实验七

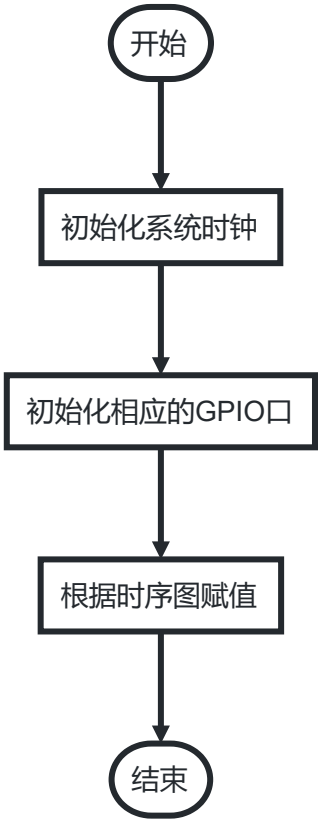
实验目的

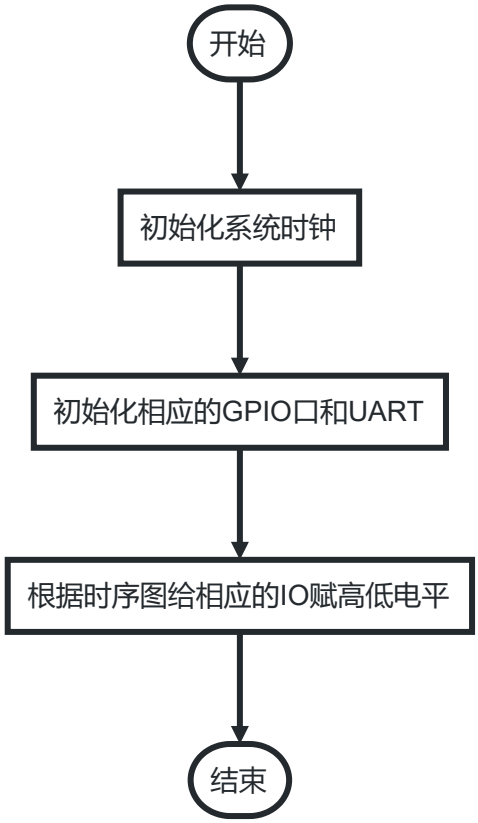
1. 熟悉并口 AD/DA 芯片的结构及工作方式
2. 熟悉并行口的扩展编程

实验内容

学习并使用并行ADC和DAC,能够根据时序图操作GPIO口

实验流程图





实验源代码

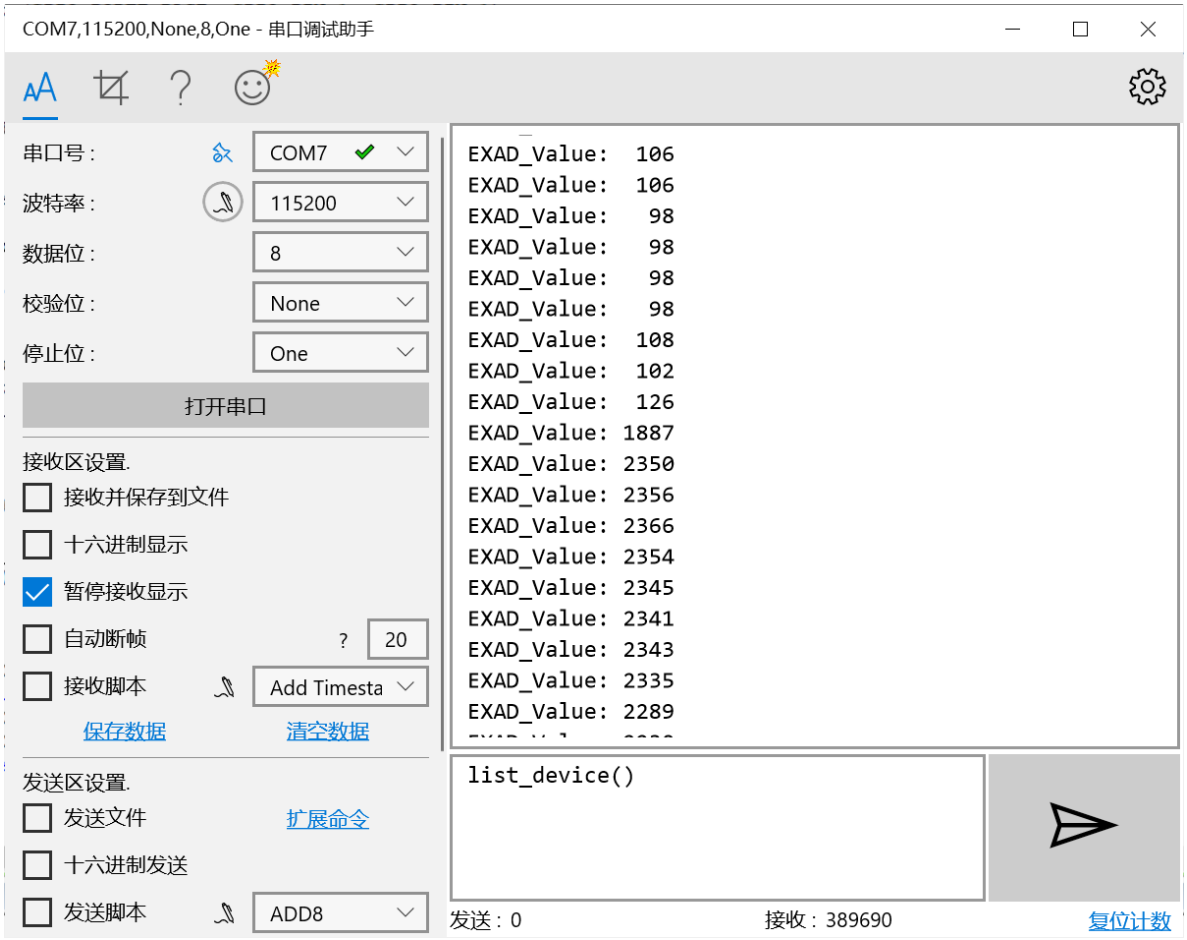
```
//DAC初始化
void DAC_Init()
{
    //CS  ~ PE1
    //WR  ~ PE2
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
    SysCtlGPIOAHBEnable(SYSCTL_PERIPH_GPIOE);
    GPIOPinTypeGPIOOutput(GPIO_PORTE_BASE, GPIO_PIN_1|GPIO_PIN_2);

    /*
        DATA0  ~ PK3
        DATA1  ~ PK2
        DATA2  ~ PK1
        DATA3  ~ PK0
        DATA4  ~ PC7
        DATA5  ~ PC6
        DATA6  ~ PC5
        DATA7  ~ PC4
        DATA8  ~ PA6
        DATA9  ~ PA7
        DATA10 ~ PG1
        DATA11 ~ PG0
    */

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOK);
    SysCtlGPIOAHBEnable(SYSCTL_PERIPH_GPIOK);
    GPIOPinTypeGPIOOutput(GPIO_PORTK_BASE, GPIO_PIN_3|GPIO_PIN_2|GPIO_PIN_1|GPIO_PIN_0);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC);
    SysCtlGPIOAHBEnable(SYSCTL_PERIPH_GPIOC);
    GPIOPinTypeGPIOOutput(GPIO_PORTC_BASE, GPIO_PIN_7|GPIO_PIN_6|GPIO_PIN_5|GPIO_PIN_4);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
    SysCtlGPIOAHBEnable(SYSCTL_PERIPH_GPIOA);
    GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_6|GPIO_PIN_7);
```

