GZP6816D

Pressure Sensor

Digital Output（IIC）

Datasheet

Version：V1.8

Issued Date：2025.06.12

Table of Contents

Document Revision History

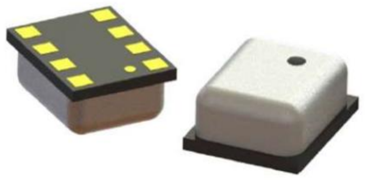
Revision	**Description**	**Date**
V1.0	Initial version	2021.04.02
V1.1	Add cover and table of contents	2021.09.23
V1.2	Increase the number of registers and provide detailed instruction descriptions	2021.11.18
V1,3	Update the program range and delay time	2021.12.24
V1.4	Adjust product classification	2022.3.16
V1.5	Product structure size annotations have been corrected, company information has been updated, and parameters have been corrected.	2022.11.28
V1.6	Format adjustment	2023.09.27
V1,7	Change the application circuit diagram	2023.09.27
V1.8	Change in precision resolution and long-term stability	2025.06.12

1 Product Description

The GZP6816D is a compact digital pressure sensor with high accuracy and low current consumption, capable of measuring both pressure and temperature. An internal signal processor converts the output of the pressure and temperature sensor elements into 24-bit and 16-bit data, respectively. Each pressure sensor is individually calibrated and includes calibration coefficients. The coefficients are used in the application to convert measurement results into true pressure and temperature values. The sensor measurements and calibration coefficients are accessible via the serial I2C interface.

The GZP6816D pressure module features high integration, a compact size, and easy installation, facilitating system integration. It is widely used in applications such as drones, weather monitoring, and pneumatic control.

1.1 Features



- Multiple range: 30kPa~110kPa
- Absolute pressure type
- Power supply voltage: 1.8V ~ 3.6V
- Current consumption: <80uA (maximum oversampling rate for one measurement)
- Standby current: 0.1uA (at 25°C)
- Absolute pressure accuracy: ±1hPa (8.3m)
- Relative pressure accuracy: ±0.12hPa (1m)
- IIC Interface
- Wide temperature compensation

1.2 Applications

- Drones, weather stations, navigation
- Sports wearable devices
- Mobile phones and other mobile devices
- Clocks, watches, home appliances
- Medical equipment such as monitors and oxygen concentrators
- Portable and fixed barometers

2 Function Description

This product is manufactured using advanced micro-electromechanical principles. Its core technologies are a MEMS pressure sensor chip based on the silicon piezoresistive effect and a high-performance signal conditioning AISC chip. The silicon micro-piezoresistive MEMS pressure sensor chip forms a Wheatstone bridge through four strain-sensitive resistors. The output signal is amplified, temperature-compensated, and linearized by the ASIC chip. The linearization and temperature compensation of the transfer function are implemented by the digital processing circuit in the ASIC. Through the polynomial compensation algorithm and multi-point pressure calibration technology at multiple temperatures, high-precision pressure measurement is achieved over the entire operating temperature range.

2.1 Block Diagram

The functional block diagram of the pressure sensor is shown in Figure 1.

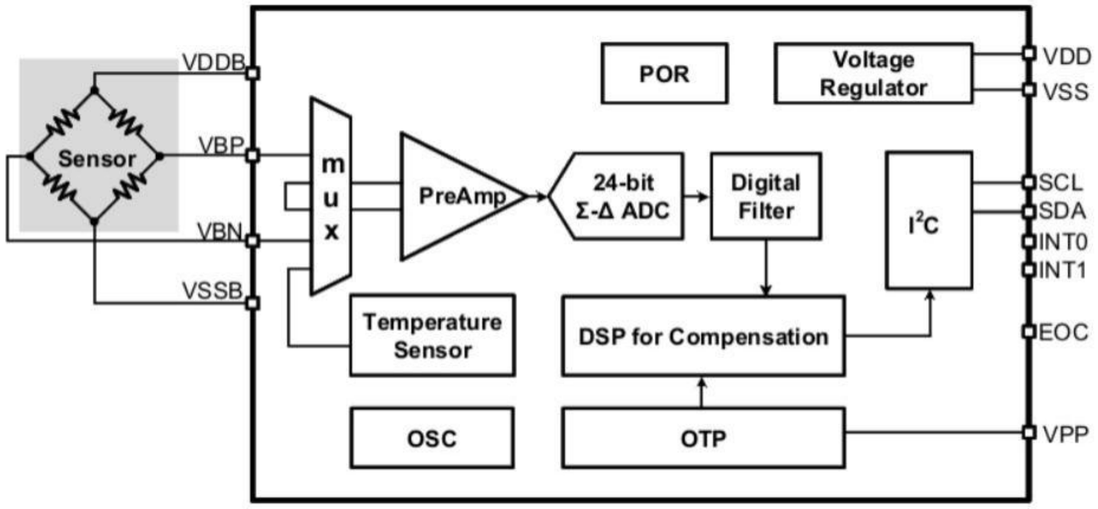


Fig.1 Block Diagram

2.2 Pin Definition

The pin configuration of the pressure sensor is shown in Figure 2.

