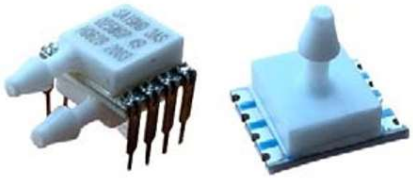
LW54 系列 压力传感器

硅陶瓷系列

数字输出



应用领域

- 呼吸机、麻醉机
- 雾化器
- HVAC 滤清器堵塞检测
- 肺活量计
- 风道静压
- HVAC（暖通空调）变送器
- 医院室内气压控制
- VAV 调节系统

产品概述

LW54 高精度硅陶瓷系列为压阻硅压力传感器，可提供指定满量程压力范围和温度范围读取压力的数字输出。LW54 系列通过使用板载专用集成电路 (ASIC) 针对传感器偏移、 灵敏度、温度效应和非线性进行了充分校准和温度补偿。经校准的压力输出值会在 1 kHz 左右更新。LW54 系列在 0~60°C 的温度范围内进行校准，也可以在-40~85℃间按客户指定温度校准。该传感器可在 1.8~3.3 Vdc 单电源条件下工作。这些传感器测量绝压、差压和表压。绝压型号的传感器具备内部真空参照以及与绝压成比例的输出值。差压型号的传感器允许向感应模片的任意一侧加压。表压型号的传感器以大气压力为参考，提供与大气压力变化成比例的输出值。LW54 压力传感器适用于无腐蚀性、非离子气体 (例如空气和其他干燥气体)。提供的选件可延伸这些传感器的性能，使其适用于无腐蚀性、非离子的液体或经特殊工艺处理后 pH 值在 5.5~10 之间的液体。 所有产品均遵循 ISO 9001 标准设计和制造。

产品特点

- 多样化的封装：LW54 系列压力传感器为硅压阻精密压力传感器，采用模块化设计，具有 SMT 封装类型
- 小型尺寸：10mm * 10mm 微型封装
- 工作电压较低，能耗极小，供电电压可以为 1.8~3.3V 输入
- 业界领先的长期稳定性：通过压力敏感芯片的优选和封装工艺的技术处理，作为微压力传感器与业内其他传感器相比表现出色，具有优异的长期稳定性
- 达到 0.25% FSS BFSL（满刻度量程最佳直线）的极高精确度
- 总误差带为 1% 的满刻度量程最大值（1~3% 不同量程偏差不同）
- 所有这些产品都同样具备业界领先的性能规格
- 与 I2C 兼容的高达 24 位数字输出，压力输出 24 位，可以同时监控温度，温度输出 24 位
- 数字输出提供 10%~90% 输出，或经客户指定，可以在 5%~95% 输出
- 标准产品在 0~60℃ 温度范围内进行精密 ASIC 调节和温度补偿，可在-40~85℃温度补偿范围定制
- 符合 RoHS 标准
- 绝压、差压和表压类型
- 压力口特点：直径 3.175MM 倒钩状压力口可以稳固的和 2.38MM 内径的压力管牢固连接测试压力
- 客户定制：精度、总偏差和温度补偿范围以及信号输出方式等可根据客户需求定制，非标准产品请联系我们

标准压力范围（PSI，KPA）

2 英寸水柱	表压、差压	数字输出
5 英寸水柱	表压、差压	数字输出
10 英寸水柱	表压、差压	数字输出
1 PSI	表压、差压	数字输出
2 PSI	表压、差压	数字输出
5 PSI	表压、差压	数字输出
15 PSI	表压、差压、绝压	数字输出
30 PSI	表压、差压、绝压	数字输出

2.5 mbar	表压、差压	数字输出
5 mbar	表压、差压	数字输出
10 mbar	表压、差压	数字输出
50 mbar	表压、差压	数字输出
5 Kpa	表压、差压	数字输出
10 Kpa	表压、差压	数字输出
35 Kpa	表压、差压	数字输出
100 Kpa	表压、差压、绝压	数字输出
200 Kpa	表压、差压、绝压	数字输出

额定值

参数	最小值	最大值	单位
电源电压 (Vsupply)	-0.4	3.6	VDC
任意引脚上的电压	-0.3	Vsupply +0.3	V
数字接口时钟频率：I2C SPI	1	3.4 20	MHz
ESD 敏感度 (人体模式)		4	Kv
储存温度	-40	125	℃
焊接时间和温度 铅焊料温度 (SIP、DIP) 回流峰值温度 (SMT)	最多 5 秒，在 250℃ 时 最多 15 秒，在 250℃ 时		

