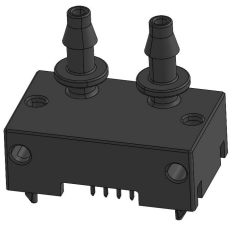
XGZP6810D PRESSURE SENSOR

FEATURES

- Wide Ranges: ±500Pa/±125Pa
- 3.0V ~ 5.5V Power Supply
- Differential Pressure Type
- Air or Nitrogen, Oxygen(non-condensing)
- I2C Interface
- Temp. Compensated: 0℃ ~ +50℃
- Excellent repeatability, extremely low drift
- Affordable Cost, Easy-to-use



APPLICATIONS

- Medical&Healthcare(such like respiratory machine, CPAP)
- Industrial&Automation(such like HVAC)
- Domestic Appliance(such like Gas boilers)
- New energy (fuel cells, heat recovery systems)
- Fire Residual Pressure Monitoring

INTRODUCTION

The XGZP6810D pressure sensor is CFSensor's series of differential pressure sensors designed for high-volume applications with an affordable cost and perfect performance. The pressure is measured by a thermal sensor element using flow-through technology. The XGZP6810D provide ±500 Pa and ±125 Pa pressure range for measuring air and non-aggressive gases with high accuracy and extremely low offset drift

XGZP6810D is a prefect silicon pressure sensor offering a ratiometric digital data(I2C interface) for reading differential pressure over the specified full scale pressure span.

The XGZP6810D is fully calibrated and temperature compensated for specified span, so XGZP6899D pressure sensor satisfy the perfect accuracy, which is designed for a wide range of application in medical care&health, home appliances, industry, air flow measurement, IoT and other pneumatic devices/meters etc by utilizing a microcontroller or microprocessor with I2C interface.

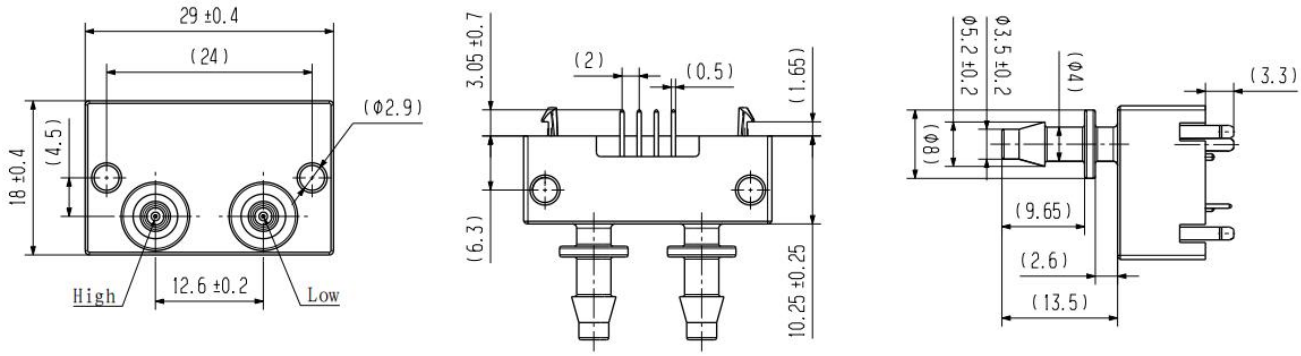
PERFORMANCE SPECIFICATION

Unless otherwise specified, measurements were taken by 5Vdc with temperature of 25±1℃ and humidity from 40% ~ 60 % RH.