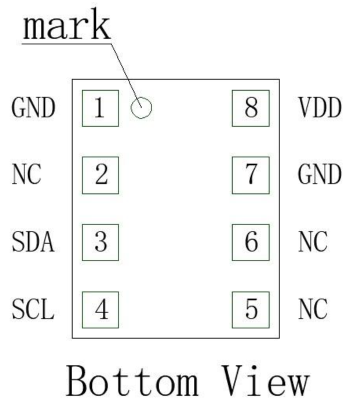


Fig.2 Pin configuration diagram

The corresponding relationship of the pressure sensor pins is shown in Table 1.

Tab.1 Pin Definition

PIN No.	Description	Remark
1	GND	Power input negative
2	NC	Floating pin
3	SDA	I2C bidirectional data line
4	SCL	I2C clock line
5	NC	Floating pin
6	NC	Floating pin
7	GND	Power input negative
8	VDD	Power input positive

3 Technical Indicators

The following indicators of the sensor are measured with (3.3)V DC and 25°C.

3.1 Maximum Ratings

The maximum rated parameters of the sensor are shown in Table 2.

Tab.2 The maximum rated parameters

Parameter	**Min.**	**Typical Value**	**Max.**	**Unit**	**Remark**
Supply Voltage	-0.3		3.6	V	
ESD Protection		2		kV	HBM
Working Temp.	-30		105	°C	
Storage Temp.	-40		125	°C	

3.2 Performance Indicators

The sensor performance indicators are shown in Table 3

Tab.3 Performance indicator

Parameter	**Conditions**	**Min.**	**Typ.**	**Max.**	**Units**
Temp.Measure Range	Interior temp.sensor	-40		150	°C
Absolute Pressure Accuracy		-1		1	hPa
Relative Pressure Accuracy		-0.12		0.12	hPa
Temp. Measure Accuracy		-1		1	°C
Over Pressure			2x		Rated
Burst Pressure			3x		Rated
Compensated Temp.		-10		60	°C
Long Term Stability			±1		Pa/year
Resolution			0.1		Pa

3.3 Electrical Characteristics

The sensor electrical characteristicsare shown in Table 4

Tab.4 Electrical Characteristics

Parameter	Conditions		Min	Typ	Max	Units
	At VDD=2V		32			dB
Average current during 1Hz conversion rate measurement	OSR_P	Oversampling rate 128x		80		μA
		Oversampling rate 64x		42		
		Oversampling rate 32x		23		
		Oversampling rate 16x		13		
		Oversampling rate 8x		8		
		Oversampling rate 3x		6		
		Oversampling rate 2x		4		
Peak Current				0.3		mA
Standby Current	Standby current in sleep state at 25°C			50	250	nA
Single measurement time (Including external bridge and temperature measurement time, the OSR of temperature measurement is 1024x)	OSR_P	Oversampling rate 128x		203		ms
		Oversampling rate 64x		105		
		Oversampling rate 32x		56		
		Oversampling rate 16x		31		
		Oversampling rate 8x		19		
		Oversampling rate 4x		13		
		Oversampling rate 2x		10		
ADC Conversion rate	OSR as 2x ~ 128x		20		1350	Hz
I2C Clock frequency					3.4	MHz
Temperature resolution				0.003		K/LSB
Start Time	VDD to the time when the interface communication starts				1	ms
	VDD to the time when the measurement starts				2.5	ms
Wake up Time	Sleep status to the time when the interface communication starts				0.5	ms
	Sleep status to the time when the measurement starts				2	2

4 Application Circuit

The recommended application circuit of the sensor is shown in Figure 3.

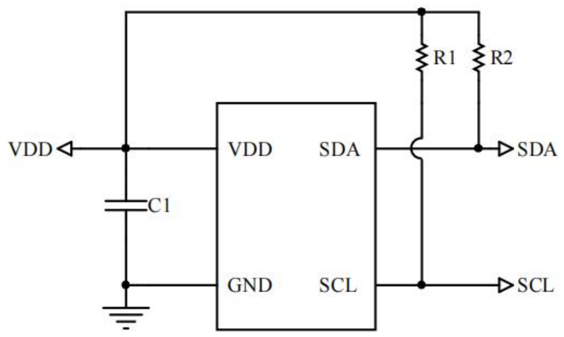


Fig.3 Recommended sensor application circuit diagram

Notice :

- The recommended value of C1 is 100nF, and the recommended values of R1 and R2 are 4.7k.
- Please confirm the electrical definition before assembly.
- Do not have any electrical connection to the NC pin, otherwise it may cause product failure.
- Provide anti-static protection during welding.
- Overload voltage (6.5Vdc) may burn out the circuit chip.
- This product has no reverse polarity protection, please pay attention to the power polarity during assembly.

5 I²C Communication Protocol

The I2C bus uses SCL and SDA as signal lines, both of which are connected to VDD through pull-up resistors (typically 4.7K) and remain high level when not communicating.

