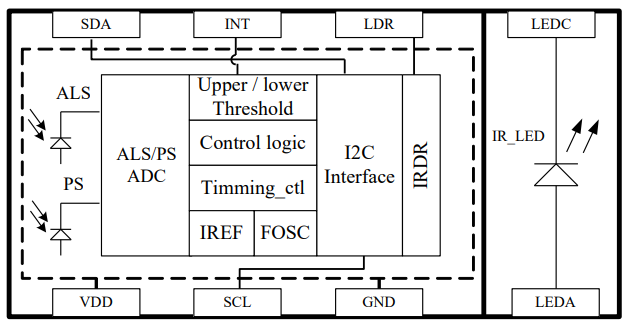
# AP3216C传感器

## 1.AP3216C介绍

AP3216C是敦南科技推出的一款传感器，在单个封装里包含了一个数字环境光传感器（ALS）、一个接近传感器（PS）和一个红外LED,环境光传感器具有16位分辨率，接近传感器和红外传感器具有10位分辨率。芯片通过IIC接口通信，支持IIC快速模式（400kHz）。

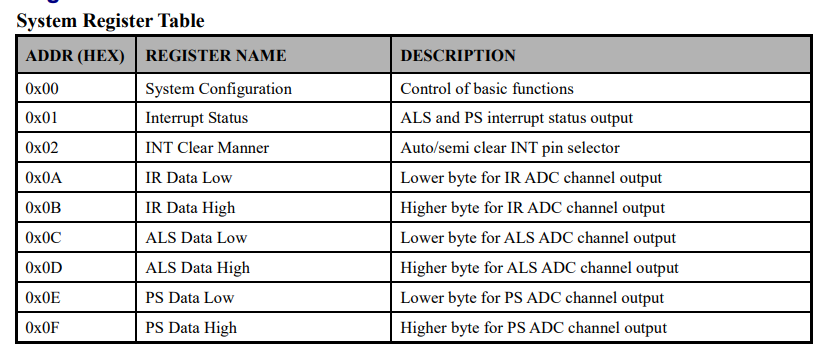
引脚框图如下所示：

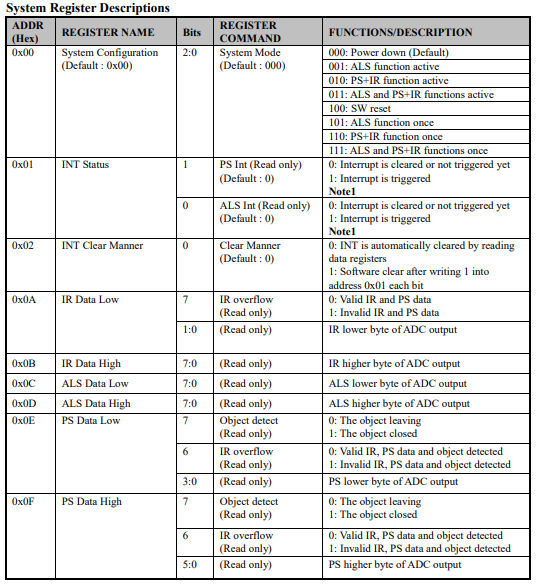


|  |  |  |
| --- | --- | --- |
| 引脚号 | 引脚名 | 功能 |
| 1 | VDD | 数字模拟电源，接3.3V |
| 2 | SCL | IIC串行时钟信号 |
| 3 | GND | 接地 |
| 4 | LEDA | LED阳极，接3.3V |
| 5 | LEDC | LED阴极，连接LDR |
| 6 | LDR | LED驱动输出引脚，一般连接LEDC |
| 7 | INT | 中断输出引脚 |
| 8 | SDA | IIC串行数据信号 |

