AUAV Dual Pressure Sensor Series
Precision Airspeed and Altitude Sensing

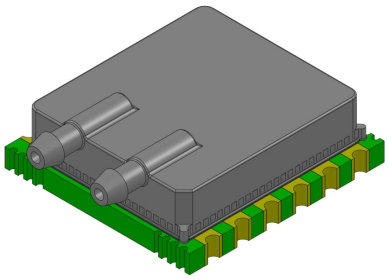


Table of Contents

The Amphenol All Sensors AUAV Series is purpose-built to meet the demanding requirements of Unmanned Aerial Vehicle (UAV) applications across a wide range of markets, including Commercial, Defense, Consumer, and Industrial Drones, as well as emerging platforms in Urban Air Mobility (UAM), Swarming Drones, and Amphibious Drone Systems. These compact sensors combine high-accuracy differential and absolute pressure sensing with advanced temperature compensation and linearity correction, enabling precise and stable airspeed and altitude measurements in mission-critical environments.

Optimized for seamless integration with flight control systems, the AUAV Series features a compact PCB-mountable module compatible with automated SMT assembly, along with both I²C and SPI digital interfaces. Multiple tubing port configurations are offered, including a low-profile side-port design requiring only two connections—ideal for minimizing space and weight in UAV builds. An optional evaluation board supports rapid prototyping and development.

Whether you're designing for high-end tactical drones, commercial delivery UAVs, hobbyist quadcopters, or next-gen autonomous air systems, the AUAV Series delivers the performance and flexibility needed to elevate your drone applications.

Contact us today to learn how the AUAV Series can enhance your next drone design.

Features

- Simultaneous Precision Measurement of Airspeed, Altitude, and Temperature
- Wide Operating Temperature Range for Harsh environments
- 18-bit high resolution I2C or SPI Output Interfaces for both sensors
- Compact and Lightweight Design weighing less than 1.5 grams [TYP]
- Surface Mount Design compatible with SMT Automated Assembly
- Fully Customizable for OEM applications

Applications

- Aerospace Systems
- Defense Drones
- Commercial Drones
- Unmanned Aerial Systems (UAS)

Standard Pressure Ranges

Device	Differential Operating Range [A]	Absolute Operating Range [A]	Proof Pressure		Burst Pressure	
			inH2O	kPa	inH2O	kPa
AUAV-L05D-M25125	± 5 inH20	250 mbarA - 1250 mbarA	270	67	415	103
AUAV-L10D-M25125	± 10 inH20	250 mbarA - 1250 mbarA	300	75	415	103
AUAV-L30D-M25125	± 30 inH20	250 mbarA - 1250 mbarA	350	87	415	103
AUAV-L60D-M25125	± 60 inH20	250 mbarA - 1250 mbarA	350	87	415	103
AUAV-L100D-M25125	± 100 inH20	250 mbarA - 1250 mbarA	350	87	415	103

Note A: Consult Factory for Additional Pressure Ranges

Pressure Sensor Maximum Ratings

Supply Voltage (Vs)	3.63 Vdc
Common Mode Pressure	15 psig
SMT Device Temperature	245°C

Environmental Specifications

Temperature Ranges

Compensated:	-40°C to 85°C
Operating	-40°C to 85 °C
Storage	-40°C to 125 °C

Humidity Limits (non condensing) 0 to 95% RH

Electrical Block Diagram

Performance Characteristics for AUAV Series High Accuracy Pressure Sensors

All Parameters are measured at 3.3V ±10% excitation and 25°C unless otherwise specified (Note 9). Differential Pressure measurements are with positive pressure applied to Port H. Absolute Pressure measurements refer to Port L.