The sensor is calibrated at the factory. Sending the 0xAC command to get the calibrated data.

⬜	From master to slave	S	Start	A	Acknowledge
🟦	From slave to master	P	Stop	N	Not Acknowledge

Write Command

S	0XF0	A 🟦	0XAC	A 🟦	P

0xF0 means that the default 7bits I2C sensor slave device address is 0x78, and the last 1bit is 0 means that the master device MCU writes to the slave device. 0xAC is the command word to start the slave device sensor to perform a measurement. (The write address is 0X78<<1+0=0XF0, and the read address is 0X78<<1+1=0XF1)

Read Command

S	0XF1	A	Status	N	P

After sending the write command, need to wait for a while till measurement finish from the slave device sensor, and then send the read command to read the measurement data. Then sending the 0XF1 command to determine whether the sensor data acquisition has been completed

The waiting time depends on the settings of [13:11] Pressure Oversampling Rate of OTP (Address: 0x14) and [15:14] Temperature Oversampling Rate of OTP (Address: 0x14). The waiting time is =Tp+Tt.

Pressure Oversampling Rate and Measurement Time Comparison Table

OSR_Pressure[13:11]（Binary）	OSR	Measurement Time Tp(ms)
000	32768	203
001	16384	105
010	8192	56
011	4096	31
100	2048	19
101	1024	13
110	512	10

Temperature Oversampling Rate and Measurement Time Comparison Table

OSR Temperature[15:14]（Binary）	OSR	Measurement Time Tt(ms)
00	2048	19
01	4096	31
10	8192	56
11	16384	105

Read Pressure Value

The read calibration data consists of 6 bytes, which are 1-byte status word, 3-byte pressure calibration value, and 2-byte temperature calibration value.

S	0XF1	A	Status	A	BridgeDat [23:16]	A	BridgeDat [15:8]	A	BridgeDat [7:0]	A	TempDat [15:8]	A	TempDat [7:0]	N	P

Tab.5 Status of Bits

Bit	Significancy	Description
Bit7	Reserved	Absolute value 0
Bit6	Power indication	1: Power on; 0: Power off
Bit5	Busy indication	1: Data collection incomplete 0: Data collection complete, data ready for reading.
Bit4	Reserved	Absolute value 0
Bit3	Reserved	Absolute value 0
Bit2	Reserved	Absolute value 0
Bit1	Reserved	Absolute value 0
Bit0	Reserved	Absolute value 0

Tab.6 I2C Command

Command(byte)	Return	Description	NOR Mode	CMD Mode
0x00~0x1F	16-bit data	Read data in the OTP that address matching command	Support	Support
0x40~0x5F Followed command byte: 0x0000 ~0xFFFF	—	Write data to OTP; Address is Command value subtract 0x40 (Address is 0x00 to 0x1F)	Support	Support
0xA0~0xA7 Followed command byte: 0xXXXX	24-bit raw data Get_Raw	Get_Raw Conduct one measurement, and write the raw ADC data to registers. See table 6-3 for further interpretation	Support	Support
0xA8	24-bit raw data Get_Raw	Start_NOM Quit CMD mode, enter NOR mode	No-Support	Support
0xA9	—	Start_CM Quit NOR mode, enter CMD mode	Support	No-Support
0xAA	—	Write_ChecksumC If CRC values are not wrote to OTP,	Support	Support

		the command check data in OTP register and writes CRC values to OTP		
0xAC	24-bit compensated bridge data and 16-bit compensated temperature data	Get_Cal Measure based on OTP settings(AZBM, BM,AZTM and TM), write conpensated bridge and temperature data to I2C interface	Support	Support
0xB0~0xBF	24-bit compensated bridge data and 16-bit compensated temperature data	Get_Cal_S and Get_Cal are the same except that Get_Cal measures based on OTP defined OSR and Get_Cal_S measures based on command defined OSR, see following table	Support	Support

Tab.7 Get_Cal_S Command

Command 0xBX(HEX)	**Function**	**Detail**
X [3] Bit	OSR_T, ADC OSR of temperature measurement	0: 4x OSR 1: 8x OSR
X [2:0] Bit	OSR_P, ADC OSR of pressure measurement	000: 128x OSR 100: 8x OSR 001: 64x OSR 101:4x OSR 010: 32x OSR 110: 2x OSR 011: 16x OSR 000: 1x OSR

For example, to set the temperature ADC to 4x oversampling and the piezoelectric panel ADC to 1x oversampling, the command format is 0xB7,Just replace 0xAC with 0xB7

Tab.8 OPT Register

Addr	**Bit Range**	**Description**	**Notes/Explanations**
0x00~0x13		Calibration Coefficient	
0x14	15:14	Temperature_OSR	00:8X 01:16 X 10: 32X 11:64X
	13:11	Pressure_OSR	000: 128X 001: 64X 010: 32X 011: 16X 100: 8X 101: 4X 110: 2X 111: 1X

