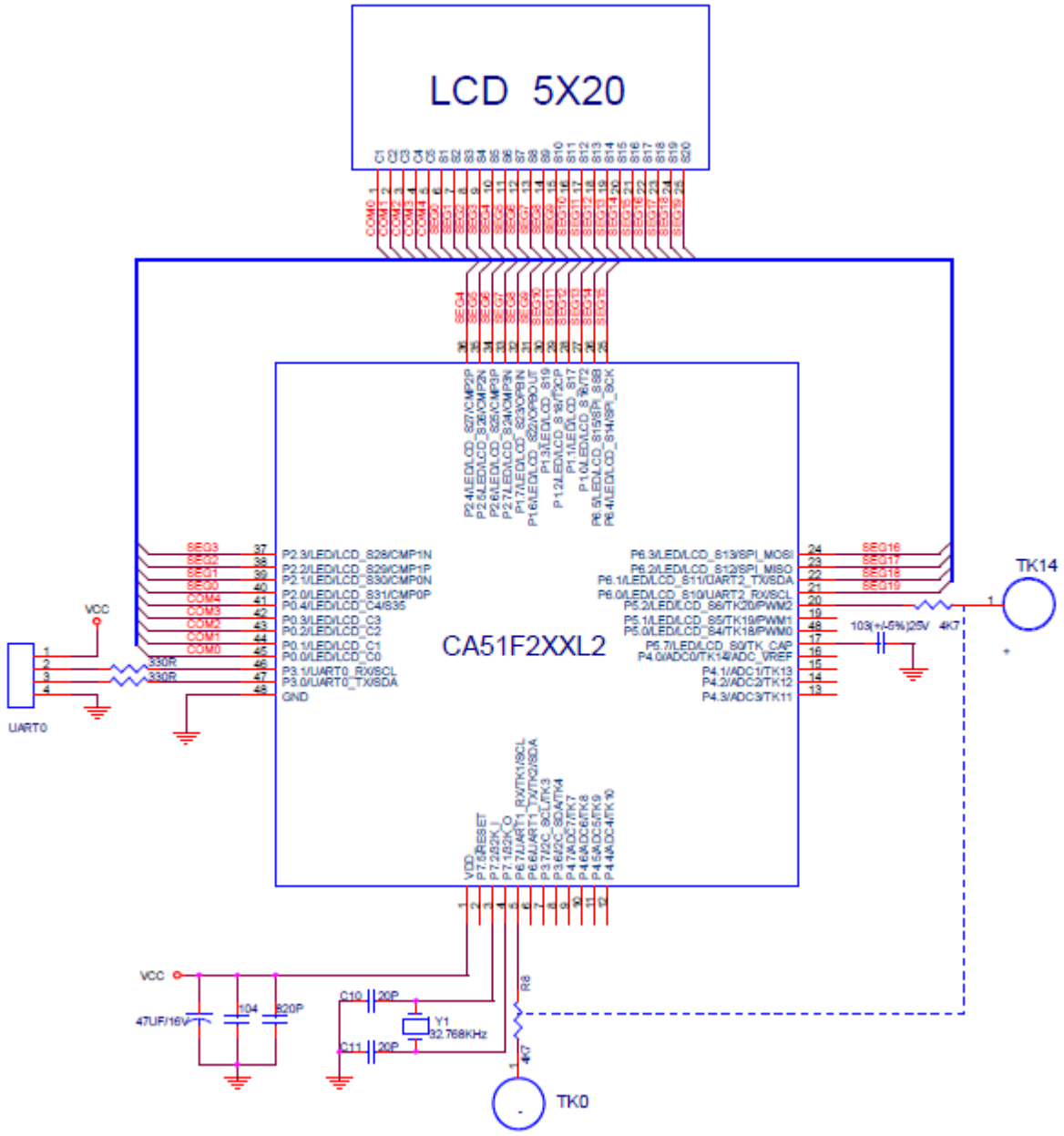
序号	最小值	标准值	最大值
A	————	————	1.60
A1	0.05	————	0.15
A2	1.35	1.40	1.45
A3	0.59	0.54	0.69
b	0.18	————	0.27
D	8.80	9.00	9.20
D1	6.90	7.00	7.10
E	8.80	9.00	9.20
E1	6.90	7.00	7.10
e	0.50		

