NOVOSENSE

NSPAD1N Series
Automotive Absolute Pressure Sensor
Datasheet (EN) 1.0

Product Overview

NSPAD1N is a calibrated absolute pressure sensor series product launched by NOVOSENSE which uses an automotive grade ASIC to calibrate and compensate the MEMS sensor element. The pressure signal from 10kPa to 400kPa can be converted into SPI/I²C output signal or analog output signal (0~5V) with a customizable output range. While ensuring the reliability of the product, the two chips are integrated and packaged, reduces the package size greatly. This series provides outstanding performance in terms of high initial accuracy and small drift over lifetime.

Key Features

- High precision pressure sensing
  Digital output better than ±1%F.S. (-20°C to 115°C)
  Analog output better than ±1.5%F.S. (-20°C to 115°C)
- Large temperature range (-40°C to 125°C)
- Pressure range can be customized
- 24-bit ADC and 12-bit DAC
- I²C/SPI/Analog output
- Surface-mountable DFN-8 package with Wettable Flank
- AEC-Q100 qualified
- RoHS & REACH Compliance
- Moisture sensitivity level: MSL1

Applications

- Automotive seat comfort system
- Automotive barometric absolute pressure (ECU-BAP)
- Industrial control

Device Information

Part Number	Package	Body Size
NSPAD1N	DFN-8	3mm × 3mm × 1.54mm

Outline

INDEX

1. Pin Configuration and Functions

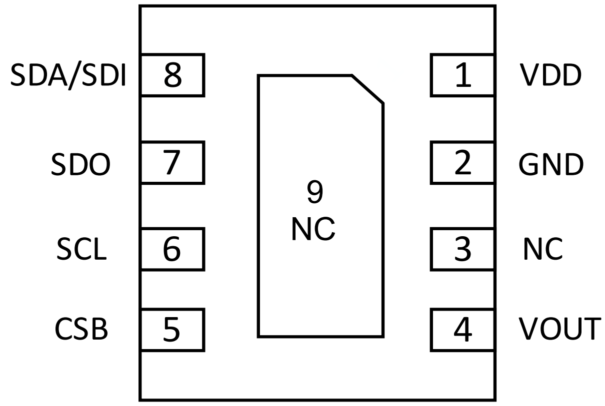


Figure 1.1 NSPAD1N Pin Definition (Bottom View)

Table 1.1 NSPAD1N Pin Description

Pin No.	Symbol	Function
1	VDD	Power supply
2	GND	Ground
3	NC	No connect
4	VOUT	Analog output
5	CSB	Chip select
6	SCL	Serial clock
7	SDO	Serial data output in SPI mode(SDO)
8	SDA/SDI	Serial data input/output in I^2C mode(SDA)
		Serial data input in SPI mode(SDI)
9	NC	No connect

2. Absolute Maximum Ratings

Parameters	*Symbol*	*Min*	*Typ*	*Max*	*Unit*	*Comments*
Supply voltage	VDD	-0.3		6.5	V	
Analog pin voltage	VOUT	-0.3		5.3	V	
Proof pressure	P_{proof}	1000			kPa	
Burst pressure	P_{burst}	1500			kPa	
Storage temperature	Tstg	-40		125	°C	

3. ESD Ratings

	Ratings	*Value*	*Unit*
Electrostatic discharge	Human body model (HBM), per AEC-Q100-002-RevE	±2	kV
	Charged device model (CDM), per AEC-Q100-011-RevB	±750	V

4. Recommended Operating Conditions

Parameters	*Symbol*	*Min*	*Typ*	*Max*	*Unit*	*Comments*
Supply voltage	VDD	3.0	3.3	3.6	V	VDD=3.3V
		4.5	5.0	5.5	V	VDD=5.0V
Operating pressure	P_{amb}	10		400	kPa	
Operating temperature	Topr	-40		125	°C	

5. Specifications

5.1. Electrical Characteristics

Parameters	Symbol	Min	Typ	Max	Unit	Comments
Accuracy pressure[1, 2, 3, 4]	Acc_P	-1.0%		1.0%	%F.S.	Digital output@-20°C ~115°C
		-1.5%		1.5%	%F.S.	Analog output@-20°C ~115°C
Power on reset	V_{POR_VDD}		2.0		V	
Operating current	I_{avdd}	1.5	2.6	3.5	mA	Analog output
			0.3	30	uA	Standby mode in digital output
Output RMS noise	V_{rms}		0.5		mV	
Output load resistance	R_{load}	1			kOhm	
Output load capacitance	C_{load}			15	nF	
Power up time	T_{UP}		100		ms	@25°C
Response time	T_{RESP}		2		ms	@25°C
EEPROM data retention	T_{live}	10			years	@125°C

1. Accuracy includes non-linearity, temperature, pressure hysteresis, temperature hysteresis.
2. Accuracy based on the part number NSPAD1N115DR08 HTOL, LTOL, HTSL, HAST and TCT testing.
3. For pressure accuracy of different part number, please refer to complete part number list at chapter 9. Unless otherwise specified, the accuracy is based on typical operating voltage.
4. The ratiometric analog output also include ±0.5% ratiometric error. The ratiometric error is defined as the difference between the ratio that VDD changed and the ratio that VOUT changed. Ratiometric signal error is not included in the overall accuracy. Absolute analog output and I^2C output are not applicable.

5.2. I²C Timing Diagram

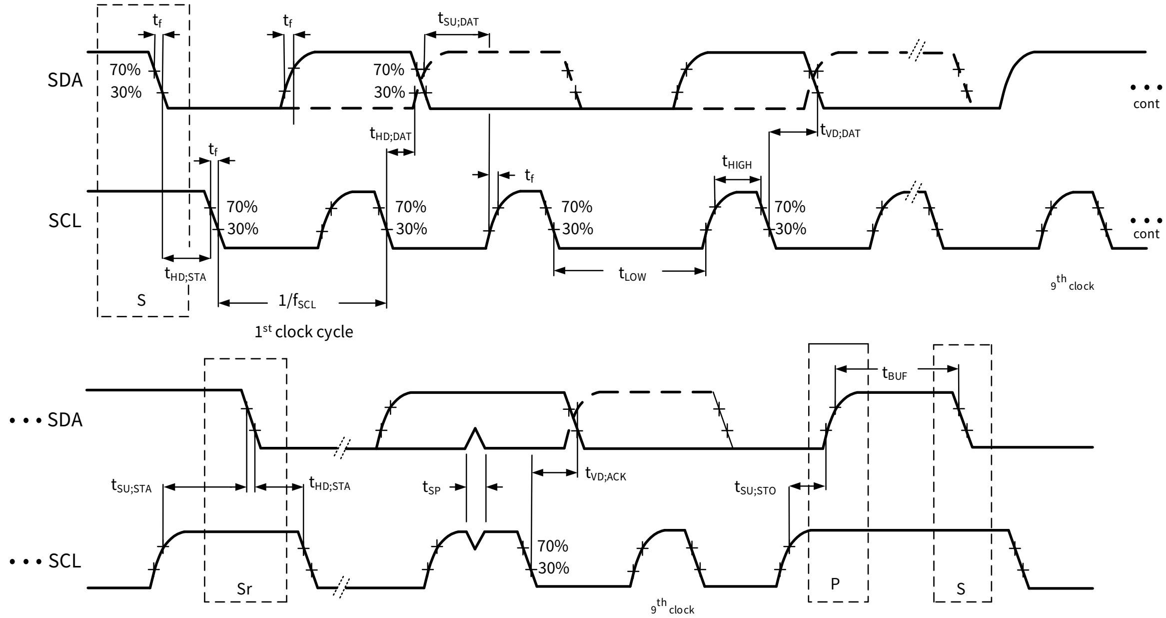


Figure 5.1 I²C Timing Diagram

5.3. I²C Electrical Characteristics

Parameters	***Symbol***	***Min***	***Typ***	***Max***	***Unit***	***Comments***
Clock frequency	f_{scl}			400	kHz	
SCL low pulse	t_{LOW}	1.3			us	
SCL high pulse	t_{HIGH}	0.6			us	
SDA setup time	t_{SUDAT}	0.1			us	
SDA hold time	t_{HDDAT}	0.0			us	
Setup time for a repeated start condition	t_{SUSTA}	0.6			us	
Hold time for a start condition	t_{HDSTA}	0.6			us	
Setup time for a stop condition	t_{SUSTO}	0.6			us	
Time before a new transmission can start	t_{BUF}	1.3			us	

5.4. SPI Timing Diagram

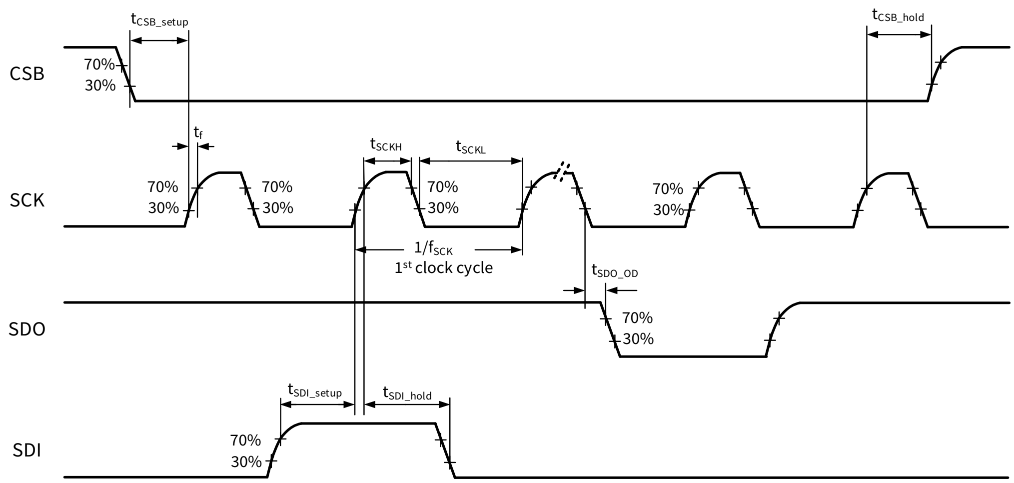


Figure 5.2 SPI Timing Diagram

5.5. SPI Electrical Characteristics

Parameters	*Symbol*	*Min*	*Typ*	*Max*	*Unit*	*Comments*
Clock frequency	f_{SCK}			10	MHz	Max load on SDIO or SDO = 25pF
SLCK low pulse	t_{SCKL}	20			ns	
SLCK high pulse	t_{SCKH}	20			ns	
SDI setup time	t_{SDI_setup}	20			ns	
SDI hold time	t_{SDI_hold}	20			ns	
SDO/SDI output delay	t_{SDO_OD}			30	ns	Load = 25pF
				40	ns	Load = 250pF
CSB setup time	t_{CSB_setup}	20			ns	
CSB hold time	t_{CSB_hold}	40			ns	