CHARACTERISTIC		MIN.	TYP.	MAX	UNIT
Power Supply		3.0		5.5	Vdc
Working Current		-	18	21	mA
Accuracy	Offset	-	0.3	0.5	Pa
	Span	-	-	3	%Reading
Repeatability	Offset	-	0.1	0.2	Pa
	Span	-	0.5	-	%Reading
Temperature Measurement Range		-20	-	85	℃
Temperature Accuracy（-10~60℃）		-2	-	2	℃
Temperature Repeatability		-	0.3	-	℃
Burst Pressure		5	-	-	Bar
Compensation Temperature		0	-	50	℃
Operating Temperature		-20	-	65	℃
Storage Temperature		-40	-	85	℃
Power on Time		-	25	-	ms
Resolution		-	16	-	bit
SCL Frequency		-	200	400	KHz
Data Refresh Rate		-	1000	-	Hz
Response Time		-	10	-	mS
Electronic Port		2.0mm 4 PIN			

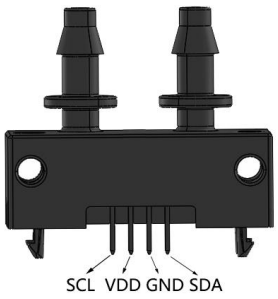
Note: The temperature referred to in the table is the temperature inside the sensor. The temperature value depends not only on the gas temperature but also on the ambient temperature around the sensor.

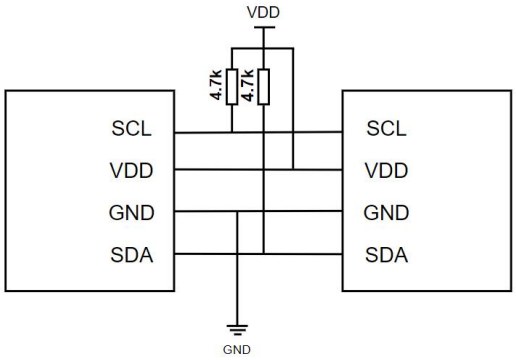
DIMENSION(Unit:mm Unspecified Tolerances:±0.1mm)



PIN DEFINITION

SCL	VDD	GND	SDA
SCL	Clock Line		
VDD	Voltage supply		
GND	Ground		
SDA	Data signal(Send& Receive)		





APPLICATION CUICUIT

CONTACT CFSENSOR FOR ABOVE FILE IF REQUIRED.

ORDER GUIDE (100kPa=0.1MPa=1bar≈14.5PSI)

XGZP6810 D 500 PA D

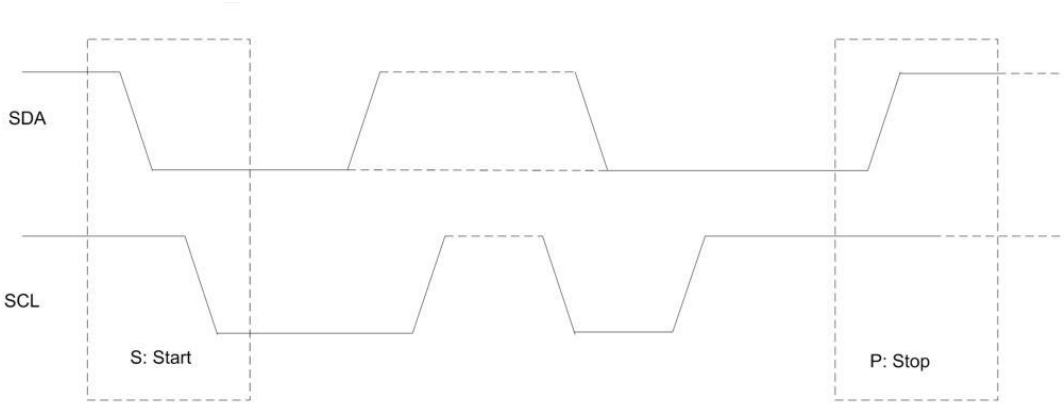
- D: Pressure Type(D:Differential Pressure Type)
- PA: Pressure Unit(PA)
- 500: Pressure Range(500 or 125)
- D: Output Signal(A: Analog D:I2C)
- XGZP6810: Series

Note: Custom requirement or parameter, please consult CFSensor and comment custom code herewith Part number.