性能参数

参数	最小值	典型值	最大值	单位	注释
电源电压 (Vsupply) (根据所选料号) 3.3	1.68	3.3	3.6	Vdc	2
电源电流 3.3 Vdc 电源		3.0		mA	
补偿温度范围	0		60	℃	3
操作温度范围	-40		125	℃	4
启动时间(从加电到数据准备就绪)		2.5	4	ms	
响应时间		1		ms	
SPI/I2C 低电平			0.2	Vsupply	
SPI/I2C 高电平	0.8			Vsupply	
SDA/MISO, SCL/SCLK, SS 上拉电阻	1			Kohm	
精度			±0.25	% FSS	5、7
位置灵敏度			±0.15	% FSS	
综合偏差 (TEB)	-1%		1%	% FSS	6
过载压力		>3		倍	
爆破压力		>5		倍	
输出分辨率	24			位	

环境规格

参数	特性	注释
湿度：仅干燥气体	0% 到 95% RH，非冷凝	
振动	MIL-STD-202F，方法 214，条件 F (20.7g 随机)	
冲击	MIL-STD-202F，方法 213B，条件 F	
寿命	100 万次循环	8
回流焊	J-STD-020D.1 MSL 湿度灵敏度级别 1	

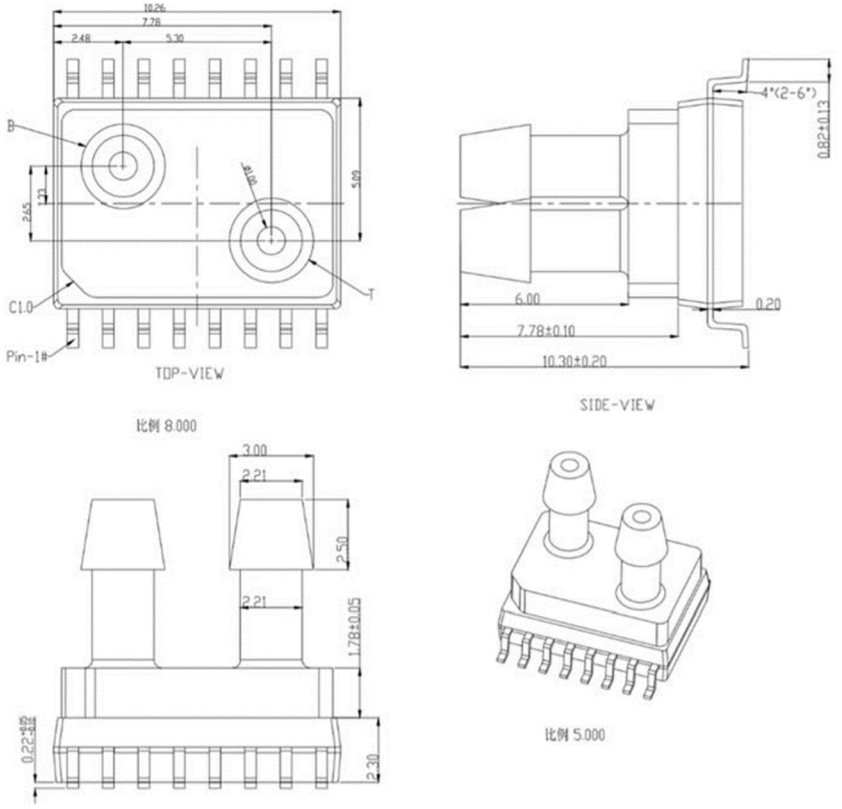
压力类型

压力类型	说明
差压	输出与施加到各压力口的压力差成比例
表压	输出与施加到压力和大气（环境）压力之间 0 psiG 的差值成比例
绝压	输出与施加压力和真空绝对零压 0 psiA 之间的差值成比例
复合压	输出与施加压力和-15 psiG 压力之间的差值成比例

脚位定义

输出类型/脚位	Pin1-3	Pin 4	Pin 5	Pin 6	Pin8-9	Pin 10	Pin 11	Pin 12	Pin13-16
数字输出(I2C)	Blank	SDA	SCL	Blank	Blank	VSS	VDD	Blank	Blank

尺寸[英寸（毫米）]



备注:

1、差压时，当压力口 A 的压力大于压力口 B 时，输出大于16777215 的 50%；
当压力口 A 的压力等于压力口 B 时，输出等于 16777215 的 50% 。

