IP5209/IP5109/IP5207/IP5108 寄存器文档

1 I2C 介绍

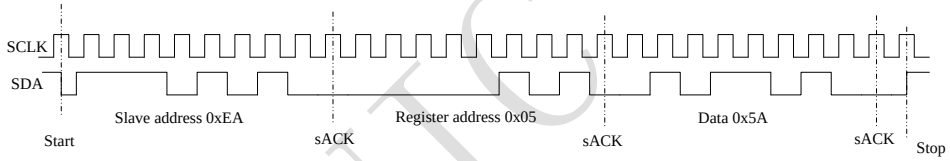
The i2c speed support 400Kbps.Support 8 bit address width and 8bit data width. Transmit and receive MSB first. The default slave address is 0Xea。

I2C acts as slave and is controlled by the master. The SCK line of the I2C interface is driven by the master. The SDA line could be pulled up to VCC by a 2.2Kohm resister and pulled

down by either the master or the slave.A typical WRITE sequence for writing 8bits data to a register is shown in below figure. A start bit isgiven by the master, followed by the slave address, register address and 8-bit data. After each 8-bit address or data transfer, the IP5209/IP5109/IP5108/IP5207 gives an ACK bit. The master stops writing by sending a stop bit.

All 8 bits data must be written before the register is updated.

Example: Write 8bit data 0x5a to register 0x05, and the slave address is 0Xea



Note：Sack generated by Slave, Mack generated by Master, and Mnack is a NACK generated by Master

Figure1 I2C WRITE

A typical READ sequence is shown in below figure. First the master has to write the slave address,followed by the register address. Then a restart bit and the slave address specify that a READ is generated. The master then clocks out 8 bits at a time to read data.

Example: Read 8bit data 0x5A from register 0x05, and the slave address is 0Xea

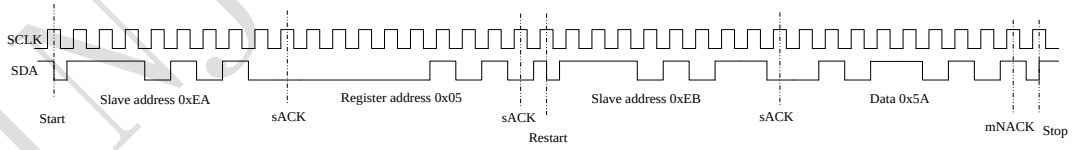
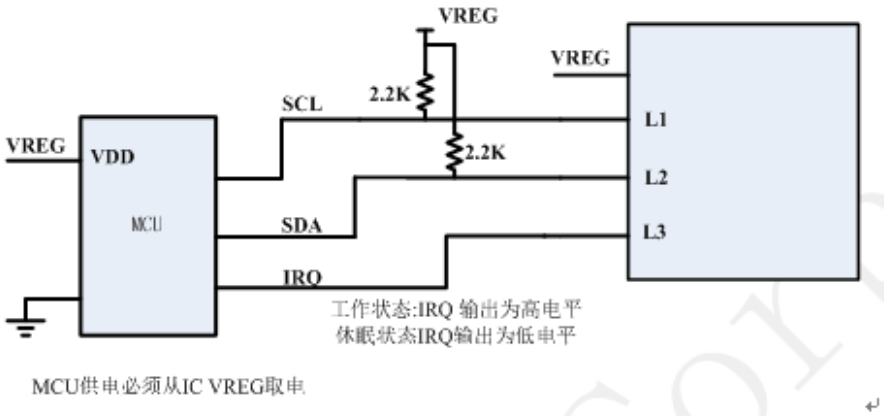