6. Function Description

6.1. Overview

NSPAD1N uses a MEMS piezoresistive absolute pressure sensor element as a pressure sensitive component that provide an original signal output that is proportional to ambient pressure. The built-in conditioning IC drives the sensitive component and amplifies, temperature compensates, and linearizes the original signal to output a voltage signal that is linear with the applied pressure.

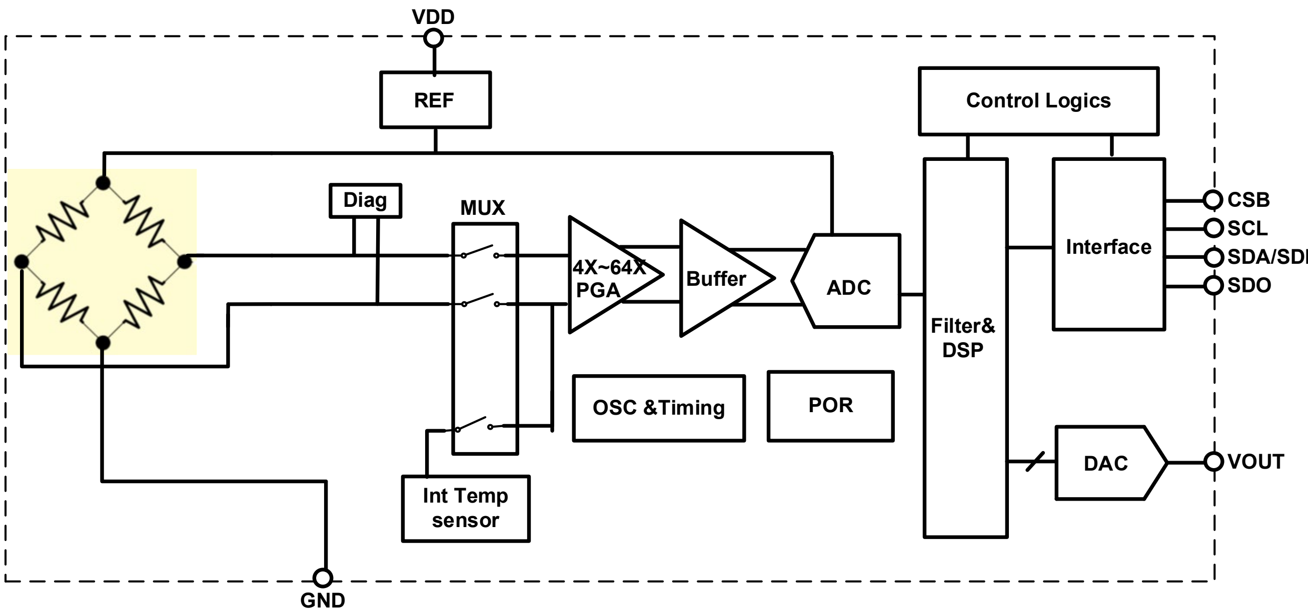


Figure 6.1 Product Function Block Diagram

6.2. Accuracy

Factors affecting the accuracy of NSPAD1N series products include power supply voltage (ratiometric error), pressure, temperature and aging effects. Standard output refers to the theoretical voltage output calculated by the transfer function of the pressure in the range. The error equals the deviation between the measured output voltage value and the specified output voltage value. The accuracy in the following analysis is in a typical application circuit.

6.2.1. Ratiometric Error

Ideally the sensor is ratiometric - the output (VOUT) scales by the same ratio that VDDHV increases or decreases. The ratiometric error is defined as the difference between the ratio that VDDHV changed and the ratio that VOUT changed, expressed as a percentage. The calculation formula is as follows:

$$E_{RAT}(\%) = \frac{VOUT(@VDD) - VOUT(@5V) \times \frac{VDD}{5V}}{5V} \times 100\%$$

The output voltage VOUT is ratiometric to VDDHV. VDDHV must be in the operating range.

Table 6.1 Ratiometric Output Error

Supply Voltage (V)	***Max. Ratiometric Error E_{RAT}(%) @ VDD_{TYP}***
VDD_{MIN}	±0.5%
VDD_{TYP}	0
VDD_{MAX}	±0.5%

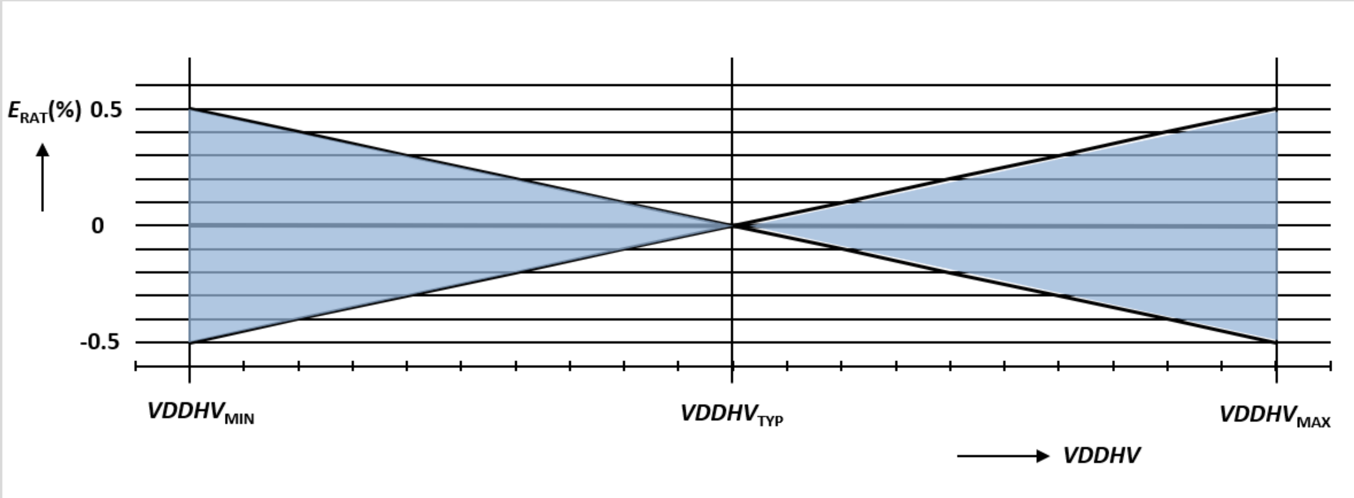


Figure 6.2 Ratiometric Error

6.2.2. Overall Accuracy

The accuracy error includes errors introduced by all influencing factors within the operating range of pressure and temperature, including:

Pressure: Output deviation from target transfer function over the specified pressure range

Temperature: Output deviation over the temperature range

Aging: Parameter drift over life time

Ps: Ratiometric signal error is not included in the overall accuracy. For error measurements, the supply voltage must have the nominal value (VDD = 5V).

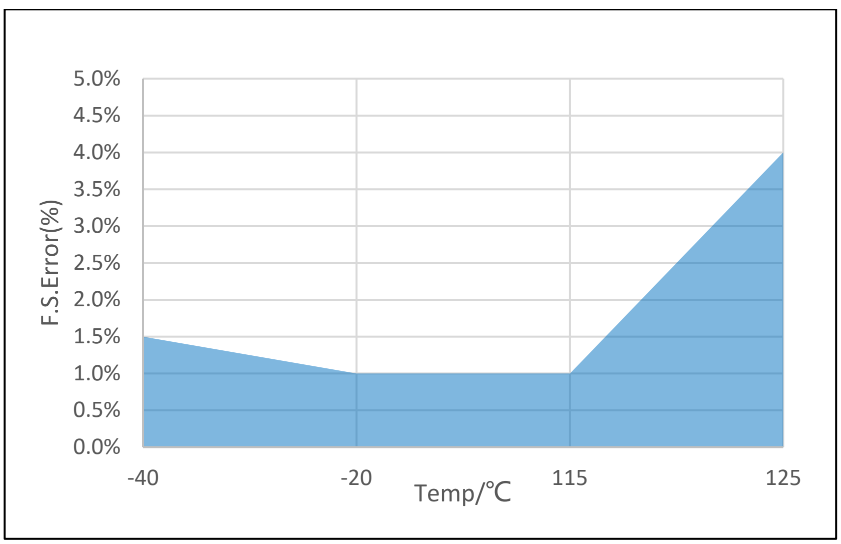


Figure 6.3 Digital Output Accuracy

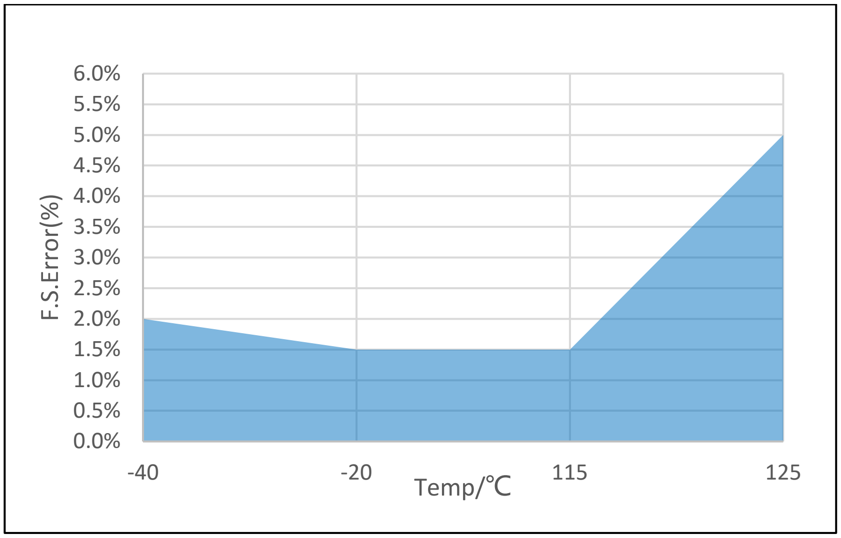


Figure 6.4 Analog Output Accuracy

6.2.3. Actual Accuracy

HTOL: High Temperature Operating Life. 130°C for 1000 hours, VDD at 5.5V.

