GZP6829DC

Pressure Sensor

Digital Output(IIC)

Datasheet

Version：V1.0

Issued Data：2025.7.4

Table of Contents

1. Product Description 4
1.1 Features 4
1.2. Application Areas 4
2. Functional Description 5
2.1 Pin Definition 5
2.2 Block Diagram 6
2.3 Accuracy 6
3. Technical Indicators 7
3.1 Maximum Rated Parameters 7
3.2 Performance Indicators 8
4. Application Circuit 8
5. I^2C Communication Protocol 9
6. Register Description 9
6. Working Mode Description 11
8. Structure Specification (unit: mm) 13
9. Order Guide 14
10. Models Example: 14
11. Instructions for Use 15
11.1 Soldering 15
11.2 Cleaning Requirements 16
11.3 Storage and Transportation 16
11.4 Other Precautions 16
12.Packing Information 17
Safety Precautions 18
IIC Example Code 20

Document Revision History

Revision	Description	Date
V1.0	Initial release	2025.07.04

The company reserves the right to make changes in the specifications contained herein without further notice.

The copyright of the product specification and the final right of interpretation of the product belong to Sencoch.

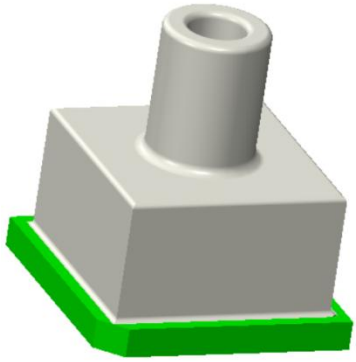
1. Product Description

The GZP6829DC type pressure sensor is packaged in a PCB board format, containing a sealed pressure sensor and a signal conditioning chip. It digitally compensates for the offset, sensitivity, temperature drift and non-linearity of the sensor. It uses a 21-bit ADC and the conditioning chip is equipped with a built-in temperature sensor to output high-precision pressure and temperature values. It also provides an IIC communication protocol interface, with strong anti-interference capability. All measurement data has been fully calibrated and temperature compensated.

The GZP6829DC type pressure sensor has fast response, high accuracy, good linearity and long-term stability.

The GZP6829DC type pressure sensor is small in size, easy to install, and convenient for system integration. It is widely used in medical electronics, wearable devices, sports and fitness equipment and other fields.

1.1 Features



- Measuring range: -100...0 ~ 10...200kPa
- Gauge pressure type
- Suitable for non-corrosive gases
- 3~5.5V power supply
- IIC output

1.2. Application Areas

- Electronic blood pressure monitors, ventilators, oxygen generators, monitoring devices, vacuum cupping therapy devices, etc. in the field of medical care
- Wearable devices such as blood pressure watches and wristbands
- Household appliances such as washing machines and coffee machine etc.
- Vacuum pumps, air pumps, vacuum preservation, water purifiers, pressure gauges, pneumatic components, etc. fields

2. Functional Description

This product is made with advanced micro-electromechanical principles. The crucial technology is the MEMS pressure sensor chip based on the silicon piezoresistive effect and the high-performance signal conditioning AISC chip. The silicon micro-piezoresistive MEMS pressure sensor chip is connected to a Wheatstone bridge composed of four strain-sensitive resistors. The output signal is amplified, temperature compensated and linearized by the ASIC chip. The linearity and temperature compensation of the transfer function are realized by the digital processing circuit in the ASIC. The high-precision pressure measurement in the full operating temperature range is achieved through the polynomial compensation algorithm and multi-point pressure calibration technology at multiple temperatures.

2.1 Pin Definition

The pin configuration of the pressure sensor is shown in Figure 1.

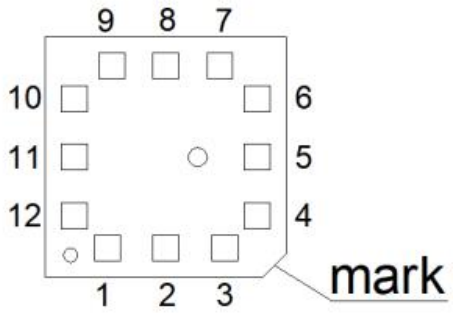


Fig.1 Pin Diagram

Tab.1 Pin Definition

Pin No.	Description	Remark
1/4/5/6/7/8/9/11	NC	Floating pin
2	SDA	Data output pin
3	SCL	Signal output pin
10	GND	Power negative
12	VDD	Power input positive

2.2 Block Diagram

The sensor functional block diagram is shown in Figure 2.