## 2.AP3216C寄存器

主要寄存器





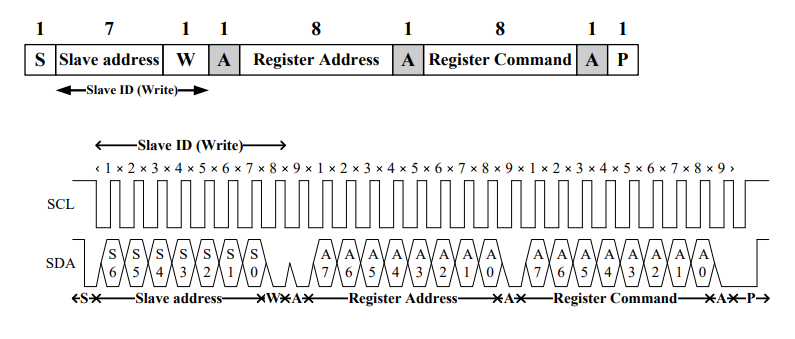


篇幅有限，其余寄存器的功能如需了解，可参看AP3216C手册。

## 3.通信协议

AP3216C使用标准IIC协议通讯（关于IIC协议这里不再具体阐述），设备地址为：0x1E。IIC协议使用的设备地址是7位，在通信时，还需要在设备地址上附加一个数据传输方向位。例如写操作时，要在设备地址的基础上左移1位，然后将最低位置0，即（0x1E<<1|0x00）。读操作时，要在设备地址的基础上左移1位，然后将最低位置1，即（0x1E<<1|0x01）。

AP3216C写操作时序如下：



S:起始信号

Slave address:AP3216C设备地址,7bit

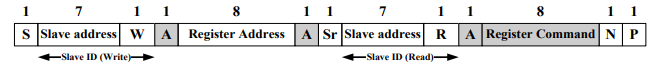
W:写标志，1bit,0

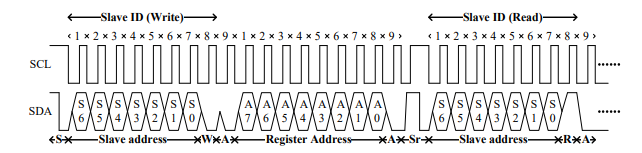
A:ACK信号

Register Address:AP3216C的寄存器地址

P:STOP信号

AP3216C读操作时序如下：





Sr:重复发送起始信号

R:读标志，1bit,1

N：NACK信号

## 4.程序实现

以下例程使用的MCU为沁恒的CH32V307VCT6，频率72MHz。本次使用硬件IIC控制AP3216C获取光照值，步骤如下：

1. 初始化IIC,设置主机地址、通信频率等基本参数。
2. 写入AP3216C系统配置寄存器（0x00），软件复位AP3216C，开启ALS功能.
3. 读取ALS数据寄存器（0x0C，0x0D）,整合为光照强度测量值

本次控制AP3216C使用硬件IIC，选用IIC2，SCL引脚是PB10,SDA引脚是PB11。

首先是IIC的初始化，需要先初始化GPIO和IIC2的时钟，GPIO配置成复用开漏，需要注意的是GPIO配置成复用开漏后需要有外部上拉，否则这两引脚无法输出高电平，影响通信。

    GPIO\_InitTypeDef GPIO\_InitStructure;

    I2C\_InitTypeDef I2C\_InitTSturcture;

    RCC\_APB2PeriphClockCmd( USER\_IIC\_GPIO\_CLK , ENABLE );

    RCC\_APB1PeriphClockCmd( USER\_IIC\_CLK, ENABLE );

    GPIO\_InitStructure.GPIO\_Pin = USER\_IIC\_SCL\_PIN;

    GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_OD;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

    GPIO\_Init( USER\_IIC\_GPIO\_PORT, &GPIO\_InitStructure );

    GPIO\_InitStructure.GPIO\_Pin = USER\_IIC\_SDA\_PIN;

    GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_OD;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

    GPIO\_Init( USER\_IIC\_GPIO\_PORT, &GPIO\_InitStructure );