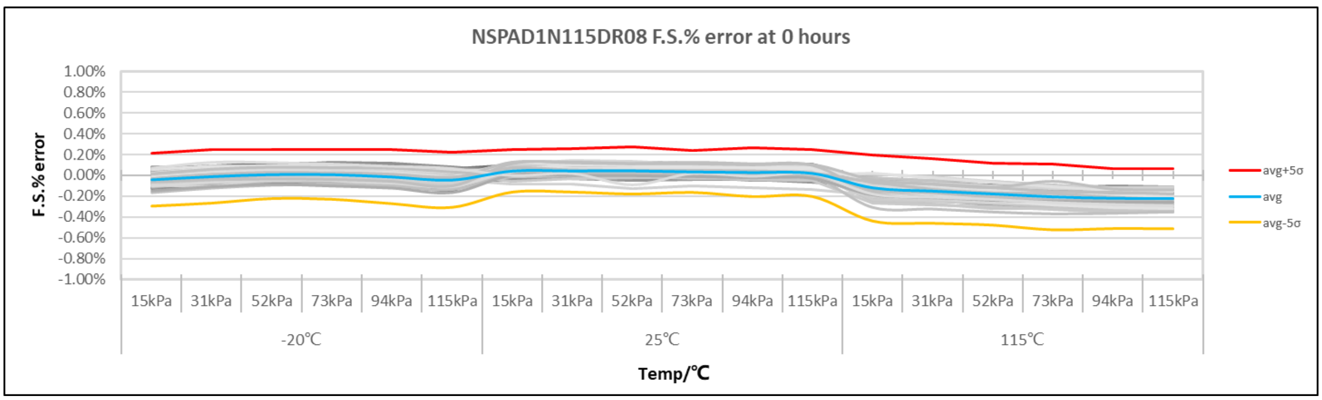


Figure 6.5 NSPAD1N115DR08 Full Scale Error at 0-hours

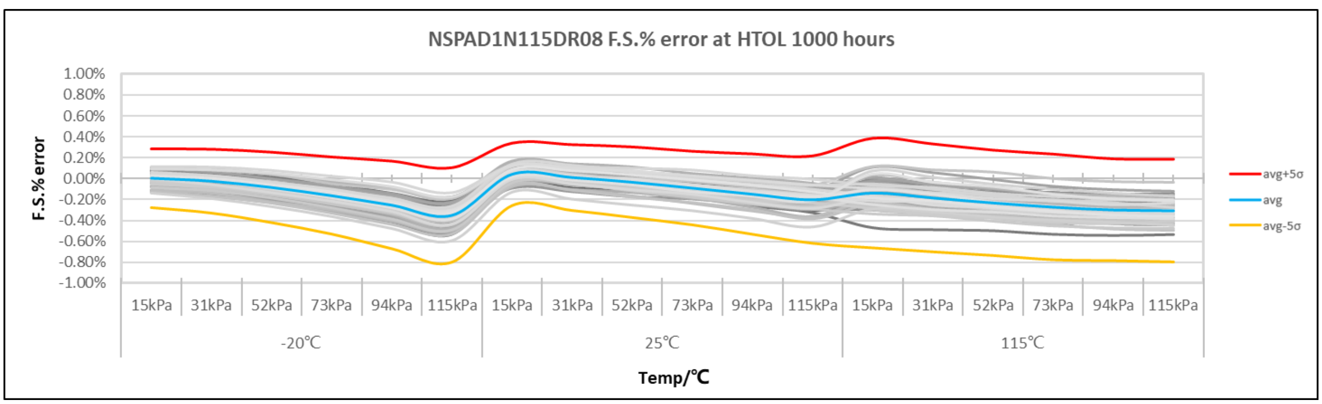


Figure 6.6 NSPAD1N115DR08 Full Scale Error at HTOL 1000-hours

6.3. Analog Output Transfer Function

NSPAD1N device is fully calibrated on delivery. The sensor has a linear transfer function between the applied pressure and the output signal:

Ratiometric: VOUT = (A * P + B) * VDD

Absolute: VOUT = (A * P + B) * 5 @VDD=5V

Note:

1) P is the pressure value, absolute pressure, range: 10kPa~400kPa; the transfer function is only established in the pressure range.

2) VDD must in the operating voltage range.

Table 6.2 NSPAD1N115RR01 Transfer Function Coefficient

Product Type	Pressure Range		Output Range		Gain and Offset	
	P_L	P_H	O_L	O_H	A	B
NSPAD1N115RR01	10kPa	115.00kPa	0.400V	4.650V	0.008095	-0.000952

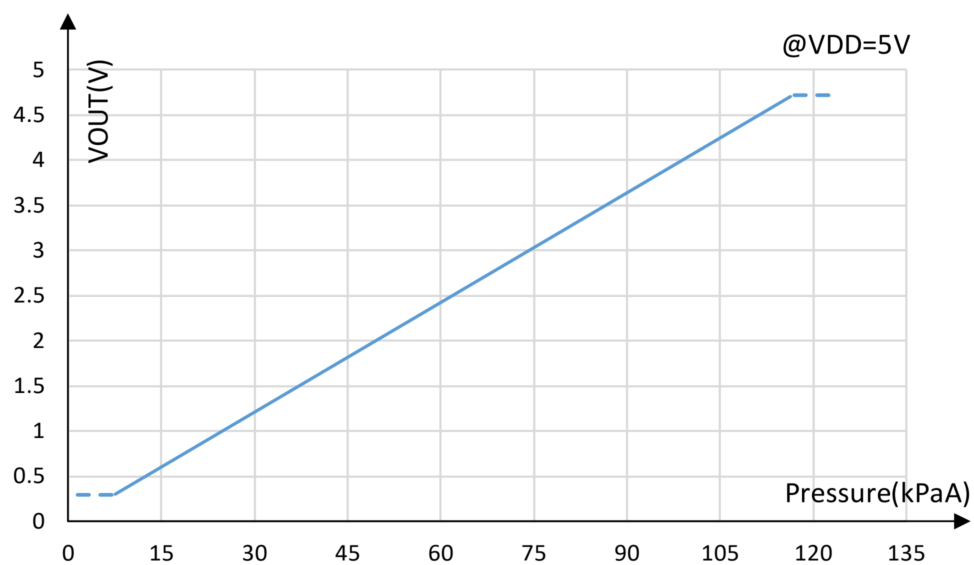


Figure 6.7 NSPAD1N115RR01 Transfer Function

6.4. Digital Output Transfer Function

P = *A* * *code*/8388607+B

Code is the register 0x06~0x08 value;

P is the pressure value, gauge pressure, unit is kPa;

Table 6.3 Digital Output Transfer Function Coefficient

Product Type	Pressure Range		Output Range		Gain and Offset	
	P_L	P_H	O_L	O_H	A	B
NSPAD1N200DR04	15kPa	200kPa	838861	7549746	231.250021	-8.125010

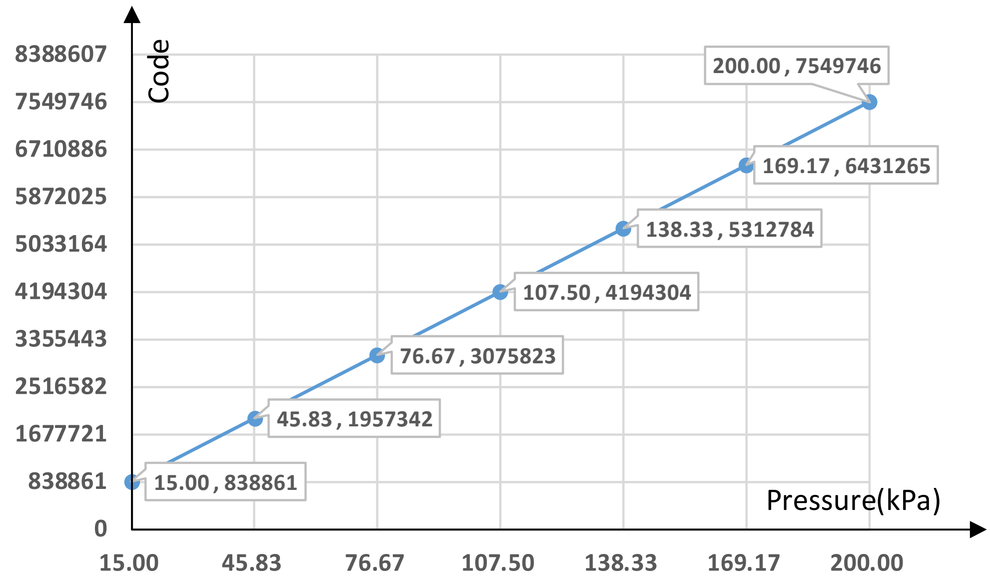


Figure 6.8 NSPAD1N200DR04 Transfer Function

6.5. Register Map

Addr	***Bit Addr***	***Description***	***Default***	***Description***
0x30	7 – 4	Reserve	4'b0000	Write with 0x0A to start a conversion, automatically come back to 0x02 after conversion ends.
	3	Sco	1'b0	
	2 – 0	Measurement_ctrl<2:0>	3'b000	
0x06	7 – 0	PDATA<23:16>	0x00	Output Pressure Data.
0x07	7 – 0	PDATA<15:8>	0x00	Code = Data0x06*2^16+ Data0x07*2^8+ Data0x08；
0x08	7 – 0	PDATA<7:0>	0x00	

For example:

If the value of the registers 0x06、0x07、0x08 are 0x3F, 0xFF, 0xFF, according to NSPAD1N200DR04 transfer function, Code = 4194303, P(kPa) = A*4194303/8388607+B, and finally get the value of pressure about 107.50kPa.