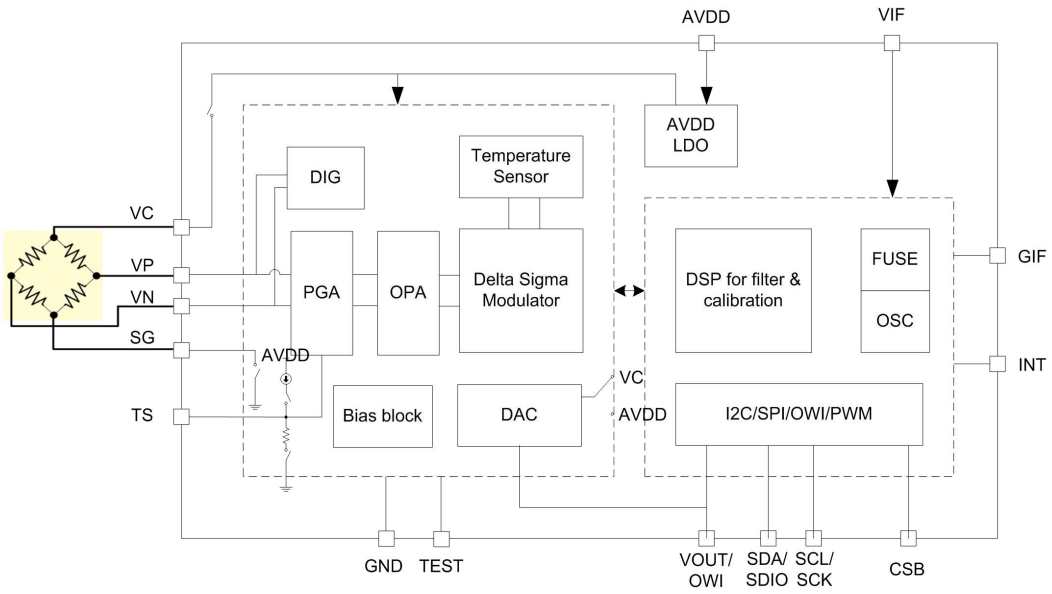


Fig.2 Block Diagram

2.3 Accuracy

The accuracy of the GZP6829DC pressure sensor is composed of its linearity, repeatability, and hysteresis errors. The value calculated by the transfer function is the specified value of the sensor and also the theoretical value. The error of the sensor is equal to the difference between the actual output value of the sensor under the specified input pressure and the specified output value.

Overall Accuracy

The overall error includes more accuracy sources based on the product accuracy :

Pressure drift: The output deviation between the actual output voltage at zero point and full scale and the specified output voltage within the specified pressure range.

Temperature effect: The output deviation of zero point and full scale at different temperatures within the temperature range.

The overall accuracy is expressed by error band, and the data are shown in Figure 3 and Table 2 shown.

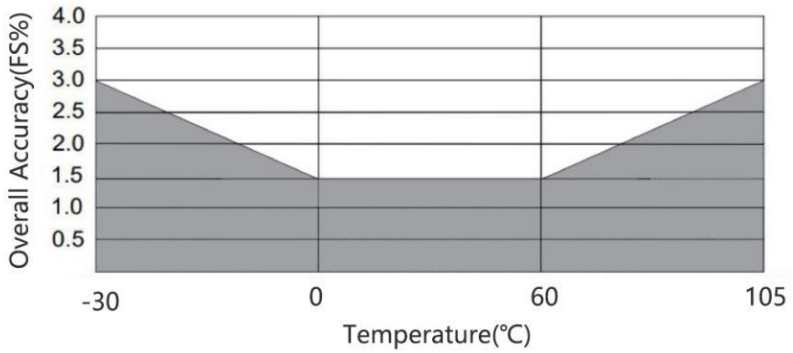


Fig.3 Relationship between overall accuracy and temperature

Tab.2 Overall Accuracy

Temperature (℃)	Overall Accuracy(Full Span)
-30~105	±2.0%
0~60	±1.0%

*Different pressure ranges have different overall errors. Please consult Sencoch for more details.

3. Technical Indicators

Measured at a power supply of (5±0.25) V DC and a temperature of 25℃.

3.1 Maximum Rated Parameters

The maximum rated parameters of the sensor are shown in Table 3.

Tab.3 The maximum rated parameters

Parameter	Min.	Typical Value	Max.	Unit	Remark
Supply Voltage	-0.3		6.5	V	VDD/VDDIO
ESD Protection		±2		kV	HBM
Overload Pressure		5X (Range≤10kPa)		Rated	
		3X (10kPa<Range≤100kPa)			
Bursting Pressure		6X (Range≤60kPa)		Rated	
		4X (10kPa<Range≤100kPa)			
Working Temperature	-30		105		
Storage Temperature	-40		125		

* Different pressure range may have different overload pressure and burst pressure, please consult Sencoch for more details.

3.2 Performance Indicators

The sensor performance indicators are shown in Table 4.