接着是配置IIC,这里使用IIC2，时钟频率配置为100KHz,配置成400KHz也是可以，主机地址就配置为0x01。

    I2C\_InitTSturcture.I2C\_ClockSpeed = USER\_IIC\_BOUND;

    I2C\_InitTSturcture.I2C\_Mode = I2C\_Mode\_I2C;

    I2C\_InitTSturcture.I2C\_OwnAddress1 = USER\_IIC\_ADDR;

另外一提的是IIC的DutyCycle，这个指的是SCL线上低电平和高电平占的时间的比例，可以配置为两种，一种是2:1，一种是16:9,I2C设备通信时需要根据对方的Duty Cycle调整，不匹配的话通信可能会出问题，这里选择16:9是可以正常通信。

   I2C\_InitTSturcture.I2C\_DutyCycle = I2C\_DutyCycle\_16\_9;

接下来使能I2C应答，设置为使能才可以发送响应信号，禁止应答可能导致通信问题，一般使能。

    I2C\_InitTSturcture.I2C\_Ack = I2C\_Ack\_Enable;

最后配置IIC的寻址模式，这个可以配置成7为或10位，AP3216C使用的是7位地址，因此寻址模式配置成7位。

    I2C\_InitTSturcture.I2C\_AcknowledgedAddress = I2C\_AcknowledgedAddress\_7bit;

IIC初始化完成后，开始编写发送和接收函数。因为使用硬件IIC，我们就只需要按照IIC时序调用库函数就可以，每调用一次IIC库函数会产生相应的事件，我们需要对事件返回值作出判断，确保程序稳定执行。

发送一个字节流程如下，发送开始信号

I2C\_GenerateSTART(USER\_IIC, ENABLE);

等待事件五（表示主模式选择事件）发生，超时就跳出

        if (IIC\_WaitEvent(USER\_IIC, I2C\_EVENT\_MASTER\_MODE\_SELECT)) {

            res = 1;

            break;

        }

发送从设备地址，最后一位置0置1都行，第三个参数填发送，函数会自动修改。

I2C\_Send7bitAddress(USER\_IIC, (addr<<1), I2C\_Direction\_Transmitter); // 发送器件地址+写命令

等待事件六发生

    if(IIC\_WaitEvent(USER\_IIC, I2C\_EVENT\_MASTER\_TRANSMITTER\_MODE\_SELECTED))

        return 1; // 等待应答

等待发送数据寄存器空闲

while (I2C\_GetFlagStatus(USER\_IIC, I2C\_FLAG\_TXE) == RESET);

发送寄存器地址

I2C\_SendData(USER\_IIC, reg); // 写寄存器地址

再等待发送数据寄存器空闲

 while (I2C\_GetFlagStatus(USER\_IIC, I2C\_FLAG\_TXE) == RESET);

发送1字节数据

I2C\_SendData(USER\_IIC, data);

等待发送数据寄存器空闲，确保发送完毕

        while (I2C\_GetFlagStatus(USER\_IIC, I2C\_FLAG\_TXE) == RESET);

        break;

最后发送停止信号

I2C\_GenerateSTOP(USER\_IIC, ENABLE);

读取数据也是类似流程，依照IIC读操作时序读取数据。

需要注意的是，在接收最后一个字节之前关闭自动应答，返回NACK。发送停止信号之后，再重新启动自动ACK。

I2C\_AcknowledgeConfig(USER\_IIC, DISABLE);

I2C\_AcknowledgeConfig(USER\_IIC, ENABL);

完成IIC初始化后需要初始化AP3216C，首先复位模块，往系统配置寄存器(0x00)发送0x04，延时20ms后开启光照检测功能，往系统配置寄存器发送0x01,完成AP3216C的配置。

void AP3216C\_Init(void)

{

    IIC\_Init();                      //初始化IIC

    IIC\_WriteByte(AP3216C\_ADDR, 0x00, 0x04);  //复位模块

    Delay\_Ms(20);

