SAS19

MEDICAL：
-Anesthesia machines
-Spirometers
-Nebulizers
-Hospital room air pressure

INDUSTRIAL：
-Variable Air Volume control
- Static duct pressure
- HVAC transmitters
-Clogged HVAC filter detection

DESCRIPTION

SAS19 High Accuracy Silicon Ceramic sensor is a piezoresistive silicon pressure sensor ，offering an analog/digital output for reading pressure over the specified full scale pressure span and temperature range. SAS19 Series is fully calibrated and temperature compensated for sensor offset, sensitivity, temperature effects, and non-linearity using an on-board Application Specific Integrated Circuit (ASIC). Calibrated output values for pressure are updated at approximately 1 kHz.

SAS19 Series is calibrated over the temperature range of -10 °C to 60 °C. The sensor is characterized for operation from a single power supply of either 3.3 Vdc or 5.0 Vdc. These sensors measure differential and gage pressures. Differential versions allow application of pressure to either side of the sensing diaphragm. Gage versions are referenced to atmospheric pressure and provide an output proportional to pressure variations from atmosphere. SAS19 Series sensors are intended for use with non-corrosive, non-ionic working fluids. They are designed and manufactured according to standards in ISO 9001.

FEATURES

- Various package：SAS19 series pressure sensor is designed with various package such as Side port, DIP or SMT. Basis substrate is optional with ceramic or FR4 PCB. Pressure port is optional with either ceramic or PPS material.
- Small size：10mm*12.5mm compact package.
- Energy efficient: Extremely low power consumption，Supply voltage is 3.3 or 5Volts
- RoHS compliant.
- Absolute, Differential and Gage pressure type.
- Wide variety of pressure ranges: Low pressure from ±1 mbar to±75 mbar, medium pressure from 1psi to 300psi, provide support for many unique applications.
- The 1/8” barbed pressure ports mate securely with 3/32” ID tubing.

- Customer orientation：Accuracy, Total error band and compensated temperature can be customized.
- Provides the sensor's true accuracy over a compensated range of -10 °C to 60 °C.
- Industry-leading long-term stability：Even after long-term use and thermal extremes, these sensors perform substantially better relative to stability than any other pressure sensor available in the industry today.
- Industry-leading accuracy: Extremely tight accuracy of ±0.25 %FSS BFSL (Full Scale Span Best Fit Straight Line)
- Industry-leading Total Error Band (TEB): Stalbesens International specifies TEB—the most comprehensive, clear, and meaningful measurement—that provides the sensor's true accuracy over a compensated range of -10 °C to 60 °C.
- I2C- or SPI-compatible 14-bit digital output (min. 12-bit sensor resolution) accelerates performance through reduced conversion requirements and the convenience of direct interface to microprocessors or microcontrollers;
- Digital output types can offer 10%~90% output or 5%~95% output for optional.

STANDARD PRESSURE RANGE（INH2O,PSI,KPA,MBAR）

2 inH2O	Gauge, Differential	Digital output
5 inH2O	Gauge, Differential	Digital output
10 inH2O	Gauge, Differential	Digital output
20 inH2O	Gauge, Differential	Digital output
1PSI	Gauge, Differential	Digital output
2PSI	Gauge, Differential	Digital output
5PSI	Gauge, Differential	Digital output
15PSI	Gauge, Differential, Absolute	Digital output
30PSI	Gauge, Differential, Absolute	Digital output
50PSI	Gauge, Differential, Absolute	Digital output
100PSI	Gauge, Differential, Absolute	Digital output
150PSI	Gauge, Differential, Absolute	Digital output
300PSI	Gauge, Differential, Absolute	Digital output

5 mbar	Gauge, Differential	Digital output
10 mbar	Gauge, Differential	Digital output
25 mbar	Gauge, Differential	Digital output
50 mbar	Gauge, Differential	Digital output
5Kpa	Gauge, Differential	Digital output
10Kpa	Gauge, Differential	Digital output
35Kpa	Gauge, Differential	Digital output

100Kpa	Gauge, Differential	Digital output
200Kpa	Gauge, Differential	Digital output
400Kpa	Gauge, Differential	Digital output
1000Kpa	Gauge, Differential	Digital output
2000Kpa	Gauge, Differential	Digital output

MAXIMUM RATINGS[1]

Parameter	Min	Max	Unit
Supply Voltage (Vsupply)	-0.3	5.5	Vdc
Voltage on any pin	-0.3	Vsupply +0.3	V
Digital Interface clock frequency： I²C SPI	 100 50	 400 800	KHZ
ESD susceptibility (Human body mode)		4	Kv
Storage Temperature	-40	125	℃
Soldering time and temperature Solder temperature（DIP） Peak reflow temperature (SMT)	5s max，at 250℃ 15s max，at250℃		

OPERATING SPECIFICATIONS

Parameter	Min	Typical	Max	Unit
Supply Voltage (Vsupply) 3.3 5.0 Sensors are either 3.3 Vdc or 5.0 Vdc based on listing selected	 3.0 4.75	 3.3 [2] 5.0 [2]	 3.6 5.25	 Vdc Vdc
Supply current 3.3 Vdc supply 5.0 Vdc supply		2.1 3		 mA mA
Compensated temperature range[3]	-10	-	60	℃
Operating temperature range [4]	-40	-	125	℃
Startup time（power up to data ready）	-	2.8	7.3	ms
Response time	-	0.46	-	ms
I ²C/SPI voltage level low	-	-	0.2	V_{supply}
I ²C/SPI voltage level low	0.8	-	-	V_{supply}
Pull up on SDA/MISO, SCL/SCLK, SS	1	-	-	Kohm
Accuracy [5]	-	-	±0.25	%FSS [7]
Orientation Sensitivity[6]	-	-	±0.15	%FSS [8]

Total Error Band (TEB)[7]	-1%	-	1%	%FSS
Over Pressure		>3		Times
Burst Pressure		>5		Times
OUTPUT RESOLUTION	14	-	-	Bits

ENVIRONMENT SPECIFICATIONS

Parameter	Characteristic
Humidity： Gases only	0% to 95% RH，non condensing
Vibration	MIL-STD-202F，Method 214，Condition F（20.7 g random）
Shock	MIL-STD-202F，Method 213B，Condition F
Life [9]	1million cycles to working pressure min
Solder reflow	J-STD-020D. Moisture sensitivity level 1

WETTED MATERIAL[9]

Parameter	PortA（Pressure port）	PortB（Reference）
Cover	PPS or Ceramic	PPS or Ceramic
Substrate	Alumina ceramic or FR4	Alumina Ceramic or FR4
Adhesives	Epoxy, silicone	Epoxy, silicone
Electrical components	Ceramic, solder, silicon	Ceramic, solder, silicon

Notes:

1. Maximum ratings are the extreme limits the device can withstand without damage to the product. Stresses above these ratings may cause permanent damage. Exposure to absolute maximum conditions for extended periods may degrade device reliability.
2. The sensor is not reverse polarity protected. Incorrect application of supply voltage or ground to the wrong pin may cause electrical failure.
3. The compensated temperature range is the temperature range over which the sensor will produce an output proportional to pressure within the specified performance limits.
4. The operating temperature range is the temperature range over which the sensor will produce an output proportional to pressure but may not remain within the specified performance limits.
5. Accuracy: The maximum deviation in output from a Best Fit Straight Line (BFSL) fitted to the output measured over the pressure range at 25 °C Includes all errors due to pressure non-linearity, pressure hysteresis, and non-repeatability.
6. Orientation sensitivity: The maximum change in offset of the sensor due to a change in position or orientation relative to Earth’s gravitational field.
7. Total Error Band: The maximum deviation from the ideal transfer function over the entire compensated temperature and pressure range. Includes all errors due to offset, full scale span, pressure non-

linearity, pressure hysteresis, repeatability, thermal effect on offset, thermal effect on span, and thermal hysteresis.