	10:8	ADC offset	000 : 1/16 —> [-1/16, 15/16] 001 : 2/16 —> [-2/16, 14/16] 010 : 3/16 —> [-3/16, 13/16] 011 : 4/16 —> [-4/16, 12/16] 100 : 5/16 —> [-5/16, 11/16] 101 : 6/16 —> [-6/16, 10/16] 110 : 7/16 —> [-7/16, 9/16] 111 : 8/16 —> [-8/16, 8/16]
	7:6	Reserved	
	5	Signal Polarity	1：No Inversion，0：Inversion
	4:0	Reserved	
0x15~0x16		Internal Test	
0x17	4	Interrupt Enable	0：disable, 1: enable
	3:2	Interrupt 0 Configuration Bits	00 : Invalid 01 : Calibration value exceeds preset upper limit (TH_H) 10 : Calibration value falls below preset lower limit (TH_L) 11 : Calibration value exceeds preset upper limit (TH_H) or falls below preset lower limit (TH_L)
	1:0	Interrupt 1 Configuration Bits	00 : Invalid 01 : Calibration value exceeds preset upper limit (TH_H) 10 : Calibration value falls below preset lower limit (TH_L) 11 : Calibration value exceeds preset upper limit (TH_H) or falls below preset lower limit (TH_L)

6 Value Conversion and Calculation

- **Pressure conversion example**：

After reading the calibration data, need to convert simply the unsigned number in the form of AD value.

To facilitate understanding, assuming that the calibration data read is: 0x00, 0x9B, 0xB0, 0xC5, 0x56, 0xAA

0x00 is the status, and Bit5 is 1 indicating that the I2C is busy last time, and it needs to wait for a period of time. If Bit5 is 0, it indicates that the device is not busy and data can be read. For a detailed description of each bit of the status word, check Table 5

The three bytes of 0x9B, 0xB0, and 0xC5 are pressure calibration value.

The pressure calibration value is converted as follows:

Assuming that the range used in this calculation and calibration is 30Kpa-110Kpa, the corresponding AD output is 1677722~15099494 (10%AD~90%AD).

Then convert 0x9B, 0xB0, and 0xC5 into decimal numbers, assuming it is 10203333.
According to the formula: pressure_kpa = ((PMAX-PMIN)/(DMAX-DMIN)*(Dtest-DMIN)+PMIN),
actual pressure value = (110-30)/(15099494-1677722)*(10203333-1677722)+30= 80.816 Kpa

The temperature calibration value is converted as follows:

Converting 0x56, 0xAA to a decimal number is 22186. Since the read calibration data is expressed as a percentage, this percentage is numerically equal to the ratio of the decimal number we converted to the maximum value of 16bits unsigned number (65535, full AD value of whole temperature range from -40°C to 150°C). Therefore, the following calculations can be made when converting percentages: 22186/65536*100%=33.85%
The temperature calibration range is specified as -40℃ to 150°C, so the calibration value=
(150-(-40))*33.85%-40=24.32°C

7 Structure Specification (unit:mm)

Refer to Figure 8 for the sensor's dimensions (error is ±0.1mm if not specified).

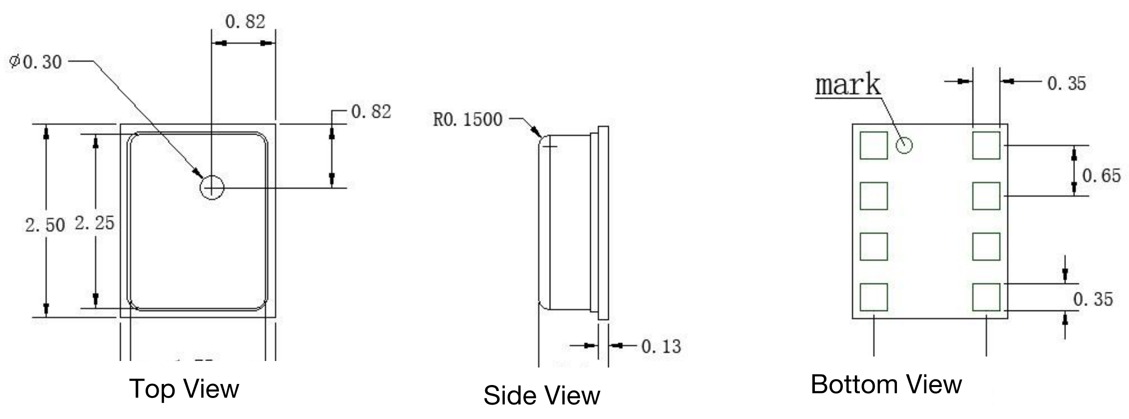


Fig.8 Product dimensions

8 Order Guide

GZP 6816 D B 01 WX

Tab.8 Order rules

GZP	Pressure Sensor Series
6816	Product Series
D	Output type A: Analog output D: IIC output
B 01	Packaging Method: B01: Reel&Tape
WX	Company interior code

9 Ordering Instructions

If you have special requirements for product performance parameters and functions, please contact us.