    IIC\_WriteByte(AP3216C\_ADDR, 0x00, 0x01);   //开启ALS

}

读取数据就调用IIC读取函数，获取AP3216C光照强度低位数据寄存器（0x0C）和光照强度高位数据寄存器（0x0D）的数据即可。

void AP3216C\_ReadData(uint16\_t\* alsData)

{

    uint8\_t temp[2];

    temp[0] = IIC\_ReadByte(AP3216C\_ADDR,0X0C);//光照强度低位

    temp[1] = IIC\_ReadByte(AP3216C\_ADDR,0X0D);//光照强度高位

    \*alsData = ((uint16\_t)temp[1] << 8) | temp[0];                //读取ALS传感器的数据

}

**完整例程如下：**

**IIC.h**

#ifndef \_\_IIC\_H

#define \_\_IIC\_H

#include "debug.h"

#define USER\_IIC              I2C2

#define USER\_IIC\_CLK          RCC\_APB1Periph\_I2C2

#define USER\_IIC\_GPIO\_CLK     RCC\_APB2Periph\_GPIOB

#define USER\_IIC\_GPIO\_PORT    GPIOB

#define USER\_IIC\_SCL\_PIN      GPIO\_Pin\_10

#define USER\_IIC\_SDA\_PIN      GPIO\_Pin\_11

#define USER\_IIC\_BOUND        100000        //100KHz

#define USER\_IIC\_ADDR         0x01

void IIC\_Init();

u8 IIC\_WaitEvent(I2C\_TypeDef\* I2Cx, uint32\_t I2C\_EVENT);

u8 IIC\_ReadByte(u8 addr,u8 reg);

u8 IIC\_WriteLen(u8 addr, u8 reg, u8 len, u8 \*buf);

u8 IIC\_ReadLen(u8 addr, u8 reg, u8 len, u8 \*buf);

u8 IIC\_WriteByte(u8 addr,uint8\_t reg, uint8\_t data);

#endif

**IIC.c**

#include "IIC.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name  : IIC\_Init

\* Description    : Initializes the IIC peripheral.

\* Input          : None

\* Return         : None

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void IIC\_Init()

{

    GPIO\_InitTypeDef GPIO\_InitStructure;

    I2C\_InitTypeDef I2C\_InitTSturcture;

    RCC\_APB2PeriphClockCmd( USER\_IIC\_GPIO\_CLK , ENABLE );

    RCC\_APB1PeriphClockCmd( USER\_IIC\_CLK, ENABLE );

    GPIO\_InitStructure.GPIO\_Pin = USER\_IIC\_SCL\_PIN;

    GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_OD;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

    GPIO\_Init( USER\_IIC\_GPIO\_PORT, &GPIO\_InitStructure );

    GPIO\_InitStructure.GPIO\_Pin = USER\_IIC\_SDA\_PIN;

    GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_OD;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

    GPIO\_Init( USER\_IIC\_GPIO\_PORT, &GPIO\_InitStructure );

    I2C\_InitTSturcture.I2C\_ClockSpeed = USER\_IIC\_BOUND;

    I2C\_InitTSturcture.I2C\_Mode = I2C\_Mode\_I2C;

    I2C\_InitTSturcture.I2C\_DutyCycle = I2C\_DutyCycle\_16\_9;

    I2C\_InitTSturcture.I2C\_OwnAddress1 = USER\_IIC\_ADDR;

    I2C\_InitTSturcture.I2C\_Ack = I2C\_Ack\_Enable;

    I2C\_InitTSturcture.I2C\_AcknowledgedAddress = I2C\_AcknowledgedAddress\_7bit;

    I2C\_Init( USER\_IIC, &I2C\_InitTSturcture );

    I2C\_Cmd( USER\_IIC, ENABLE );

}

u8 IIC\_WaitEvent(I2C\_TypeDef\* I2Cx, uint32\_t I2C\_EVENT){

    u16 counter = 0xffff;

    while( !I2C\_CheckEvent( I2Cx, I2C\_EVENT ) ){

        counter--;

        if(counter == 0){

            return 1;