COMMUNICATION PROTOCOL

I2C bus uses SCL and SDA as signal lines. Both lines are connected to VDD externally via pull-up resistors(Typ value:**4.7kΩ**) so that they are pulled high when the bus is free. I2C device factory setting slave address: **0X25.**

I2C Time Diagram

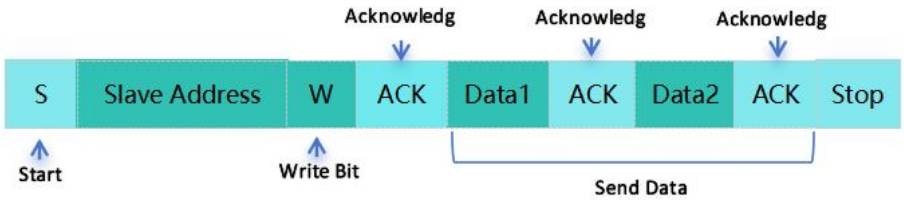


The I2C interface protocol has special bus signal conditions. Start (S), stop (P) and binary data conditions are shown above. At start condition, SCL is high and SDA has a falling edge. Then the slave address is sent. After the 7 address bits, the direction control bit R/W selects the read or write operation. When a slave device recognizes that it is being addressed, it should acknowledge by pulling SDA low in the ninth SCL (ACK) cycle.
At stop condition, SCL is also high, but SDA has a rising edge. Data must be held stable at SDA when SCL is high. Data can change value at SDA only when SCL is low.

Slave Address							R/W
A6	A5	A4	A3	A2	A1	A0	1/0

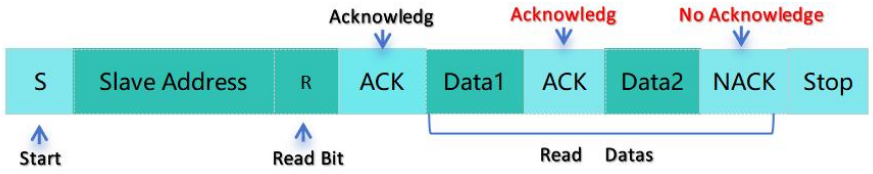
Write Data

The master device to write multiple data transmission format to the slave device is shown below, the host sends the write address, the slave responds to the address, the host sends multiple 8-bit data, the slave sends back the response signal in turn, and the host sends a stop signal to indicate the end of the data transmission after sending the data.



Read Data

After the host sends the read address, the slave sends a response signal, and then the slave returns the corresponding data in sequence according to the SCL signal provided by the slave, note that after returning the data, the host needs to respond to the data, and the slave decides whether it wants to continue to send the data according to the response signal of the host, when the host sends a NACK indicating that the host no longer needs to read the data, and finally the host sends the end-of-transmission signal.



COMMANDS

Command	Command Desc.	Return from Slave
36 1E	Continuous Measurement	6 bits
E1 02	Sensor Type	3 bits

Below are the details about Continuous Measurement Command and Sensor Type Command:

Continuous Measurement Command

Send Format(Host to Slave)

	Address	Continuous Measurement Command	
Send Format	Byte1	Byte2	Byte3
	0x4A	0x36	0x1E

Return Data Format

	Pressure Raw Data		CRC	Temperature Raw Data		CRC	Pressure Scaling Factor		CRC
Return Format	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8	Byte9
	xx	xx	xx	xx	xx	xx	xx	xx	xx

Note: xx indicates the data returned by the slave
Example:
Host sends: 4A 36 1E
Slave returns: 08 60 0D 13 66 DF 00 3C 39
Where 0x0860 is the raw data of differential pressure, 0x1366 is the raw data of temperature, and their corresponding CRC8 checksum values are 0x0D and 0xDF respectively.

According to the formula in the following table,
the differential pressure is 0x0860/60 = 2400/60 = 35 (Pa)
the temperature is 0x1366/200 = 4966/200 = 24.83 (℃)

Pressure Calculation Formula	Pressure Raw Data/60
Temperature Calculation Formula	Pressure Raw Data/200

Sensor Type Command

Send Format(Host to Slave)

	Address	Sensor Type Command	
Send Format	Byte1	Byte1	Byte2
	0x4A	0x4A	0xE1

Return Data Format

	Sensor Type Data		CRC
Send Format	Byte1	Byte2	Byte3
	xx	xx	xx

Example:

Host sends: 4A E1 02

Slave returns: 68 10 A0

Where 0x6810 is the product type code of XGZP6810 and 0xA0 is the checksum value of CRC8 of 0x6810.