Figure2 I2C Read

2、I2C 应用注意事项



I2C 连接示意图

1、IP5209/IP5109/IP5108 标准品默认支持 I2C，不需要单独定制 I2C 版本；

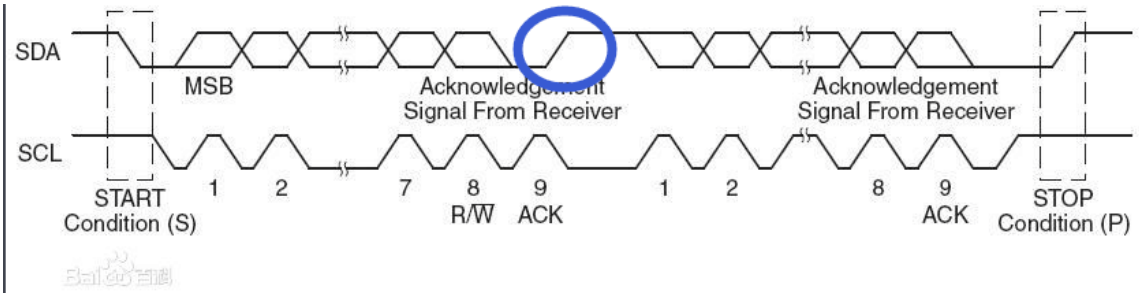
2、IP5209/IP5109/IP5108/IP5207 从休眠状态转入工作状态（按键、负载接入、5V 充电接入）时，IP5209/IP5109/IP5108/IP5207 内部首先会检测 L1、L2 脚的是否被上拉到 3.1V（VREG），如果 L1 L2 同时被上拉到 3.1V 则进入 I2C 模式，L3 输出一个 3.1V 的高电平；如果没有检测到 L1 L2 同时上拉则进入 LED 灯显模式，每次从休眠进入工作状态都会进行检测；

3、由于 IP5209/IP5109/IP5108/IP5207 由休眠进入工作状态时会进行 I2C 检测，所以 MCU 在休眠的时候需要将 SDA 和 SCK 配置为输入或者高阻状态，直到检测到 INT 为高时才开始读写 I2C 数据，否则会导致 IC 在由休眠进入工作状态时检测到 L1 或者 L2 没有被上拉而无法进入 I2C 状态

4、由于 IP5209/IP5109/IP5108/IP5207 由休眠进入工作状态时会进行 I2C 检测以及 IP5209/IP5109/IP5108/IP5207 内部的数字电平都是 3.1V 所以 MCU 供电必须有 VREG 供电，如果 MCU 用外部的 LDO 供电，当 BAT 没电或者小于 2V 时 VIN 接入 5V 给 IP5209/IP5109/IP5108/IP5207 供电，VREG 有电系统会进行 I2C 检测，但是 MCU 没有电,SDA 和 SCK 的状态不定，可能导致 L1 和 L2 没检测到上拉无法进入 I2C 模式；

5、如果要修改 IP5209/IP5109/IP5108/IP5207 某个寄存器的时候需要先将相应寄存器的值读出来对需要修改的 BIT 位进行与或运算后再把计算的值写进这个寄存器，**确保只修改需要修改的 bit 其他 bit 的值不能随意改动**

6、I2C 通讯波形介绍



I2c master 写的时候，先传 8bit 数据，第 9 个 bit 读 slave 返回的 ack，ack 为低代表写入成功，为高代表写入不成功。

I2c master 读的时候，最后一个 byte 传输是 slave 返回数据，master 返回 nack（高电平），代表读结束；如果 master 返回的是 ack（低电平），则说明读没有结束，master 会继续读。

所以第九个 bit 的 ack 信号要看 master 端是读操作还是写操作：

因为 IP5209/IP5109/IP5108/IP5207 只能做 slave：

如果往 IP5209/IP5109/IP5108/IP5207 寄存器写入数据，IP5209/IP5109/IP5108/IP5207 返回 ack 为低电平；

如果从 IP5209/IP5109/IP5108/IP5207 读取数据，IP5209/IP5109/IP5108/IP5207 返回 nack 高电平），（**master 必须发 NACK，否则会有异常）代表读结束**

