SV SensTech

华景传感科技

Data Sheet

Version 1.0/Oct. 2024

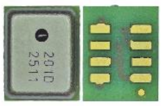
SVP-3730D-201A

拥有核心芯片技术的MEMS传感技术公司

A MEMS Sensor Company with Advanced Core Chip Technology

Product Specification

SVP-3730D-201A
Pressure Sensor Module



Features

- Pressure range: 100 ~300kPaA
- Temperature Range: -40…+105°C
- Supply voltage: 1.8 ... 3.6V
- Standby current：＜0.1μA
- Average current consumption: Low power: 6 μA @1Hz
- Digital I2C interface, Embedded 24-bit ADC

Applications

- Air pressure switch
- Altimeter
- Multi-function barometer
- Sweeper
- Unmanned aerial vehicle
- Absolute pressure control system

Definition

The SVP-3730D-201A digtal absolute pressure sensor consists of diffused silicon piezoresistive pressure sensor chip and special signal processing integrated circuit chip. Each module undergoes linear calibration and temperature compensation, ensuring that its output signal is linearly proportional to the input pressure.

The SVP-3730D-201A absolute pressure sensor is a compact digital pressure sensor designed to measure pressure and temperature values with high accuracy and low current consumption. Each pressure sensor has been individually calibrated for pressure and temperature and includes calibration coefficients before leaving the factory. Coefficients are used in applications to convert measurements into real pressure and temperature values. It is widely used in wearable devices, inflator pump, automotive electronics and other fields.

Contents

1. Device Internal Function Block Diagram

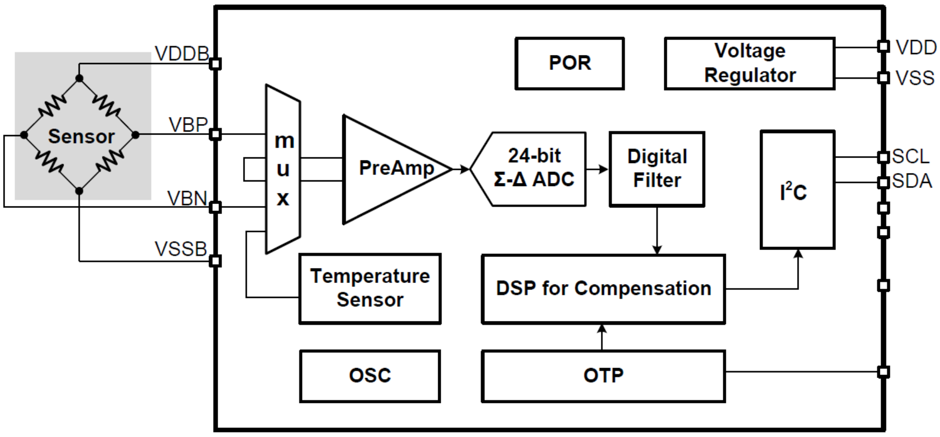


Figure1: Block Diagram

2. Absolute Maximum Ratings

Table1: Absolute Maximum Ratings

Parameter	Condition	Min	Max	Units
Storage temperature		-40	+125	°C
Supply voltage	All pins	-0.3	+3.6	V
Voltage at all IO pins	All pins	-0.3	+3.6	V
ESD	HBM,R=1.5kohm,C=100pF	-2	2	kV
Over pressure			1000	kPa

3. Electrical Characteristics

Table2: Electrical Characteristics

（Vs=3.3VDC，T_A=25±3℃）

Param eter	Symbol	Condition	Min	Type	Max	Units
Performance index						
Operating Pressure	P		100		300	kPaA
Operating temperature	T_A		-40		105	℃