10 Instruction for Use

10.1 Soldering

Since this product has a small structure with low heat capacity, please minimize the influence of heat from the outside. Otherwise, it may be damaged due to thermal deformation and cause changes in characteristics. Please use non-corrosive rosin type flux. In addition, since the product is exposed to the outside, please be careful not to allow flux to penetrate into the inside.

(1) Manual soldering

- Use a soldering iron with a head temperature between 260 and 300°C (30 W) and perform the work within 5 seconds.
- Please note that the output may change when soldering with a load applied to the terminals.
- Please keep the soldering iron tip clean.

(2) Reflow soldering (SMD terminal type)

- The recommended reflow oven temperature setting conditions are shown:

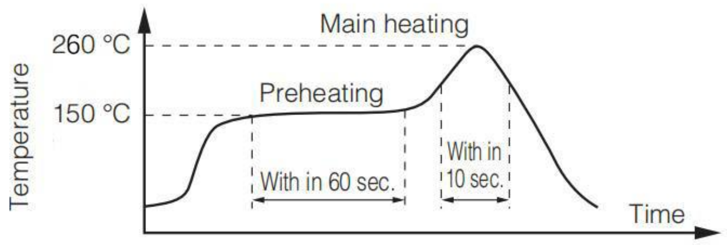


Fig.9 Remelting temperature setting conditions

(3) The warping of the printed circuit board relative to the entire sensor should be kept below 0.05mm. Please manage this.

(4) After installing the sensor, be careful not to generate stress on the solder joint when cutting and bending the substrate.

(5) Since the sensor terminals are exposed, contact with metal pieces or other objects may cause abnormal output. Be careful not to touch the terminals with metal pieces or your hands.

(6) When applying coating to prevent insulation degradation of the substrate after soldering, be careful not to allow chemicals to adhere to the sensor.

10.2 Cleaning Requirements

(1) Since the product is open type, please be careful not to allow cleaning fluid to enter the interior.

(2) Please avoid using ultrasonic cleaning as it may cause product failure.

10.3 Storage and Transportation

(1) This product is not drip-proof, so do not use it in places where it may be splashed with water.

(2) Do not use in an environment where condensation occurs. In addition, if moisture attached to the sensor chip freezes, it may cause fluctuations in sensor output or damage.

(3) Due to the structure of the pressure sensor chip, the output will fluctuate when it is exposed to light. Especially when applying pressure through a transparent cover, etc., please avoid light from reaching the sensor chip.

(4) Normally packaged pressure sensors can be transported by ordinary transportation vehicles. Please note: The product must be protected from moisture, shock, sunburn and pressure during transportation.

10.4 Other Precautions

(1) If the installation method is incorrect, it may cause an accident, so please be careful.

(2) Avoid using the product in a manner that applies high-frequency vibrations, such as ultrasonic waves.

(3) The only pressure medium that can be used directly is dry, non-corrosive gas. Other media, especially corrosive media or media containing moisture or foreign matter, may cause malfunction and damage. Therefore, please avoid using it in the above environment.

(4) A pressure sensor chip is located inside the pressure inlet. Inserting a needle or other foreign object into the pressure inlet can damage the chip and clog the inlet, so please avoid such an operation.

(5) Regarding the operating pressure, please use it within the rated pressure range. Using it outside the range may cause damage.

(6) Since static electricity may cause damage, please be careful to ground charged objects on the table and operators when using it to allow the surrounding static electricity to discharge safely.

If you have any questions, please feel free to ask.

11 Packaging Information

Carrier tape information as shown in Figure 10 (unit: mm) Quantity per tray 10000 PCS.