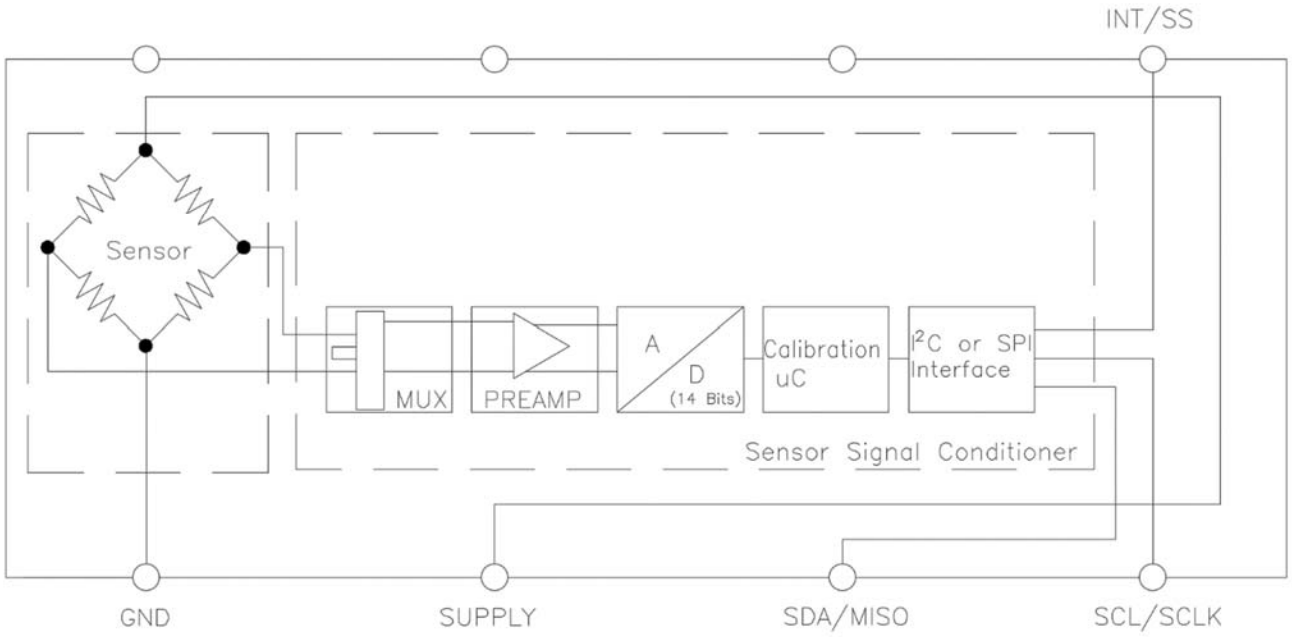
8. Full Scale Span (FSS): The algebraic difference between the output signal measured at the maximum (Pmax.) and minimum (Pmin.) limits of the pressure range.
9. Life may vary depending on specific application in which sensor is utilized.
10. Contact Stalbesens International Sales and Service for detailed material information.
11. Total Error Band After Auto-Zero: The maximum deviation from the ideal transfer function over the entire compensated pressure range at a constant temperature and supply voltage for a minimum of 24 hours after an auto-zero operation. Includes all errors due to full scale span, pressure non-linearity, pressure hysteresis, and thermal effect on span.
12. Working Pressure: The maximum pressure that may be applied to any port of the sensor in continuous use. This pressure may be outside the operating pressure range limits (Pmin. to Pmax.) in which case the sensor may not provide a valid output until pressure is returned to within the operating pressure range. Tested to 1 million cycles, min.
13. Overpressure: The absolute maximum rating for pressure which may safely be applied to the product for it to remain in specification once pressure is returned to the operating pressure range. Exposure to higher pressures may cause permanent damage to the product. Unless otherwise specified this applies to all available pressure ports at any temperature with the operating temperature range. Tested to 10,000 cycles, minimum.
14. Burst Pressure: The maximum pressure that may be applied to any port of the product without causing escape of pressure media. Product should not be expected to function after exposure to any pressure beyond the burst pressure.
15. Common Mode Pressure: The maximum pressure that can be applied simultaneously to both ports of a differential pressure sensor without causing changes in specified performance.
16. Customized design please contact Stalbesens International sales.

Warning：

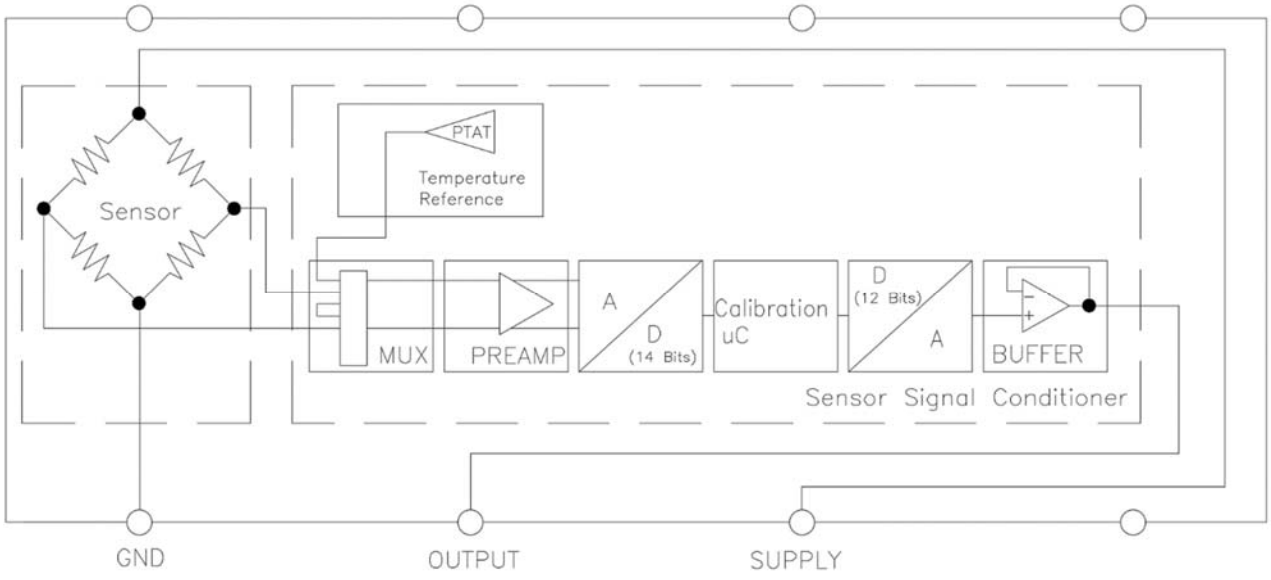
Please follow below instructions to avoid possible product damage

- Make sure fluid only be used for port A, port B is not compatible with liquid, need contact customer service for liquid compatibility customization for port B.
- Make sure liquid fluid do not contain particles. All SAS19 sensor is connected to sealed device, the particles will accumulate inside the sensor and cause device damage or impact the sensor output.
- Suggest to place the port A upside down so the particles in the system will not enter into port A and will not stay in the sensor.
- Make sure the liquid fluid will not cause any residual after dry out, accumulated residuals may impact the sensor output, it is very hard to clean and remove the residuals.