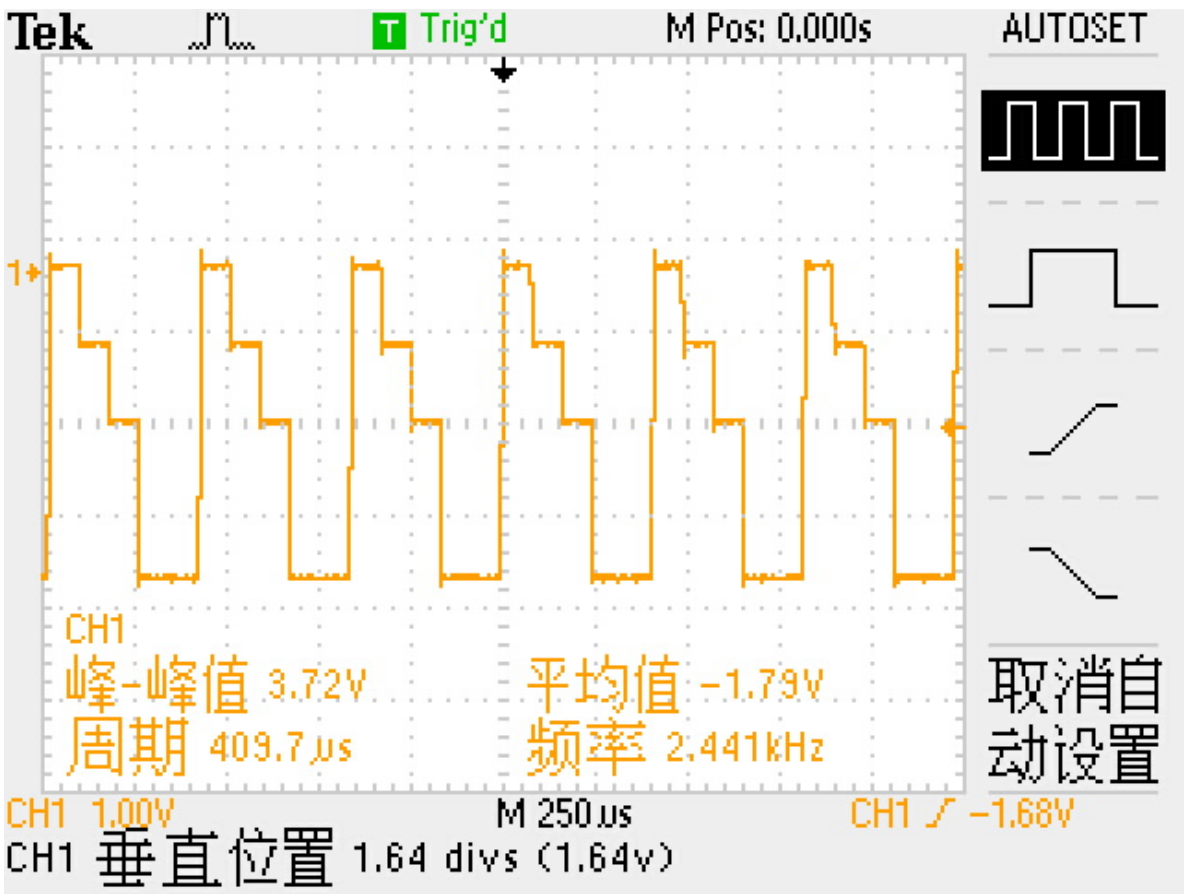
```
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOG);
    SysCtlGPIOAHBEnable(SYSCTL_PERIPH_GPIOG);
    GPIOPinTypeGPIOOutput(GPIO_PORTG_BASE, GPIO_PIN_1|GPIO_PIN_0);

}

//DAC输出电压
void DAC_Set_Data(uint16_t out)
{
    bool out_buffer[12];
    uint16_t temp;
    for(int i=0;i<12;i++)
    {
        out_buffer[i] = (out<<(4+i))>>15;
    }

    GPIOPinWrite(GPIO_PORTG_BASE, GPIO_PIN_0, out_buffer[0]?GPIO_PIN_0:0);  //DATA11
    GPIOPinWrite(GPIO_PORTG_BASE, GPIO_PIN_1, out_buffer[1]?GPIO_PIN_1:0);  //DATA10
    GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_7, out_buffer[2]?GPIO_PIN_7:0);  //DATA9
    GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_6, out_buffer[3]?GPIO_PIN_6:0);  //DATA8
    GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_4, out_buffer[4]?GPIO_PIN_4:0);  //DATA7
    GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_5, out_buffer[5]?GPIO_PIN_5:0);  //DATA6
    GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6, out_buffer[6]?GPIO_PIN_6:0);  //DATA5
    GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_7, out_buffer[7]?GPIO_PIN_7:0);  //DATA4
    GPIOPinWrite(GPIO_PORTK_BASE, GPIO_PIN_0, out_buffer[8]?GPIO_PIN_0:0);  //DATA3
    GPIOPinWrite(GPIO_PORTK_BASE, GPIO_PIN_1, out_buffer[9]?GPIO_PIN_1:0);  //DATA2
    GPIOPinWrite(GPIO_PORTK_BASE, GPIO_PIN_2, out_buffer[10]?GPIO_PIN_2:0);  //DATA1
    GPIOPinWrite(GPIO_PORTK_BASE, GPIO_PIN_3, out_buffer[11]?GPIO_PIN_3:0);  //DATA0
}

//out = 3.3*out/4096 (0<out<4096)
void DAC_Output(uint16_t out)
{
    //PULL UP CS
    GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_1, GPIO_PIN_1);
    SysCtlDelay(1);

    //PULL DOWN WR
    GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_2, 0);
    SysCtlDelay(1);

    //PULL DOWN CS
    GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_1, 0);
    SysCtlDelay(1);

    //SET DATA0~11
    DAC_Set_Data(out);
    SysCtlDelay(1);

    //PULL UP CS
    GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_1, GPIO_PIN_1);
    SysCtlDelay(1);

    //PULL up WR
    GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_2, GPIO_PIN_2);