Parameter	Symbol	Min	Typ	Max	Units	Notes
Output Span						
Differential Channel	$Span_{dig}$	-	$\pm 0.4 * 2^{24}$	-	Dec Count	1
Absolute Channel	$Span_{dig}$	-	$0.8 * 2^{24}$	-	Dec Count	1
Offset Output @ Zero Diff./ Abs. Pressure						
Differential Channel	OS_{dig}	-	$0.5 * 2^{24}$	-	Dec Count	-
Absolute Channel	OS_{dig}	-	$0.1 * 2^{24}$	-	Dec Count	-
Error Summary						
L05D - Differential Channel						
Total Error Band [85C to -20C]	TEB_P	-	-	±0.25	%FSS	2, 6
Total Error Band [-20C to -40C]	TEB_P	-	-	±0.45	%FSS	2, 6
Accuracy	Acc_P	-	-	±0.10	%FSS	3, 6
L10D - Differential Channel						
Total Error Band [85C to -20C]	TEB_P	-	-	±0.20	%FSS	2, 6
Total Error Band [-20C to -40C]	TEB_P	-	-	±0.40	%FSS	2, 6
Accuracy	Acc_P	-	-	±0.10	%FSS	3, 6
L30D / L60D / L100D - Differential Channel						
Total Error Band [85C to -20C]	TEB_P	-	-	±0.15	%FSS	2, 6
Total Error Band [-20C to -40C]	TEB_P	-	-	±0.35	%FSS	2, 6
Accuracy	Acc_P	-	-	±0.10	%FSS	3, 6
Absolute Channel - All variants						
Total Error Band [85C to -20C]	TEB_P	-	-	±0.25	%FSS	2, 6
Total Error Band [-20C to -40C]	TEB_P	-	-	±0.45	%FSS	2, 6
Accuracy	Acc_P	-	-	±0.10	%FSS	3, 6
Offset Position Sensitivity (±1g)	Sen_{Pos}	-	±0.10	-	%FSS	-
Offset Long Term Drift (one year)	**LTOS**	-	±0.25	-	%FSS	-
Pressure Digital Resolution - No Missing Codes	Res_{Padc}	17.2	-	-	bit	-
Temperature Output						
Resolution (internal)	Res_{Tadc}	-	13	-	bit	-
Overall Accuracy	Acc_T	-	-	±2	°C	-
Electrical Specification:						
Supply Voltage	V_S	2.7	3.3	3.6	Vdc	9
Supply Current Requirement						5, 7, 8
During Active State	ICC_{Active}	-	5.3	7.3	mA	-
During Idle State	ICC_{Idle}	-	1.7	7	µA	-
Power On Delay	t_{pwron}	-	-	2.5	ms	5
Memory Read Access Time	t_{rd}	-	100	2000	µs	4
Data Update Time	t_{du}		(see table below)			5, 7

Channel	Measurement Command										Units
	Single		Average2		Average4		Average8		Average16		
	Typ	Max	Typ	Max	Typ	Max	Typ	Max	Typ	Max	
Absolute	4.9	5.4	9.6	10.6	19.0	21.1	37.8	41.9	75.2	83.3	ms
Differential	2.0	2.2	3.8	4.2	7.4	8.2	14.6	16.2	29.0	32.1	ms

Specification Notes

NOTE 1: THE SPAN IS THE ALGEBRAIC DIFFERENCE BETWEEN FULL SCALE DECIMAL COUNTS AND THE OFFSET DECIMAL COUNTS. THE FULL SCALE PRESSURE IS THE MAXIMUM POSITIVE CALIBRATED PRESSURE.

NOTE 2: TOTAL ERROR BAND (TEB) IS THE COMBINATION OF ERRORS INCLUDING OFFSET, SPAN, LINEARITY, PRESSURE HYSTERESIS, TEMPERATURE EFFECT ON OFFSET, AND TEMPERATURE EFFECT ON SPAN.

NOTE 3: ACCURACY INCLUDES PRESSURE HYSTERESIS, REPEATABILITY AND BEST-FIT STRAIGHT LINE LINEARITY, EVALUATED AT 25C.

NOTE 4: DELAY BETWEEN END OF MEMORY READ REQUEST COMMUNICATION AND START OF MEMORY DATA READ COMMUNICATION.

NOTE 5: PARAMETER IS CHARACTERIZED AND NOT 100% TESTED.

NOTE 6: EVALUATED FOLLOWING CORRECTIONS DESCRIBED IN EXTENDED COMPENSATION SECTION.

NOTE 7: DATA UPDATE TIME IS EXCLUSIVE OF COMMUNICATIONS, FROM COMMAND RECEIVED TO END OF BUSY STATUS. THIS CAN BE OBSERVED AS EOC PIN LOW- STATE DURATION ON ABSOLUTE CHANNEL, OR AS DELAY TO EOC PULSE ON DIFFERENTIAL CHANNEL.

NOTE 8: AVERAGE CURRENT CAN BE ESTIMATED AS : ICC_{Idle} + (t_{DU} / READING INTERVAL) * ICCACTIVE). REFER TO FIGURE 2 FOR ACTIVE AND IDLE CONDITIONS OF THE SENSOR.