        }

    }

    return 0;

}

u8 IIC\_WriteLen(u8 addr, u8 reg, u8 len, u8 \*buf)

{

    u8 i = 0;

    I2C\_GenerateSTART(USER\_IIC, ENABLE);

    if(IIC\_WaitEvent(USER\_IIC, I2C\_EVENT\_MASTER\_MODE\_SELECT))

        return 1;

    I2C\_Send7bitAddress(USER\_IIC, (addr<<1) | 0, I2C\_Direction\_Transmitter); // 发送器件地址+写命令

    if(IIC\_WaitEvent(USER\_IIC, I2C\_EVENT\_MASTER\_TRANSMITTER\_MODE\_SELECTED))

        return 1; // 等待应答

    while (I2C\_GetFlagStatus(USER\_IIC, I2C\_FLAG\_TXE) == RESET)

        ;

    I2C\_SendData(USER\_IIC, reg); // 写寄存器地址

    while (i < len)

    {

        if (I2C\_GetFlagStatus(USER\_IIC, I2C\_FLAG\_TXE) != RESET)

        {

            I2C\_SendData(USER\_IIC, buf[i]); // 发送数据

            i++;

        }

    }

    while (I2C\_GetFlagStatus(USER\_IIC, I2C\_FLAG\_TXE) == RESET);

    I2C\_GenerateSTOP(USER\_IIC, ENABLE);

    return 0;

}

u8 IIC\_ReadLen(u8 addr, u8 reg, u8 len, u8 \*buf)

{

    u8 i = 0;

    I2C\_GenerateSTART(USER\_IIC, ENABLE);

    if(IIC\_WaitEvent(USER\_IIC, I2C\_EVENT\_MASTER\_MODE\_SELECT))return 1       ;

    I2C\_Send7bitAddress(USER\_IIC, (addr << 1) | 0X00, I2C\_Direction\_Transmitter); //发送器件地址+写命令

    if(IIC\_WaitEvent(USER\_IIC, I2C\_EVENT\_MASTER\_TRANSMITTER\_MODE\_SELECTED))return 1       ; //等待应答

    I2C\_SendData(USER\_IIC, reg); //写寄存器地址

    I2C\_GenerateSTART(USER\_IIC, ENABLE);

    if(IIC\_WaitEvent(USER\_IIC, I2C\_EVENT\_MASTER\_MODE\_SELECT))return 1       ;

    I2C\_Send7bitAddress(USER\_IIC, ((addr << 1) | 0x01), I2C\_Direction\_Receiver); //发送器件地址+读命令

    if(IIC\_WaitEvent(USER\_IIC, I2C\_EVENT\_MASTER\_RECEIVER\_MODE\_SELECTED))return 1       ; //等待应答

    while (i < len)

    {

        if (I2C\_GetFlagStatus(USER\_IIC, I2C\_FLAG\_RXNE) != RESET)

        {

            if (i == (len - 1))

            {

                I2C\_AcknowledgeConfig(USER\_IIC, DISABLE);//关闭自动ACK,不IIC总线不会被释放

                buf[i] = I2C\_ReceiveData(USER\_IIC); //读数据,发送nACK

            }

            else

            {

                buf[i] = I2C\_ReceiveData(USER\_IIC); //读数据,发送ACK

            }

            i++;

        }

    }

    I2C\_GenerateSTOP(USER\_IIC, ENABLE); //产生一个停止条件

    I2C\_AcknowledgeConfig(USER\_IIC, ENABLE);//重新启动自动ACK,不然程序会停在接收

    return 0;

}

u8 IIC\_ReadByte(u8 addr,u8 reg)

{

    u8 res;

    I2C\_GenerateSTART( USER\_IIC, ENABLE );

    while( !I2C\_CheckEvent( USER\_IIC, I2C\_EVENT\_MASTER\_MODE\_SELECT ) );