BLOCK DIAGRAM



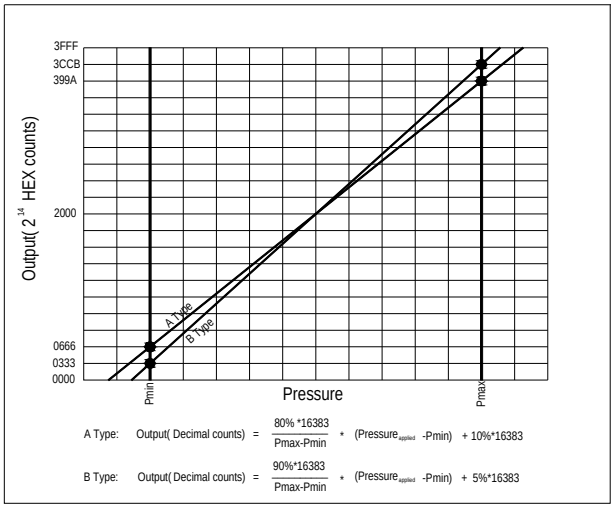
SAS19 Digital output



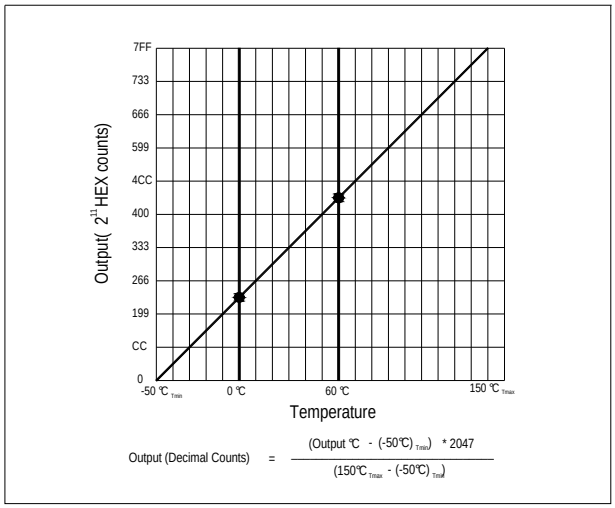
SAS19 Analog

PRESSURE AND TEMPERATURE TRANSFER

Pressure Transfer Functions



Temperature Transfer Functions

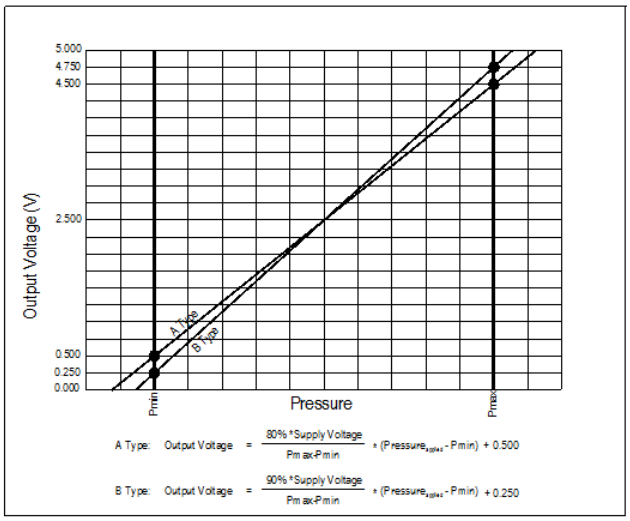


Digital Output

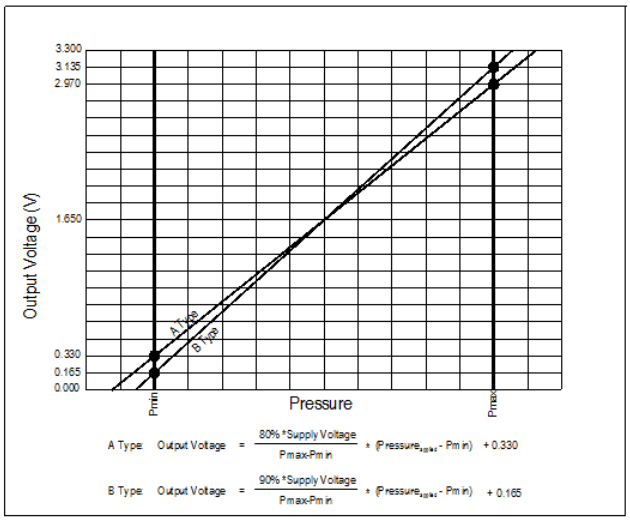
Temperature Output vs Counts

Output °C	Digital Counts (decimal)	Digital Counts (hex)
−50	0	0 X 0000
0	511	0 X 01FF
10	614	0 X 0266
25	767	0 X 02FF
50	1023	0 X 03FF
85	1381	0 X 0565
150	2047	0 X 07FF

Pressure Transfer Functions, Supply=5V



Pressure Transfer Functions, Supply=3.3V



Analog Output

PRESSURE TYPES

Pressure type	Comments
Gauge	Output is proportional to the difference between applied pressure and atmospheric (ambient) pressure. Pmin. is set at atmospheric pressure.
Differential	Output is proportional to the difference between the pressures applied to each port (Port A ~Port B). 50% point of transfer function set at Port A = Port B.
Absolute	Output is proportional to the difference between applied pressure and absolute zero pressure pressure. Pmin. is set at absolute zero pressure.