3、寄存器功能描述

标示为“Reserved”的寄存器位有特殊控制作用，不可改变原有的值，否则会出现无法预期的结果。对寄存器的操作必须按照“读-->修改-->写”来进行，只修改要用到的 bit，不能修改其他未用 bit 的值。

1.1 SYS_CTL0

Offset = 0x01

Bit(s)	Name	Description	R/W	Reset
7 ： 5		Reserved		
4		手电筒检测是否使能 1：enable 0：disable	RW	1
3		Light enable 0: disable 1: enable	RW	1
2		Boost enable 0: disable 1:enable	RW	1
1		Charger enable 0: disable 1:enable	RW	1
0		Reserved		

1.2 SYS_CTL1

Offset = 0x02

Bit(s)	Name	Description	R/W	Reset
7 ： 2		Reserved		
1		轻载关机使能（0x0c 可设定轻载关机阈值） 1:使能 IBATLOW 轻载关机功能 0:关闭 IBATLOW 轻载关机功能	R/W	1
0		负载插入自动开机 1:使能 0:关闭	R/W	1

1.3 SYS_CTL2

Offset = 0x0c

Bit(s)	Name	Description	R/W	Reset
7:3		轻载关机电流阈值设定 n * 12mA 当 BAT 电流小于设定阈值时，持续 32s 关机 注意：此电流设定阈值需要大于 100mA	RW	00100
2:0		Reserved		

1.4 SYS_CTL3

Offset = 0x03

Bit(s)	Name	Description	R/W	Reset
7:6		长按按键时间选择 00：1S 01：2S 10：3S 11：4S		01
5		1：连续两次短按（两次短按在 1s 内）关机功能使能 0：连续两次短按（两次短按在 1s 内）关机功能关闭	R/W	1
4：0		Reserved		

1.5 SYS_CTL4

Offset = 0x04

Bit(s)	Name	Description	R/W	Reset
7:6		关机时间设定 11:64s 10:32s 01:16s 00: 8s	R/W	10
5		VIN 拔出是否开启 BOOST 1：开启 0：不开启		1
4：0		Reserved		

1.6 SYS_CTL5

Offset = 0x07

Bit(s)	Name	Description	R/W	Reset
7		Reserved	R/W	
6		NTC 功能使能 0：使能 1：关闭	R/W	0
5:2		Reserved	R/W	
1		按键开关 WLED 手电筒方式选择： 0：长按 2S 1：短按两次按键	R/W	0
0		按键关机方式选择： 0：短按两次按键 1：长按 2S	R/W	0

1.7 Charger_CTL1

Offset=0x22

Bit(s)	Name	Description	R/W	Reset
7:5		Reserved		
3:2		充电欠压环设定（充电时输出端 VOUT 的电压） 11:4.83V 10:4.73V 01:4.63V 00:4.53V 注：在充电的时候 IC 会检测输出 VOUT 的电压来自动调整充电电流，当 VOUT 的电压大于设置值时就以最大电流对充电充电，小于设定值时就自动减小充电电流以维持此电压； 如果客户要求边充边放状态下可在输出端加采样电阻检测边充边放状态输出端的负载电流大于 100mA 时可把欠压环设置为最高，优先对外部负载充电。	R/W	01
1:0		Reserved		

1.8 Charger_CTL2

Offset=0x24

Bit(s)	Name	Description	R/W	Reset
7		Reserved		
6:5		BAT 电池类型选择 11：RESERVED 10：4.35V 电池 01：4.3V 电池 00: 4.2V 电池	R/W	00
4:3		RESERVED		
2：1		恒压充电电压加压设置 11：加压 42mV 10：加压 28mV 01：加压 14mV 00: 不加压 注：4.30V/4.35V 建议加压 14mV； 4.2V 建议加压 28mV； 如果客户需要支持 4.4V 的电芯，可以在 4.35V 电池的基础上选择加压 48mV，充饱由 MCU 检测到电池电压大于 4.4V，电流小于 200MA 才认为是电芯充饱了；	R/W	10
0		Reserved		