NOTE 9: THE SENSOR IS CALIBRATED WITH A 3.3V SUPPLY HOWEVER, AN INTERNAL REGULATOR ALLOWS A SUPPLY VOLTAGE OF 2.7V TO 3.6V TO BE USED WITHOUT AFFECTING THE OVERALL SPECIFICATIONS. THIS ALLOWS DIRECT OPERATION FROM A BATTERY SUPPLY.

I2C / SPI Electrical Parameters

Parameter	Symbol	Min	Typ	Max	Units	Notes
Input High Level	-	70	-	100	% of VS	5
Input Low Level	-	0	-	30	% of VS	5
Output Low Level (at 3mA sink)	-	-	-	10	% of VS	5
I2C Pull-Up Resistor	-	1000	-	-	Ω	5
I2C Load Capacitance on SDA, @ 400 kHz	C_{SDA}	-	-	200	pF	5
I2C Input Capacitance (each pin)	C_{I2C_IN}	-	-	10	pF	5
I2C Address:						
Differential Channel	Iadr $_{Diff}$	-	38	-	dec	-
Absolute Channel	Iadr $_{Abs}$	-	39	-	dec	-

Pressure Output Transfer Function - Example

$$\textit{Differential Pressure} = 1.25 \times \left(\frac{P_{dig} - 2^{23}}{2^{24}} \right) \times \text{Cal Range}$$

Where:

P dig is the sensor 24-bit output, following corrections applied by extended compensation
CalRange is 2x the calibrated differential pressure range: eg. for L05D is 10 inH20

$$\textit{Absolute Pressure} = 250 \text{ mbar} + 1.25 \times \left(\frac{P_{dig} - (0.1 \times 2^{24})}{2^{24}} \right) \times 1000 \text{ mbar}$$

Where:

P dig is the sensor 24-bit output, following corrections applied by extended compensation

Temperature Output Transfer Function

$$\textit{Temperature (*C)} = \left(\frac{Tout_{dig} * 155}{2^{24}} \right) - 45$$

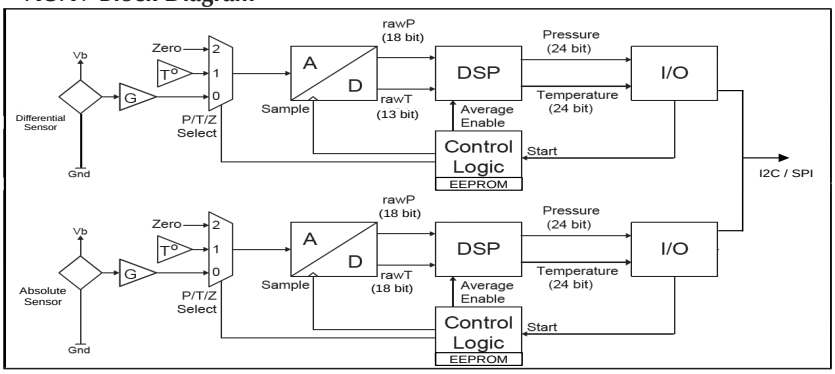
Where:

Tout (dig) The sensor 24-bit digital temperature output.

Functional Overview

The AUAV is a digital sensor with two independent signal paths that each include a sensing element, a differential amplifier, an analog to digital converter, a DSP and an IO block that supports either an I2C or SPI interface (see Figure 1 below). Each channel of the sensor also includes an internal temperature reference and associated control logic to support the configured operating mode. A multiplexer at the front end of the ADC selects the signal source for conversion.

Figure 1 - AUAV Block Diagram



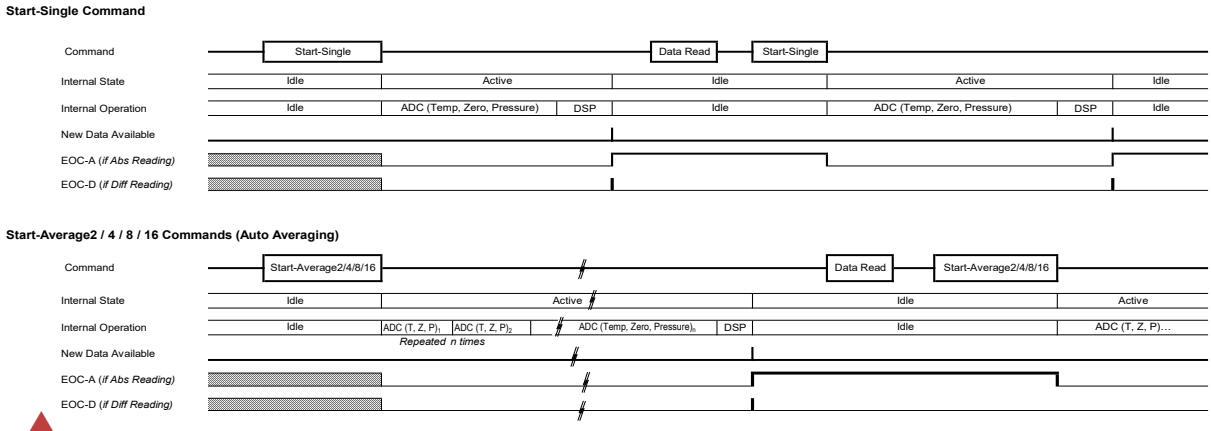
The ADC performs conversions on the raw sensor signal (P), the temperature reference (T) and a zero reference (Z) during the ADC measurement cycle.