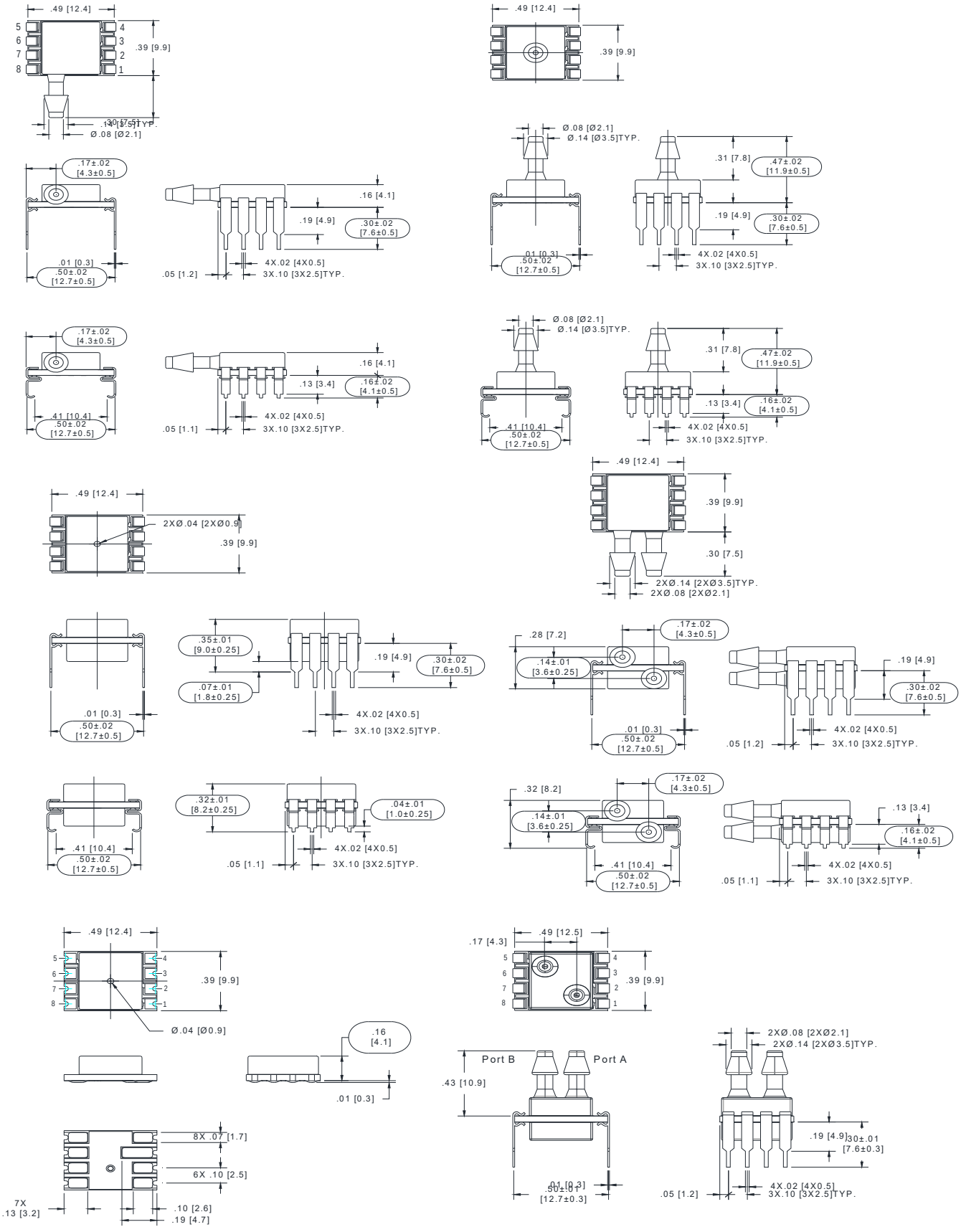
SENSOR OUTPUT AT SIGNFICANT PERCENTAGES

% Output	Digital Counts (decimal)	Analog (V)
0	0	0
5	819	0.25
10	1638	0.5
50	8192	2.5
90	14746	4.5
95	15565	4.75
100	16383	5

CONNECTION DIAGRAM

Output type	1	2	3	4	5	6	7	8
Analog	Blank	V_{supply}	Signal	Ground	Blank	Blank	Blank	Blank
Digital	Ground	V_{supply}	SDA/MISO	SCL/SCLK	INS/SS	Blank	Blank	Blank

DIMENSION DRAWING（INCH［MILLIMETER］）



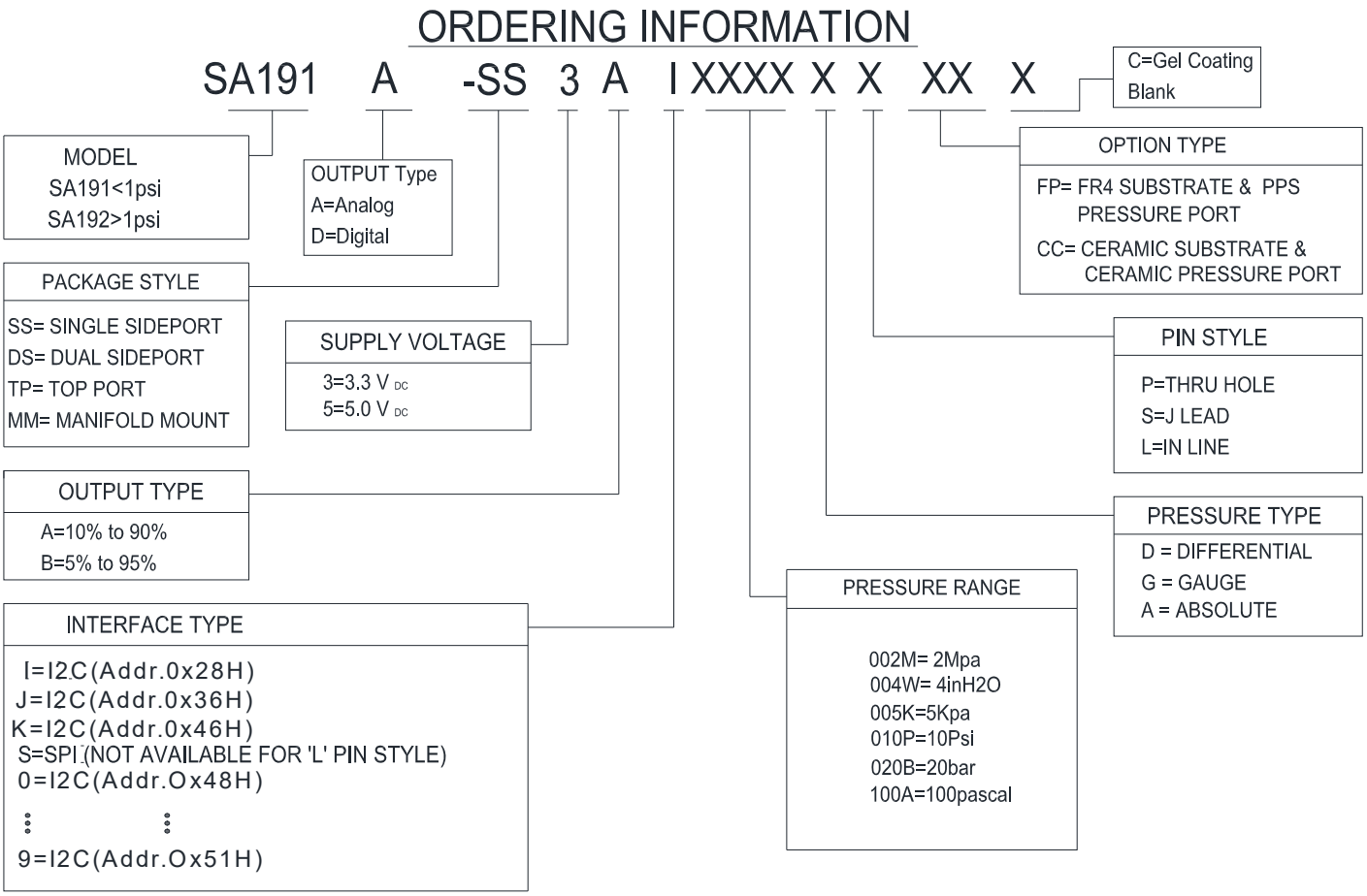
NOTES：

-For differential，when pressure in port A is greater than pressure in port B, output is greater than 50% of 16777215；when pressure in port A is equal to the pressure in port B, output is equal to 50% of 16777215.