Tab.4 Sensor performance indicators

Parameter	Value	Unit	Remark
Pressure Range	-100...0～10...200kPa	kPa	Customizable
Power Supply	3.0～5.5	V	
Standby Current	100	nA	
Accuracy	±1	%Span	
Pressure Resolution	24	Bits	
Built-in Temperature Accuracy	±1	°C	@0~70°C
Temperature Resolution	16	Bits	LSB = (1/256) °C
Compensation Temperature	0～60	°C	
Pull-up Resistors	4.7	K ohm	
Clock Frequency	400	KHz	Max.
Response Time	2.5	ms	OSR_P=512X

* The different range may have different accuracy, please consult Sencoch for more details.

4. Application Circuit

The recommended application circuit is shown in Figure 4.

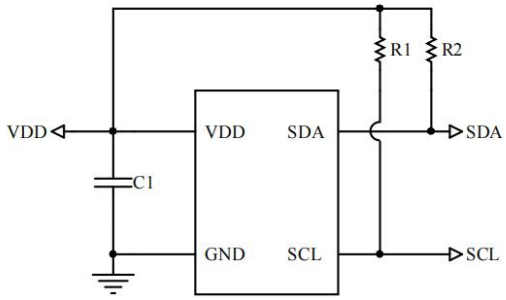


Fig.4 Application circuit

Notice :

1. Please confirm the electrical definition before soldering and assembly
2. Do not have any electrical connection with NC pin, otherwise it may cause sensor failure
3. Provide anti-static protection during soldering
4. Overload voltage (6.5Vdc) may burn out the circuit chip
5. Please add a 0.1uf capacitor between VDD and GND
6. Please pay attention to the power polarity during assembly

5. I²C Communication Protocol

The I²C bus uses SCL and SDA as signal lines. Both lines are connected to VDD through pull-up resistors (typical value 4.7K) and remain high when not communicating. The I²C device address is 0x58.

The I²C communication protocol has specific start (S) and stop (P) conditions. While SCL is high, a falling edge on SDA signals the start of data transmission. The I²C master device sequentially transmits the slave device's address (7 bits) and the read/write control bit. When the slave device recognizes this address, it generates an acknowledge signal and pulls SDA low in the ninth cycle. After receiving the slave device's acknowledgement, the master device continues to transmit the 8-bit register address and, upon receiving the acknowledgement, continues to send or read data. A rising edge on SDA while SCL is high signals the end of I²C communication . In addition to the start and stop signals, data transmitted by SDA must remain stable while SCL is high. The value transmitted by SDA can change while SCL is low. All data transmission in I²C communication is in 8-bit units, and an acknowledge signal is required after every 8 bits of data transmission to ensure continued transmission.

The I²C timing diagrams are shown in Figures 5 and 6.