	T_c	Full-scsle accuracy			0	25	65	℃
Absolute pressure accuracy	P_A	0~200kPa @ 0~65 ℃			-1.0	±0.5	1.0	kPa
Relative pressure accuracy	P_R					±0.02		kPa
Noise in pressure	P_noise					3.0		PaRMS
Resolution of output data	P_res					1.0		Pa
Solder drift		24h after reflow			-0.3		0.3	kPa
Long term stability		12 months			-0.1		0.1	kPa
Supply voltage and current consumption								
Supply voltage	VDD				1.8	3.3	3.6	V
PSRR	PSR_{VDD}	VDD=1.8V			17			dB
Average current consumption （1 Hz）	I_{VDDAVP}	Operating mode	OSR_P	OSR_T				
		Low-power mode	4X	4X		6	8	uA
		Standard mode	16X	4X		17	22	uA
		High-precision mode	64X	4X		42	50	uA
Peak current	I_{PEAK}					760		uA
Standby current	I_{VDD}	Hibernation at 25°C					0.1	uA
ADC Conversion								
Pressure/Temperature measurement rate	FS,raw	OSR 为 2X~128X			20		1350	Hz
Pressure/temperature measurement time	Tc	Operating mode	OSR_P	OSR_T				ms
		Low-power mode	4X	4X		13		ms
		Standard mode	16X	4X		31		ms
		High-precision mode	64X	4X		105		ms
ADC resolution		Pressure				24		Bit
		Temperature				16		Bit
Serial data clock	F_{I2C}						3.4	MHz
Temperature sensor								
Resolution of output data	T_{res}					0.003		K/LSB
Measurement range	T_{txt_sp}	Built-in temperature sensor			-40		150	℃
Operating temperature accuracy						±2.0		℃
Power-on sequence								
Run-up time	T_{STA1}	VDD rise to the time when the interface started communication					1.0	ms

	T_{STA2}	VDD rise to the time when the measurement began			2.5	ms
Wake-up Time	T_{WUP1}	Time between the sleep state and the interface started communication			0.5	ms
	T_{WUP2}	Time between the sleep state and the measurement began			2.0	ms

4. I2C Communication Protocol

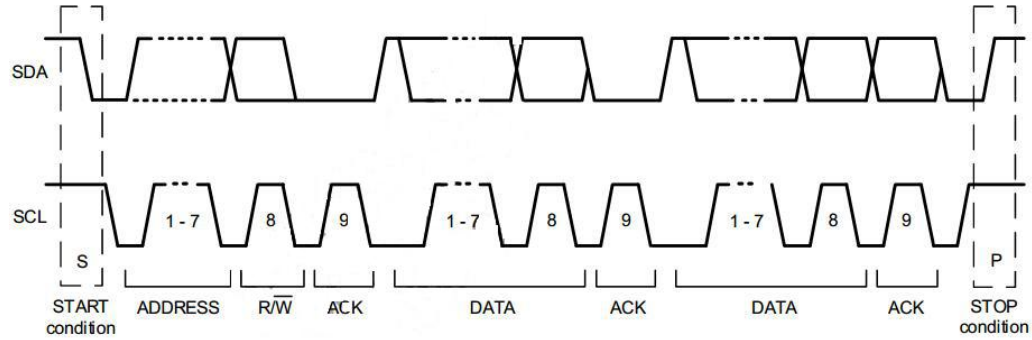


Figure2: I2C Communication Protocol

- **START Condition**

  SDA transitions from the idle high state to the low state when SCL stays high. This can also be repeated during the transmission start condition, which indicates that the transaction will start again without an intermediate stop bit.

- **Address Bits**

  During 1st byte data transfer, the first 7-bit provide defined device address, default value is 0x78. Device with this address will response the communication.

- **Read/Write Direction Bit**

  During 1st byte data transfer, last bit indicates communication direction. 0 indicates master device writes data, 1 indicates master device reads data.

- **Data Byte**

  All bytes transferred at SDA line except address bits and read/write bit are recognized as data bytes.

- **Acknowledge or Not Acknowledge Bit**

  Acknowledge bit is used to tell sender that bytes received. The ACK bit acknowledges the receiving of the byte. The device needs to feedback ACK bit for each received data byte including the address byte. At this moment, the device that sent the data stops driving the SDA line and the SDA line is pulled high. The receiving device does not need to do anything if no ACK is required. The receiving device needs to pull SDA low if the ACK is required. A slave device does not need to reply ACK if the slave device is not matched. The master does not send ACK bit if the master wants to end the communication. If no ACK is sent by the master device, a stop bit needs to be generated by the master device for ending the transmission.