The DSP receives the converted pressure and temperature information and applies a multi-order transfer function to compensate the pressure output. This transfer function includes compensation for span, offset, temperature effects on span, temperature effects on offset and second order temperature effects on offset. This compensated output is further improved by applying additional external correction, as described later in the Extended Compensation instructions section.

Sensor Commands: Five Measurement commands are supported, returning values of either a single pressure / temperature reading or an average of 2, 4, 8, or 16 readings. Each of these commands wakes the sensor from Idle state into Active state, and starts a measurement cycle. For the Start-Average commands, this cycle is repeated the appropriate numper of times, while the Start-Single command performs a single iteration. When the DSP has completed calculations and the new values have been made available to the I/O block, the sensor returns to Idle state.
The sensor remains in this low-power state until another Measurement command is received. Each channel of the AUAV operates independently in this manner, providing flexibility for system data acquisition design.

After completion of the measurement, the result may then be read using the Data Read command. The ADC and DSP remain in Idle state, and the I/O block returns the 7 bytes of status and measurement data (See Figure 2). At any time, the host may request current device status with the Status Read command.

Figure 2 - AUAV Communication Model



I2C / SPI Interface Overview

I/O Interface Configuration: The sensor automatically selects SPI or I2C serial interface, based on the following protocol: If the /SS input is set low by the host (as occurs during a SPI command transaction), the I/O interface will remain configured for SPI communications until power is removed. Otherwise, once a valid device address and command have been received over the I2C interface, the I/O interface will remain configured for I2C until power is removed.

See Initialization Instructions (page 12) for further information.

I2C Bus Communications Overview

The 2-wire I2C interface uses a standard set of signal sequences for communication.
Refer to Figure 3 for the associated usage of the following signal sequences; timing requirements are shown in Figure 5.

Bus not Busy (I): During idle periods both data line (SDA) and clock line (SCL) remain HIGH.

START condition (ST): A HIGH to LOW transition of SDA line while the clock (SCL) is HIGH is interpreted as START condition. START conditions are always set by the master. Each command request has to begin with a start condition.

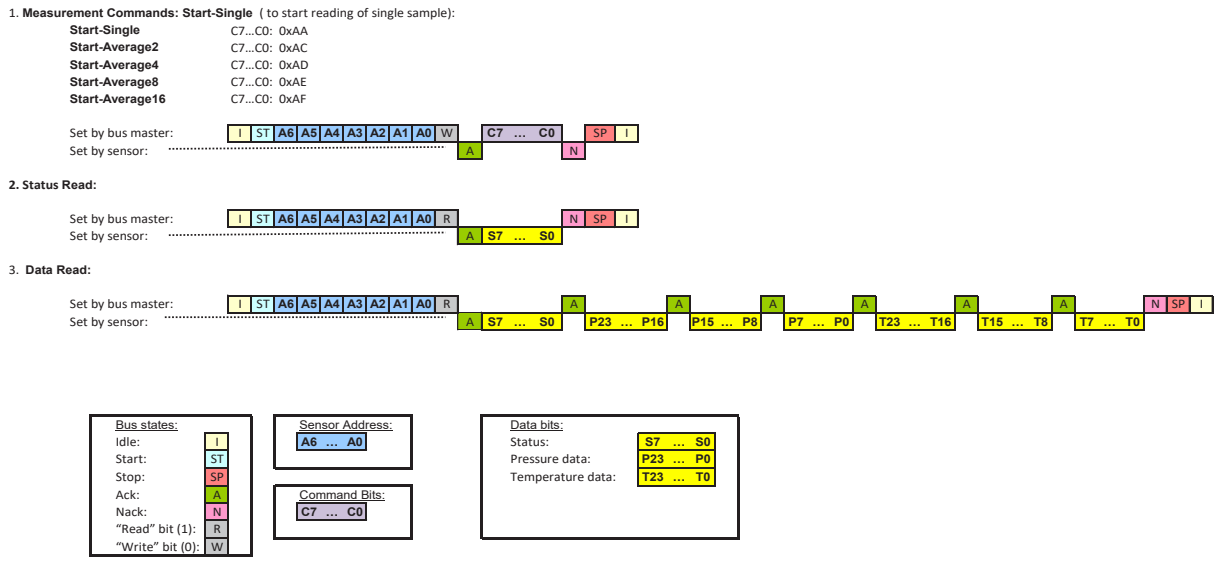
Slave address (An): The I2C-bus requires a unique address for each device. The AUAV sensor has a preconfigured slave address for each channel (defined previously in the Specifications section). After setting a START condition the master sends the address byte containing the 7 bit sensor address followed by a data direction bit (R/W). A "0" indicates a transmission from master to slave (WRITE), a "1" indicates a device-to master request (READ).