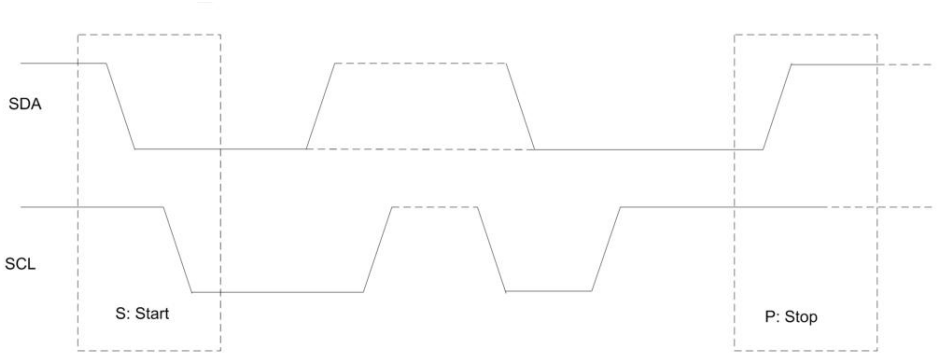


Fig.5. I²C Timing Diagram 1

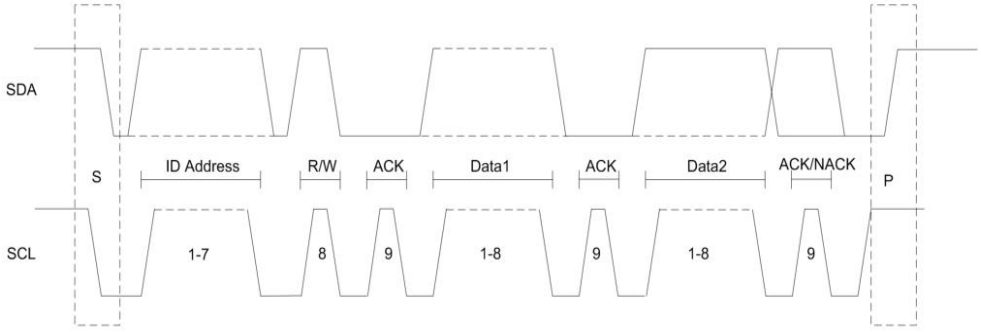


Fig.6 I²C Timing Diagram 2

6. Register Description

Tab.5 Register Description

Add.	Description	R/W	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0x00	ID	R	ID<7:0>							
0x01	Chip_Control	R/W	/		data_ Ready	/	data_out	measurement_ctrl	Active<1:0>	
0x02	CFG_OSR	R/W	OSR_T<7:5>			OSR_P<4:2>			MODE[1:0]	
0x03	CFG_MEAS	R/W	/		T_SB[5:3]			PT_R[2:0]		
0x04	P_data	R	Data out<23:16>							
0x05	P_data	R	Data out<15:8>							
0x06	P_data	R	Data out<7:0>							
0x07	T_data	R	Temp out<15:8>							
0x08	T_data	R	Temp out<7:0>							
0x24	CFG_OPER	R/W	reserved<7:1>						DAC_EN	

Reg0x00 I^2C device address, the default address is 0x58.

Reg0x01 (factory pre-configured)

Chip Control Register

active<1:0>: 00, the chip is powered off; 01, the chip is powered on;

measurement_ctrl: 0, pressure measurement; 1, temperature measurement;

data_out: 0, output calibration data; 1, output original data;

data_ready: 0, data conversion is not completed; 1, data conversion is completed.

Reg0x02 (factory pre-configured)

MODE[1:0]: 00: Sleep mode, 01: Normal mode, 10: One shot mode

OSR_P[4:2]: (pressure oversampling):

000: over sampling x 256

001: over sampling x 512

010: over sampling x 1024

011: over sampling x 2048

100: over sampling x 4096

101: over sampling x 8192

110: over sampling x 16384

111: over sampling x 32768

OSR_T[7:5] (temperature oversampling):

000: over sampling x 256

001: over sampling x 512

010: over sampling x 1024

011: over sampling x 2048

100: over sampling x 4096

101: over sampling x 8192

110: over sampling x 16384

111: over sampling x 32768

Reg0x03 (factory pre-configured)

PT_R[2:0]: 000: 64/1, 001: 32/1, 010: 16/1, 011: 8/1, 100: 4/1, 101: 1/1, Others: 128/1 (pressure/temperature measurement ratio in normal mode)

T_SB[5:3]: 000: 0ms, 001: 62.5ms, 010: 125ms, 011: 250ms, 100: 500ms, 101: 750ms, 110: 1000ms, 111: 2000ms (standby time setting in normal mode)

Reg0x04-Reg0x06

Pressure Data Register

Reg0x07-Reg0x08

Temperature Data Register

Reg0x24

DAC_EN: 0: disable DAC, 1: enable DAC

7. Working Mode Description

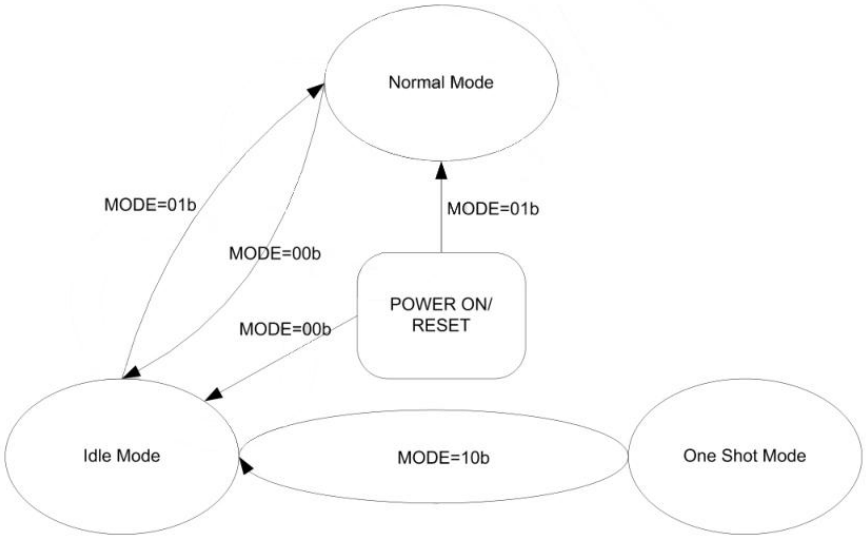


Fig.7 Working Mode

Normal Mode:

When powered on, the sensor automatically enters Normal Mode. If switching from another mode to Normal Mode, it can be enabled by writing 01b to the MODE register (0x02[1:0]). The pressure and temperature sensor signals output measurement data cyclically at a predetermined frequency(Normally the standby time was pre-configured with 0mS).

One Shot Mode:

It can be enabled by writing 10b to the MODE register (0x02[1:0]). The user can specify whether to measure the temperature or pressure signal by clearing or setting the measurement_ctrl bit (0x01[2]). After completing a single measurement, the sensor enters Idle Mode to await the next command.