➢ **Stop condition**

SDA from low voltage level to high voltage level and SCL keeps high voltage level, end I2C communication.

5. Operating Mode

After the product is powered on, the pressure and temperature measurement and calibration process will only be started once the corresponding I2C command is received, After the measurement is complete, it automatically enters deep sleep to save power consumption. In deep hibernation, All internal circuits except the I2C interface are closed, Its current consumption is only about 0.1μA.

Sensor I2C address: The default 7bits device address is 0x78；

Operation sequence:

①Power-on

②I2C write in

③Wait

④I2C read

⑤Data Conversion

5.1 I2C write In

Send the 0xAC command to get the data calibrated using the internal algorithm, To obtain the calibration value, run the 0xAC command as follows：

5.1.1. Send the Get_Cal command (0xAC)

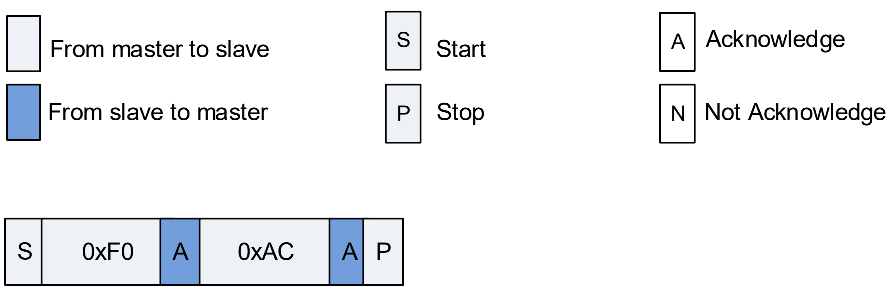


Figure3： I2C Write Command

5.1.2 Get_Cal_S Command

Get_Cal_S command (0xB0 ~ 0xBF) is almost the same as Get_Cal command (0xAC) , In addition, the ADC oversampling rate is set differently during measurement. The oversampling rate of the Get_Cal_S command is determined by the content of the command itself. After the user calibrates the sensor, After burning the OTP data, The Get_Cal_S command allows the user to complete the measurement with different sample rates, To achieve high, medium or low precision measurements.

Table3：Get_Cal_S Command

Command 0xBX(HEX)	Function	Detail
The [3] Bit of X	The ADC oversampling rate OSR_T when measuring temperature	0: 4x 1: 8x
The [2:0] Bit of X	The ADC oversampling rate OSR_P when measuring the external bridge	000: 128X 100: 8X 001: 64X 101: 4X 010: 32x 110: 2X 011: 16X 111: 1X

5.2. Waiting

You need to wait a while after sending the write command before sending the read command, because it takes a while to complete the entire measurement internally. The wait time depends on the setting of the pressure oversampling rate and the temperature oversampling rate.

The waiting period does not need to be calculated. You can determine whether the collection is complete by reading the IIC status word continuously.

Different oversampling rates correspond to different waiting times. The following time is the time for temperature measurement + pressure measurement:

Table4： Measurement time

ADC Conversion							
Pressure/Temperature measurement rate	FS,raw	OSR 为 2X~128X		20		1350	Hz
Pressure/temperature measurement time	Tc	OSR_P	OSR_T				ms
		2X	4X		10		
		4X			13		
		8X			19		
		16X			31		
		32X			56		
		64X			105		
		128X			203		

Table5：Status of Bits

Bit	Meaning	Description
Bit7	Reserved	Constant 0
Bit6	Power indication	1: Power on; 0: Power off
Bit5	Busy indication	1: Busy, the device is measuring pressure and temperature and the results are not ready yet. New I^2C command will not be proceeded; 0: Idle, the recent I2C command has been executed and the data to be read is ready.
Bit4	Reserved	Constant 0
Bit3	Mode Status	Constant 0
Bit2	Memory integrity/error flag	1: The integrity check(CRC) of the NVM is failed. Some of the data in the NVM is error; 0: The integrity check (CRC) of the NVM is passed. All the data in the NVM is correct.
Bit1	Reserved	Constant 0
Bit0	Reserved	Constant 0

5.3. Read Data

To ensure that the written instructions and read the time interval is greater than the measurement of the length of the data calibration can read, reading format as shown in figure 4:

Pressure and temperature data need to read together,If you do not need the temperature value, you can not perform the temperature conversion calculation.