Acknowledge (A or N): Data is transferred in units of 8 bits (1 byte) at a time, MSB first. Each data-receiving device, whether master or slave, is required to pull the data line LOW to acknowledge receipt of the data. The Master must generate an extra clock pulse for this purpose. If the receiver does not pull the data line down, a NACK condition exists, and the slave transmitter becomes inactive. The master determines whether to send the last command again or to set the STOP condition, ending the transfer.

DATA valid (Dn): State of data line represents valid data when, after a START condition, data line is stable for duration of HIGH period of clock signal (SCL). Data line SDA must be changed during LOW period of SCL. There is one clock pulse per data bit, with data value on SDA captured on the rising edge of SCL.

STOP condition (P): LOW to HIGH transition of the SDA line while clock (SCL) is HIGH indicates a STOP condition. STOP conditions are always generated by the master.

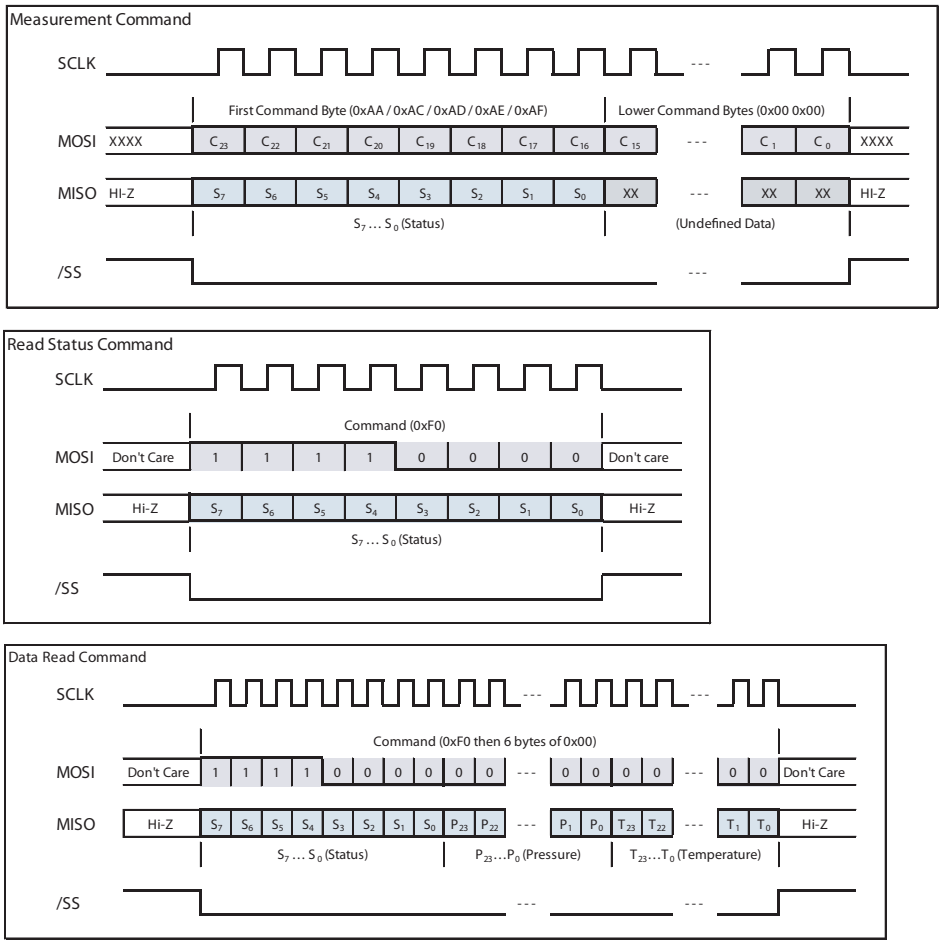
Figure 3 - I2C Communication Diagram



SPI Bus Communications Overview

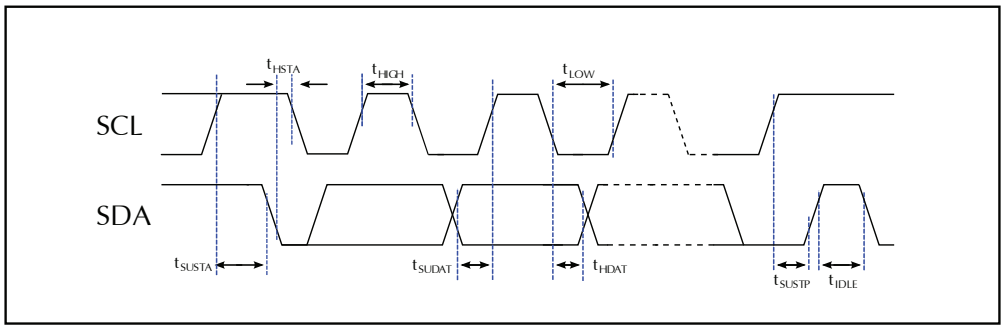