Idle Mode: The sensor keep the low-consumption sleep situation till it is activated.

In normal mode, data is read in the following order:

1) After Power-up, read 5 bytes continuously from 0x04 to 0x08 (ASIC will automatically refresh the data)
2) The first 3 bytes are the pressure data, later 2 bytes are temperature data.

Pressure Calculation

Sum = (0x04 value * 2^16 + 0x05 value * 2^8 + 0x06 value),

If sum < 8388608, P=sum / 2^21*(PMAX - PMIN) (Unit: Pa)

If sum ≥8388608，P=(sum-2^24)/2^21*(PMAX - PMIN) (Unit: Pa)

※PMAX is the upper value of pressure range, PMIN is the lower value of pressure range.

Take -100 to 100kPa range as example, PMAX is 100000, PMIN is -100000

Temperature Calculation

Final_T = (Inter_T - Byte1) / 2 ^ Shift_N + 25

The specific calculation steps are as follows:

1) Inter_T = RAW_T - 65536 (RAW_T > 32768); otherwise, Inter_T = RAW_T, where

   RAW_T = 0x07 value * 2^8 + 0x08 value

2) Byte1 is calculated from the 0x20 register value, where

   Bits [6:0] are used as the exponent of 2,

   Bit [7] is the sign bit, when bit [7] = 1, Byte1 is a negative value, when bit [7] = 0, Byte1 is a positive value.

Assumed the value of 0x20 register is 0x8D (i.e. 10001101), according to the above description, the exponent is 13,the sign bit is negative, so the value of Byte1 is -(2^13), i.e. -8192.

Shift_N = Byte2/10, where Byte2 is the 0x21 register value. Assumed the value of 0x21 register is 0x46, then Shift_N=70/10=7.

8. Structure Specification (unit: mm)

Refer to Figure 6 for sensor dimensions. (Tolerance ±0.5mm)

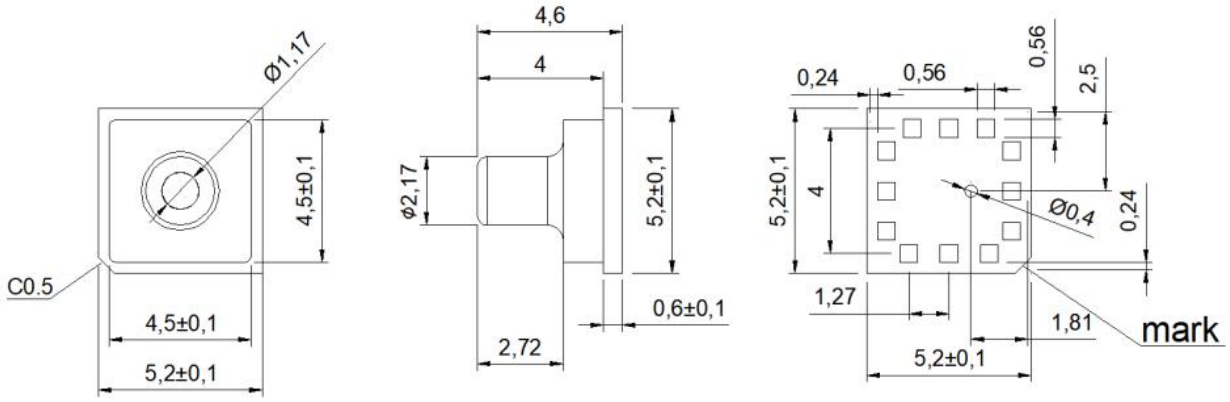


Fig.8 Sensor Dimensions

Recommended Footprint Layout refer to Figure 9.

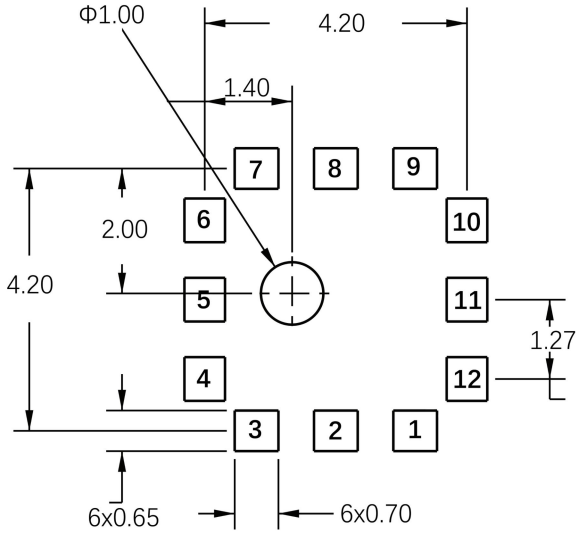


Fig.9 Recommended Footprint

9. Order Guide

GZP 6829 DC 040KPP B01

Tab.6 Order Guide

GZP	Pressure Sensor Series
6829	Product Series
D	Output type A: Analog output D: IIC output
C	IIC Communication Content Format
040KPP	Pressure range: 040 means the measured pressure value is 40 (including 0~40, -40~0, -40~40) Pressure unit: KP: KPa MP: MPa PS: PSI BA: Bar Pressure type: P: positive pressure (such as 0~40) N: negative pressure (such as -40~0) W: negative pressure to positive pressure (such as -40~40) Therefore, 006KPP means the measured pressure from 0kPa to 40kPa
B01	Packaging Method: B01: Tape