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

Figure4： I2C Read Out 3-byte Original Bridge Data or Temperature Data

5.4. Revaluation of data

When you read the calibration data, you need to do a simple conversion of the unsigned numbers expressed as percentages.

For ease of understanding, we assume that the calibration data we read is：0x00 0x9B 0xB0 0xC5 0x56 0xAA

0x00 indicates that the status word Bit5 is 1, indicating that the latest I2C is busy and needs to wait for a while. If Bit5 is 0, the device is not busy and can read the data.

0x9B 0xB0 0xC5 The three bytes are the bridge calibration values

0x56 0xAA The two bytes are temperature calibration values

Pressure value conversion: Convert 0x9B 0xB0 0xC5 to decimal number 10203333

The calculation assumptions used in the calibration range of 0 ~ 200 kPa, the corresponding AD output is 2516582.4 ~ 14260633.6 (15% ~ 85% AD) according to the calibration P2 input/output relation formula is:

Actual pressure value =（200-0）/（14260633.6-2516582.4）*（10203333-2516582.4）+0 = 130.905hPa

Temperature value conversion: Convert 0x56 0xAA to a decimal number of 22186. Since the calibration data read is expressed in the form of a percentage, this percentage is numerically equal to the ratio of the decimal number we converted to the maximum value of the 16 bits unsigned number (65535), so the following calculation can be performed when converting the percentage

22186/65536*100% = 33.85%

The temperature calibration range is specified as -40℃~150℃, So the calibration value = (150-(-40))*33.85% - 40 = 24.32℃

Barometric value conversion: input-output relationship:

$$\boldsymbol{Pressure} = \frac{\mathbf{P_{MAX}} - \mathbf{P_{MIN}}}{\mathbf{D_{MAX}} - \mathbf{D_{MIN}}} \cdot (\mathbf{D_{test}} - \mathbf{D_{MIN}}) + \mathbf{P_{MIN}}$$

Pressure: actual pressure value; **Dtest**: digital output value of the sensor; **PMIN**: sensor zero pressure value; **PMAX**: sensor full scale pressure value; **DMIN**: the digital output value corresponding to the sensor zero; **DMAX**: the corresponding digital output value of the sensor at full scale;

