

PS-33D

Pressure Sensor-Package
With I²C Digital Output

- Supply Voltage : 3.0V
- I²C Interface
- 24 Bit Σ - Δ ADC
- Pressure range : -1,000Pa to 1,000Pa
- Piezo-resistive silicon micro-machined sensor
- RoHS compliant and Halogen-free Package
- Approximately 200Hz data output rate

The PS-33D is the pressure sensor which measures gauge pressures. It consists of a silicon micro-machined sensing element chip and a signal conditioning ASIC. The pressure sensor element and the ASIC are mounted inside a system-in-package and wire-bonded to appropriate contacts. The PS-33D provides the digital output data with the format of I²C interface. It can achieve ESD robustness, fast response time, high accuracy and linearity as well as long-term stability. All measurement data is fully calibrated and temperature compensated. In addition, it allows for easy system integration.



☐ Recommended Operating Conditions

Specifications	
Pressure range	-1,000Pa to +1,000Pa (Option)
Supply Voltage	3VDC
Operating mode current	~0.6mA (typical)
Sleep Mode current	20nA (typical)
Output	I ² C interface
ADC resolution	24 Bit
ESD withstand	+/- 2KV
Humidity Limits	0 to 95% RH
I ² C clock frequency	400KHz

Performance & Environmental	
Operating Temp	-20 to 80°C
Accuracy	±0.1% of F.S
Total error band	±0.5% of F.S(0 to 50°C) ±1.0% of F.S(-20 to 80°C)
Cycle life	1,000,000 / cycle
Proof pressure	x 2.0
Burst pressure	x 3.0
Water protection	IP-60
External Material	PBT

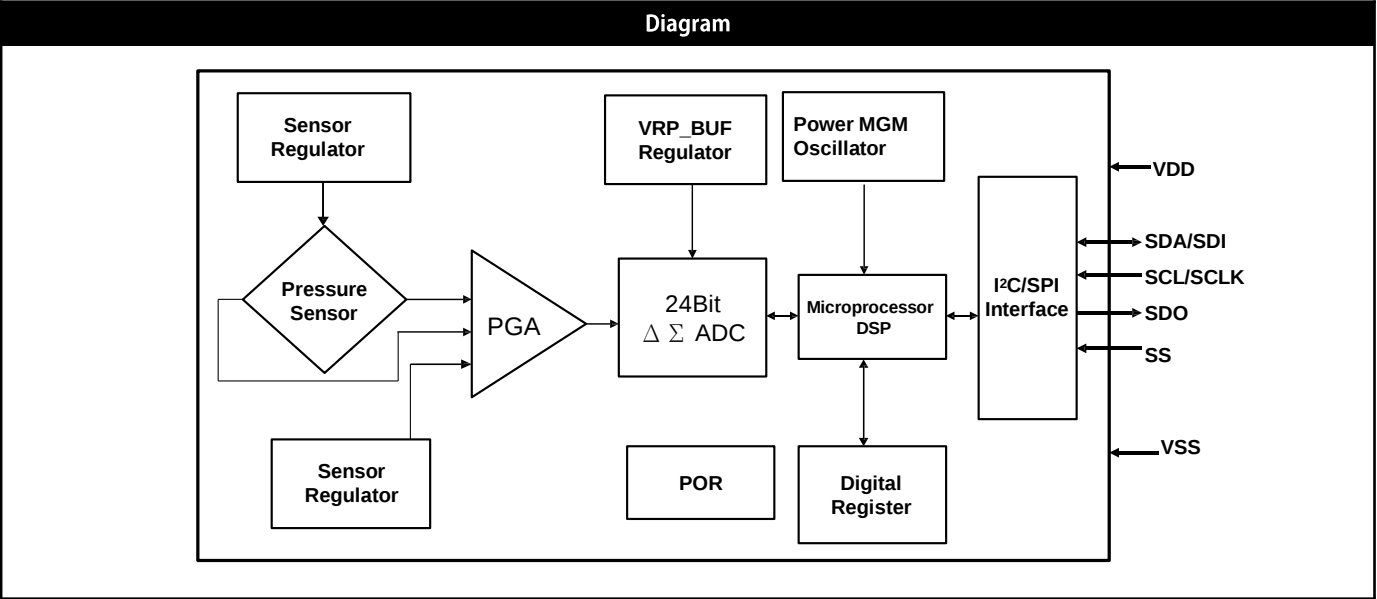
Maximum Ratings

PARAMETER	MIN	TYP	MAX	UNITS
VDD Supply Voltage	-0.4	-	+3.63	V
Voltages at Digital I/O Pins	-0.5	-	VDD+0.5	V
Operating Temperature Range	- 40	-	+ 85	° C
Storage Temperature Range	-40	-	+125	° C
Electrostatic Discharge Tolerance - Human Body Model	-2	-	+2	kV