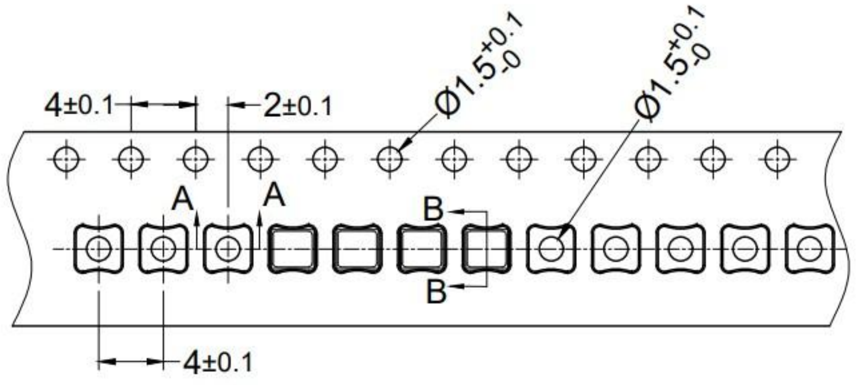


Fig.10 Tape

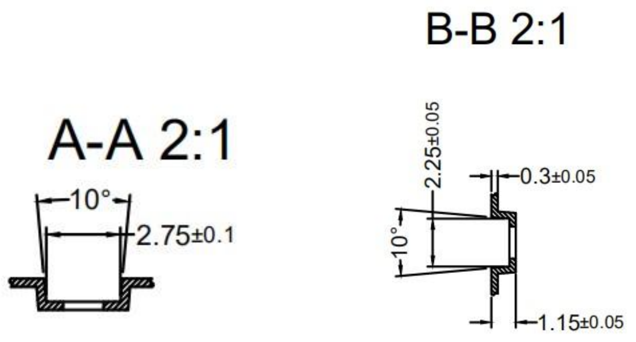


Fig.11 Detail of the tape

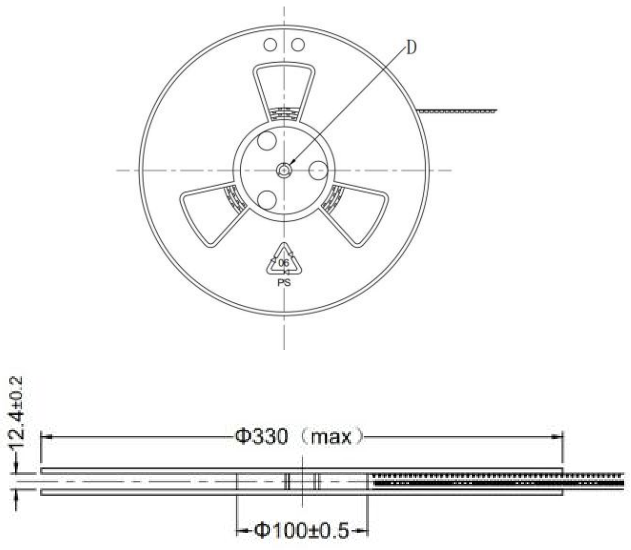


Fig.12 Tape Carrier

Safety Precautions

This product is made of semiconductor components for general electronic equipment (communication equipment, measuring equipment, working machinery, etc.). Products using these semiconductor components may malfunction and fail due to external interference and surges, so please confirm the performance and quality under actual use. To be on the safe side, please perform safety design on the device (setting of protection circuits such as fuses and circuit breakers, multiple devices, etc.) so that life, body, property, etc. will not be harmed in the event of a malfunction. To prevent injuries and accidents, please be sure to comply with the following matters:

·The driving current and voltage should be used below the rated values.

Please wire according to the electrical definition . In particular, reverse connection of the power supply may cause accidents due to circuit damage such as heat, smoke, and fire, so please be careful.

·Be careful when fixing the product and connecting the pressure inlet .

Warranty and Disclaimer

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. Sencoch Technology reserves the right to make changes without further notice to any product herein. Sencoch Technology makes no warranty, representation or guarantee regarding the suitability of its product for any particular purpose, nor does Sencoch Technology 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. Sencoch Technology does not convey any license under its patent rights nor the rights of others.