2、表压时，当压力口 A 的压力大于压力口 B 时，对于 A 型，输出大于16777215 的 10%；
对于 B 型，输出等于16777215 的 10% 。

3、压力口 A 在调试的时候始终加的是正压。

订购指南

LW54 D-DS 3 A I XXXX X S X

Model
LW54

Output Type
D = Digital

Package Style
DS = Dual Side Port

Supply Voltage
3 = 3.3 Vdc
5 = 5.0 Vdc

Output Type
A = 10% to 90%
B = 5% to 95%

Interface Type
I = I2C (Addr. 0x28H)
J = I2C (Addr. 0x36H)
K = I2C (Addr. 0x46H)
S = SPI (Not Available for 'L' Pin Style)
0 = I2C (Addr. Ox48H)
.....
9 = I2C (Addr. Ox51H)

Pressure Range
002 M = 2 Mpa
004 W = 4 InH2O
005 K = 5 Kpa
010 P = 10 Psi
020 B = 20 Bar
100 A = 100 Pascal

Pressure Type
D = Differential
G = Gauge
A = Absolute

Pin Style
S = SMT

Humidity Protection
C = Gel Coating
Black

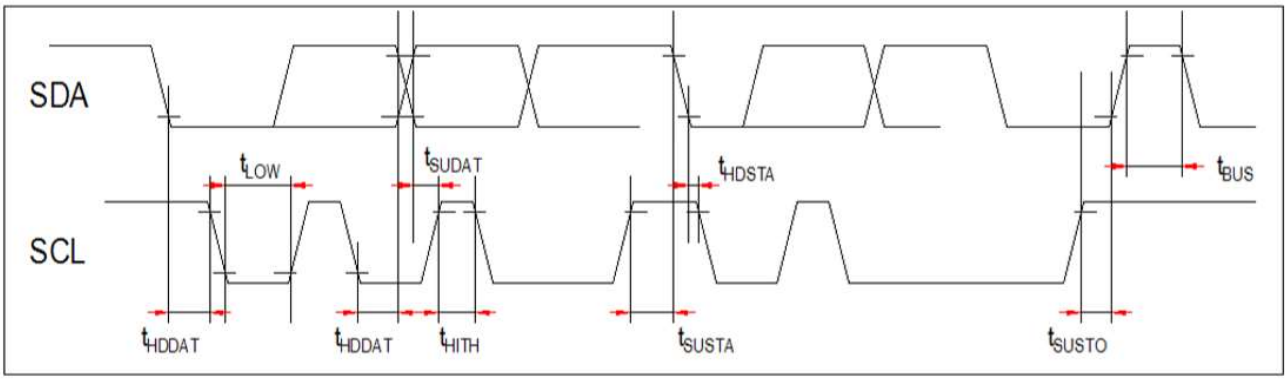
零壹智控 INTELLIGENT

零壹智能控制技术（深圳）有限公司
www.linkwon.com.cn
Email：sales1@bill-well.com
深圳市 南山区 南光路 17号 现代城华庭 4栋25A
Tel：0755-2664 7945 / 189 2389 8109　　Fax：0755-2641 9680