6.6. I²C Interface

I²C bus uses SCL and SDA as signal lines. Both lines are connected to VDD externally via pull-up resistors so that they are pulled high when the bus is free. The I²C device address of NSPAD1N is shown below.

Table 6.4 I²C Address

A7	A6	A5	A4	A3	A2	A1	W/R
1	1	1	1	1	1	1	0/1

The I²C interface protocol has special bus signal conditions. Start (S), stop (P) and binary data conditions are shown below. 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.

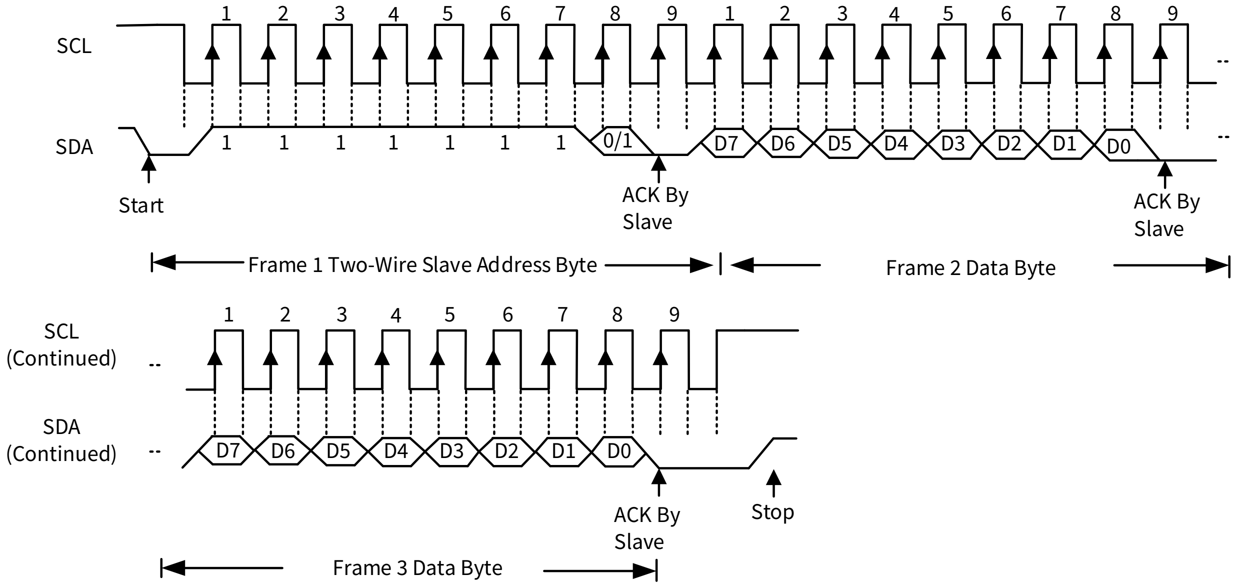


Figure 6.9 I²C Protocol

Byte Write

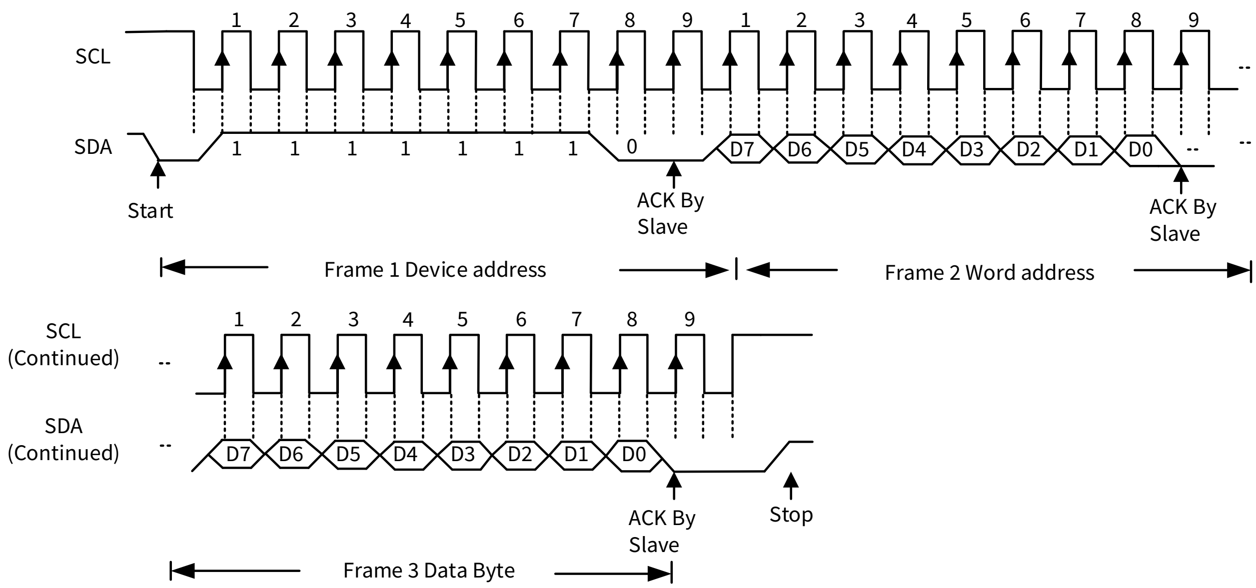


Figure 6.10 I²C Write Byte

Random Read

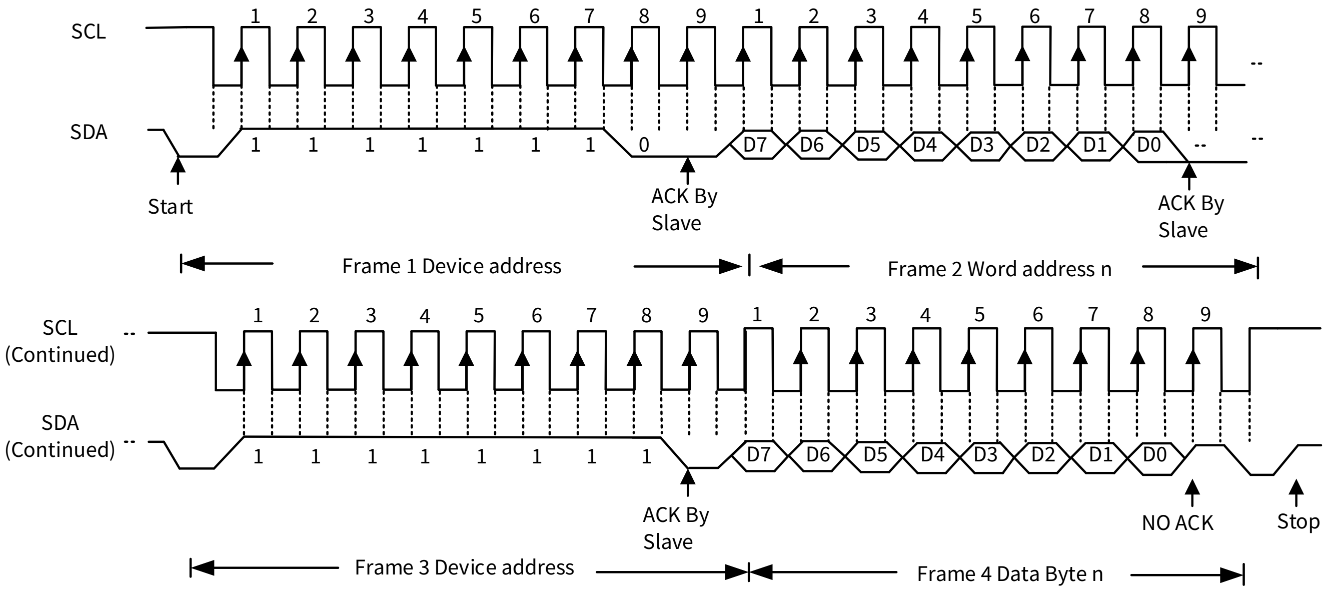


Figure 6.11 I²C Read Byte

6.7. SPI Interface

SPI clock and polarity are defined and should only be applied as: CPOL = 0, CPHA =0. (CPOL= Clock Polarity and CPHA = Clock Phase). The falling edge of CSB, in conjunction with the rising edge of SCK, determines the start of framing. Once the beginning of the frame has been determined, timing is straightforward. The first phase of the transfer is the instruction phase, which consists of 16 bits followed by data that can be of variable lengths in multiples of 8 bits. If the device is configured with CSB tied low, framing begins with the first rising edge of SCK.