The sequence of bits and bus signals are shown in the following illustration (Figure 4). Refer to Figure 5 in the Interface Timing Diagram section for detailed timing data. Note that /SS in Figure 4 refers to /SSA or /SSD.

Figure 4 - SPI Communications Diagram



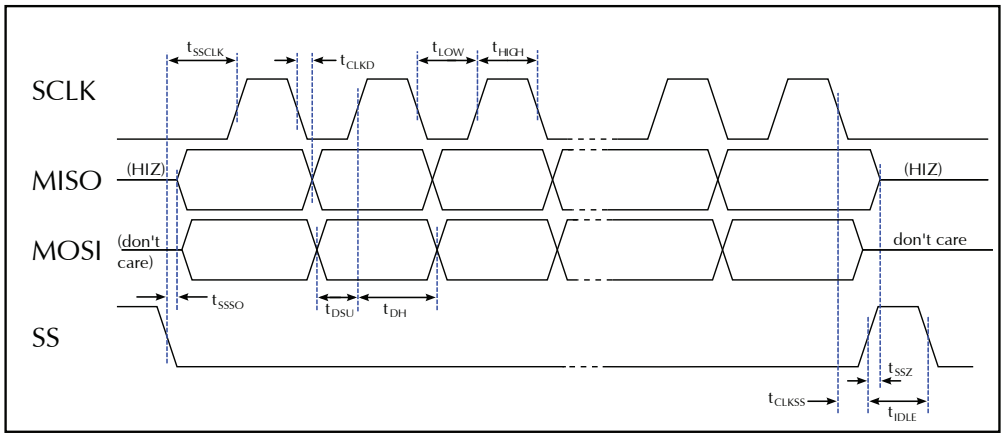
Interface Timing Diagrams

Figure 5 - I2C Timing Diagram



PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS
SCL frequency	f_{SCL}	100	-	400	KHz
SCL low width	t_{LOW}	1.3	-	-	us
SCL high width	t_{HIGH}	0.6	-	-	us
Start condition setup	t_{SUSTA}	0.6	-	-	us
Start condition hold	t_{HSTA}	0.6	-	-	us
Data setup to clock	t_{SUDAT}	0.1	-	-	us
Data hold to clock	t_{HDAT}	0	-	-	us
Stop condition setup	t_{SUSTP}	0.6	-	-	us
Bus idle time	t_{IDLE}	2.0	-	-	us

Figure 6 - SPI Timing Diagram



PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS
SCLK frequency	f_{SCLK}	0.05	1	5	MHz
SCLK duty cycle [$t_{HIGH}/(t_{LOW}+t_{HIGH})$]	D_{SCLK}	40	50	60	%
/SS low to first rising clock edge	t_{SSCLK}	5	50	--	ns
/SS low to MISO enabled	t_{SSSO}	--	--	20	ns
/SS high to MISO disabled	t_{SSZ}	--	--	20	ns
Clock to MISO data out	t_{CLKD}	8	--	20	ns
MOSI setup to rising clock	t_{DSU}	20	--	--	ns
MOSI hold after rising clock	t_{DH}	20	--	--	ns
Last falling clock to rising /SS	t_{CLKSS}	0	--	--	ns
Bus idle time	t_{IDLE}	10	--	--	µs