I2C SPI INTERFACE PARAMETERS & TIMING DIAGRAM

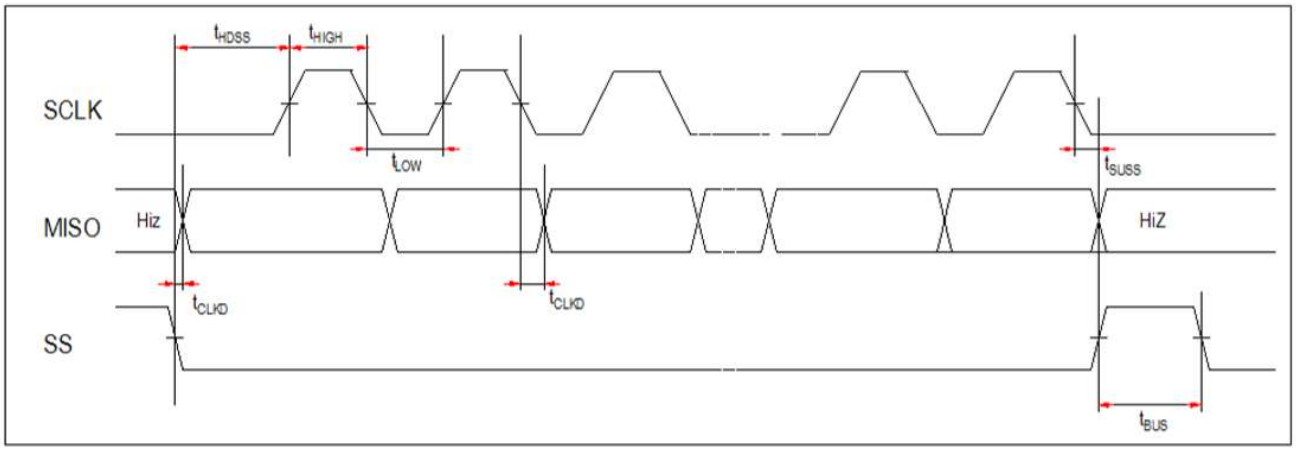
I2C INTERFACE PARAMETERS					
PARAMETERS	SYMBOL	MIN	TYP	MAX	UNITS
SCLK CLOCK FREQUENCY	FSCL	100		400	KHz
START CONDITION HOLD TIME RELATIVE TO SCL EDGE	tHDSTA	0.1			uS
MINIMUM SCL CLOCK LOW WIDTH @1	tLOW	0.6			uS
MINIMUM SCL CLOCK HIGH WIDTH @1	tHIGH	0.6			uS
START CONDITION SETUP TIME RELATIVE TO SCL EDGE	tSUSTA	0.1			uS
DATA HOLD TIME ON SDA RELATIVE TO SCL EDGE	tHDDAT	0			uS
DATA SETUP TIME ON SDA RELATIVE TO SCL EDGE	tSUDAT	0.1			uS
STOP CONDITION SETUP TIME ON SCL	tSUSTO	0.1			uS
BUS FREE TIME BETWEEN STOP AND START CONDITION	tBUS	2			uS
@1 COMBINED LOW AND HIGH WIDTHS MUST EQUAL OR EXCEED MINIMUM SCL PERIOD.					

I2C INTERFACE TIMING DIAGRAM



SPI INTERFACE PARAMETERS					
PARAMETERS	SYMBOL	MIN	TYP	MAX	UNITS
SCLK CLOCK FREQUENCY	FSCL	50		800	KHz
SS DROP TO FIRST CLOCK EDGE	tHDSS	2.5			uS
MINIMUM SCL CLOCK LOW WIDTH @1	tLOW	0.6			uS
MINIMUM SCL CLOCK HIGH WIDTH @1	tHIGH	0.6			uS
CLOCK EDGE TO DATA TRANSITION	tCLKD	0		0.1	uS
RISE OF SS RELATIVE TO LAST CLOCK EDGE	tSUSS	0.1			uS
BUS FREE TIME BETWEEN RISE AND FALL OF SS	tBUS	2			uS
@1 COMBINED LOW AND HIGH WIDTHS MUST EQUAL OR EXCEED MINIMUM SCLK PERIOD.					

SPI INTERFACE TIMING DIAGRAM



ADC Conversion Times for a Single Analog-to-Digital Conversion

Resolution (Bits)	Conversion Time in µs (typical)
12	140
13	185
14	250
15	335
16	470
17	640
18	890
19	1250
20	1760
21	2460
22	3480
23	4890
24	6940