CRC8 Check

There are three general standards of CRC: CRC8, CRC16, and CRC32, and CRC8 is used in the GZP6810. Taking the polynomial x8 + x5 + x4 + 1 (0x31) as an example, the CRC8 checksum byte is generated by the CRC algorithm with the attributes shown as below:

Property	Value
Length	8bit
Multinomial	0x31(x8 + x5 + x4 + 1)
Initial value	0xFF
Whether the input needs to be inverted	False
Whether the Output needs to be inverted	False
Final XOR value	0x00
Example	CRC(0xBEEF)=0x92

The C code to calculate the CRC code is as follows:

```
//*****************************************************
//Function name : Calc_CRC8
///Function : CRC8 calculation, initial value: 0xFF , polynomial: 0x31(x8 + x5 + x4 +1)
//Parameters: u8 *data: the first number of CRC validation; u8 Num: the length of the CRC validation data.
//Return : crc: the value of the calculated crc8.
//*****************************************************
u8 Calc_CRC8(u8 *data, u8 Len)
{
u8 bit, byte, crc = 0xFF.
for( byte = 0; byte < Len; byte++)
{
crc^ = (data[ byte ] ).
for( bit = 8 ;bit > 0; --bit)
{
if( crc & 0x80 ) crc = ( crc << 1 )^ 0x31;
else crc = ( crc << 1 );
}
}
return crc; }
                }
```

PACKING INFORMATION

Packing	Tray	Inner Box	Outer Box	Note
Quantity	80PCS per Tray	240pcs(3pcs Tray)	960pcs(4pcs Inner Box)	Anti-static bag

Note: 1, The sensor should be stored in an ESD protective container before using them.
2, The packing information may be not quite same with above for other different quantity and samples.

OVERALL NOTES

Unless otherwise specified, following notes are general attention or presentation for all products from CFSensor.

Mounting

The following steps is for transmitting the air pressure to sensor after sensor soldering on PCB.

▼ For some sensors that come with inlet tube, select the flexible pipe to suit the pressure inlet that is firm enough to prevent the pressure leaks.

▼ Atmosphere hole (for Gauge type sensors) and Inlet pipe/hole can't be blocked with gel or glue etc,..

▼ Avoiding excessive external force operation

Soldering

Due to its small size, the thermal capacity of the pressure sensor is low. Therefore, take steps to minimize the effects of external heat. Damage and changes to characteristics may occur due to heat deformation. Use a non-corrosive resin type of flux. Since the pressure sensor is exposed to the atmosphere, do not allow flux to enter inside. Reflow soldering is not feasible and may damage the sensor, so the manual soldering is recommended.

⊙Raise the temperature of the soldering tip between 260 and 300°C/500 and 572°F (30 W) and solder within 5 seconds.⊙ The sensor output may vary if the load is applied on the terminal during soldering.

⊙Keep the soldering tip clean.

Connecting

▼ Correctly wire as in the connection diagram. Reverse connection may damage the product and degrade the performance.

▼ Do not use idle terminals(N/C) to prevent damages to the sensor.

Cleaning

▼ Since the pressure sensor is exposed to the atmosphere, do not allow cleaning fluid to enter inside from atmosphere hole (for Gauge type sensors) and inlet pipe.

▼ Avoid ultrasonic cleaning since this may cause breaks or disconnections in the wiring.

Environment

▼Please avoid using or storing the pressure sensor in a place exposed to corrosive gases (such as the gases given off by organic solvents, sulfurous acid gas, hydrogen sulfides, etc.) which will adversely affect the performance of the pressure sensor chip.

▼Since this pressure sensor itself does not have a water-proof construction(even available media can be liquid), please do not use the sensor in a location where it may be sprayed with water, etc.

▼Avoid using the pressure sensors in an environment where condensation may form. Furthermore, its output may fluctuate if any moisture adhering to it freezes.