10. Models Example

Tab.7 Models Example

Pressure Range	**Model**
0 ~ 40kPa	GZP6829DC040KPP B01
-40 ~ 40kPa	GZP6829DC040KPW B01

*For more customized ranges and special parameter part numbers, please consult Sencoch.

11. Instructions for Use

11.1 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

· Please use a soldering iron with a head temperature of 260 to 300°C (30 W) and perform the work within 5 seconds.

· When soldering with a load applied to the terminals, the output may change, so be careful.

Please keep the soldering iron tip clean.

2) Reflow soldering (SMD terminal type)

The recommended reflow oven temperature setting conditions are show:

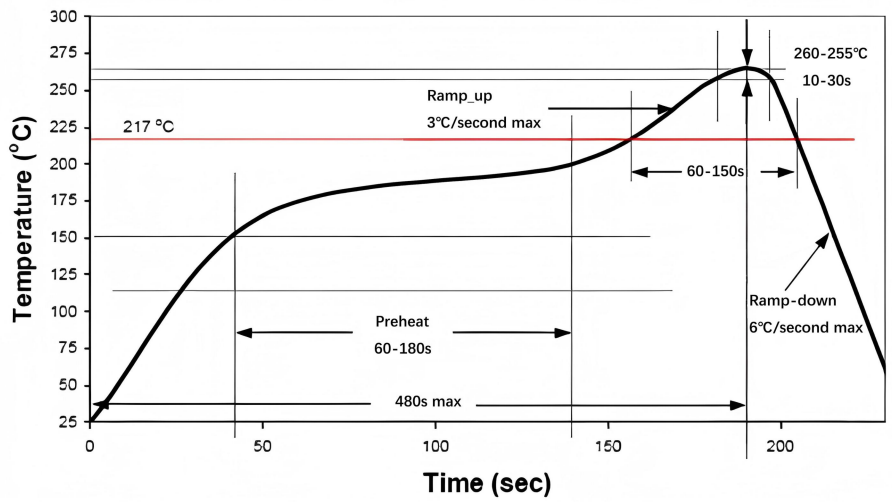


Fig.10 Remelting temperature setting conditions

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, when cutting and bending the substrate, be careful not to cause stress on the soldered parts.

5) Since the sensor terminals are exposed, if metal pieces touch the terminals, abnormal output will occur. Be careful not to touch them with metal pieces or hands.

6) After soldering, when coating is applied to prevent insulation degradation of the substrate, be careful not to allow chemicals to adhere to the sensor.

11.2 Cleaning Requirements

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

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

11.3 Storage and Transportation

1) This product is not drip-proof, so do not use it in a location 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 changes in sensor output or damage.

3) The chip of the pressure sensor is structurally exposed to light, and the output will change. Especially when applying pressure through a transparent cover, please avoid light from reaching the chip of the sensor.

4) Normally packaged pressure sensors can be transported by ordinary transportation tools. Please note: The product should be protected from moisture, impact, sunburn and pressure during transportation.

11.4 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 media that can be used directly are air and non-corrosive gases. Other media, especially when used in corrosive media or media containing foreign matter, may cause malfunctions and damage, so please avoid using it in the above environment .

4) There is a pressure sensor chip inside the pressure inlet. If a needle or other foreign object is inserted into the pressure inlet, the chip will be damaged and the inlet will be blocked, so please avoid such operations. In addition, please avoid blocking the atmosphere inlet when using it.

5) Please use the pressure within the rated pressure range. Using outside the range may cause damage.

6) Since static electricity may cause damage, please pay attention to the following matters when using.

Please ground the charged objects and workers on the table to discharge the static electricity around safely.