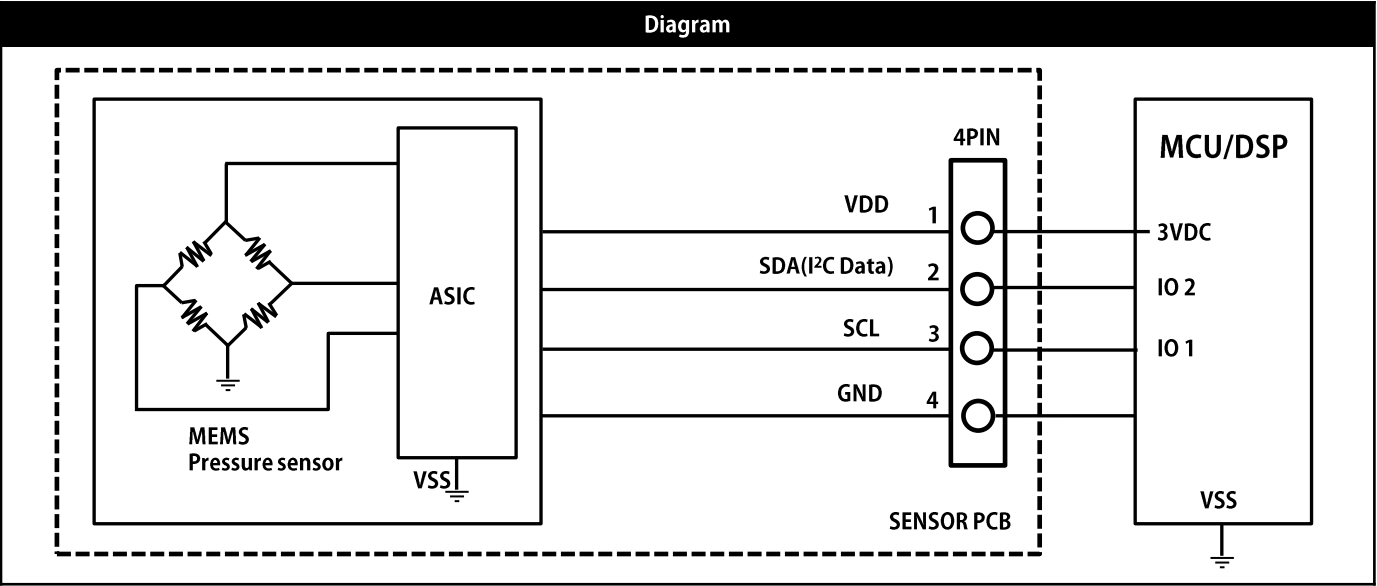
Electrical Characteristics

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
V _{DD}	Operating Voltage	—	—	1.68	3.0	3.6	V
I _{DD}	Operating Current	3.0V	—	—	0.6	1.2	mA
I _{STB1}	Standby Current	3.0V	Temperature ≤ 85°C	—	20	250	nA
P _{PRO}	Proof pressure	—	—	—	—	+20,000	Pa
P _{OP}	Operating pressure	3.0V	—	-1,000	—	+1,000	Pa
P _{RACC}	Relative Accuracy	3.0V	P: +/- 1000 Pa, T: 0 ~ +50°C	-10	—	+10	pa
P _{AACC1}	Absolute Accuracy 1	3.0V	P: +/- 1000 Pa, T: +25 °C	-20	—	+20	Pa
P _{AACC2}	Absolute Accuracy 2	3.0V	P: +/- 1000 Pa, T: 0 ~ +50 °C	-30	—	+30	Pa
P _{RES}	ADC Resolution	—	By internal option	—	20	—	bit
P _{SEN}	Pressure Sensitivity	3.0V	—	—	0.0027	—	Pa/LSB
P _{NS}	Pressure Resolution (RMS)	3.0V	—	—	0.1	—	Pa
t _{ADC}	AD Conversion Time	3.0V	—	—	1.7	—	ms
t _{update}	Data Update Time	3.0V	Full measurement and temperature compensation	—	7	—	ms
t _{ST}	Start-Up Time	3.0V	VDD ramp up to communication	—	—	1	ms
			VDD ramp up to Analog operation	—	—	2.5	ms
t _{SLP}	Wake-UP Time	3.0V	Sleep to communication	—	—	0.5	ms
			Sleep to analog operation	—	—	2	ms
F _{sys}	System Frequency	3.0V	—	—	4	—	MHz
F _{I2C}	I ² C Clock Frequency	3.0V	—	—	—	3.4	MHz
F _{SPI}	SPI Clock Frequency	3.0V	—	—	1	20	MHz