▼The pressure sensor is constructed in such a way that its output will fluctuate when it is exposed to light. Especially when pressure is to be applied by means of a transparent tube, take steps to prevent the pressure sensor chip from being exposed to light.

▼Avoid using pressure sensor where it will be susceptible to ultrasonic or other high-frequency vibration.

▼Keeping the sensors sealed in static shielding bags with an oxygen-free condition and use the sensor as soon as possible once unfold the package, because the sensors' PINs may be oxidated a bit under atmosphere environment(slight oxidation wouldn't affect soldering and performance)

More Precautions

▼That using the wrong pressure range or mounting method may result in accidents.

▼The only direct pressure medium you can use is non-corrosive gas or air as illuminated above(Note: some sensors are compatible with liquid media). The use of other media, in particular, corrosive gases and liquid (organic solvent based, sulfurous acid based, and hydrogen sulfide based, etc.) or contains foreign substances will cause malfunction and damage. Please do not use them and check with CFSensor.

▼The pressure sensor is positioned inside the pressure inlet. Never poke wires or other foreign matter through the pressure inlet since they may damage the sensor or block the inlet. Avoid use when the atmospheric pressure inlet(only for Gauge type pressure sensor) is blocked.

▼Use an operating pressure which is within the rated pressure range. Using a pressure beyond this range may cause damage.

▼Since static charge can damage the pressure sensor, bear in mind the following handling precautions.

⊙When storing the pressure sensor, use a conductive material to short the pins or wrap the entire sensor in aluminum foil. Common plastic containers should not be used to store or transport the sensor since they readily become charged.

⊙When using the pressure sensor, all the charged articles on the bench surface and the work personnel should be grounded so that any ambient static will be safely discharged.

▼Based on the pressure involved, give due consideration to the securing of the pressure sensor.

【 SAFETY NOTES 】

Using these sensors products may malfunction due to external interference and surges, therefore, please confirm the performance and quality in actual use. Just in case, please make a safety design on the device (fuse, circuit breaker, such as the installation of protection circuits, multiple devices, etc.), so it would not harm life, body, property, etc even a malfunction occurs. To prevent injuries and accidents, please be sure to observe the following items:

●The driving current and voltage should be used below the rated value.

●Please follow the terminal connection diagram for wiring. Especially for the reverse connection of the power supply, it will cause an accident due to circuit damage such as heat, smoke, fire, etc.

●In order to ensure safety, especially for important uses, please be sure to consider double safety circuit configuration.

●Do not apply pressure above the maximum applied pressure. In addition, please be careful not to mix foreign matter into the pressure medium. Otherwise, the sensor will be discarded, or the media will blew out and cause an accident.

● Be careful when fixing the product and connecting the pressure inlet. Otherwise, accidents may occur due to sensor scattering and the blowing out of the media.

●If the sensor come with sharp PIN, please be careful not to hurt your body when using it.

【 WARRANTY 】

The information in this sheet has been carefully reviewed and is believed to be accurate; however, no responsibility is assumed for inaccuracies. Furthermore, this information does not convey to the purchaser of such devices any license under the patent rights to the manufacturer. CFSensor reserves the right to make changes without further notice to any product herein. CFSensor makes no warranty, representation or guarantee regarding the suitability of its product for any particular purpose, nor does CFSensor assume any liability arising out of the application or use of any product or circuit and specifically disclaims any and all liability, including without limitation consequential or incidental damages. Typical parameters can and do vary in different applications. All operating parameters must be validated for each customer application by customer's technical experts. CFSensor does not convey any license under its patent rights nor the rights of others.