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

12. Packing Information

Tape Packing

Product Series		Tray (PCS)	Box (PCS)	Box (PCS)	Remark
GZP6829DC	LGA package	1000	2000	16000	Vacuum packaging

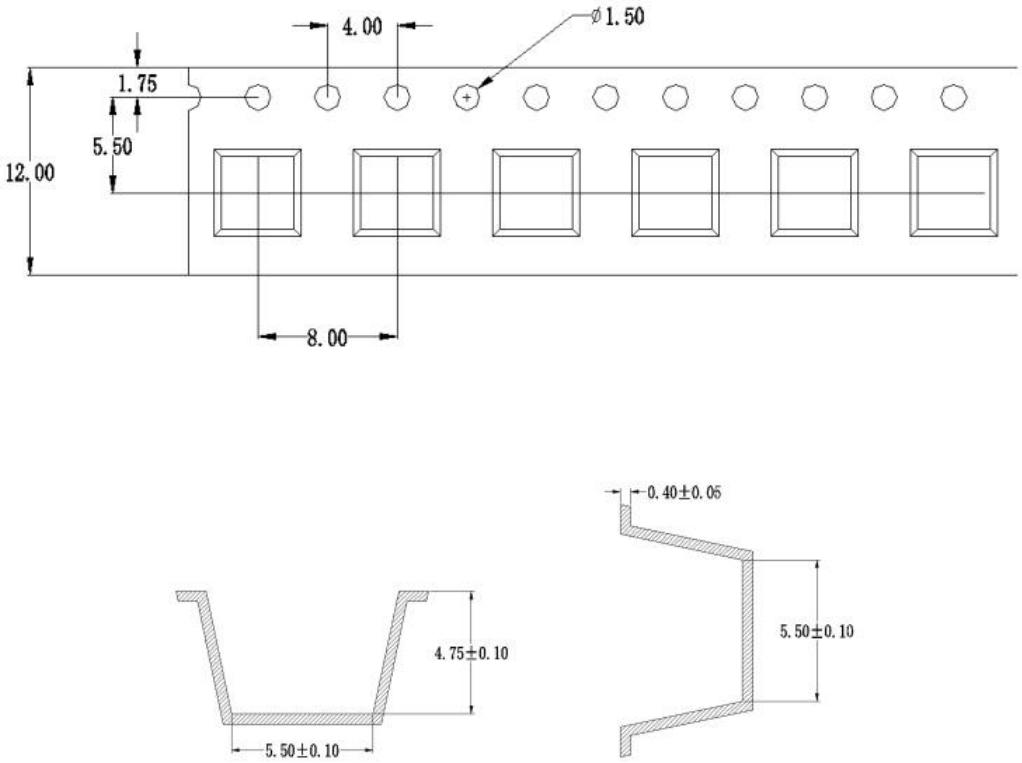


Fig.11 The drawing of tapes

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)

```
#include "stm32f10x.h"

// Define SCL and SDA pins
#define SCL_PIN GPIO_Pin_6
#define SDA_PIN GPIO_Pin_7
#define I2C_GPIO_PORT GPIOB

// Pressure range
#define PMIN -100000.0
#define PMAX 200000.0
/*
Common range                PMIN            PMAX
0 ~ 40kPa                    0.0             40000.0
-40 ~ 0kPa                  -40000.0         0.0
*/
*/

// Delay function
void delay_us(uint32_t us)
{
uint32_t count = us * 7;
while (count--);
}

// I2C initialization function
void I2C_Init(void)
{
GPIO_InitTypeDef GPIO_InitStructure;
// Enable GPIOB clock
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
// Configure the SCL and SDA pins as open-drain outputs
GPIO_InitStructure.GPIO_Pin = SCL_PIN | SDA_PIN;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_OD;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
```

```
GPIO_Init(I2C_GPIO_PORT, &GPIO_InitStructure);
// Initial state, SCL and SDA are high
GPIO_SetBits(I2C_GPIO_PORT, SCL_PIN | SDA_PIN);
}

// void I2C_Start(void)
{
    GPIO_SetBits(I2C_GPIO_PORT, SDA_PIN);
    delay_us(5);
    GPIO_SetBits(I2C_GPIO_PORT, SCL_PIN);
    delay_us(5);
    GPIO_ResetBits(I2C_GPIO_PORT, SDA_PIN);
    delay_us(5);
    GPIO_ResetBits(I2C_GPIO_PORT, SCL_PIN);
    delay_us(5);
}

// void I2C_Stop(void)
{
    GPIO_ResetBits(I2C_GPIO_PORT, SDA_PIN);
    delay_us(5);
    GPIO_SetBits(I2C_GPIO_PORT, SCL_PIN);
    delay_us(5);
    GPIO_SetBits(I2C_GPIO_PORT, SDA_PIN);
    delay_us(5);
}



// void I2C_SendByte(uint8_t byte)
{
    for (uint8_t i = 0; i < 8; i++) {
        if (byte & 0x80) {
            GPIO_SetBits(I2C_GPIO_PORT, SDA_PIN);
        } else {
```

```
            GPIO_ResetBits(I2C_GPIO_PORT, SDA_PIN);
        }
        byte <<= 1;
        delay_us(5);
        GPIO_SetBits(I2C_GPIO_PORT, SCL_PIN);
        delay_us(5);
        GPIO_ResetBits(I2C_GPIO_PORT, SCL_PIN);
        delay_us(5);
    }
}

// I2C reads a byte and sends ACK or NACK
uint8_t I2C_ReadByte(uint8_t ack)
{
uint8_t byte = 0;
GPIO_SetBits(I2C_GPIO_PORT, SDA_PIN);
for (uint8_t i = 0; i < 8; i++) {
byte <<= 1;
GPIO_SetBits(I2C_GPIO_PORT, SCL_PIN);
delay_us(5);
if (GPIO_ReadInputDataBit(I2C_GPIO_PORT, SDA_PIN)) {
byte |= 0x01;
}
GPIO_ResetBits(I2C_GPIO_PORT, SCL_PIN);
delay_us(5);
}
if (ack) {
GPIO_ResetBits(I2C_GPIO_PORT, SDA_PIN); // Send ACK
} else {
        GPIO_SetBits(I2C_GPIO_PORT, SDA_PIN); // 发送 NACK
    }
    delay_us(5);
    GPIO_SetBits(I2C_GPIO_PORT, SCL_PIN);
    delay_us(5);
    GPIO_ResetBits(I2C_GPIO_PORT, SCL_PIN);
    delay_us(5);
```

```
    return byte;
}

// uint8_t I2C_WaitAck(void)
{
    uint8_t ack = 0;
    GPIO_SetBits(I2C_GPIO_PORT, SDA_PIN);