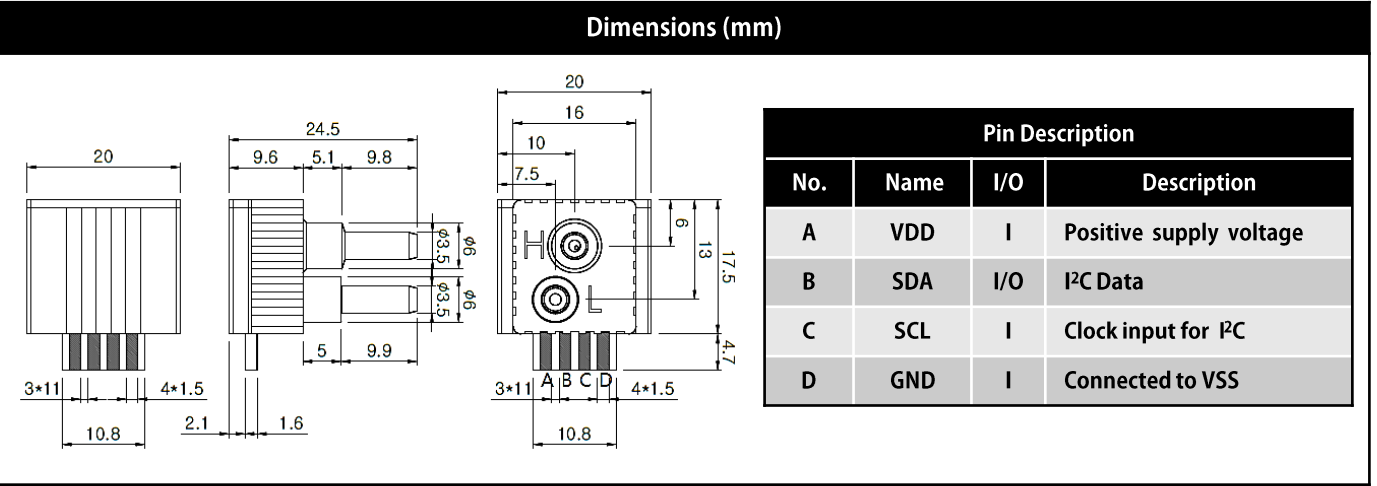
❑ Main ASIC Part Diagram



❑ Block Diagram



❑ Outline Drawing Dimension (mm)



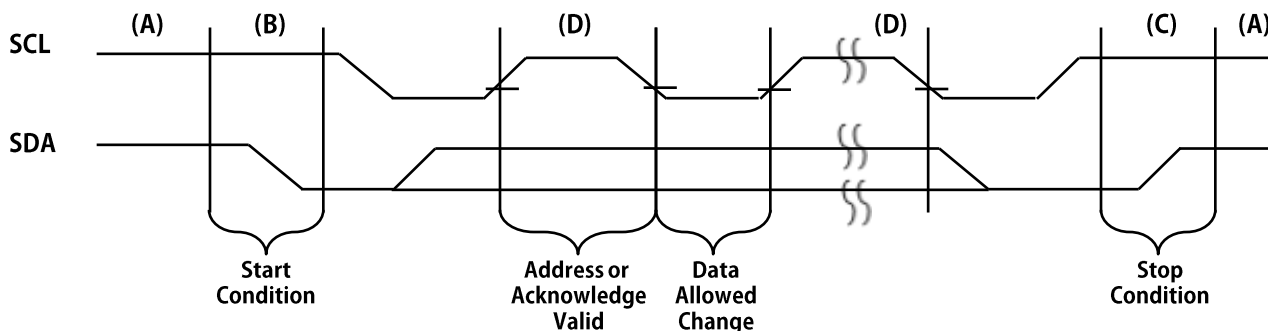
I²C Operation

I²C Operation

The first command must be I²C command, the I²C mode will be selected. PS-33D supports a bi-direction two wire bus and data transmission protocol to output data. A processor sends data onto the bus is defined as transmitter, PS-33D receives data is defined receiver. The bus must be controlled by a master processor which generates the serial clock (SCL), controls the bus access, and generates the START and STOP conditions, while the PS-33D works as slave.