6. Typical Application Circuit

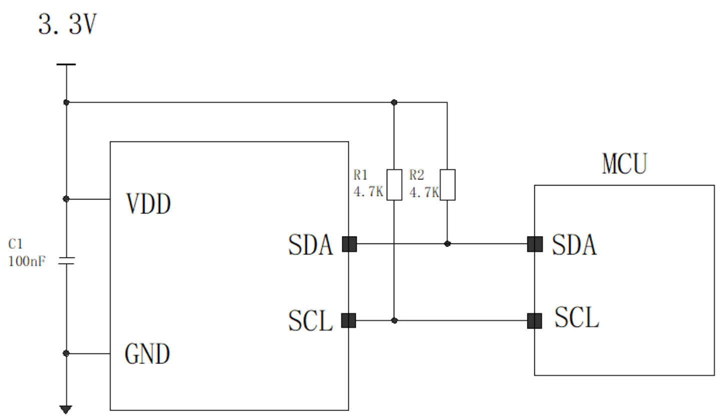


Figure5：Typical Application Circuit

7. Mechanical Characteristics

7.1 Outline Dimensions   Unit: mm

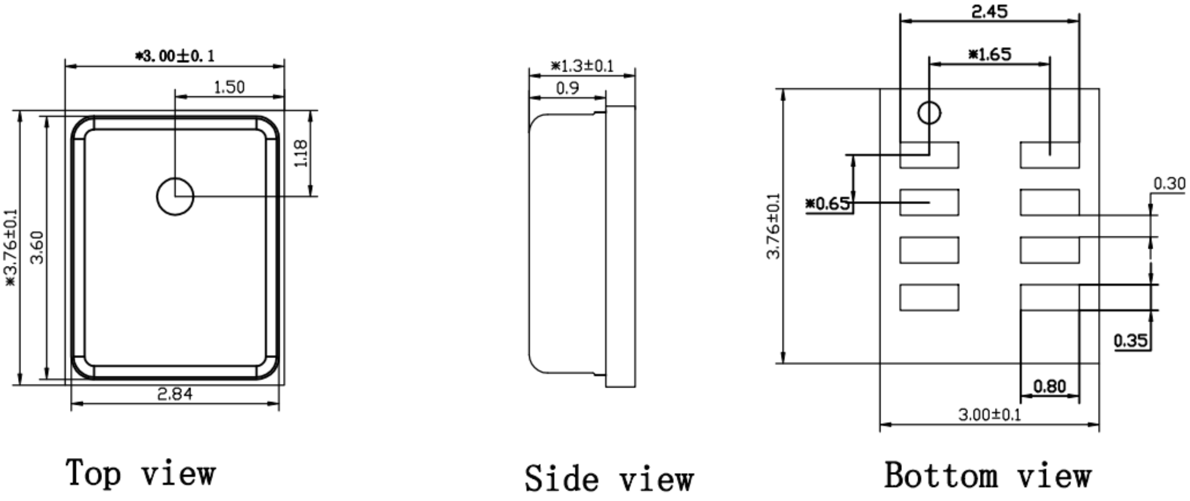


Figure6：Outline Dimensions

7.2 Pin Configurations

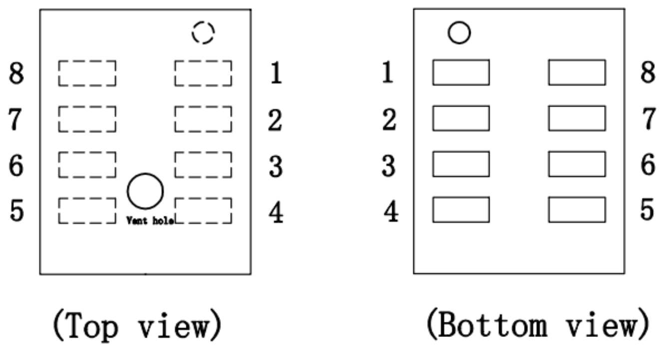


Figure7：Pin Configurations

Table6：Pin Configurations

Pin#	1	2	3	4	5	6	7	8
Name	GND	NC	SDA	SCL	NC	NC	GND	VDD

Note： Keep the NC foot in the air; otherwise, the product function may fail.