1.9 CHG_DIG_CTL4

Offset = 0x26

Bit(s)	Name	Description	R/W	Reset
7		Reserved		
6		电池类型内部寄存器设定还是外部 Vset PIN 设定选择 1：外部 VSET PIN 设置 0：内部寄存器设置 如果是该 bit 为 0，可通过 0x24 寄存器的 bit6：5 来设定电池类型	RW	1
5：0		Reserved		

1.10 CHG_DIG_CTL4

Offset=0x25

Bit(s)	Name	Description	R/W	Reset
7：5		Reserved		
4：0		充电电流设置(设置为电池端电流)： Ibat=b0*0.1+b1*0.2+b2*0.4+b3*0.8+b4*1.6A 注：默认值为 10111 2.3A 左右	R/W	10111

1.11 MFP_CTL0

Offset = 0x51

Bit(s)	Name	Description	R/W	Reset
7:6	-	Reserved	-	-
5:4	LIGHT_sel	LIGHT 功能选择 00：WLED 01：GPIO2 10： VREF 11： Reserved	R/W	00
3:2	L4_sel	L4 的功能选择 00：L4 01：GPIO1 10： Reserved 11： Reserved	R/W	00
1：0	L3_sel	L3的功能选择 00：L3 01：GPIO0 10： Reserved 11： Reserved	R/W	00

1.12 MFP_CTL1

Offset = 0x52

Bit(s)	Name	Description	R/W	Reset
7:4		Reserved	R/W	00
3:2	VSET_sel	VSET功能选择 00：电池电压选择PIN	R/W	00

		01：GPIO4 10： Reserved 11： Reserved		
1:0	RSET_sel	RSET功能选择 00：电池内阻选择PIN 01：GPIO3 10： Reserved 11： Reserved	R/W	00

1.13 GPIO_CTL2

Offset = 0x53 default 0x00

Bit(s)	Name	Description	R/W	Reset
7:5		Reserved		
4:0	GPIO_INEN	GPIO[4:0]input enable 0：Disable 1：Enable	RW	0

1.14 GPIO_CTL2

Offset = 0x54 default 0x00

Bit(s)	Name	Description	R/W	Reset
7:5		Reserved		
4:0	GPIO_OUTEN	GPIO[4:0]output enable 0：Disable 1：Enable 在开启 Output 之前，需要先将 data 配好。	RW	0

1.15 GPIO_CTL3

Offset = 0x55

Bit(s)	Name	Description	R/W	Reset
7:5		Reserved		
4:0	GPIO_DAT	GPIO[4:0] DATA	R/W	0

2.1 BATVADC_DAT0

Offset = 0xa2

Bit(s)	Name	Description	R/W	Reset
7:0	BATVADC[7：0]	BATVADC 数据的低 8bit	R	X

2.2 BATVADC_DAT1

Offset = 0xa3

Bit(s)	Name	Description	R/W	Reset
7:6		Reserved		
5:0	BATVADC[13：8]	BATVADC 数据的高 6bit VBAT=BATVADC*0.26855mv+2.6V	R	X

```
BATVADC_VALUE_low =I2C_ReadByte(I2C_SLAVE_ADDR,0xa2); //low    8bit
BATVADC_VALUE_high=I2C_ReadByte(I2C_SLAVE_ADDR,0xa3); //high 6bit
if((BATVADC_VALUE_high & 0x20)==0x20)//补码
    {
    BATVOL[i]=2600-((~BATVADC_VALUE_low)+(~(BATVADC_VALUE_high & 0x1F))*256+1)*0.26855;
    }
else   //原码
    {
BATVOL[i]= 2600+(BATVADC_VALUE_low+BATVADC_VALUE_high*256)*0.26855; //mv 为单位
    }
```