The following bus protocol has been defined:

- ❖ Data transfer may be initiated only when the bus is not busy.
- ❖ During data transfer, the data line must keep stable whenever the clock is HIGH level. Changes in the data line while the clock line is HIGH will be interpreted as a start or stop condition.
- ❖ Following bus conditions has been define.



- ❖ Bus not busy as condition(A)
Both data and clock lines remain HIGH.
- ❖ Start data transfer as condition(B)
A HIGH to LOW transition of the SDA line while the clock(SCL) is HIGH determines a START condition. Reading data must be began by START condition.
- ❖ Stop data transfer as condition(C)
A LOW to HIGH transition of the SDA line while the clock(SCL) is HIGH determines a STOP condition. All operation must be ended by a STOP condition.
- ❖ Data valid as condition(D)
After a START condition, the data line is stable for the duration of the HIGH period of the clock signal. The data on the line must be changed during the LOW period of the clock signal. There is one clock pulse per bit of data. The number of valid data bytes transferred between the START and STOP conditions.
- ❖ Acknowledge signal Each PS-33D receiving, when addressed, is obliged to generate an acknowledge after the reception of each byte. The processor must generate an extra clock pulse which is associated with this acknowledge bit. The PS-33D has to pull down the SDA line during the acknowledge clock pulse. The way is the SDA line is stable LOW during the HIGH period of acknowledge related clock pulse. A processor must signal an end of data to the slave by not generating an acknowledge bit on the last byte.

❑ PS-33D control address

Chart

The seven bit is as slave address after START condition. The PS-33D slave address is 1001100B (7 bits). The eighth bit of control address is read or written bit that processor wants. The processor read data sequence refers as below :

Read/Write

START

Slave address

R/W 0/1

SAK

1

0

0

1

1

0

0

I2C command Format

Writing one Byte to slave

Master	S	SAD+W <1001100 0>		Command		P
Slave			SAK		SAK	

S

START

P

STOP

SAD+W

Slave Address (1001 100) + Write bit (0)

SAD+R

Slave Address (1001 100) + Read bit (1)

A

Master acknowledge

~A

Master non-acknowledge

SAK

Slave acknowledge

I2C reading format for pressure data:

Example : Full measurement command (0xAA, Force Mode)

After written a command (0xAA), the master will start to read pressure value through I2C interface. Reading format as below :

Write 0xAA Command (Force Mode)						Conversion Time Delay	Read Pressure Data										
Master	S	SAD+W <1001100 0>	Command <10101010>	P		Delay >10ms	S	SAD+R <1001100 1>	Status <Bit 7:0>	A	Pressure Data < Bit 23:16 >	A	Pressure Data < Bit 15:8 >	A	Pressure Data < Bit 7:0 >	~A	P
Slave			SAK	SAK				SAK									

Model [PS-33D] 1000Pa_ver1.2

5 of 13

I2C Command

I2C Command

The command of pressure measurement as shown below:

Command	Description
0xAA Hex (Force Mode)	(1) When the PS-33D is written a 0xAAHex, it will turn into force mode and start measurement pressure. (2) After pressure measurement is done, the PS-33D will turn into sleep mode automatically.

NOTE:

(1) Command Delay Time:

Before next command (0xAA) write to PS-33D, the PS-33D must has a delay time is more than 10 ms for ADC conversion time, as show below:

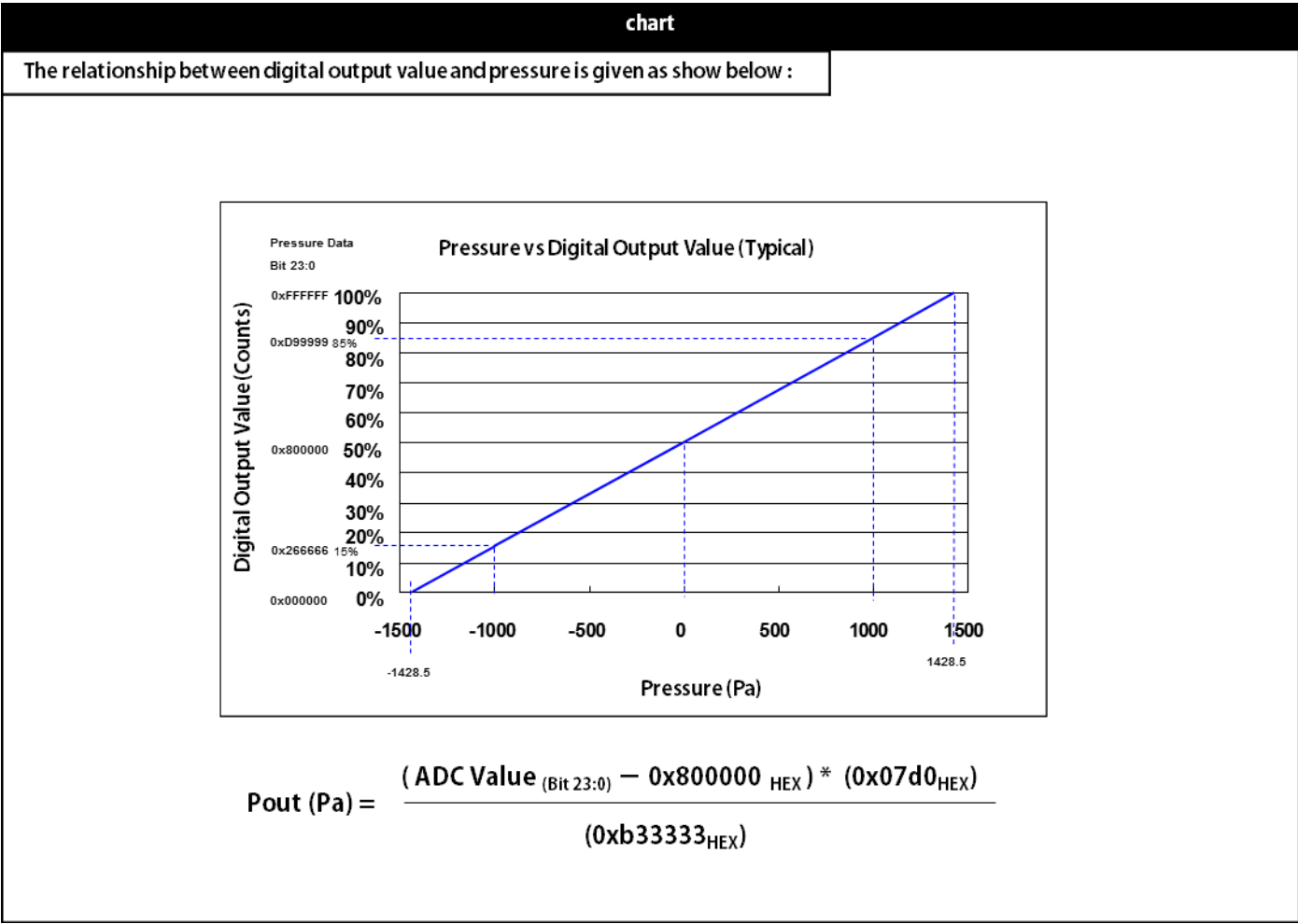
Command Delay Time Diagram

Command_1 (0xAA)	ADC Conversion Time Delay 10ms	Read Status and Pressure Data	Command_2 (0xAA)	ADC Conversion Time Delay 10ms	Read Status and Pressure Data	Command_3 (0xAA)		Command_N (0xAA)
---------------------	-----------------------------------	----------------------------------	---------------------	-----------------------------------	----------------------------------	---------------------	-------	---------------------