8. Matters Need Attention

8.1. Soldering Recommendation

8.1.1. Solder Reflow

Recommended Solder Reflow：

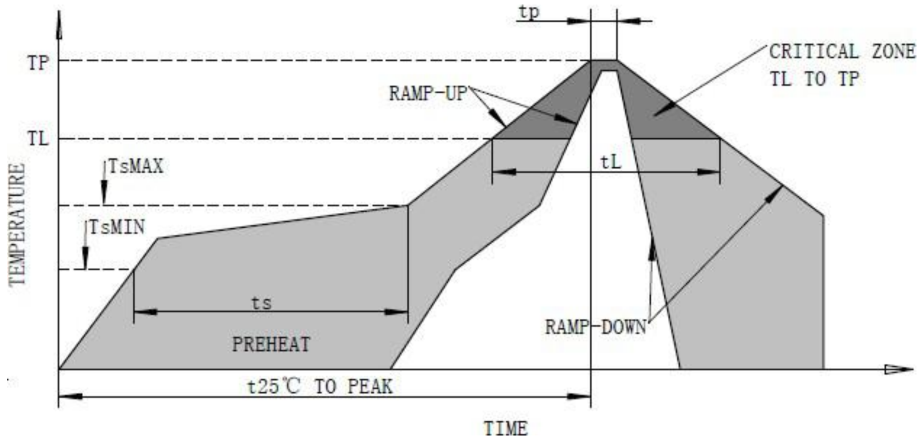


Figure8： Reflow Profile

Table7：Reflow Parameter Configuration

Profile Feature		Pb-Free Assembly
Average ramp-up rate(TsMAX to TP)		3°C /seconds max
Preheat	Temperature Min.(TsMIN)	150℃
	Temperature Max.(TsMAX)	200℃
	Time(TsMIN to TsMAX)(Ts)	60～80 seconds
Time maintained above	Temperature(TL)	217℃
	Time(tL)	60～150 seconds
Peak temperature(TP)		260℃
Time within 5 of actual peak temperature(TP)		20～40seconds
Ramp-down rate		4°C /seconds max
Time 25℃ to peak temperature		8 minutes max

8.1.2. Hand Soldering

1. Select 25W external heat or 20W internal heat type electric soldering iron for welding, set temperature 260~320℃, single point welding does not exceed 5 seconds. The metal surface to be welded should be kept clean. Oxides, dust, oil, etc. prevent solder from infiltrating the metal surface to be welded, which can be removed by mechanical or chemical methods before welding;
2. When the welding is completed, pay attention to the timing, direction and speed of the welding head to avoid the risk of poor welding caused by product displacement;

8.1.3. Soldering Rework

1. Do not use heat gun to repair the pressure sensor products and its surrounding devices. For SMD products, choose the method of using solder paste over reflux;
2. When using a soldering iron for manual maintenance, avoid contact with the sensor PAD for a long time, so as to avoid product failure or the pad (including PCB pad) falling off;
3. Pay attention to the protection of the pressure sensor during the welding process of the connector, do not touch the device directly with your hand, and prevent the solder from falling into the device;

8.2.Cleaning

1. After welding, use special printed board cleaning agent as far as possible;
2. It is strictly prohibited to clean or wipe the pressure sensor with liquid, even if the device around it is cleaned or wiped with liquid, it should be noted that no liquid can enter the inside of the barometer to cause abnormal output;
3. Pay attention to the protection of the pressure sensor during the welding process of the connector, do not touch the device directly with your hand, and prevent the solder from falling into the device to avoid product failure;

8.3. Design considerations

1. The pressure sensor is designed away from the stress concentration area;
2. Pressure sensor design away from heat source devices (including PCB double-sided);
3. The pressure sensor can not be enclosed by non-breathable materials, and the intake or vent can not be blocked;