IIC Example Code (Attachment: IIC Code Example)

```
/*****************************************************************************************************
//*********************************************Digital tube displays pressure and temperature
//**********************************************STC12+MAX7219*******************************
//*****************************CLK=P2^2 CS=P2^1
DIN=P2^0*****************************************************************************************
//*********************************************SCL=P1^7
SDA=P1^6****************************************************************************************
//****************************************************************************************************
#include <STC12C5A60S2.H>
#include <stdio.h>
#include <math.h>
#include "MAX7219.h"
#include "GZP6816D.h"
#include "IIC.h"

extern float pressure_kpa ;//, temp = 0.0;//float 4byte
extern unsigned long pressure_pa ;
extern unsigned long temp ;
void Delay300ms()       //@11.0592MHz
{
        unsigned char i, j, k;

        i = 13;
        j = 156;
        k = 83;
        do
        {
                do
                {
                        while (--k);
                } while (--j);
        } while (--i);
}

void main()
```

```
{
        unsigned char dis[8] = {0,0,0,0,0,0,0,0};
        Delay_Ms(DELAY_TIME);
        Init_MAX7219();
    while(1)
    {
    GZP6816D_get_cal();
        dis[0] = (unsigned char)(pressure_pa / 10000000);
            dis[1] = (unsigned char)(pressure_pa % 10000000 / 1000000);
                dis[2] = (unsigned char)(pressure_pa % 1000000 / 100000);
                    dis[3] = (unsigned char)(pressure_pa % 100000 / 10000);
                        dis[4] = (unsigned char)(pressure_pa % 10000 / 1000);
                            dis[5] = (unsigned char)(pressure_pa % 1000 / 100);
                                dis[6] = (unsigned char)(pressure_pa % 100 / 10);
                                    dis[7] = (unsigned char)(pressure_pa % 10);

            Write_Max7219(8, dis[0]);
                Write_Max7219(7, dis[1]);
                    Write_Max7219(6, dis[2]);
                        Write_Max7219(5, dis[3]);
                            Write_Max7219(4, dis[4]|0x80);   //Display decimal point
                                Write_Max7219(3, dis[5]);
                                    Write_Max7219(2, dis[6]);
                                        Write_Max7219(1, dis[7]);
      Delay300ms();

            GZP6816D_get_cal();
            temp=temp*10;
            dis[0] = (unsigned char)(temp / 10000000);
            dis[1] = (unsigned char)(temp % 10000000 / 1000000);
                dis[2] = (unsigned char)(temp % 1000000 / 100000);
                    dis[3] = (unsigned char)(temp % 100000 / 10000);
                        dis[4] = (unsigned char)(temp % 10000 / 1000);
                            dis[5] = (unsigned char)(temp % 1000 / 100);
                                dis[6] = (unsigned char)(temp % 100 / 10);
                                    dis[7] = (unsigned char)(temp % 10);
                    Write_Max7219(8, dis[0]);
                          Write_Max7219(7, dis[1]);
                                Write_Max7219(6, dis[2]);
                                      Write_Max7219(5, dis[3]);
                                            Write_Max7219(4, dis[4]);        //Display decimal point
                                                  Write_Max7219(3, dis[5]);
                                                        Write_Max7219(2, dis[6]|0x80);
                                                              Write_Max7219(1, dis[7]);
                    Delay300ms();
      }
}

#include "GZP6816D.h"
#include <math.h>
// Define the upper and lower limits of the calibration pressure
#define PMIN 30    //Zero range pressure for example 30Kpa
#define PMAX 110    //Full Point Pressure Value, for example 110Kpa
#define DMIN 1677721.6    //AD value corresponding to pressure zero, for example
10%AD=2^24*0.1
#define DMAX 15,099,494.4 //AD Value Corresponding to Full Pressure Range, for example
90%AD=2^24*0.9

float pressure_kpa = 0.0;//, temp = 0.0;
unsigned long pressure_pa = 0;
unsigned long temp = 0.0;

//The 7-bit IIC address of the JHM1200 is 0x78
unsigned char Device_Address = 0x78 << 1;

//Read the status of IIC and judge whether IIC is busy
unsigned char GZP6816D_IsBusy(void)
{
      unsigned char status;
      GZP6816D_IIC_Read(Device_Address, &status, 1);
      status = (status >> 5) & 0x01;
```

```
        return status;
}

void GZP6816D_get_cal(void)
{
        unsigned char buffer[6] = {0};
        unsigned long Dtest = 0;
        unsigned int temp_raw = 0;
        //Send 0xAC command and read the returned six-byte data
        buffer[0] = 0xAC;
        GZP6816D_IIC_Write(Device_Address, buffer, 1);
        Delay_Ms(DELAY_TIME);
        while (1)
        {
                if (GZP6816D_IsBusy())
                {
                        Delay_Ms(DELAY_TIME);
                }
                else
                        break;
        }
        GZP6816D_IIC_Read(Device_Address, buffer, 6);

        //The returned pressure and temperature values are converted into actual values
according to the calibration range
        Dtest = (unsigned long)((((unsigned long)buffer[1]) << 16) | (((unsigned int)buffer[2]) << 8) |
((unsigned char)buffer[3]));
        temp_raw = ((unsigned int)buffer[4] << 8) | (buffer[5] << 0);
        if (Dtest != 0)
        {
                pressure_kpa = (float) ((PMAX-PMIN)/(DMAX-DMIN)*(Dtest-DMIN)+PMIN);    //单位：
KPa
                pressure_pa = (unsigned long) (pressure_kpa * 1000.0);     //unit：Pa
        }
        else
```

```
    {
        pressure_kpa = 0.0;     //unit：KPa
        pressure_pa = 0;    //unit：Pa
    }
    temp = (double)temp_raw /65536   * 190 - 40;
}

//Write a byte of data through IIC
unsigned char GZP6816D_IIC_Write(unsigned char address, unsigned char *buf, unsigned
char count)
{
    unsigned char timeout, ack;
    address &= 0xFE;
    Start();
    Delay_Ms(DELAY_TIME);
    SendByte(address);
    /*    Set_SDA_INPUT(); */
    Delay_Ms(DELAY_TIME);
    timeout = 0;
    do
    {
        ack = Check_ACK();