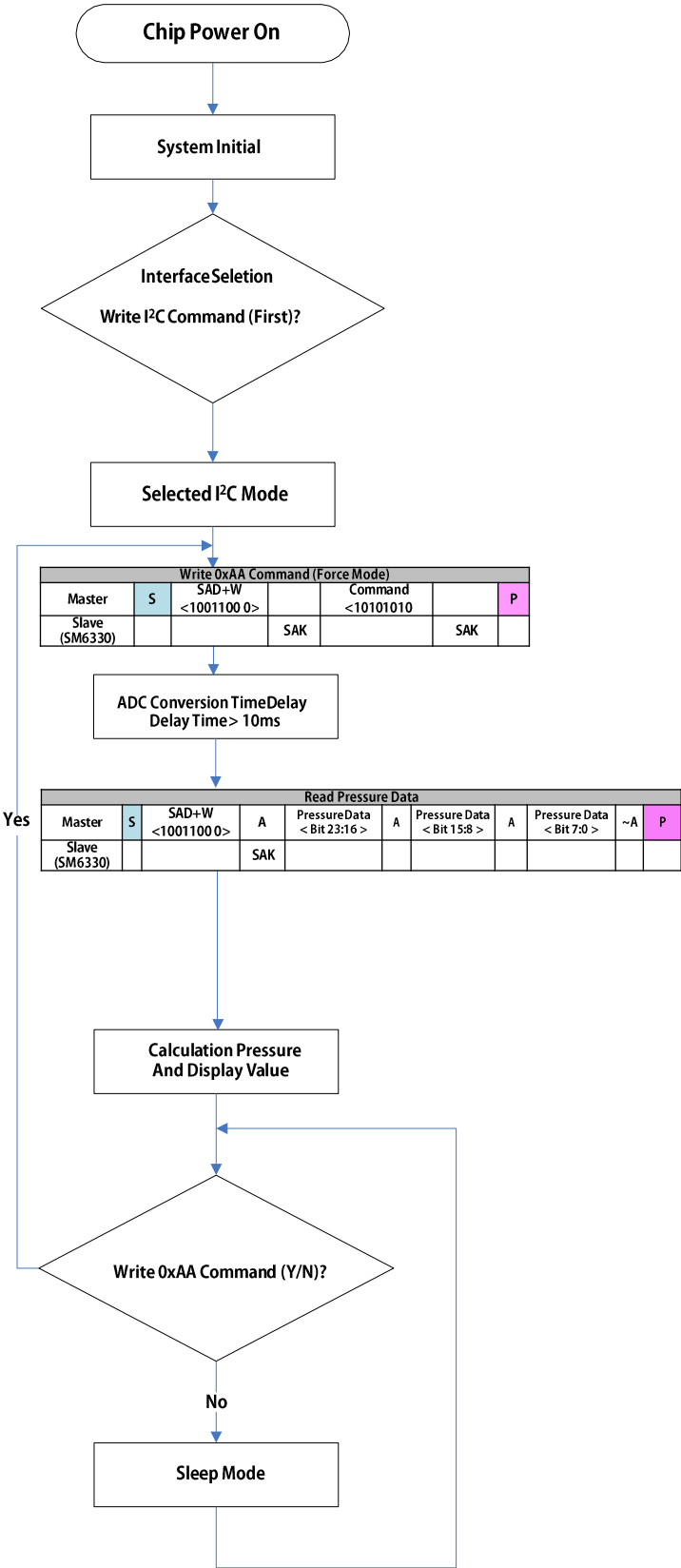
Status Register

STATUS			
Bit	Description	Attr	Default
7	Reserved	R	0
6	Power supply for ADC reference voltage: 1: Power on. 0: Power off.	R	0
5	Busy: 1: Measurement is active. 0: Sleep mode (Default after POR) * This bit is reset to zero automatically after pressure sensor measurement done in force mode.	R	0
4	Reserved	R	0
3	Reserved	R	0
2	Reserved	R	0
1	Reserved	R	0
0	Reserved	R	0

Pressure versus digital out value



❑ Pressure Measurement Operating low (I²C Mode)