    SysCtlDelay(1);
}

//DAC测试函数
void DAC_Test()
{
    uint16_t i =0;
    for(;;)
    {
        DAC_Output(i*1000);
        i++;
        if(i==5)i=0;
        SysCtlDelay(SysCtlClockGet()/3000);
    }
}


//ADC初始化
```

```c
void EXADC_Init()
{
    //RD~PE0    CS~PE1
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
    SysCtlGPIOAHBEnable(SYSCTL_PERIPH_GPIOE);
    GPIOPinTypeGPIOOutput(GPIO_PORTE_BASE, GPIO_PIN_1|GPIO_PIN_0);

    //AD_BUSY~PD6
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
    SysCtlGPIOAHBEnable(SYSCTL_PERIPH_GPIOD);
    GPIOPinTypeGPIOInput(GPIO_PORTD_BASE, GPIO_PIN_6);

    //AD_BYTE~PD5   AD_CONVST~PD4
    GPIOPinTypeGPIOOutput(GPIO_PORTD_BASE, GPIO_PIN_5|GPIO_PIN_4);

    /*
        DATA0  ~ PK3
        DATA1  ~ PK2
        DATA2  ~ PK1
        DATA3  ~ PK0
        DATA4  ~ PC7
        DATA5  ~ PC6
        DATA6  ~ PC5
        DATA7  ~ PC4
        DATA8  ~ PA6
        DATA9  ~ PA7
        DATA10 ~ PG1
        DATA11 ~ PG0
    */

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOK);
    SysCtlGPIOAHBEnable(SYSCTL_PERIPH_GPIOK);
    GPIOPinTypeGPIOInput(GPIO_PORTK_BASE, GPIO_PIN_3|GPIO_PIN_2|GPIO_PIN_1|GPIO_PIN_0);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC);
    SysCtlGPIOAHBEnable(SYSCTL_PERIPH_GPIOC);
    GPIOPinTypeGPIOInput(GPIO_PORTC_BASE, GPIO_PIN_7|GPIO_PIN_6|GPIO_PIN_5|GPIO_PIN_4);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
    SysCtlGPIOAHBEnable(SYSCTL_PERIPH_GPIOA);
    GPIOPinTypeGPIOInput(GPIO_PORTA_BASE, GPIO_PIN_6|GPIO_PIN_7);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOG);
    SysCtlGPIOAHBEnable(SYSCTL_PERIPH_GPIOG);
    GPIOPinTypeGPIOInput(GPIO_PORTG_BASE, GPIO_PIN_1|GPIO_PIN_0);
}

//ADC读取
uint16_t EXADC_Read()
{

    uint16_t exad_value=0;
    uint16_t input_buffer[12]={};

    //PULL CS & RD HIGH
    GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_0, GPIO_PIN_0);
    GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_1, GPIO_PIN_1);

    //PULL AD_CONVST LOW  AD_BYTE LOW
    GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_4, 0);
    GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_5, 0);