Operational Overview

Digital Interface Command Formats

When requesting the start of a measurement or an EEPROM data reading, the command length for I2C is 1 byte, for SPI it is 3 bytes.
When requesting sensor status over I2C, the host simply performs a 1-byte read transfer.
When requesting sensor status over SPI, the host must send the Status Read command byte while reading 1 byte.
When reading sensor data over I2C, the host simply performs a 7-byte read transfer.
When reading sensor data over SPI, the host must send the 7-byte Data Read command while reading the data.

SENDING UNDOCUMENTED COMMANDS TO SENSOR WILL CORRUPT CALIBRATION AND IS NOT COVERED BY WARRANTY.

See Table 1 below for Measurement Commands, Sensor Data read and Sensor Status read details.
Note that each channel of the AUAV sensor independently processes commands, as selected by I2C address or by SPI/SS line.

Table 1 - AUAV Sensor Reading Command Set

Measurement Commands				
Description	**SPI (3 bytes)**			**I2C (1 byte)**
Start-Single	0xAA	0x00	0x00	0xAA
Start-Average2	0xAC	0x00	0x00	0xAC
Start-Average4	0xAD	0x00	0x00	0xAD
Start-Average8	0xAE	0x00	0x00	0xAE
Start-Average16	0xAF	0x00	0x00	0xAF

Read Sensor Data	
I2C	Read of 7 bytes from device
SPI	Read of 7 bytes from device ***Host must send [0xF0], then 6 bytes of [0x00] on MOSI*** *Sensor Returns 7 bytes on MISO*

Read Sensor Status	
I2C	Read of 1 byte from device.
SPI	Read of 1 byte from device ***Host must send [0xF0] on MOSI*** *Sensor Returns 1 byte on MISO*

Digital Interface Command Formats (Cont'd)

The Memory Read Command is used to retrieve the extended Compensation Coefficients from internal memory of the sensor. As described more fully in the Extended Compensation section, these are referred to as A, B, C, D, Es, TC50H, TC50L.
Values (A, B, C, and D) are 32-bit signed integers, stored in eight 16-bit registers in the EEPROM of each channel.
Values Es, TC50H, TC50L are 8 bit signed integers, stored in two 16-bit EEPROM locations.
The memory map of each channel is shown in Table 2:

Table 2 - Coefficient Memory Map

Absolute Channel:

Address	47 (0x2F)	48 (0x30)	49 (0x31)	50 (0x32)	51 (0x33)	52 (0x34)	53 (0x35)	54 (0x36)	55 (0x37)	56 (0x38)
Coeff. Word	[AHW] A (High Word)	[ALW] A (Low Word)	[BHW] B (High Word)	[BLW] B (Low Word)	[CHW] C (High Word)	[CLW] C (Low Word)	[DHW] D (High Word)	[DLW] D (Low Word)	[TC50H} [TC50L}	[0] [Es}

Differential Channel:

Address	43 (0x2B)	44 (0x2C)	45 (0x2D)	46 (0x2E)	47 (0x2F)	48 (0x30)	49 (0x31)	50 (0x32)	51 (0x33)	52 (0x34)
Coeff. Word	[AHW] A (High Word)	[ALW] A (Low Word)	[BHW] B (High Word)	[BLW] B (Low Word)	[CHW] C (High Word)	[CLW] C (Low Word)	[DHW] D (High Word)	[DLW] D (Low Word)	[TC50H} [TC50L}	[0] [Es}

The sequence of commands to retrieve these values is in the form of a Memory Read Request (See Table 3) followed by a Memory Data Read (See Table 4). Note that the Memory Read Access Time delay must be observed between the request and the read operations.

Table 3 - Memory Read Request Command

Memory Commands: I2C or SPI:			
Description	**Command (3 bytes)**		
Read Request *Absolute Channel* *Differential Channel*	<EEPROM Address> *(Values 47 -56 only)* *(Values 43 -52 only)*	0x00	0x00

It must be emphasized that these commands be used accurately and carefully. Errors in forming or transmitting these commands can result in overwriting calibration data and degraded sensor operation.