❑ I2C Reading Code Example

Code Example

```

/* MAIN PRORGAM */

void main()
{
    SYSTEM_INITIAL(); // IO(I2C Mode: SS Pin Output High) , LCM Display, Memory Initail
    ms_DELAY(5); // Delay 5ms
    while (1) // Read pressure loop in force mode
    {
        CMD_WRITE(0x98,0x0aa); // Force Mode & Full Measurement.
        ms_DELAY(10); // ADC Conversion Time Delay 10ms
        IIC_read_pressure(0x99); // Read Pressure data
    }
}
//*****Sub-Program *****//
void CMD_WRITE(uint16_t sub_address,uint16_t wr_data)
{
    //===== Slave Address + Bit 0 (write=0) =====
    start();
    if((sub_address >>7) & 0x01) {SDA_DOUTSET;}
    else {SDA_DOUTCLR;} ; clock();
    if((sub_address >>6) & 0x01) {SDA_DOUTSET;}
    else {SDA_DOUTCLR;} ; clock();
    if((sub_address >>5) & 0x01) {SDA_DOUTSET;}
    else {SDA_DOUTCLR;} ; clock();
    if((sub_address >>4) & 0x01) {SDA_DOUTSET;}
    else {SDA_DOUTCLR;} ; clock();
    if((sub_address >>3) & 0x01) {SDA_DOUTSET;}
    else {SDA_DOUTCLR;} ; clock();
    if((sub_address >>2) & 0x01) {SDA_DOUTSET;}
    else {SDA_DOUTCLR;} ; clock();
    if((sub_address >>1) & 0x01) {SDA_DOUTSET;}
    else {SDA_DOUTCLR;} ; clock();
    if((sub_address >>0) & 0x01) {SDA_DOUTSET;}
    else {SDA_DOUTCLR;} ; clock();
    SACK();
    //=====write data=====
    if((wr_data >>7) & 0x01) {SDA_DOUTSET;}
    else {SDA_DOUTCLR;} ; clock();
    if((wr_data >>6) & 0x01) {SDA_DOUTSET;}
    else {SDA_DOUTCLR;} ; clock();
    if((wr_data >>5) & 0x01) {SDA_DOUTSET;}
    else {SDA_DOUTCLR;} ; clock();
    if((wr_data >>4) & 0x01) {SDA_DOUTSET;}
    else {SDA_DOUTCLR;} ; clock();
    if((wr_data >>3) & 0x01) {SDA_DOUTSET;}
    else {SDA_DOUTCLR;} ; clock();
    if((wr_data >>2) & 0x01) {SDA_DOUTSET;}
    else {SDA_DOUTCLR;} ; clock();
    if((wr_data >>1) & 0x01) {SDA_DOUTSET;}
    else {SDA_DOUTCLR;} ; clock();
    if((wr_data >>0) & 0x01) {SDA_DOUTSET;}
    else {SDA_DOUTCLR;} ; clock();
    SACK();
    stop();
}
//=====//

```

❑ I²C Reading Code Example

Code Example

```
void IIC_read_pressure(uint16_t read_address)
{
    status_reg=0; counts1=0;
    start(); // start condition
    //===== Slave Address + Bit 0(Read=1) =====
    if((read_address>>7) & 0x01) {SDA_DOUTSET;}
    else {SDA_DOUTCLR;} ; clock();
    if((read_address>>6) & 0x01) {SDA_DOUTSET;}
    else {SDA_DOUTCLR;} ; clock();
    if((read_address>>5) & 0x01) {SDA_DOUTSET;}
    else {SDA_DOUTCLR;} ; clock();
    if((read_address>>4) & 0x01) {SDA_DOUTSET;}
    else {SDA_DOUTCLR;} ; clock();
    if((read_address>>3) & 0x01) {SDA_DOUTSET;}
    else {SDA_DOUTCLR;} ; clock();
    if((read_address>>2) & 0x01) {SDA_DOUTSET;}
    else {SDA_DOUTCLR;} ; clock();
    if((read_address>>1) & 0x01) {SDA_DOUTSET;}
    else {SDA_DOUTCLR;} ; clock();
    if((read_address>>0) & 0x01) {SDA_DOUTSET;}
    else {SDA_DOUTCLR;} ; clock();
    SACK(); // master ack low
    SDA_IN ; // set sda input
    //=====Read status =====
    if (SDA==1 ) status_reg+=128; clock();
    if (SDA==1 ) status_reg+=64; clock();
    if (SDA==1 ) status_reg +=32; clock();
    if (SDA==1 ) status_reg +=16; clock();
    if (SDA==1 ) status_reg+=8; clock();
    if (SDA==1 ) status_reg+=4; clock();
    if (SDA==1 ) status_reg+=2; clock();
    if (SDA==1 ) status_reg+=1; clock();
    MACK();
    //=====Read pressure Data Bit 23:16 =====
    if (SDA==1 ) counts1+=0x800000; clock();
    if (SDA==1 ) counts1+=0x400000; clock();
    if (SDA==1 ) counts1+=0x200000; clock();
    if (SDA==1 ) counts1+=0x100000; clock();
    if (SDA==1 ) counts1+=0x080000; clock();
    if (SDA==1 ) counts1+=0x040000; clock();
    if (SDA==1 ) counts1+=0x020000; clock();
    if (SDA==1 ) counts1+=0x010000; clock();
    //===== Read pressure Data Bit 15:8=====
    MACK(); // master ack low
    if (SDA==1 ) counts1+=0x8000; clock();
    if (SDA==1 ) counts1+=0x4000; clock();
    if (SDA==1 ) counts1+=0x2000; clock();
    if (SDA==1 ) counts1+=0x1000; clock();
    if (SDA==1 ) counts1+=0x0800; clock();
    if (SDA==1 ) counts1+=0x0400; clock();
    if (SDA==1 ) counts1+=0x0200; clock();
    if (SDA==1 ) counts1+=0x0100; clock();
    MACK(); // master ack low
```