2.3 BATIADC_DAT0

Offset = 0xa4

Bit(s)	Name	Description	R/W	Reset
7:0	BATIADC[7 ： 0]	BATIADC 数据的低 8bit	R	X

2.3 BATIADC_DAT1

Offset = 0xa5

Bit(s)	Name	Description	R/W	Reset
7:6		Reserved		
5:0	BATIADC[13： 8]	BATIADC 数据的高 6bit VBAT=BATiADC*0.745985mA	R	X

```
BATIADC_VALUE_low =I2C_ReadByte(I2C_SLAVE_ADDR,0xa4);
BATIADC_VALUE_high=I2C_ReadByte(I2C_SLAVE_ADDR,0xa5);
if((BATIADC_VALUE_high & 0x20)==0x20)//负值
        {
        char a=~BATIADC_VALUE_low;
        char b=(~(BATIADC_VALUE_high & 0x1F) & 0x1f);
        int c=b*256+a+1;
        BATCUR=-c*0.745985;
        //BATCUR=-(int)(((~BATIADC_VALUE_low)+(~(BATIADC_VALUE_high & 0x1F))*256+1)*0.745985);
        }
else //正值
        {
        BATCUR= (BATIADC_VALUE_high*256+BATIADC_VALUE_low)*0.745985;    //mA 为单位
```

2.4 BATOCV_DAT0

Offset = 0xa8

Bit(s)	Name	Description	R/W	Reset
7:0	BATOCV[7：0]	BATOCV 数据的低 8bit	R	X

2.5 BATOCV _DAT1

Offset = 0xa9

Bit(s)	Name	Description	R/W	Reset
7:6		Reserved		
5:0	BATOCV [13：8]	BATOCV 数据的高 6bit OCV=BATOCV*0.26855mv+2.6V	R	X

```
BATOCVADC_VALUE_low =I2C_ReadByte(I2C_SLAVE_ADDR,0xa8); //low    8bit
BATOCVADC _VALUE_high=I2C_ReadByte(I2C_SLAVE_ADDR,0xa9); //high 6bit
if((BATOCVADC _VALUE_high & 0x20)==0x20)//补码
    {
    BATVOL[i]=2600-((~ BATOCVADC _VALUE_low)+(~( BATOCVADC _VALUE_high & 0x1F))*256+1)*0.26855;
    }
else   //原码
    {
BATOCV= 2600+( BATOCVADC _VALUE_low BATOCVADC _VALUE_high*256)*0.26855; //mv 为单位
    }
```

注意：

BATOCV=BATVADC+BATIADC*预设电池内阻（BATIADC 有方向）

2.6 Reg_READ0

Offset = 0x71

Bit(s)	Name	Description	R/W	Reset
7:5		充电状态指示标志 000: idle 001: 涓流充电状态 010: 恒流充电状态 011: 恒压充电状态 100: 恒压停止充电检测 101: 充饱状态 110: 充电超时状态	R	X
4		Chgop 0: 1:	R	X
3		充电结束标志 0: 1: 充电结束	R	X
2		恒压超时标志 0: 1：恒压超时	R	X
1		充电超时 0: 1：充电超时	R	X
0		涓流充电超时 0: 1：涓流充电超时	R	X

2.7 Reg_READ1

Offset = 0x72

Bit(s)	Name	Description	R/W	Reset
7		Light LED 是否有接灯 0：无 light LED 灯 1：有 light LED 灯	R	X
6		轻载标志位 0：重载状态(输出端电流大于 75mA) 1：轻载状态(输出端电流小于 75mA)		
5		输入过压状态 0：VIN 电压小于 5.6V 1：VIN 电压大于 5.6V，输入过压状态		

4：0		Reserved	R	X

2.8 Reg_READ2

Offset = 0x77

Bit(s)	Name	Description	R/W	Reset
7:4		Reserved	R	X
3		按键按键标志 0：当前按键没有按键 1：当前按键按下	R	X
2		Reserved		
1		按键长按标志位 0： 1：onoff 发生长按	R	X
0		按键短按标志位 0： 1：onoff 发生短按	R	X