The instruction phase is the first 16 bits transmitted. As shown in Figure 6.12, the instruction phase is divided into a number of bit fields.

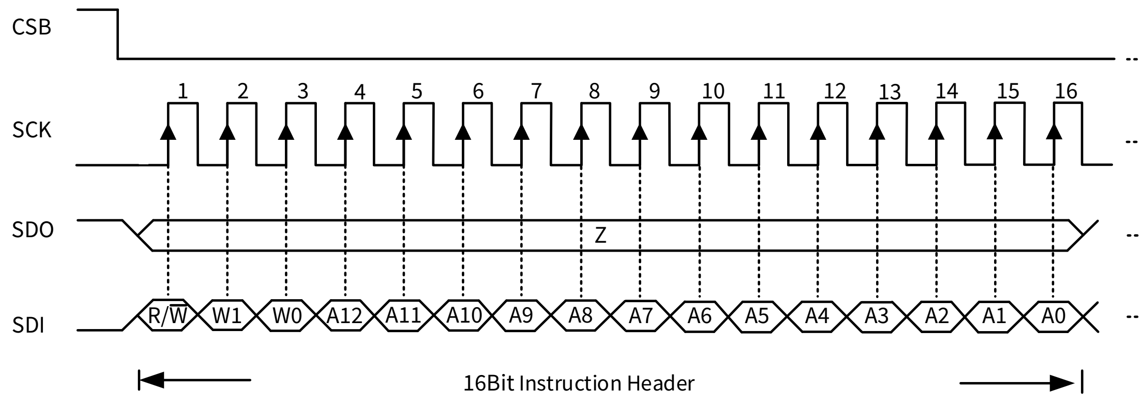


Figure 6.12 Instruction Phase Bit Field

The first bit in the stream is the read/write indicator bit (R/$\overline{W}$). When this bit is high, a read is being requested, otherwise indicates it is a write operation.

W1 and W0 represent the number of data bytes to transfer for either read or write (Table 6.5). If the number of bytes to transfer is three or less (00, 01, or 10), CSB can stall high on byte boundaries. Stalling on a nonbyte boundary terminates the communications cycle. If these bits are 11, data can be transferred until CSB transitions high. CSB is not allowed to stall during the streaming process.

The remaining 13 bits represent the starting address of the data sent. If more than one word is being sent, sequential addressing is used, starting with the one specified, and it either increments (LSB first) or decrements (MSB first) based on the mode setting.

Table6.5 W1 and W0 Settings

W1:W0	*Action*	*CSB Stalling*
00	1 byte of data can be transferred.	Optional
01	2 bytes of data can be transferred.	Optional
10	3 bytes of data can be transferred.	Optional
11	4 or more bytes of data can be transferred. CSB must be held low for entire sequence; otherwise, the cycle is	No

Data follows the instruction phase. The amount of data sent is determined by the word length (Bit W0 and Bit W1). This can be one or more bytes of data. All data is composed of 8-bit words.

Data can be sent in either MSB-first mode or LSB-first mode (by setting ‘LSB_first’ bit). On power up, MSB-first mode is the default. This can be changed by programming the configuration register. In MSB-first mode, the serial exchange starts with the highest-order bit and ends with the LSB. In LSB-first mode, the order is reversed. (Figure 6.13)