    GPIO_SetBits(I2C_GPIO_PORT, SCL_PIN);
    delay_us(5);
    if (GPIO_ReadInputDataBit(I2C_GPIO_PORT, SDA_PIN)) {
        ack = 1;
    } else {
        ack = 0;
}
GPIO_ResetBits(I2C_GPIO_PORT, SCL_PIN);
delay_us(5);
return ack;
}

// Read multiple bytes from the specified slave address and register address
void I2C_ReadBytes(uint8_t slave_addr, uint8_t reg_addr, uint8_t *data, uint8_t len)
{
I2C_Start();
//Send slave address, write operation
I2C_SendByte(slave_addr << 1);
I2C_WaitAck();
//Send register address
I2C_SendByte(reg_addr);
I2C_WaitAck();
I2C_Start();
//Send slave address, read operation
I2C_SendByte((slave_addr << 1) | 0x01);
I2C_WaitAck();
for (uint8_t i = 0; i < len; i++) {
if (i == len - 1) {
// Last byte, send NACK
```

```
data[i] = I2C_ReadByte(0);
} else {
// Send ACK
data[i] = I2C_ReadByte(1);
}
}
I2C_Stop();
}


int main(void)
{
uint8_t received_data[5];
uint8_t temp_coeff_data[2];
uint32_t pressure_data;
    uint16_t temperature_data;
    float shiftN;
int EOFFout;
float actual_pressure,actual_temperature;
I2C_Init();
// Slave address 0x58, register address 0x04, read 5 bytes continuously
I2C_ReadBytes(0x58, 0x04, received_data, 5);
// Combined pressure data
pressure_data = ((uint32_t)received_data[0] << 16) | ((uint32_t)received_data[1] << 8) |
(uint32_t)received_data[2];
// Combined temperature data
temperature_data = ((uint16_t)received_data[3] << 8) | (uint16_t)received_data[4];
// Read two bytes from register 0x20 as temperature coefficient
I2C_ReadBytes(0x58, 0x20, temp_coeff_data, 2);
// Calculate shiftN
shiftN = (float)temp_coeff_data[1] / 10.0;
// Set EOFFout according to temperature coefficient [0]
    switch (temp_coeff_data[0])
    {
        case 0x0C:
            EOFFout = 4096;
```

```
            break;
        case 0x8C:
            EOFFout = -4096;
            break;
        case 0x0D:
            EOFFout = 8192;
            break;
        case 0x8D:
            EOFFout = -8192;
            break;
        case 0x0E:
            EOFFout = 16384;
            break;
        case 0x8E:
            EOFFout = -16384;
            break;
        default:
// You can add default processing, temporarily set it to 0 here
EOFFout = 0;
break;
}

// Process pressure data
if (pressure_data > 8388608)
    {
pressure_data -= 16777216;
}
// Process temperature data
if (temperature_data > 32768)
    {
temperature_data -= 65536;
}
// Calculate actual pressure
actual_pressure = ((float)pressure_data / (1 << 21)) * (PMAX -PMIN); // Unit is Pa
// Calculate actual temperature
actual_temperature = ((float)(temperature_data - EOFFout) / (1 << (int)shiftN)) + 25; //Unit
```

```
is °C
while (1)
    {
// Code to process received pressure, temperature, and temperature coefficient data can be added here
// For example, print data
// Note: To use printf you need to configure the serial port first
// printf("Pressure: %f, Temperature: %f,\n",actual_pressure,actual_temperature);
}
}
```