❑ I²C Reading Code Example

Code Example

```
//===== Read pressure Data Bit 7:0=====
if (SDA==1 ) counts1+=0x80; clock();
if (SDA==1 ) counts1+=0x40; clock();
if (SDA==1 ) counts1+=0x20; clock();
if (SDA==1 ) counts1+=0x10; clock();
if (SDA==1 ) counts1+=0x08; clock();
if (SDA==1 ) counts1+=0x04; clock();
if (SDA==1 ) counts1+=0x02; clock();
if (SDA==1 )) counts1+=0x01; clock();
NACK(); // master ack High
stop();

//=====Calculation Pressure output value =====//
Offset =0x800000; // offset value
negative_flag= 0; // clear negative_flag
if (counts1>= Offset)
{
// ===== Calculation pressure value , Display resolution : 0.1Pa =====//
counts1= (counts1- Offset)* (0x07d0)*10/(0xb33333);}
else
{
counts1= ~(( Offset -counts1)*(0x07d0)*10/(0xb33333))+1; // Calculation negative pressure value
negative_flag= 1;
}
Display_Pressure(counts1, negative_flag); // display pressure value
}
//=====
//
void start() // start condition //
{
SDA_OUT; // SDA set to Output mode.
SDA_DOUTSET; // SDA Output high.
SCL_DOUTSET; // SCL Output high.
NOP_DELAY(30);
SDA_DOUTCLR; // SDA output low.
NOP_DELAY(30);
SCL_DOUTCLR; // SCL output low.
NOP_DELAY(30);
}
void stop()
{
SDA_OUT; // SDA set to Output mode.
SDA_DOUTCLR;
SCL_DOUTCLR;
NOP_DELAY(30);
SCL_DOUTSET;
NOP_DELAY(30);
SDA_DOUTSET;
NOP_DELAY(30);
}
void clock()
{
SCL_DOUTSET;
NOP_DELAY(2);
NOP_DELAY(2);
SCL_DOUTCLR;
NOP_DELAY(1);
NOP_DELAY(1);
}
```

❑ I²C Reading Code Example

Code Example

```
void MACK()
{
  SDA_OUT ; // SDA set to output mode
  SDA_DOUTCLR;
  NOP_DELAY(30);
  clock();
  NOP_DELAY(30);
  SDA_IN ; // set sda input
  NOP_DELAY(30);
}
void NACK()
{
  SDA_OUT ; // SDA set to output mode
  SDA_DOUTSET; // SDA output high
  NOP_DELAY(30);
  clock();
  NOP_DELAY(30);
}
void SACK()
{
  SDA_IN ; // SDA set to input mode
  NOP_DELAY(30);
  SCL_DOUTSET;
  NOP_DELAY(30);
  ack_error_flag=SDA_N; // read ACK Signal
  clock();
  NOP_DELAY(30);
  SDA_OUT;
  NOP_DELAY(30);
}
//=====End =====///
```

☐ Document Revision History

Revision	DATE	Description
1.0	2018.01.16	First Release.
1.1	2018.09.27	SPEC changed to page11 and added page12.
1.2	2021.01.07	SPEC changed to page3.

DISCLAIMER

We hereby expressly disclaims any liability of us to any customer, licensee or any other third party, and any such customer, licensee and any other third party hereby waives any liability of us for any damages in connection with or arising out of the furnishing, performance or use of this technical data, whether based on contract, warranty, tort (including negligence), strict liability, or otherwise.

This datasheet is applies to a product under development. Its characteristics and specifications are subject to change without notice and we assumes no obligation regarding future manufacture unless otherwise agreed to in writing.

However, under no circumstances shall us be liable to any customer, licensee, or any other third party for any special, indirect, incidental, or consequential damages of any kind or nature whatsoever arising out of or in any way related to the furnishing, performance, or use of this technical data. The datasheet is furnished hereby is believed to be true and accurate.