9. Package Specifications

Carrier Tape Information：Unit: mm

Quantity per reel: 5K pcs

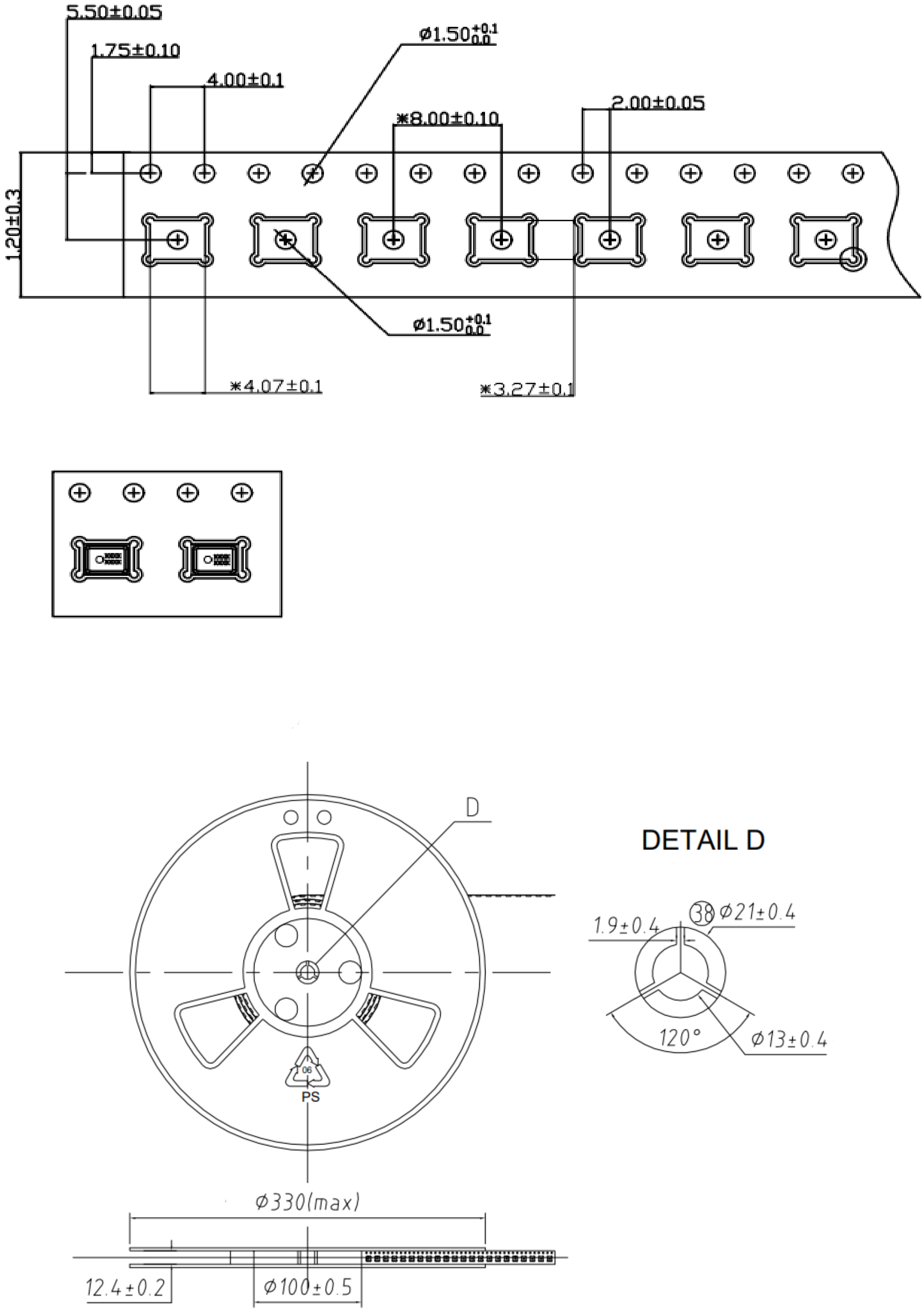


Figure9： Carrier Tape

10. Revision History

Version	Changes	Date
1.0	Initial release	08-Oct-2024

11. Contact Information

SV SensTech （wuxi）Co ., Ltd.