-For Gauge type A output, when pressure in port A is greater than pressure in port B, output is greater than 10% of 16777215；when pressure in port A is equal to the pressure in port B, output is equal to 10% of 16777215.

-Port A is always given positive pressure during calibration.

ORDERING INFORMATION



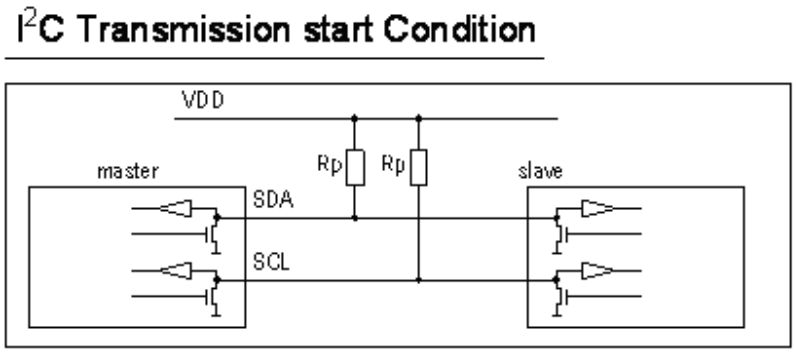
I2C SPI COMMUNICATION SPECIFICATIONS

1. I2C Interface Specification

HM192 series is optimized in terms of sensor readout and power consumption when the factory setting for I2C. For detailed specifications of the I2C protocol, see The I2C Bus Specification.

1.1 Interface Connection-External

Bi-directional bus lines are implemented by the devices (master and slave) using open-drain output stages and a pull-up resistor connected to the positive supply voltage. The recommended pull-up resistor value depends on the system setup (capacitance of the circuit or cable and bus clock frequency). In most cases, 4.7kΩ is a reasonable choice. The capacitive loads on SDA and SCL line have to be the same. It is important to avoid asymmetric capacitive loads.



1.2 I2C Address

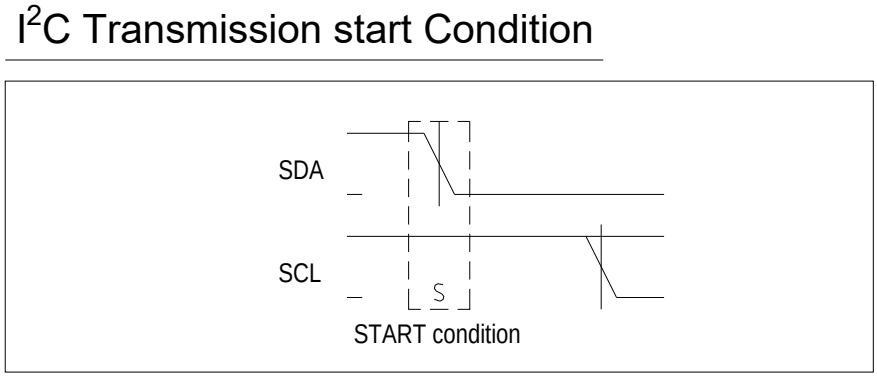
The I2C address consists of a 7-digit binary value. The factory setting for I2C slave address is 0x28, 0x36 or0x46 depending on the interface type selected from the ordering information. The address is always followed by a write bit (0) or read bit (1).The default hexadecimal I2C header for read access to the sensor is therefore 0x51, 0x6D, 0x8D respectively, based on the ordering information.

1.3 INT/SS pin

When programmed as an I2C device, the INT/SS pin operates as an interrupt. The INT/SS pin rises when new output data is ready and falls when the next I2C communication occurs.

1.4 Transfer sequences

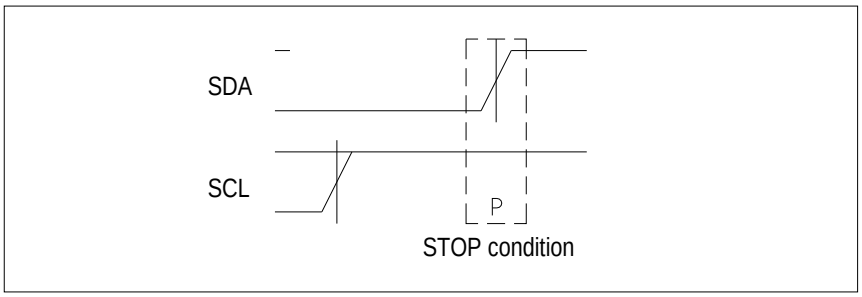
Transmission START Condition (S): The START condition is a unique situation on the bus created by the master, indicating to the slaves the beginning of a transmission sequence (the bus is considered busy after a START).



A HIGH to LOW transition on the SDA line while SCL is HIGH

Transmission STOP Condition (P): The STOP condition is a unique situation on the bus created by the master, indicating to the slaves the end of a transmission sequence (the bus is considered free after a STOP).

I²C Transmission stop Condition

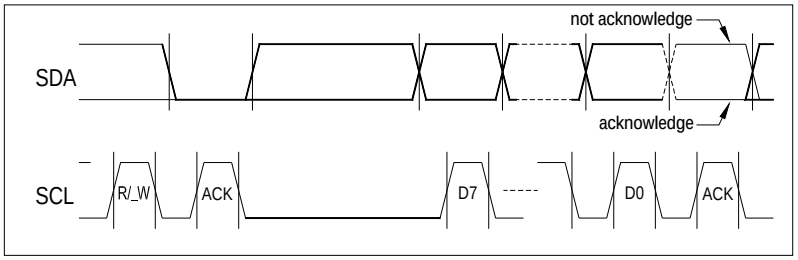


A LOW to HIGH transition on the SDA line while SCL is HIGH

Acknowledge (ACK) / Not Acknowledge (NACK): Each byte (8 bits) transmitted over the I2C bus is followed by an acknowledge condition from the receiver. This means that after the master pulls SCL low to complete the transmission of the 8th bit, SDA will be pulled low by the receiver during the 9th bit time. If after transmission of the 8th bit the receiver does not pull the SDA line low, this is considered to be a NACK condition.

If an ACK is missing during a slave to master transmission, the slave aborts the transmission and goes into idle mode.