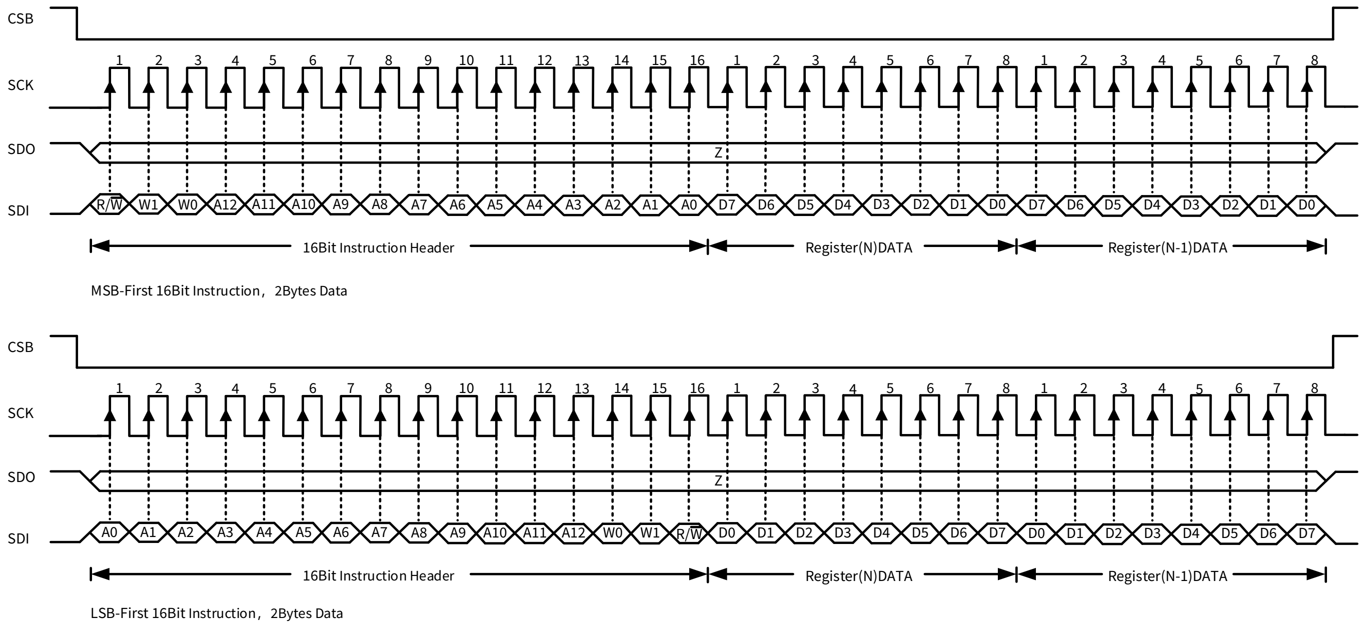


Figure 6.13 MSB First and LSB First Instruction and Data Phases

Byte Write

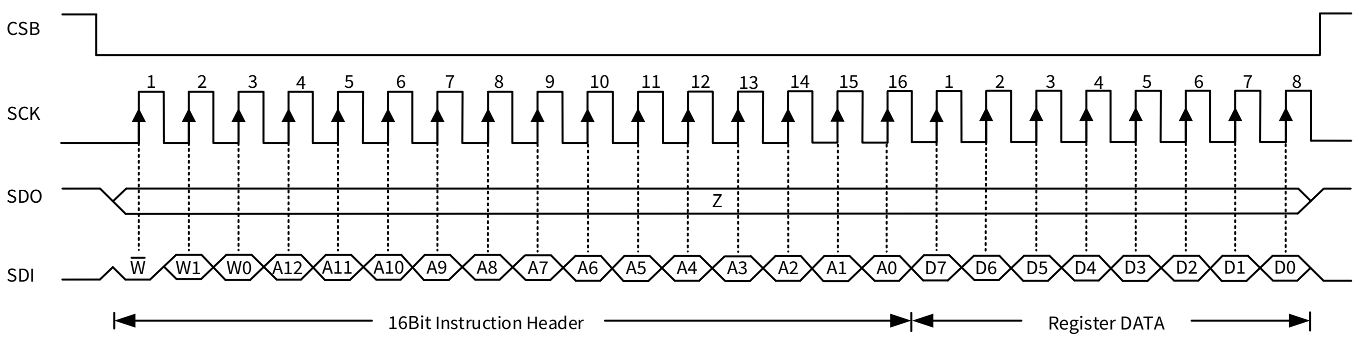


Figure 6.14 SPI Write Byte

Byte Read

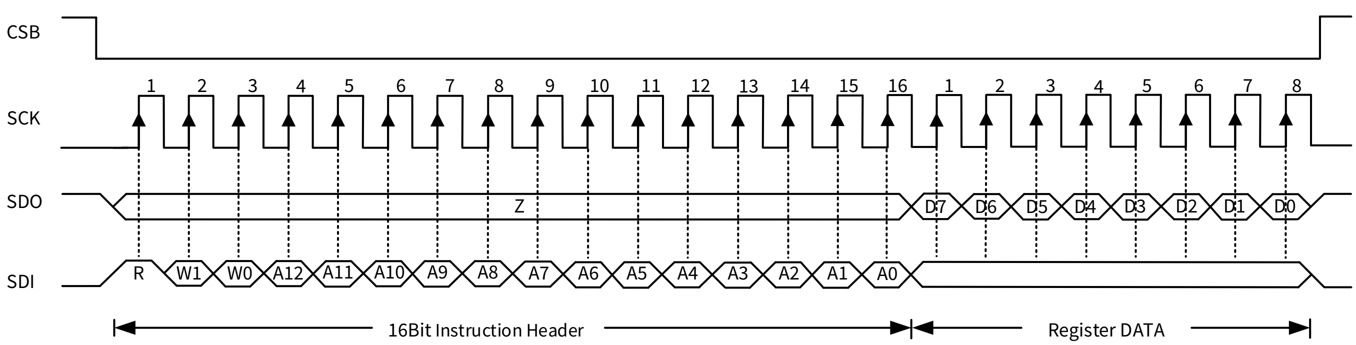


Figure 6.15 SPI Read Byte

7. Typical Application

7.1. Application Circuit

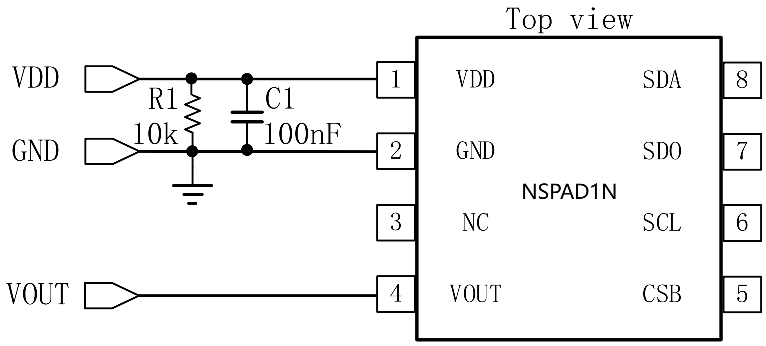


Figure 7.1 Analog Output Application Circuit

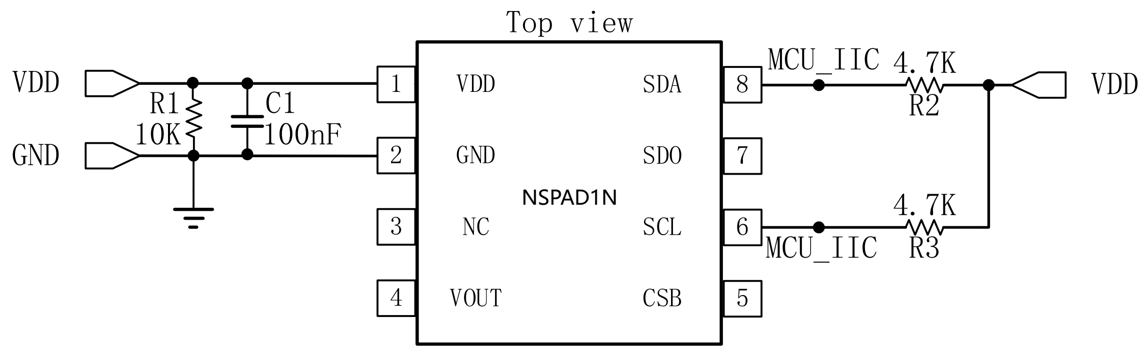


Figure 7.2 I^2C Output Application Circuit

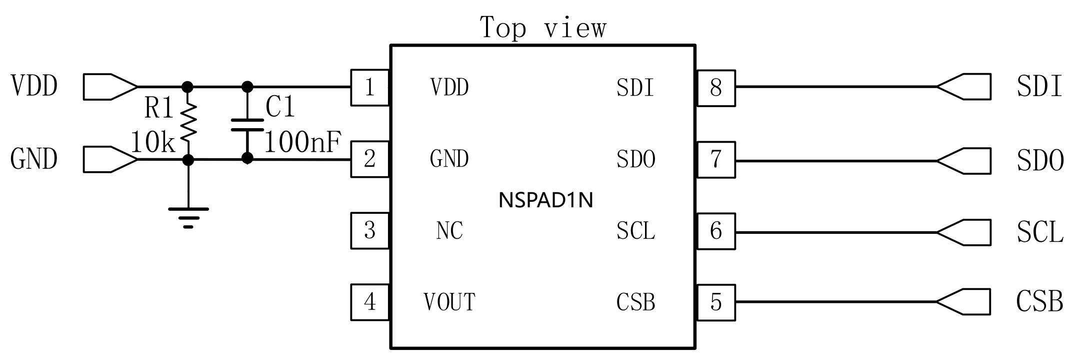


Figure 7.3 SPI Output Application Circuit

Note:
1)For applications with higher ESD requirements, can add TVS between VOUT and GND and between VDD and GND.
2)Please contact NOVOSENSE for detailed peripheral recommended circuit.

7.2. Recommended Footprint

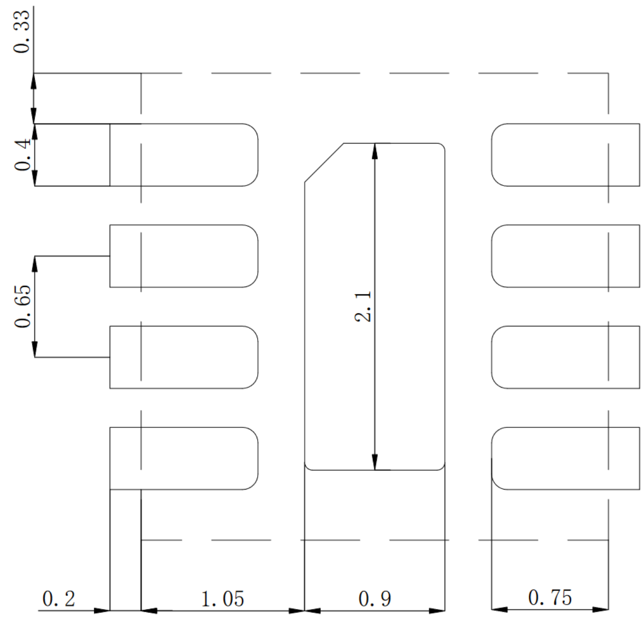


Figure 7.4 Footprint (unit: mm)

7.3. Soldering Parameters

Table 7.1 Soldering Parameters

Reflow Condition		***Lead–free Assembly***
Pre Heat	Temperature Min (Ts(min))	150°C
	Temperature Max (Ts(max)	200°C
	Time (min to max) (ts)	60 – 180 secs
Average ramp up rate (Liquidus Temp (T_L) to peak		3°C/second max
T_S(max)to T_L - Ramp-up Rate		3°C/second max
Reflow	Temperature (T_L) (Liquidus)	217°C
	Time (min to max) (t_L)	60 – 150 seconds
Peak Temperature (T_P)		260°C
Time within 5°C of actual peak Temperature (tp)		20 – 40 seconds
Ramp-down Rate		6°C/second max
Time 25°C to peak Temperature (T_P)		8 minutes Max.
Do not exceed		260°C

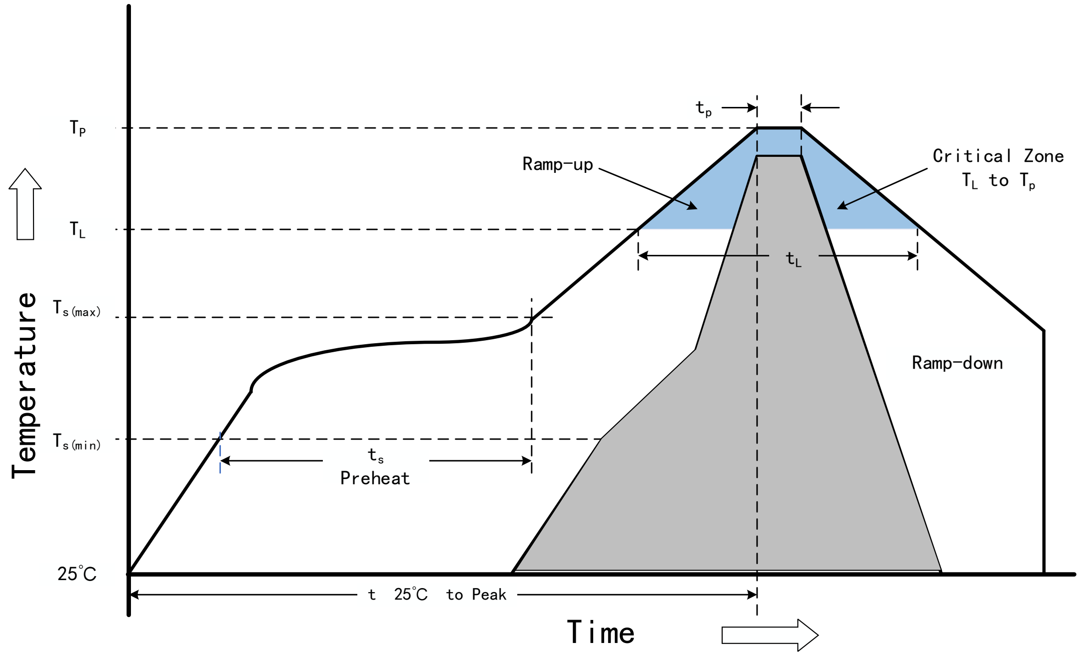
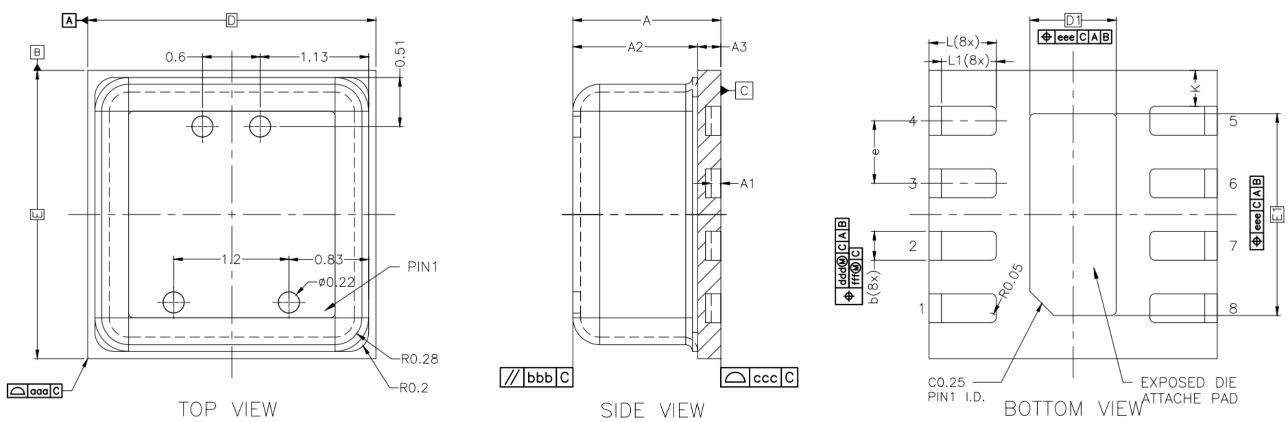
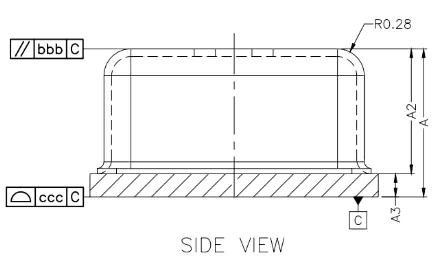