【 CONTACT 】

CFSensor

22F/14Bldg High-Tech Park High-Tech Area Wuhu P.R.C.241000

Tel/Fax:+86 18226771331 Email:INFO@CFSensor.com

North America II Europe II Southeast Asia II Middle East II Latin America

Code for reference

```
#define IIC_SCL_GPIO_PORT                GPIOB
#define IIC_SCL_GPIO_PIN                 GPIO_PIN_6
#define IIC_SCL_GPIO_CLK_ENABLE()        do{ __HAL_RCC_GPIOB_CLK_ENABLE(); }while(0)
#define IIC_SDA_GPIO_PORT                GPIOB
#define IIC_SDA_GPIO_PIN                 GPIO_PIN_7
#define IIC_SDA_GPIO_CLK_ENABLE()        do{ __HAL_RCC_GPIOB_CLK_ENABLE(); }while(0)
#define IIC_SCL(1)      GPIO_WritePin(IIC_SCL_GPIO_PORT,IIC_SCL_GPIO_PIN, GPIO_PIN_SET)
#define IIC_SCL(0)      GPIO_WritePin(IIC_SCL_GPIO_PORT,IIC_SCL_GPIO_PIN, GPIO_PIN_RESET)
#define IIC_SDA(1)      GPIO_WritePin( IIC_SDA_GPIO_PORT,IIC_SDA_GPIO_PIN, GPIO_PIN_SET)
#define IIC_SDA(0)      GPIO_WritePin(IIC_SDA_GPIO_PORT,IIC_SDA_GPIO_PIN, GPIO_PIN_RESET)
#define IIC_READ_SDA    GPIO_ReadPin(IIC_SDA_GPIO_PORT, IIC_SDA_GPIO_PIN)

void IIC_Init(void)
{
    GPIO_InitTypeDef gpio_init_struct;

    IIC_SCL_GPIO_CLK_ENABLE();          //SCL 引脚时钟使能
    IIC_SDA_GPIO_CLK_ENABLE();          // SDA 引脚时钟使能

    gpio_init_struct.Pin = IIC_SCL_GPIO_PIN;
    gpio_init_struct.Mode = GPIO_MODE_OUTPUT_PP;        // 推挽输出
    gpio_init_struct.Pull = GPIO_PULLUP;                // 上拉
    gpio_init_struct.Speed = GPIO_SPEED_FREQ_HIGH;      // 高速
    HAL_GPIO_Init(IIC_SCL_GPIO_PORT, &gpio_init_struct);    // SCL

    gpio_init_struct.Pin = IIC_SDA_GPIO_PIN;
    gpio_init_struct.Mode = GPIO_MODE_OUTPUT_OD;        // 开漏输出
    HAL_GPIO_Init(IIC_SDA_GPIO_PORT, &gpio_init_struct);    //  SDA
    IIC_Stop();                                         // 停止总线上所有设备
}

static void IIC_Delay(void)
{
    delay_us(3);
}

void IIC_Start(void)
{
    IIC_SDA(1);
    IIC_SCL(1);
```

```
    IIC_Delay();
    IIC_SDA(0);                // START 信号: 当 SCL 为高时, SDA 从高变成低, 表示起始信号
    IIC_Delay();
    IIC_SCL(0);
    IIC_Delay();
}

void IIC_Stop(void)
{
    IIC_SDA(0);                // STOP 信号: 当 SCL 为高时, SDA 从低变成高, 表示停止信号
    IIC_Delay();
    IIC_SCL(1);
    IIC_Delay();
    IIC_SDA(1);
    IIC_Delay();
}

uint8_t IIC_Wait_Ack(void)
{
    uint8_t Waitime = 0;
    uint8_t Rack = 0;

    IIC_SDA(1);              //  主机释放 SDA 线(此时外部器件可以拉低 SDA 线)
    IIC_Delay();
  IIC_Delay();            //第 8 个下降沿之后需要等待~9us, 否则会有数据丢失的概率
    IIC_SCL(1);            //SCL=1, 此时从机可以返回 ACK
    IIC_Delay();
  IIC_Delay();
    while (IIC_READ_SDA)       //等待应答
    {
        Waitime++;
        if (Waitime > 250)
        {
            IIC_Stop();
            Rack = 1;
            break;
        }
    }

    IIC_SCL(0);        // SCL=0, 结束 ACK 检查
    IIC_Delay();
    return Rack;
```

```
}

void IIC_Ack(void)
{
    IIC_SDA(0);        // SCL 0 -> 1  时  SDA = 0,表示应答
    IIC_Delay();
    IIC_SCL(1);        //  产生一个时钟
    IIC_Delay();
    IIC_SCL(0);
    IIC_Delay();
    IIC_SDA(1);        //  主机释放 SDA 线
    IIC_Delay();