I²C Acknowledge / Not acknowledge



Each byte is followed by an acknowledge or a not acknowledge, generated by the receiver

1.5 Data Transfer Format

Data is transferred in byte packets in the I2C protocol, which means in 8-bit frames. Each byte is followed by an acknowledge bit. Data is transferred with the most significant bit (MSB) first.

A data transfer sequence is initiated by the master generating the Start condition (S) and sending a header byte. The I2C header consists of the 7-bit I2C device address and the data direction bit (R/_W).

The value of the R/_W bit in the header determines the data direction for the rest of the data transfer sequence. If R/_W = 0 (WRITE) the direction remains master-to-slave, while if R/_W = 1 (READ) the direction changes to slave-to-master after the header byte.

1.6 Command Set and data Transfer Sequences

The I2C master command starts with the 7bit slave address with the 8th bit =1 (READ).The sensor as the slave sends an acknowledge (ACK) indicating success. The sensor has four I2C read commands: Read_MR, Read_DF2, Read_DF3, and Read_DF4.Figure 1.6 shows the structure of the measurement packet for three of the four I2C read commands, which are explained in sections 1.6.1 and 1.6.2

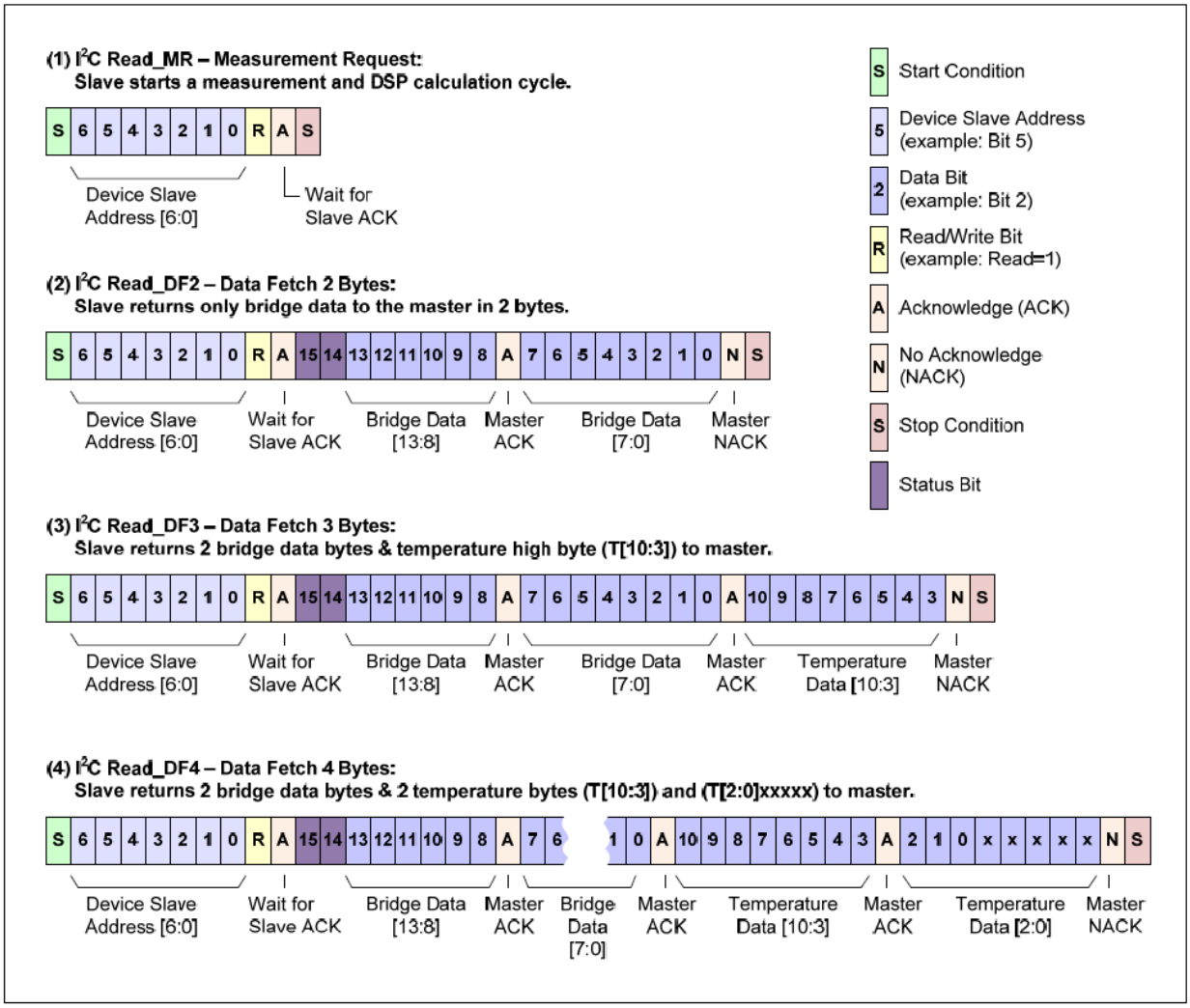


Figure 1.6 – I2C Measurement Packet Reads

I2CRead_DF (Data Fetch)

For Data Fetch commands, the number of data bytes returned by the RBiciLite™ is determined by when the master sends the NACK and stop condition. For the Read_DF3 data fetch command (Data Fetch 3 Bytes; see example 3 in Figure 1.6), the sensor returns three bytes in response to the master sending the slave address and the READ bit (1): two bytes of bridge data with the two status bits as the MSBs and then 1 byte of temperature data (8-bit accuracy). After receiving the required number of data bytes, the master sends the NACK and stop condition to terminate the read operation. For the Read_DF4 command, the master delays sending the NACK and continues reading an additional finalbyte to acquire the full corrected 11-bit temperature measurement. In this case, the last 5 bits of the final byte of the packet are undetermined and should be masked off in the application.The Read_DF2 command is used if corrected temperature is not required. The master terminates the READ operation after the two bytes of bridge data (see example 2 in Figure 1.6).

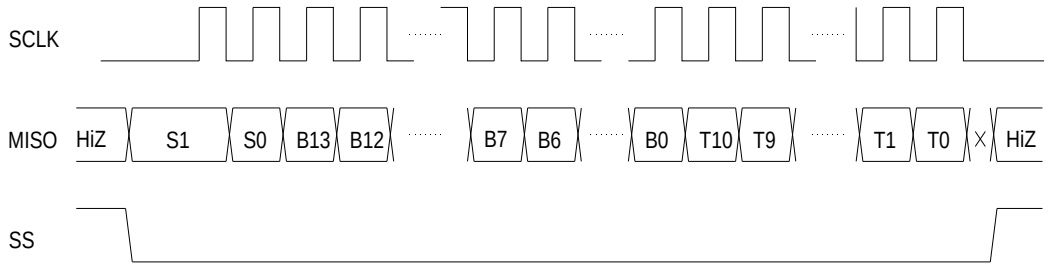
The two status bits (Bit 15 and Bit 14) give an indication of stale or valid data depending on their value. A returned value of 00 indicate “normal operation and a good data packet” while a returned value of 10 indicates “stale data that has been already fetched”. Users that use “status bit” polling should select a frequency slower than 20% more than the response time.