Figure 7.5 Soldering Profile

8. Package Information





DESCRIPTION		SYMBOL	MILLIMETER MIN	MILLIMETER NOM	MILLIMETER MAX
TOTAL THICKNESS		A	1.475	1.54	1.635
WETTABLE FLANK HEIGHT		A1	0.10	---	---
METAL LID HEIGHT		A2	1.25	1.30	1.35
L/F THICKNESS		A3	0.24 REF		
BODY SIZE	X	D	2.95	3.00	3.05
	Y	E	2.95	3.00	3.05
LEAD PITCH		e	0.65 BSC		
LEAD WIDTH		b	0.25	0.30	0.35
LEAD LENGTH		L	0.65	0.70	0.75
		L1	0.52	0.57	0.62
EP SIZE		D1	0.85	0.90	0.95
		E1	2.05	2.10	2.15
LEAD EDGE TO PKG EDGE		K	0.375		
Tolerance of form and position					
GEOMETRY TOLERANCE		aaa	0.05		
METAL LID FLATNESS		bbb	0.10		
COPLANARITY		ccc	0.08		
LEAD OFFSET		ddd	0.10		
LEAD OFFSET		fff	0.05		
EXPOSED PAD OFFSET		eee	0.10		

NOTES
1.0 COPLANARITY APPLIES TO LEADS, CORNER LEADS AND DIE ATTACH PAD.

Figure 8.1 Package Outline (unit: mm)

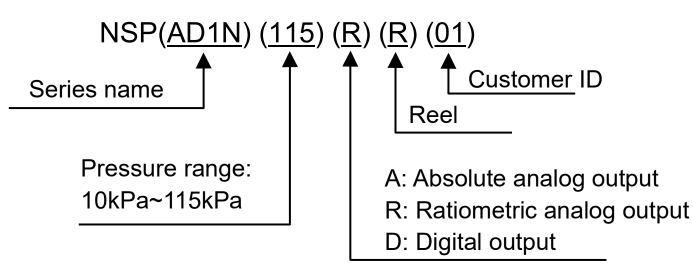
9. Ordering Information

Product No.	Output Type	Pressure Range		Output Range		Clamp Level		Gain and Offset		Supply Voltage	Accuracy (-20~115℃)
		P_L	P_H	O_L	A	V_{CL}	V_{CH}	A	B		
NSPAD1N115RR01	Ratiometric	10.00kPa	115.00kPa	0.400V	4.650V	0.30V	4.70V	0.008095	-0.000952	5.00V	±1.5%
NSPAD1N200RR02	Ratiometric	15.00kPa	200.00kPa	0.400V	4.650V	NA	NA	0.004595	0.011081	5.00V	±1.5%
NSPAD1N115DR03	IIC/SPI	40.00kPa	115.00kPa	838861	7549746	NA	NA	93.750008	30.624996	3.30V	±1.0%
NSPAD1N200DR04	IIC/SPI	15.00kPa	200.00kPa	838861	7549746	NA	NA	231.250021	-8.125010	5.00V	±1.0%
NSPAD1N165DR05	IIC/SPI	60.00kPa	165.00kPa	838861	7549746	NA	NA	131.250012	46.874994	5.00V	±1.0%
NSPAD1N250RR06	Ratiometric	10.00kPa	250.00kPa	0.400V	4.650V	0.30V	4.70V	0.003542	0.044583	5.00V	±1.5%
NSPAD1N400DR07	IIC/SPI	50.00kPa	400.00kPa	838861	7549746	NA	NA	437.500039	6.249980	5.00V	±1.0%
NSPAD1N115DR08	IIC/SPI	10.00kPa	115.00kPa	838861	7549746	NA	NA	131.250012	-3.125006	5.00V	±1.0%

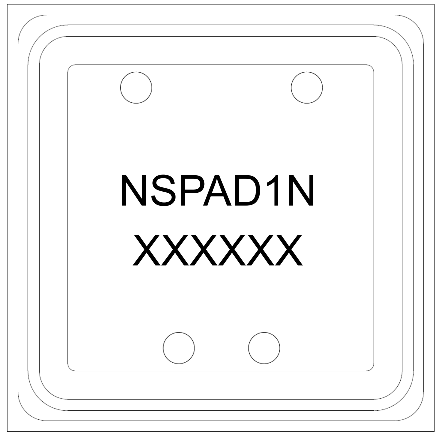
Please scan the following QR code or visit the download link for complete part number list.

https://www.novosns.com/Public/Uploads/uploadfile/files/20250626/NSPAD1N.pdf

Naming Rule:



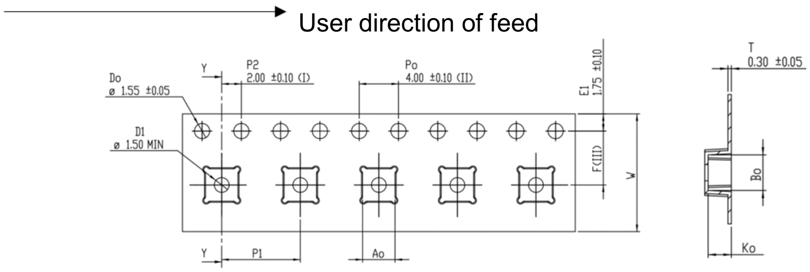
10. Identification Code



NSPAD1N: series name.

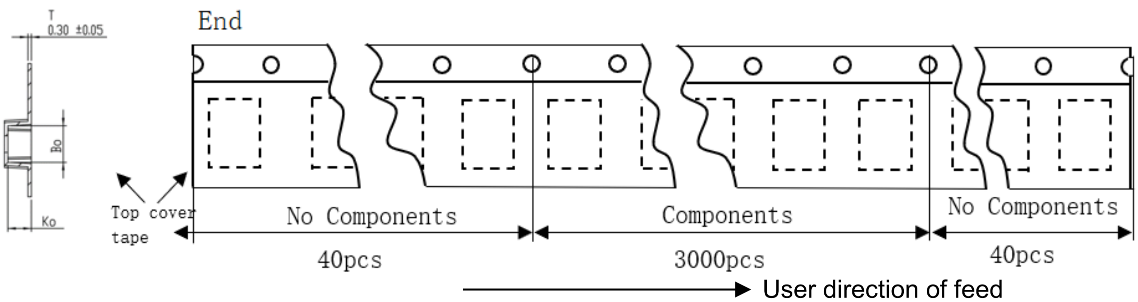
XXXXXX: assembly lot number.

11. Tape/Reel Information

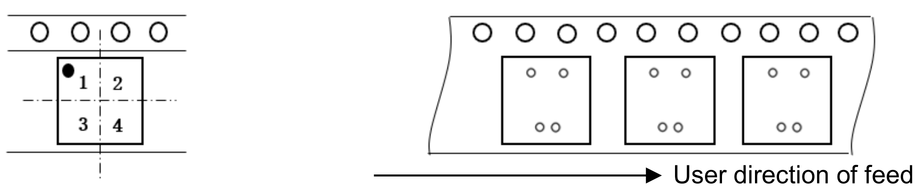


Series	Type	E1 (mm)	F (mm)	P2 (mm)	D0 (mm)	D1 (mm)	P0 (mm)	W (mm)	P1 (mm)	A0 (mm)	B0 (mm)	K0 (mm)	T (mm)
NSPAD1N	DFN8	1.75±0.10	5.5±0.10	2.0±0.10	1.55+0.05	1.5Min	4.0±0.1	12.0±0.30	8.0±0.1	3.30±0.1	3.3±0.10	2.10±0.10	0.30±0.05

There is no component at the head and the tail of each tape/reel, where the space is 40pcs, as shown in the following figure.



Pin1 is located at the first quadrant, as shown in the following figure.



Standard pack quantity(SPQ): 3000EA.