    //PULL CS LOW
    GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_1, 0);

    //PULL AD_CONVST HIGH
    GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_4, GPIO_PIN_4);
    SysCtlDelay(2);

    //PULL AD_CONVST LOW
    GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_4, 0);
    SysCtlDelay(3);

    //PULL RD LOW
```

```
    GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_0, 0);
    SysCtlDelay(1);

    //DATA READ
    input_buffer[0] = GPIOPinRead(GPIO_PORTG_BASE, GPIO_PIN_0);      //DATA11
    input_buffer[1] = GPIOPinRead(GPIO_PORTG_BASE, GPIO_PIN_1)>>1;   //DATA10
    input_buffer[2] = GPIOPinRead(GPIO_PORTA_BASE, GPIO_PIN_7)>>7;   //DATA9
    input_buffer[3] = GPIOPinRead(GPIO_PORTA_BASE, GPIO_PIN_6)>>6;   //DATA8
    input_buffer[4] = GPIOPinRead(GPIO_PORTC_BASE, GPIO_PIN_4)>>4;   //DATA7
    input_buffer[5] = GPIOPinRead(GPIO_PORTC_BASE, GPIO_PIN_5)>>5;   //DATA6
    input_buffer[6] = GPIOPinRead(GPIO_PORTC_BASE, GPIO_PIN_6)>>6;   //DATA5
    input_buffer[7] = GPIOPinRead(GPIO_PORTC_BASE, GPIO_PIN_7)>>7;   //DATA4
    input_buffer[8] = GPIOPinRead(GPIO_PORTK_BASE, GPIO_PIN_0);      //DATA3
    input_buffer[9] = GPIOPinRead(GPIO_PORTK_BASE, GPIO_PIN_1)>>1;   //DATA2
    input_buffer[10] = GPIOPinRead(GPIO_PORTK_BASE, GPIO_PIN_2)>>2;  //DATA1
    input_buffer[11] = GPIOPinRead(GPIO_PORTK_BASE, GPIO_PIN_3)>>3;  //DATA0

    //PULL CS & RD HIGH
    GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_0, GPIO_PIN_0);
    GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_1, GPIO_PIN_1);

    exad_value =
        input_buffer[0]*2048+ input_buffer[1]*1024+ input_buffer[2]*512 +\
        input_buffer[3]*256 + input_buffer[4]*128 + input_buffer[5]*64 +\
        input_buffer[6]*32  + input_buffer[7]*16  + input_buffer[8]*8 +\
        input_buffer[9]*4   + input_buffer[10]*2  + input_buffer[11];

    return exad_value;
}

//ADC测试函数
void EXADC_Test()
{
    uint16_t exad_value;
    float v;
    char s[20];
    while(1)
    {
        exad_value = EXADC_Read();
        UARTprintf("%4d\n",exad_value);
        SysCtlDelay(SysCtlClockGet()/300);
    }
}
```