    I2C\_Send7bitAddress(USER\_IIC,(addr << 1) | 0X00,I2C\_Direction\_Transmitter); //发送器件地址+写命令

    if(IIC\_WaitEvent( USER\_IIC, I2C\_EVENT\_MASTER\_TRANSMITTER\_MODE\_SELECTED ) )return 1;  //等待应答

    I2C\_SendData(USER\_IIC,reg);         //写寄存器地址

    I2C\_GenerateSTART( USER\_IIC, ENABLE );

    while( !I2C\_CheckEvent( USER\_IIC, I2C\_EVENT\_MASTER\_MODE\_SELECT ) );

    I2C\_Send7bitAddress(USER\_IIC,((addr << 1) | 0x01),I2C\_Direction\_Receiver);//发送器件地址+读命令

    while( !I2C\_CheckEvent( USER\_IIC, I2C\_EVENT\_MASTER\_RECEIVER\_MODE\_SELECTED ) ); //等待应答

    I2C\_AcknowledgeConfig( USER\_IIC, DISABLE );

    while( I2C\_GetFlagStatus( USER\_IIC, I2C\_FLAG\_RXNE ) ==  RESET );

    res = I2C\_ReceiveData( USER\_IIC ); //读数据,发送nACK

    I2C\_AcknowledgeConfig(USER\_IIC, ENABLE);

    I2C\_GenerateSTOP( USER\_IIC, ENABLE );//产生一个停止条件

    return res;

}

u8 IIC\_WriteByte(u8 addr,uint8\_t reg, uint8\_t data)

{

    uint8\_t res = 0;

    I2C\_GenerateSTART(USER\_IIC, ENABLE);

    while (1) {

        if (IIC\_WaitEvent(USER\_IIC, I2C\_EVENT\_MASTER\_MODE\_SELECT)) {

            res = 1;

            break;

        }

        I2C\_Send7bitAddress(USER\_IIC, ((addr << 1) | 0), I2C\_Direction\_Transmitter);

        if (IIC\_WaitEvent(USER\_IIC, I2C\_EVENT\_MASTER\_TRANSMITTER\_MODE\_SELECTED)) {

            res = 1;

            break;

        }

        I2C\_SendData(USER\_IIC, reg);

        if (IIC\_WaitEvent(USER\_IIC, I2C\_EVENT\_MASTER\_BYTE\_TRANSMITTED)) {

            res = 1;

            break;

        }

        while (I2C\_GetFlagStatus(USER\_IIC, I2C\_FLAG\_TXE) == RESET);

        I2C\_SendData(USER\_IIC, data);

        while (I2C\_GetFlagStatus(USER\_IIC, I2C\_FLAG\_TXE) == RESET);

        break;

    }

    I2C\_GenerateSTOP(USER\_IIC, ENABLE);

    return res;

}

**AP3216C.h**

#ifndef \_\_AP3216C\_H

#define \_\_AP3216C\_H

#include "debug.h"

#define AP3216C\_ADDR    0X1E    //AP3216C器件IIC地址

void AP3216C\_Init(void);

void AP3216C\_ReadData(uint16\_t\* alsData);

#endif

**AP3216C.c**

#include "AP3216C.h"

#include "debug.h"

#include "IIC.h"

//光照传感器

void AP3216C\_Init(void)

{

    IIC\_Init();                      //初始化IIC

    IIC\_WriteByte(AP3216C\_ADDR, 0x00, 0x04);  //复位模块

    Delay\_Ms(20);

    IIC\_WriteByte(AP3216C\_ADDR, 0x00, 0x01);   //开启ALS

}

void AP3216C\_ReadData(uint16\_t\* alsData)

{

    uint8\_t temp[2];

    temp[0] = IIC\_ReadByte(AP3216C\_ADDR,0X0C);//光照强度低位

    temp[1] = IIC\_ReadByte(AP3216C\_ADDR,0X0D);//光照强度高位

    \*alsData = ((uint16\_t)temp[1] << 8) | temp[0];                //读取ALS传感器的数据

}