}

void IIC_Nack(void)
{
    IIC_SDA(1);        // SCL 0 -> 1  时  SDA = 1,表示不应答
    IIC_Delay();
    IIC_SCL(1);        //  产生一个时钟
    IIC_Delay();
    IIC_SCL(0);
    IIC_Delay();
}

void IIC_Send_Byte(uint8_t Data)
{
    uint8_t t;
    for (t = 0; t < 8; t++)
    {
        IIC_SDA((Data & 0x80) >> 7);      //高位先发送
        IIC_SCL(1);
        IIC_SCL(0);
        Data <<= 1;        //左移 1 位,用于下一次发送
    }
    IIC_SDA(1);              //发送完成，主机释放 SDA 线
}

uint8_t IIC_Read_Byte(uint8_t Ack)
{
    uint8_t i, ReceData = 0;
    for (i = 0; i < 8; i++ )      //接收 1 个字节数据
    {
```

```
            ReceData <<= 1;   // 高位先输出,所以先收到的数据位要左移
            IIC_SCL(1);
            if (IIC_READ_SDA)
            {
                ReceData++;
            }
            IIC_SCL(0);
            delay_us(2);      // Minum  延时 2us,否则读数据异常
        }
        if (!Ack)
        {
            IIC_Nack();       // 发送 nACK
        }
        else
        {
            IIC_Ack();        // 发送 ACK
        }
        return ReceData;
}

uint8_t GZP6810_Trig_ReadData(uint16_t Cmd,uint8_t *p)
{
        uint8_t i=0,ACK=0,Ture=1,False=0;
        IIC_Start();
        IIC_Send_Byte(0x4a);
        ACK = IIC_Wait_Ack();
        IIC_Send_Byte(Cmd>>8);
        ACK = IIC_Wait_Ack();
        IIC_Send_Byte(Cmd);
        ACK = IIC_Wait_Ack();
        IIC_Stop();
        delay_ms(1);
        IIC_Start();
        IIC_Send_Byte (0x4B);
        ACK = IIC_Wait_Ack();
        for(i=0;i<9;i++)
        {
            if ( i == 8 ) p[i]=IIC_Read_Byte(False);
            else p[i]=IIC_Read_Byte(Ture);
        }
        IIC_Stop();
```

```
    return ACK;
}
uint8_t GZP6810_ReadData(uint8_t *p)
{
        uint8_t i=0,ACK=0,Ture=1,False=0;
        IIC_Start();
        IIC_Send_Byte (0x4b);
        ACK = IIC_Wait_Ack();
        for(i=0;i<9;i++)
        {
              if ( i == 8 ) p[i]=IIC_Read_Byte(False);
              else p[i]=IIC_Read_Byte(Ture);
        }
        IIC_Stop();
        return ACK;
}
void    main(void)
{
    float Pressure,Temperature;                         //差压，温度
    uint8_t p[9],IIC_Ack= 0;
       HAL_Init();                                        //初始化 HAL 库
       sys_stm32_clock_init(RCC_PLL_MUL9);              // 设置时钟, 72Mhz
       delay_init(72);                                    // 延时初始化
       usart_init(460800);                               // 串口初始化为 115200
    IIC_Init();
    delay_ms(5);
    IIC_Ack = GZP6810_Trig_ReadData(0x361E,p);
    delay_ms(40);
    while(1)
        {
              IIC_Ack = GZP6810_ReadData(p);
              Pressure=(float)(((int16_t)(p[0]*256)+p[1])/60.0);          //压力比例系数是 60
              Temperature=(float)((int16_t)((p[3]*256)+p[4])/200.0);   //温度比例系数是 200
              if ( IIC_Ack ) printf("i2c ack error\n");
              else
                   {
                        printf("Press is %f\n",Pressure);
                   printf("Temp is %f\n",Temperature);
                   }
              delay_ms(5);
        }
}
```