实验现象





思考题

已使用库函数实现

实验八

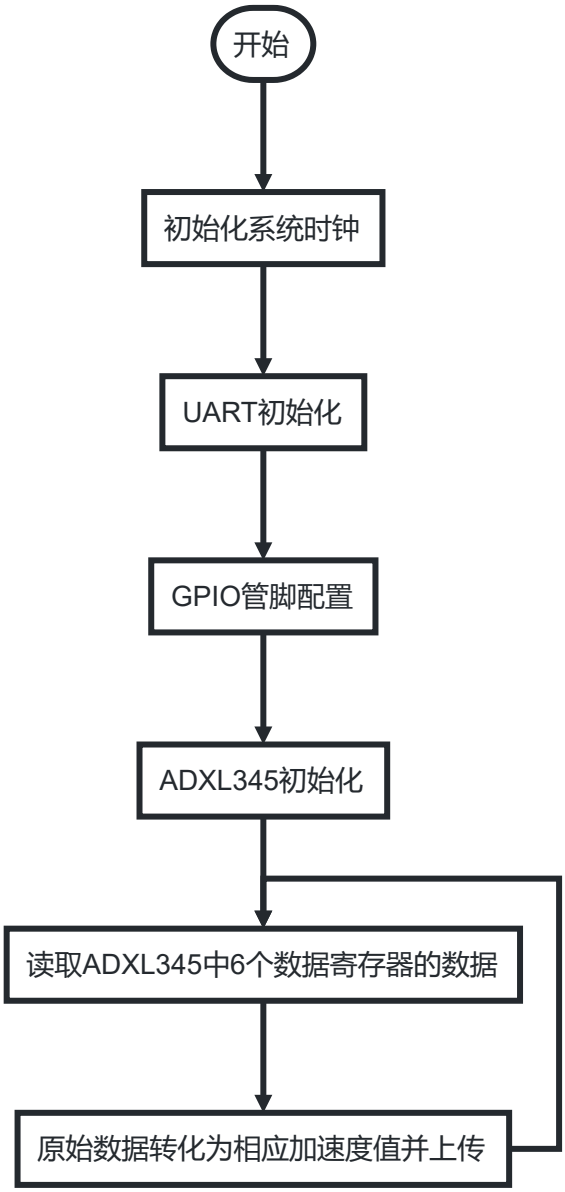
实验目的

1. 了解 I2C 总线的特点和功能；
2. 了解加速度传感器的原理；
3. 学会使用 I2C 总线对 ADXL345 芯片进行操作

实验内容

读取三轴加速计的三个数据并通过串口显示

实验流程图



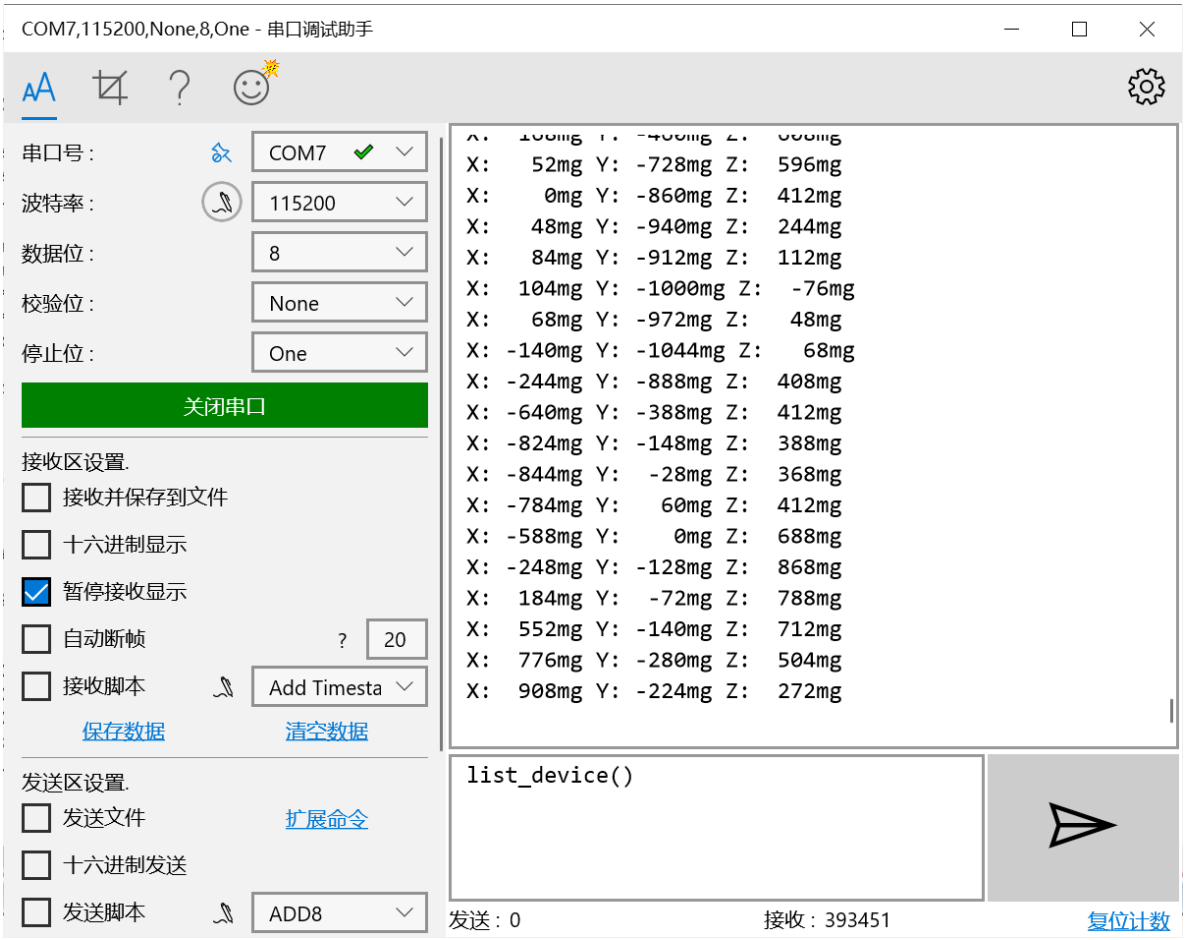
实验源代码

```
//加速计初始化
void ACCELERATOR_Init(void)
```

```
{
    //管脚配置
    I2C0GPIOBEnable();

    char buffer;

    //数据格式
    buffer = 0x0B;
    I2CSend(ACCELERATOR_W,ACCELERATOR_DATA_FORMAT,&buffer,1);
    //测量范围
    buffer = 0x18;
    I2CSend(ACCELERATOR_W,ACCELERATOR_BW_RATE,&buffer,1);
    //电源控制
    buffer = 0x08;
    I2CSend(ACCELERATOR_W,ACCELERATOR_POWER_CTL,&buffer,1);
    //中断控制
    buffer = 0x00;
    I2CSend(ACCELERATOR_W,ACCELERATOR_INT_ENABLE,&buffer,1);

    //X轴偏置
    buffer = 0x00;
    I2CSend(ACCELERATOR_W,ACCELERATOR_OFSX,&buffer,1);
    //Y轴偏置
    buffer = 0x00;
    I2CSend(ACCELERATOR_W,ACCELERATOR_OFSY,&buffer,1);
    //Z轴偏置
    buffer = 0x00;
    I2CSend(ACCELERATOR_W,ACCELERATOR_OFSZ,&buffer,1);


    uint8_t devid = I2CRead(ACCELERATOR_R,ACCELERATOR_DEVID);
    if(devid == 0xe5)
    {
        UARTprintf("accelerator init success!\n");
    }

}

//加速计读取指定轴数据
int ACCELERATOR_Read(char channel)
{
    int res;
    uint8_t register_value[2]={};
    switch(channel)
    {
        case('X'):
        {
            register_value[0] = I2CRead(ACCELERATOR_R,ACCELERATOR_X_L);
            register_value[1] = I2CRead(ACCELERATOR_R,ACCELERATOR_X_H);
            if(register_value[1]>16)
            {
                register_value[1]=0xFF-register_value[1];
                register_value[0]=0xFF-register_value[0];
                res = register_value[0]*4 + register_value[1]*1024;
                res = res*-1;
                break;
            }
            res = register_value[0]*4 + register_value[1]*1024;