12. Revision History

Revision	Description	Date
1.0	Formal release.	2025/7/23

Notes：

1. I^2C code

```
#define ACK       1
#define NACK    0
uchar REG06=0,REG07=0,REG08=0;
uchar number=1;
uchar Reg30[1];
int PCode=0, Pdata=0;
float Pressure=0.0;
void IIC_Start(void)              //Start the IIC, SDA High-to-low when SCL is high
{
      IIC_SCL(1);                 //SCL output high level
      SDA_OUT(1);                 //SDA output high level
      Delay_us(2);                //Delay 2us
      SDA_OUT(0);                 //SDA output low level
      Delay_us(2);
}

void IIC_Stop(void)               //Stop the IIC, SDA Low-to-high when SCL is high
{
      IIC_SCL(0);
      Delay_us(2);
      IIC_SCL(1);
      SDA_OUT(0);
      Delay_us(2);
      SDA_OUT(1);
      Delay_us(2);
}

void IIC_ACK(void)                //Send ACK (LOW)
{
      SDA_OUT(0);
      IIC_SCL(1);
      Delay_us(2);
      IIC_SCL(0);
}

void IIC_NACK(void)               //Send No ACK (High)
{
      SDA_OUT(1);
      IIC_SCL(1);
      Delay_us(2);
      IIC_SCL(0);
}


uchar IIC_Wait_ACK(void)          //Check ACK, if return 0, then right, if return 1, then error
{
      int ErrTime=0;
      SDA_IN();                          //SDA set as input
      IIC_SCL(1);
      Delay_us(2);
```



```
    while(Read_SDA)
    {
        ErrTime++;
        if(ErrTime>200)
        {
            IIC_Stop();
            return 1;
        }
    }
    IIC_SCL(0);
    SDA_OUT(0);
    Delay_us(2);
    return 0;
}

void IIC_Send(uchar IIC_Data)                //Send a byte to IIC
{
    uchar i;
    IIC_SCL(0);
    Delay_us(2);
    for(i=0;i<8;i++)
    {
        if((IIC_Data&0x80)>>7)
            SDA_OUT(1);
        else
            SDA_OUT(0);
        IIC_Data<<=1;
        IIC_SCL(1);
        Delay_us(2);
        IIC_SCL(0);
        Delay_us(2);
    }
}

uchar IIC_Receive(uchar ACK)                 //Receive a byte from I2C
{
    uchar i,Receive_Data=0;
    SDA_IN();
    for(i=0;i<8;i++)
    {
        IIC_SCL(0);
        Delay_us(2);
        IIC_SCL(1);
        Receive_Data<<=1;
        if(Read_SDA==1)
            Receive_Data++;
        Delay_us(2);
    }
    IIC_SCL(0);
    Delay_us(2);
    if(ACK==0x01)
        IIC_ACK();
    else
        IIC_NACK();
```

```
    return Receive_Data;
}

void NSPAD1N200DR04_Write_Byte(uchar WriteAddr,uchar WriteData)
{
    IIC_Start();
    IIC_Send(0xFE|0x00);
    IIC_Wait_ACK();
    IIC_Send(WriteAddr);
    IIC_Wait_ACK();
    IIC_Send(WriteData);
    IIC_Wait_ACK();
    IIC_Stop();
}

void NSPAD1N200DR04_Read_Byte(uchar ReadAddr, uchar *pBuffer)
{
    IIC_Start();
    IIC_Send(0xFE|0x00);
    IIC_Wait_ACK();
    IIC_Send(ReadAddr);
    IIC_Wait_ACK();
    IIC_Start();
    IIC_Send(0xFE|0x01);
    IIC_Wait_ACK();
    pBuffer[0]=IIC_Receive(0);
    IIC_Stop();
}

void NSPAD1N200DR04_Read_3Byte(uchar ReadAddr,uchar *pBuffer)
{
    IIC_Start();
    IIC_Send(0xFE|0x00);
    IIC_Wait_ACK();
    IIC_Send(ReadAddr);
    IIC_Wait_ACK();
    IIC_Start();
    IIC_Send(0xFE|0x01);
    IIC_Wait_ACK();
    pBuffer[0]=IIC_Receive(ACK);
    pBuffer[1]=IIC_Receive(ACK);
    pBuffer[2]=IIC_Receive(NACK);
    IIC_Stop();
}

void main()
{
        uchar PData[3]={0,0,0};
        while(1)
        {
                NSPAD1N200DR04_Write_Byte(0x30,0x0A);
                while(1)                                        //Check whether the conversion ends
                {
                        if(number<=50)
```

```
                {
                        number++;
                        delay_ms(1);
                        NSPAD1N200DR04_Read_Byte(0x30,Reg30);
                        if(0x02==Reg30[0])
                        {
                                number=1;
                                break;
                        }
                }
                if(number>50)
                {
                        number=1;
                        //User can add his own error handler function
                        break;
                }
        }
        NSPAD1N200DR04_Read_3Byte(0x06,PData);
        REG06 = PData [0];                              //Register 0x06
        REG07 = PData [1];                              //Register 0x07
        REG08 = PData [2];                              //Register 0x08
        PCode=(REG06*65536+REG07*256+REG08);  //PCode = Data0x06*2^16+ Data0x07*2^8+ Data0x08
        if (PCode >8388607)
                Pdata= PCode-16777216;                          //Symbol processing
        else
                Pdata= PCode;
        Pressure = ((float)Pdata/8388607*231.250)+(-8.125);     //P=A*PCode/8388607+B
                                                        //A=231.250021, B=-8.125010
    }
}
```

2. SPI code

```
u8 REG06=0,REG07=0,REG08=0;
u8 number=1;
u8 PData[3]={0};
u32 PCode=0,Pdata=0;
float Pressure=0.0;
void NSPAD1N200DR04_SPI_Init(void)
{
	SPI_PORT_GPIO_Config();
	CSB(1);
	SCLK(0);
	SDI(1);
}

void NSPAD1N200DR04_SPI_Write_OneByte(u8 addr,u8 val)
{
	u8 i=0; u16 dat;

	dat=0x0000+addr;
	CSB(0);
	delay_us(2);

	for(i=0;i<16;i++)
	{
		SCLK(0);
		if(dat&0x8000)
			SDI(1);
		else
			SDI(0);
		delay_us(2);
		dat<<=1;
		SCLK(1);
		delay_us(2);
	}

	for(i=0;i<8;i++)
	{
		SCLK(0);
		if(val&0x80)
			SDI(1);
		else
			SDI(0);
		delay_us(2);
		val<<=1;
		SCLK(1);
		delay_us(2);
	}

	SCLK(0);
	CSB(1);
	delay_us(2);

}

u8 NSPAD1N200DR04_SPI_Read_OneByte(u8 addr)
{
	u8 i=0;   u16 dat; u8 val=0;

	dat=0x8000+addr;
	CSB(0);
```

```
	delay_us(2);

	for(i=0;i<16;i++)
	{
		SCLK(0);
		if(dat&0x8000)
		{
			SDI(1);
		}
		else
		{
			SDI(0);
		}
		delay_us(2);
		dat <<= 1;
		SCLK(1);
		delay_us(2);
	}

	for(i=0;i<8;i++)
	{
		SCLK(0);
		val<<=1;
		if(SPI_MISO)
			val++;
		delay_us(2);
		SCLK(1);
		delay_us(2);
	}

	SCLK(0);
	CSB(1);
	delay_us(2);

	return val;
}

void NSPAD1N200DR04_SPI_Read_3Byte(u8 addr,u8* pBuffer)
{
	u8 i=0;   u16 dat; u8 val_1=0,val_2=0,val_3=0;

	dat=0xC000+addr;
	CSB(0);
	delay_us(2);

	for(i=0;i<16;i++)
	{
		SCLK(0);
		if(dat&0x8000)
		{
			SDI(1);
		}
		else
		{
			SDI(0);
		}
		delay_us(2);
		dat <<= 1;
		SCLK(1);
		delay_us(2);
	}
```

```
        for(i=0;i<8;i++)
        {
                SCLK(0);
                val_1<<=1;
                if(SPI_MISO) val_1++;
                delay_us(2);
                SCLK(1);
                delay_us(2);
        }

        for(i=0;i<8;i++)
        {
                SCLK(0);
                val_2<<=1;
                if(SPI_MISO) val_2++;
                delay_us(2);
                SCLK(1);
                delay_us(2);
        }

        for(i=0;i<8;i++)
        {
                SCLK(0);
                val_3<<=1;
                if(SPI_MISO) val_3++;
                delay_us(2);
                SCLK(1);
                delay_us(2);
        }
        pBuffer[0]=val_1;
        pBuffer[1]=val_2;
        pBuffer[2]=val_3;

        SCLK(0);
        CSB(1);
        delay_us(2);
}

void NSPAD1N200DR04_SPI_Read_MultiByte(u8 addr,u8 len,u8 *pBuffer)
{
        u8 i=0,k=0,val=0; u16 dat;

        dat=0xE000+addr;
        CSB(0);
        delay_us(2);

        for(i=0;i<16;i++)
        {
                SCLK(0);
                if(dat&0x8000)
                {
                        SDI(1);
                }
                else
                {
                        SDI(0);
                }
                delay_us(2);
                dat <<= 1;
                SCLK(1);
                delay_us(2);
        }
```

```
                for(k=0;k<len;k++)
        {
                for(i=0;i<8;i++)
                {
                        SCLK(0);
                        val<<=1;
                        if(SPI_MISO)val++;
                        delay_us(2);
                        SCLK(1);
                        delay_us(2);
                }
                pBuffer[k]=val;

        }

        SCLK(0);
        CSB(1);
        delay_us(2);
}
int main(void)
{
        NSPAD1N200DR04_SPI_Init();
        delay_ms(100);
        NSPAD1N200DR04_SPI_Write_OneByte(0x00,0x81);

        while(1)
        {
                NSPAD1N200DR04_SPI_Write_OneByte(0x30,0x0A);
                while(1)                                        //Check whether the conversion ends
                {
                        if(number<=50)
                        {
                                number++;
                                delay_ms(1);
                                if(0x02== NSPAD1N200DR04_SPI_Read_OneByte(0x30))
                                {
                                        number=1;
                                        break;
                                }
                        }
                        if(number>50)
                        {
                                number=1;
                                break;
                        }
                }
        }

        NSPAD1N200DR04_SPI_Read_3Byte(0x08,PData);
        REG08=PData[0];                                                 //Register 0x08
        REG07=PData[1];                                                 //Register 0x07
        REG06=PData[2];                                                 //Register 0x06
        PCode=(REG06*65536+REG07*256+REG08);             //PCode = Data0x06*2^16+ Data0x07*2^8+ Data0x08
        if (PCode>8388607)
                Pdata=PCode-16777215;                           //Symbol processing
        else Pdata=PCode;
        Pressure = ((float)Pdata/8388607*231.250)+(-8.125);       //P=A*PCode/8388607+B
                                                                        //A=231.250021, B=-8.125010

        }
    }
```

IMPORTANT NOTICE

For further information on applications, products and technologies, please contact NOVOSENSE (www.novosns.com).

Suzhou NOVOSENSE Microelectronics Co., Ltd