Add：F2 ,No. 200，Linghu Avenue， Wuxi city，Jiangsu, China 214135

Tel：(86) 0510-85622282

Fax：(86) 0510-85622278

E-mail: sales@svsens.com

Web：www.svsens.com

12. I2C Sample Code

```
#include "Sensor3730D.h"
#include "Sensor3730D _IIC.h"

//IIC clock line
sbit SCL = P1 ^ 1;

//IIC data line
sbit SDA = P1 ^ 0;

//Set the input and output mode of IIC data pin
#define Set_SDA_INPUT()
        P1MDOUT &= 0xFE;
        P1 |= 0X01
#define Set_SDA_OUTPUT() P1MDOUT |= 0x01;

////Delay function needs to be defined
void DelayUs(unsigned char i)
{
}

//Start signal
void Start(void)
{
        SDA = 1;
        DelayUs(2);
        SCL = 1;
        DelayUs(2);
        SDA = 0;
        DelayUs(2);
        SCL = 0;
}

//Stop signal
void Stop(void)
{
        Set_SDA_OUTPUT();
        SDA = 0;
        DelayUs(2);
        SCL = 1;
        DelayUs(2);
        SDA = 1;
        DelayUs(2);
}

//Read ACK signal
unsigned char Check_ACK(void)
{
        unsigned char ack;
        Set_SDA_INPUT();
        SCL = 1;
        DelayUs(2);
        ack = SDA;
```

```
    SCL = 0;
    Set_SDA_OUTPUT();
    return ack;
}

//Send ACK signal
void Send_ACK(void)
{
    Set_SDA_OUTPUT();
    SDA = 0;
    DelayUs(2);
    SCL = 1;
    DelayUs(2);
    SCL = 0;
    DelayUs(2);
}

//Send one byte
void SendByte(unsigned char byte)
{
    unsigned char i = 0;
    Set_SDA_OUTPUT();
    do
    {
        if (byte & 0x80)
        {
            SDA = 1;
        }
        else
        {
            SDA = 0;
        }
        DelayUs(2);
        SCL = 1;
        DelayUs(2);
        byte <<= 1;
        i++;
        SCL = 0;
    } while (i < 8);
    SCL = 0;
}

//Receive one byte
unsigned char ReceiveByte(void)
{
    unsigned char i = 0, tmp = 0;
    Set_SDA_INPUT();
    do
    {
        tmp <<= 1;
        SCL = 1;
        DelayUs(2);
        if (SDA)
        {
```

```
			tmp |= 1;
		}
		SCL = 0;
		DelayUs(2);
		i++;
	} while (i < 8);
	return tmp;
}

//Write a byte of data through IIC
uint8 BSP_IIC_Write(uint8 address, uint8 *buf, uint8 count)
{
	unsigned char timeout, ack;
	address &= 0xFE;
	Start();
	DelayUs(2);
	SendByte(address);
	Set_SDA_INPUT();
	DelayUs(2);
	timeout = 0;
	do
	{
		ack = Check_ACK();
		timeout++;
		if (timeout == 10)
		{
			Stop();
			return 1;
		}
	} while (ack);
	while (count)
	{
		SendByte(*buf);
		Set_SDA_INPUT();
		DelayUs(2);
		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
uint8 BSP_IIC_Read(uint8 address, uint8 *buf, uint8 count)
```

```
{
    unsigned char timeout, ack;
    address |= 0x01;
    Start();
    SendByte(address);
    Set_SDA_INPUT();
    DelayUs(2);
    timeout = 0;
    do
    {
        ack = Check_ACK();
        timeout++;
        if (timeout == 4)
        {
            Stop();
            return 1;
        }
    } while (ack);
    DelayUs(2);
    while (count)
    {
        *buf = ReceiveByte();
        if (count != 1)
            Send_ACK();
        buf++;
        count--;
    }
    Stop();
    return 0;
}

// Define the upper and lower limits of the calibration pressure
#define PMIN 100000// Zero Point Pressure Value for example 100kPa
#define PMAX300000 // Full range pressure Value, for example 300kPa
#define DMIN 2516582.4 //AD value corresponding to pressure zero, for example 15%AD
#define DMAX 14260633.6 //AD Value Corresponding to Full Pressure Range, for example 85%AD

//The 7-bit IIC address of the Sensor3730D is 0x78
uint8 Device_Address = 0x78 << 1;

//Delay function needs to be defined
void DelayMs(uint8 count)
{
}

//Read the status of IIC and judge whether IIC is busy
uint8 2020D_IsBusy(void)
{
    uint8 status;
    BSP_IIC_Read(Device_Address, &status, 1);
    status = (status >> 5) & 0x01;
    return status;
}
```

```
void 2020D_get_cal(void)
{
	uint8 buffer[6] = {0};
	uint32 Dtest = 0;
	uint16 temp_raw = 0;
	double pressure = 0.0, temp = 0.0;

	//Send 0xAC command and read the returned six-byte data
	buffer[0] = 0xAC;
	BSP_IIC_Write(Device_Address, buffer, 1);
	DelayMs(5);
	while (1)
	{
		if (2020D_IsBusy())
		{
			DelayMs(1);
		}
		else
			break;
	}
	BSP_IIC_Read(Device_Address, buffer, 6);

	//The returned pressure and temperature values are converted into actual values according to the calibration
range
	Dtest = ((uint32)buffer[1] << 16) | ((uint16)buffer[2] << 8) | buffer[3];
	temp_raw = ((uint16)buffer[4] << 8) | (buffer[5] << 0);
	pressure = (PMAX-PMIN)/(DMAX-DMIN)*(Dtest-DMIN)+PMIN;
	temp = (double)temp_raw / 65536;
	temp = temp * 19000 - 4000;
}
```