            break;
        }

            case('Y'):
        {
            register_value[0] = I2CRead(ACCELERATOR_R,ACCELERATOR_Y_L);
            register_value[1] = I2CRead(ACCELERATOR_R,ACCELERATOR_Y_H);
            if(register_value[1]>16)
            {
                register_value[1]=0xFF-register_value[1];
                register_value[0]=0xFF-register_value[0];
                res = register_value[0]*4 + register_value[1]*1024;
                res = res*-1;
                break;
            }
            res = register_value[0]*4 + register_value[1]*1024;
```

```
            break;
        }
            case('Z'):
        {
            register_value[0] = I2CRead(ACCELERATOR_R,ACCELERATOR_Z_L);
            register_value[1] = I2CRead(ACCELERATOR_R,ACCELERATOR_Z_H);
            if(register_value[1]>16)
            {
                register_value[1]=0xFF-register_value[1];
                register_value[0]=0xFF-register_value[0];
                res = register_value[0]*4 + register_value[1]*1024;
                res = res*-1;
                break;
            }
            res = register_value[0]*4 + register_value[1]*1024;
            break;
        }
    }
    return res;
}

//加速剂测试函数
void ACCELERATOR_Test(void)
{
    int Gx,Gy,Gz;
    while(1)
    {
        Gx = ACCELERATOR_Read('X');
        Gy = ACCELERATOR_Read('Y');
        Gz = ACCELERATOR_Read('Z');
        UARTprintf("X: %4dmg Y: %4dmg Z: %4dmg\n",Gx,Gy,Gz);
        SysCtlDelay(1000*SysCtlClockGet()/3000);
    }
}

```

实验现象



思考题

1.

寄存器名称	寄存器数值	实现功能
BW_RATE	0x0B	正常模式,25HZ
POWER_CTL	0x08	非连接,非自动sleep,测量模式,正常模式,8HZ
DATA_FORMAT	0x0B	±16g
OFSX	0x00	偏置为0
OFSY	0x00	偏置为0
OFZ	0x00	偏置为0

2. 只需要在初始化时调整各个轴的OFFSET即可.
3. 已经实现单独读取每个轴数据