        timeout++;
        if (timeout == 10)////////////////
        {
            Stop();
            return 1;
        }
    } while (ack);
    while (count)
    {
        SendByte(*buf);
        /*    Set_SDA_INPUT(); */
        Delay_Ms(DELAY_TIME);
        timeout = 0;
        do
        {
            ack = Check_ACK();
            timeout++;
            if (timeout == 10)
            {
                return 2;
            }
        } while (0);
        buf++;
        count--;
    }
    Stop();
    return 0;
}

//Read a byte of data through IIC
unsigned char GZP6816D_IIC_Read(unsigned char address, unsigned char *buf, unsigned
char count)
{
    unsigned char timeout, ack;
    address |= 0x01;
    Start();
    SendByte(address);
    /*    Set_SDA_INPUT(); */
    Delay_Ms(DELAY_TIME);
    timeout = 0;
    do
    {
        ack = Check_ACK();
        timeout++;
        if (timeout == 10)
        {
            Stop();
            return 1;
        }
```

```
	} while (ack);
	Delay_Ms(DELAY_TIME);
	while (count)
	{
		*buf = ReceiveByte();
		if (count != 1)
			Send_ACK();
		buf++;
		count--;
	}
	Stop();
	return 0;
}

#include "IIC.h"
//*****************************************************
//MS Delay Function (Tested with 12M Crystal Oscillator)
//*****************************************************
void Delay_Ms(unsigned char n)
{
	unsigned char i,j;		//Change "char" to "int"
		for(i=0;i<n;i++)
			for(j=0;j<123;j++);
}

//Start signal
void Start(void)
{
	/*	Set_SDA_OUTPUT();	*/
	SDA = 1;
		Delay_Ms(DELAY_TIME);
			SCL = 1;
				Delay_Ms(DELAY_TIME);
					SDA = 0;
						Delay_Ms(DELAY_TIME);
							SCL = 0;
```

```
                                                Delay_Ms(DELAY_TIME);//**
}

//Stop signal
void Stop(void)
{
    /*    Set_SDA_OUTPUT();    */
    SDA = 0;
        Delay_Ms(DELAY_TIME);
            SCL = 1;
                Delay_Ms(DELAY_TIME);
                    SDA = 1;
                        Delay_Ms(DELAY_TIME);
                            SCL = 0;                    //**
                                Delay_Ms(DELAY_TIME); //**
}

//Read ACK signal
unsigned char Check_ACK(void)
{
    unsigned char ack;
    /*    Set_SDA_INPUT(); */
        SDA = 1;                        //**
            Delay_Ms(DELAY_TIME); //**
                SCL = 1;
                    Delay_Ms(DELAY_TIME / 2);
                        ack = SDA;
                            Delay_Ms(DELAY_TIME / 2);
                                SCL = 0;
                                    Delay_Ms(DELAY_TIME);//**
    /*    Set_SDA_OUTPUT();    */
                                        return ack;
    //Delay_Ms(DELAY_TIME);//**
}

//Send ACK signal
```

```
void Send_ACK(void)
{
    /*    Set_SDA_OUTPUT();    */
          SDA = 0;
                Delay_Ms(DELAY_TIME);
                      SCL = 1;
                            Delay_Ms(DELAY_TIME);
                                  SCL = 0;
                                        Delay_Ms(DELAY_TIME);
                                              SDA = 1;
                                                    Delay_Ms(DELAY_TIME);
}

//Send one byte
void SendByte(unsigned char byte1)
{
    unsigned char i = 0;
    /*    Set_SDA_OUTPUT();    */
          do
                {
                            if (byte1 & 0x80)
                      {
                                  SDA = 1;
                                  }
                                        else
                                              {
                                                    SDA = 0;
                                                          }
                                  Delay_Ms(DELAY_TIME);
                                        SCL = 1;
                                              Delay_Ms(DELAY_TIME);
                                                    byte1 <<= 1;
                                                          i++;
                                                                SCL = 0;
          //Delay_Ms(DELAY_TIME);//**
                } while (i < 8);
```

```
                        SCL = 0;
                            Delay_Ms(DELAY_TIME);
}

//Receive one byte
unsigned char ReceiveByte(void)
{
    unsigned char i = 0, tmp = 0;
    /*    Set_SDA_INPUT(); */
            do
                {
                    tmp <<= 1;
                        SCL = 1;
                            Delay_Ms(DELAY_TIME);
                                if (SDA)
                                    {
                                        tmp |= 1;
                                        }
                                            SCL = 0;
                                                Delay_Ms(DELAY_TIME);
                                                    i++;
                } while (i < 8);
                        return tmp;
}



/******Digital tube driver program************/
/******Pin configuration
            CLK=P2^2
             CS=P2^1
             DIN=P2^0
            SCL=P1^7
             SDA=P1^6
             ************************/
#include "MAX7219.h"
```

```
#include <STC12C5A60S2.H>

#define uchar unsigned char

sbit Max7219_CLK=P2^2;
sbit Max7219_CS=P2^1;
sbit Max7219_DIN=P2^0;

//-------------Write One Byte to the Max7219-------------
void Write_Max7219_byte(uchar Data)
{
    unsigned char i;
    Max7219_CS = 0;        //CS low effect
    for (i = 8; i >= 1; i--)
    {
        Max7219_CLK = 0;
        Max7219_DIN = Data & 0x80;
        Data = Data << 1;
        Max7219_CLK = 1;                //when pinCLK is high send the Data
    }
}

//-------------decide which address shows the Data-------------
void Write_Max7219(uchar address,uchar dat)
{
    Max7219_CS = 0;
    Write_Max7219_byte(address);
    Write_Max7219_byte(dat);
    Max7219_CS = 1;
}

//-------------MAX_7219 Initialization-------------
void Init_MAX7219(void)
{
    Write_Max7219(0x09, 0xff);    //Decoding method: BCD code
    Write_Max7219(0x0a, 0x01);    //luminance
```

```
Write_Max7219(0x0b, 0x07);     //Scanning range: 8 digital tubes display
Write_Max7219(0x0c, 0x01);     //Power-off mode: 0, Normal mode: 1
Write_Max7219(0x0f, 0x00);     //Display test: 1; Test completed, normal display: 0
```