32 典型应用参考电路

参考电路（一）



参考电路（二）


LCD 5X20
CA51F2XXL2
UART0
VCC
330R
330R
TK14
TK0
103(+/-5%)25V
4K7
47UF/16V
104
820P
C10
20P
C11
20P
Y1
32.768KHz
R8
4K7

33 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,tem	4

		p.14,...,temp.8)	
DIV AB	A 除以 B	QUO ← (A) / (B) .....REM (A) ← QUO (B) ← REM	4

DA A	累加器进行十进制转换	IF (A.3,...,A.0) > 9 || AC = 1
		THEN
		temp16 ← (A) + 0x06
		(A) ← (temp.7,...,temp.0)
		
		IF (temp16) > 0xFF
		THEN
		CY ← 1
		
		IF (A.7,...,A.4) > 9 || CY = 1
		THEN
		temp16 ← (A) + 0x60
		(A) ← (temp.7,...,temp.0)
		
		IF (temp16) > 0xFF
		THEN
		CY ← 1
			1

逻辑操作类指令			
ANL A, Rn	累加器与寄存器相“与”	(A) ← (A) & (Rn)	1
ANL A, direct	累加器与直接地址单元相“与”	(A) ← (A) & (direct)	1
ANL A, @Ri	累加器与间接 RAM 内容相“与”	(A) ← (A) & ((Ri))	1
ANL A, #data8	累加器与 8 位立即数相“与”	(A) ← (A) & #data	1
ANL direct, A	直接地址单元与累加器相“与	(direct) ← (direct) & (A)	1
ANL direct, #data8	直接地址单元与 8 位立即数相“与”	(direct) ← (direct) & #data	2
ORL A, Rn	累加器与寄存器相“或”	(A) ← (A) | (Rn)	1
ORL A, direct	累加器与直接地址单元相“或”	(A) ← (A) | (direct)	1
ORL A, @Ri	累加器与间接 RAM 内容相“或”	(A) ← (A) | ((Ri))	1
ORL A, #data8	累加器与 8 位立即数相“或”	(A) ← (A) | #data	1

ORL direct, A	直接地址单元与累加器相“或”	(direct) ← (direct) | (A)	1
ORL direct, #data8	直接地址单元与 8 位立即数相“或”	(direct) ← (direct) | #data	2
XRL A, Rn	累加器与寄存器相“异或”	(A) ← (A) ^ (Rn)	1
XRL A, direct	累加器与直接地址单元相“异或”	(A) ← (A) ^ (direct)	1
XRL A, @Ri	累加器与间接 RAM 内容相“异或”	(A) ← (A) ^ ((Ri))	1
XRL A, #data8	累加器与 8 位立即数相“异或”	(A) ← (A) ^ #data	1
XRL direct, A	直接地址单元与累加器相“异或”	(direct) ← (direct) ^ (A)	1
XRL direct, #data8	直接地址单元与 8 位立即数相“异或”	(direct) ← (direct) ^ #data	2
CLR A	累加器清 0	(A) ← 0	1
CPL A	累加器求反	(A) ← /(A)	1
RL A	累加器循环左移	(A) ← (A.6,A.5,...,A.0,A.7)	1
RLC A	累加器带进位循环左移	C ← A.7 (A) ← (A.6,A.5,...,A.0,C)	1
RR A	累加器循环右移	(A) ← (A.0,A.7,...,A.2,A.1)	1
RRC A	累加器带进位循环右移	C ← A.0 (A) ← (C,A.7,...,A.2,A.1)	1
控制转移类指令			
ACALL addr11	绝对短调用子程序	(PC) ← (PC) + 2 (SP) ← (SP) + 1 ((SP)) ← (PC7-0) (SP) ← (SP) + 1 ((SP)) ← (PC15-8) (PC10-0) ← page address	2
LACLL addr16	长调用子程序	(PC) ← (PC) + 3 (SP) ← (SP) + 1 ((SP)) ← (PC7-0) ((SP)) ← (PC15-8) (PC) ←addr15-0	2
RET	子程序返回	(PC15-8) ← ((SP)) (SP) ← (SP) - 1 (PC7-0) ← ((SP)) (SP) ← (SP) - 1	2
RETI	中断返回	(PC15-8) ← ((SP))	2

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

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

CJNE A, #data8, rel	累加器与 8 位立即数比较，不等则转移	(PC) ← (PC) + 3
		IF (A) <> data
		THEN
		(PC) ← (PC) + relative offset
		IF (A) < data
		THEN
		(C) ← 1
		ELSE
			2

		(C) ← 0	

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

CJNE @Ri, #data8, rel	间接 RAM 单元，不等则转移	(PC) ← (PC) + 3
		IF ((Ri)) <> data
		THEN
		(PC) ← (PC) + relative offset
		IF ((Ri)) < data
		THEN
		(C) ← 1
		ELSE
		(C) ← 0
			2

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

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