2.SPI interface Specification

The SPI interface of sensor can be programmed for falling-edge MISO change.

2.1 SPI Read_DF (Data Fetch)

For simplifying explanations and illustrations, only falling edge SPI polarity will be discussed in the following sections. The SPI interface will have data change after the falling edge of SCLK. The master should sample MISO on the rise of SCLK. The entire output packet is 4 bytes (32 bits). The high bridge data byte comes first, followed by the low bridge data byte. Then 11 bits of corrected temperature (T[10:0]) are sent: first the T[10:3]byte and then the {T[2:0],xxxxx} byte. The last 5 bits of the final byte are undetermined and should be masked off in the application. If the user only requires the corrected bridge value, the read can be terminated after the 2nd byte. If the corrected temperature is also required but only at an 8-bit resolution, the read can be terminated after the 3rd byte is read.



Packet = [{S(1:0),B(13:8)},{B(7:0)},{T(10:3)},{T(2:0),xxxxx}] Where
S(1:0) = Status bits of packet (normal, command, busy, diagnostic)
B(13:8) = Upper 6 bits of 14-bit bridge data.
B(7:0) = Lower 8 bits of 14-bit bridge data.
T(10:3) = Corrected temperature data (if application does not require corrected temperature, terminate read early)
T(2:0),xxxxx =. Remaining bits of corrected temperature data for full 11-bit resolution
HiZ = High impedance

Figure 2.1 – SPI Output Packet with Falling Edge SPI_Polarity

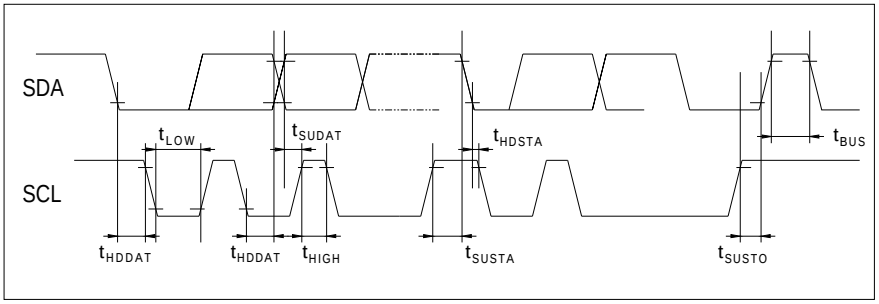
TIMING DIAGRAM

I2C INTERFACE PRRAMETERS

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 Timing Diagram

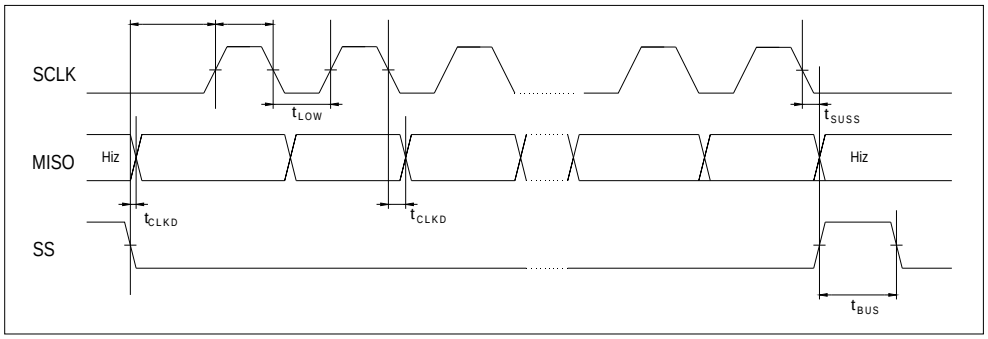


SPI INTERFACE PRRAMETERS

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 Timing Diagram



I2C/SPI PROGRAMING SAMPLE

I2C （SAS191D-DS-3BI002WDP）

C code example for I2C with Read_DF4 command

On power-up, PORTB is initialized to all inputs with the internal pull-ups turned off, the external pull-ups pull the SDA and SCL lines high and the PORTB output latch bits SCL and SDA are initialized to zero. Routines WriteSDA and WriteSCL toggle their respective data direction bit depending on the value of parameter “state”. When state is a “1” the port pin is configured as input (external pull-ups pull high). When state is a “0” the port pin is configured as an output and the latch drives the pin low. WriteSDA and WriteSCL are very simple routines that could be incorporated into their respective calling routines to further reduce the code size.

General Calling Sequence for the Routines

SendStartBit(); /*start*/

```
SendByte(byte);              /*send address or command MSB first*/
GetOneByte();                /*read one byte from serial stream */
SendStop();                  /*stop*/
```

PORTB on the ATmega164P is used to communicate with SAS191D transducer. Bit assignments are as follows:

I2C.c

```
/*PB0 = SDA*/
/*PB1 = SCL*/

#include “i2c.h”

void WriteSCL(unsigned char state)
{
  if (state)
    DDRB &= 0xfd;            /* input ... pullup will pull high or Slave will drive low */
  else
    DDRB |= 0x02;                    /* output ... port latch will drive low */
}

void WriteSDA(unsigned char state)
{
  if (state)
    DDRB &= 0xfe;            /* input ... pullup will pull high or Slave will drive low */
  else
    DDRB |= 0x01;                    /* output ... port latch will drive low */
}

unsigned char SetSCLHigh(void)
{
  WriteSCL(1);                       /* release SCL*/
  /* set up timer counter 0 for timeout */
  t0_timed_out = FALSE;              /* will be set after approximately 34 us */
  TCNT0 = 0;                 /* clear counter */
  TCCR0 = 1;                 /* ck/1 .. enable counting */
  /* wait till SCL goes to a 1 */
  while (! (PINB & 0x02) && !t0_timed_out)
  ;
  TCCR0 = 0;                 /* stop the counter clock */
  return(t0_timed_out);
}

void BitDelay(void)
{
  char delay;
  delay = 0x03;
  do
  {
    _NOP();
  } while (--delay);
}


/* Routine SendStopBit generates an TWI stop bit assumes SCL is low stop bit is a 0 to 1
transition on SDA while SCL is high

          ____________
         /
```

```
SCL  ___/

                  _______
                 /
SDA ___________/
*/


void SendStopBit(void)
{
   WriteSDA(0);
   BitDelay();
   SetSCLHigh();
   BitDelay();
   WriteSDA(1);
   BitDelay();
}

/* Routine SendStartBit generates an start bit start bit is a 1 to 0 transition on SDA while SCL is
high

             _______________
            /
SCL ___/

      ____________
                  \
SDA           \_____
*/

void SendStartBit(void)
{
   WriteSDA(1);
   BitDelay();
   SetSCLHigh();
   BitDelay();
   WriteSDA(0);
   BitDelay();
   WriteSCL(0);
   BitDelay();
}

unsigned char SendByte(unsigned char byte)
{
   unsigned char i;
   unsigned char error;
   for (i = 0; i < 8; i++)
   {
      WriteSDA(byte & 0x80);              /* if > 0 SDA will be a 1 */
      byte = byte << 1;             /* send each bit */

      BitDelay();
      SetSCLHigh();
      BitDelay();
      WriteSCL(0);
      BitDelay();
   }
   /* now for an ack */
   /* Master generates clock pulse for ACK */
   WriteSDA(1);                           /* release SDA ... listen for ACK */
   BitDelay();
   SetSCLHigh();                           /* ACK should be stable ... data not allowed to change when
```

```
SCL is high */
   /* SDA at 0 ?*/
   error = (PINB & 0x01);                  /* ack didn't happen if bit 0 = 1 */
   WriteSCL(0);
   BitDelay();
   return(error);
}

unsigned char GetOneByte(unsigned char lastbyte)
{
   /* lastbyte ==1 for last byte */
   unsingned char i;
   unsigned char data;

   DDRB &=0xfe;    /* release SDA ... listen for slave output */
   data=0;

  for (i=0; i<8;i++)
  {
     SetSCLHigh();                         /* Slave output should be stable ... data not allowed to
change when SCL is high */
     BitDelay();
     data=data<<1;
     if (PINB & 0x01)
       data=data | 1;
     WriteSCL(0);
     BitDelay();
  }

  /*send ACK*/
   WriteSDA (lastbyte); /* no ack on last byte ... lastbyte = 1 for the last byte */
   BitDelay();
   SetSCLHigh();

   BitDelay();
   WriteSCL(0);
   BitDelay();
   WriteSDA(1);
   BitDelay();

  return (data);
}
```