I2C/SPI 例程

I2C （LW54D-DS-001DP）

```
#include "I2C.h"
#include "main.h"
#include "stm32l0xx_hal.h"

float Pdisplay=0;
u32 Tdisplay=0;
float Pmax=6894.757;
float Pmin=-6894.757;
float Pspan=13421772.8;
u32 Pvalue=0;
```

```
float Tmax=85;
float Tmin=-40;
u32 Tspan=13421773;
u32 Tvalue=0;

void delay_us(long int time)
{
  long int i=8*time;
  while(i--);
}

void delay_ms(long int time)//1372@4M 686@2M 343@1M
{
  long int i=1372*time;
  while(i--);
}

void IIC_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOA_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, SCL_Pin|SDA_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pins : PAPin PAPin */
  GPIO_InitStruct.Pin = SCL_Pin|SDA_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
}

void SDA_IN()
{
        GPIO_InitTypeDef  GPIO_InitStructure;
        GPIO_InitStructure.Pin = SDA_Pin;
  GPIO_InitStructure.Mode = GPIO_MODE_INPUT;
```

```
  GPIO_InitStructure.Pull = GPIO_NOPULL;
  //GPIO_InitStructure.Alternate = GPIO_PuPd_UP;
  GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_MEDIUM;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStructure);
}

void SDA_OUT()
{
        GPIO_InitTypeDef  GPIO_InitStructure;
        GPIO_InitStructure.Pin = SDA_Pin;
  GPIO_InitStructure.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStructure.Pull = GPIO_NOPULL;
        GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_MEDIUM;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStructure);
}

void IIC_Start(void)
{
        SDA_OUT();
        IIC_SDA_ON;
        IIC_SCL_ON;
        delay_us(4);
        IIC_SDA_OFF;//START:when CLK is high,DATA change form high to low
        delay_us(4);
        IIC_SCL_OFF;
}

void IIC_Stop(void)
{
        SDA_OUT();//sda
        IIC_SCL_OFF;
        IIC_SDA_OFF;//STOP:when CLK is high DATA change form low to high
        delay_us(4);
        IIC_SCL_ON;
        IIC_SDA_ON;
        delay_us(4);
}
unsigned char IIC_Wait_Ack(void)
{
        unsigned char ucErrTime=0;
```

```
        SDA_IN();
        IIC_SDA_ON;delay_us(1);
        IIC_SCL_ON;delay_us(1);
        while(HAL_GPIO_ReadPin(GPIOA, SDA_Pin))
        {
                ucErrTime++;
                if(ucErrTime>250)
                {
                        IIC_Stop();
                        return 1;
                }
        }
        IIC_SCL_OFF;
        return 0;
}

void IIC_Ack(void)
{
        IIC_SCL_OFF;
        SDA_OUT();
        IIC_SDA_OFF;
        delay_us(2);
        IIC_SCL_ON;
        delay_us(2);
        IIC_SCL_OFF;
}

void IIC_NAck(void)
{
        IIC_SCL_OFF;
        SDA_OUT();
        IIC_SDA_ON;
        delay_us(2);
        IIC_SCL_ON;
        delay_us(2);
        IIC_SCL_OFF;
}

void IIC_Send_Byte(unsigned char txd)
{
```

```
    unsigned char t;
                SDA_OUT();
    IIC_SCL_OFF;
    for(t=0;t<8;t++)
    {
                            if(txd&0x80)
                            {IIC_SDA_ON;}
                            else
                            {IIC_SDA_OFF;}
       txd<<=1;
                delay_us(2);
                IIC_SCL_ON;
                delay_us(2);
                IIC_SCL_OFF;
                delay_us(2);
    }
}
unsigned char IIC_Read_Byte(unsigned char ack)
{
        unsigned char i,receive=0;
        SDA_IN();//SDA
    for(i=0;i<8;i++ )
        {
       IIC_SCL_OFF;
       delay_us(2);
                IIC_SCL_ON;
       receive<<=1;
       if(HAL_GPIO_ReadPin(GPIOA, SDA_Pin))receive++;
                delay_us(1);
    }
    if (!ack)
       IIC_NAck();//nACK
    else
       IIC_Ack(); //ACK
    return receive;
}

unsigned char temp[7];
void Get_Value()
{
```

```
        IIC_Start();
        IIC_Send_Byte(0x50);
        IIC_Wait_Ack();
        IIC_Send_Byte(0xaa);
        IIC_Wait_Ack();
        IIC_Stop();
        delay_ms(17);

        IIC_Start();
        IIC_Send_Byte(0x51);
        IIC_Wait_Ack();
        temp[0]=IIC_Read_Byte(1);
        temp[1]=IIC_Read_Byte(1);
        temp[2]=IIC_Read_Byte(1);
        temp[3]=IIC_Read_Byte(1);
        temp[4]=IIC_Read_Byte(1);
        temp[5]=IIC_Read_Byte(1);
        temp[6]=IIC_Read_Byte(0);
        IIC_Stop();

                if(temp[0]==0x40)
                {
                        Pvalue=temp[1]*256*256+temp[2]*256+temp[3];
                        Tvalue=temp[4]*256*256+temp[5]*256+temp[6];
                }

                Tdisplay=(Tvalue-1677721.6)/Tspan*(Tmax-Tmin)+Tmin;
                Pdisplay=(Pvalue-1677721.6)/Pspan*(Pmax-Pmin)+Pmin;
}


float pressure;
float filt(unsigned char N)
{
        u8 count;
        float sum=0;
        float value_buf[N];
        for (count=0;count<N;count++)
        {
                Get_Value();
```

```
            value_buf[count] = Pdisplay;
            sum += value_buf[count];
        }
        pressure=sum/N;

        return  pressure;
}
```