ReadWithPollingI2C.c

```
/*
ReadWithPollingI2C.c reads the digital output simply at any time and be assured the data is no
older than the selected response time specification by checking the status of the 2 MSBs of the
bridge high byte data
*/

#include “i2c.h”

extern unsigned char GetOneByte(unsigned char lastbyte);
extern unsigned char SendByte(unsigned char byte);
```

```
extern void SendStartBit(void);
extern void SendStopBit(void);
extern void BitDelay(void);
extern unsigned char SetSCLHigh(void);
extern void WriteSDA(unsigned char state);
extern void WriteSCL(unsigned char state);


unsigned char SAS191D_
Address; unsigned char bufptr[4]
;
void Init (void)
{
    __disable_interrupt();

   /* P0 = SDA - bidirectional */
   /* P1 = SCL - output */
   /* P7, P6, P5, P4, P3, P2, P1, P0 */
   /* O O O O O O O O */
   /* 1 1 1 1 1 1 1 1 */
   DDRB = 0xff;
   PORTB = 0xfc;

    /*setup SAS191D device address*/
   SAS191DDO_Address=
   0x28; /*
   The factory setting for I2C slave address is 0x28, 0x36 or 0x46 depending on the interface type
selected from the ordering information.
   For this sample code, 0x28 is used for Slave address of
   SAS191D.
   */
}
unsigned char ReadSAS191D(unsigned char DF_
Command) {
  unsigned char i;
  unsigned char error;

  SendStartBit();

  if (SendByte((SAS191D_Address<<1) + read))                    /*send salve address byte*/

  {
    return (1);    /*check error*/
  }

  for (i=0; i< (DF_Command-1); i++)
  {
    bufptr[i] = GetOneByte (0);                  /* 1 byte of read sequence */
  }

  bufptr[DF_Command-1] = GetOneByte (1);          /* 1 signals last byte of read sequence */

  SendStopBit ();

 return (0);
}


void main (void)
```

```
{
   float Pressure, Temperature;
   unsigned int Dpressure, Dtemperature;

   float P1=819.15;              /* P1= 5% * 16383 – B type*/
   float P2=15563.85;            /* P2= 95% * 16383 – B type*/

   float Pmax=2.0;
   float Pmin=-2.0;

   Init();

   do
    {

    ReadSAS191D (DF4);         /*Read_DF4 command – data fetch 4 bytes */

    If ((bufptr [0] & 0xc0) ==0x00)      /*test status of the 2 MSBs of the bridge high byte of data*/
    {

   Dpressure= ((unsigned int) (bufptr [0] & 0x3f) <<8) + (bufptr [1]);
   Dtemperature= (((unsigned int) bufptr [2]) <<3) + bufptr [3];

    Pressure= (((float) Dpressure)-P1) * (Pmax-Pmin) / P2+Pmin;
    Temperature= ((float) Dtemperature) * 200 / 2047 -50;

   }
  }  while (1);
} /* main */
```

I2C.h

```
#include “iom164p.h”

#define DF2    2
#define DF3    3
#define DF4    4

#define write   0
#define read    1
```

SPI （SAS191D-DS-3BS002WDP）

C code example for SPI with Read_DF4 command

ReadWithSPI.c

```
/*
ReadWithSPI.c reads the digital output simply at any time and be assured the data is no older
than the selected response time specification by checking the status of the 2 MSBs of the bridge
high byte data */


/*PB0 = SCLK*/
/*PB1 = MISO*/
/*PB2 = SS*/

#include “iom164p.h”
```

```
#define DF2    2
#define DF3    3
#define DF4    4

unsigned char bufptr[4];

void Init(void)
{
/* P0 = SCLK  –  output */
/* P1 = MISO  –  input */
/* P2 = SS  –  output */

/* P7, P6, P5, P4, P3, P2, P1, P0 */
/* O O O O O O I O */
/* 1 1 1 1 1 1 1 1 */
DDRB = 0xfd;
PORTB = 0xfc;

}

void BitDelay(void)
{
char delay;
delay = 0x03;
do
{
while(--delay)
;
_NOP();
return;
}

unsigned char GetOneByte (void)
{

unsigned char data=0;
  unsigned char i;

 for (i=0; i<8; i++)
{
  BitDelay();
   SCLK=1;
   BitDelay();
data=data<<1;
if (PINB & 0x02)
    data=data | 1;
   SCLK=0;
 BitDelay()
}

return (data);
}

unsigned char ReadSAS191D(unsigned char DF_
Command) {
    unsingned char i;

    SCLk=0;
```

```
   SS=0;
  BitDelay();
for (i=0; i<(DF_Command); i++)
{
    bufptr[i] = GetOneByte ();                    /* 1 byte of read sequence */
}

SS=1;
BitDelay();
}

void main (void)
{
float Pressure, Temperature;
unsigned int Dpressure,Dtemperature;

float P1= 819.15;              /* P1= 5% * 16383 – B type*/
float P2= 15563.85;     /* P2= 95% * 16383 – B type*/

float Pmax= 2.0;
float Pmin= -2.0;

Init();

do
{

  ReadSAS191D (DF4);                  /*Read_DF4 command – data fetch 4 bytes */
  If((bufptr [0] & 0xc0)==0)        /*test status of the 2 MSBs of the bridge high byte of data*/
{
      Dpressure= ((unsigned int) (bufptr [0] & 0x3f) <<8) + (bufptr [1]);
      Dtemperature= (((unsigned int) bufptr [2]) <<3) + bufptr [3];

      Pressure= (((float) Dpressure)-P1) * (Pmax-Pmin) / P2+Pmin;
      Temperature= ((float) Dtemperature) * 200 / 2047-50;
```