Table 4 - Memory Data Read Operation

Read Memory Data	
I2C	Read of 3 bytes from device.
SPI	Read of 3 bytes from device. ***Host must send [0xF0], then 2 bytes of [0x00] on MOSI.*** *Sensor returns 3 bytes on MISO.*

*Example : I2C Read of Coefficient **B** from Absolute channel EEPROM :*

Write <**0x31**> , and read back: <Status> <**BHW**>. (high 16 bits of B, as 2 bytes)
Write <**0x32**>, and read back: <Status> <**BLW**>. (low 16 bits of B, as 2 bytes)
B = [BHW:BLW], assembling BHW and BLW into a signed 32-bit integer.

*Example : SPI Read of Coefficient **D** from Differential channel EEPROM :*

Write <**0x31**><0x00><0x00>, over SPI MOSI output.
Set output buffer to <**0xF0**><0x00><0x00>, then perform 3-byte transfer.
Input buffer will then contain: <Status> < $\mathbf{DHW}_{(\text{high byte})}$ > < $\mathbf{DHW}_{(\text{low byte})}$ >.
Write <**0x32**><0x00><0x00>, over SPI MOSI output.
Set output buffer to <**0xF0**><0x00><0x00>, then perform 3-byte transfer.
Input buffer will then contain: <Status> < $\mathbf{DLW}_{(\text{high byte})}$ > < $\mathbf{DLW}_{(\text{low byte})}$ >.
D = [DHW:DLW], assembling DHW and DLW into a signed 32-bit integer.

Digital Interface Data Format

For either type of digital interface, the format of data returned from the sensor is the same. For measurement data, the first byte consists of the Status Byte followed by a 24-bit unsigned pressure value and a 24-bit unsigned temperature value. See the Pressure Output Transfer Function and Temperature Output Transfer Function definitions for converting to pressure and temperature. Note that the sensor output includes error terms that must be removed by using the Extended Compensation adjustments. Refer to *'Extended Compensation Instructions Section'* for these computations.

For memory data readings, the status byte is followed by the high byte, then low byte of the memory word.

Refer to Table 5 for the measurement data format of the sensor. Table 6 shows the form of EEPROM-read data, and Table 7 shows the Status Byte definition.

Note that a completed reading without error will return status 0x40 for the Absolute channel, 0x50 for the Differential channel.

Table 5 - Measurement Output Data Format

S[7:0]	P[23:16]	P[15:8]	P[7:0]	T[23:16]	T[15:8]	T[7:0]
Status Byte	Pressure Byte 2	Pressure Byte 1	Pressure Byte 0	Temperature Byte 2	Temperature Byte 1	Temperature Byte 0

Table 6 - Memory Data Output Format

S[7:0]	MEM[15:8]	MEM[7:0]
Status Byte	MEM High Byte	MEM Low Byte

Table 7- Status Byte Definition

Bit	Description
Bit 7 [MSB]	[Always = 0]
6	Power: [1 = Power On]
5	Busy: [1 = Processing Command, 0 = Ready]
4:3	4Mode: [00 = Normal Operation, others = Command Fault]
2	Memory Error [1 = EEPROM Checksum Fail] Sensor
1	Configuration [always = 0]
Bit 0 [LSB]	ALU Error [1 = Error: Pressure or Temperature Overrange or Underrange]

I2C and SPI Command Sequence

I2C Command Sequence

The part enters Idle state after power-up, and waits for a command from the bus master. Any of the five Measurement commands may be sent, as shown in Table 1. Following receipt of one of these command bytes, the EOC-A pin is set to Low level (for the Absolute channel only), and the sensor Busy bit is set in the Status Byte. After completion of measurement and calculation in the Active state, compensated data is written to the output registers for access through the digital interface. For the Absolute channel the EOC-A pin is set high, remaining high until the following command is received. For the Differential channel, a 5 microsecond high – level pulse is set on EOC-D. For either channel, the BUSY bit is cleared and the processing core goes back to Idle state. The host processor can then perform the Data Read operation, which for I2C is simply a 7-byte Device Read.

If the EOC pin is not monitored, the host can poll the Status Byte by repeating the Status Read command, which for I2C is a one-byte Device Read. When the Busy bit in the Status byte is zero, this indicates that valid data is ready, and a full Data Read of all 7 bytes may be performed.

DO NOT SEND COMMANDS TO SENSOR OTHER THAN THOSE DEFINED IN TABLES 1, 3 & 4.

SPI Command Sequence

As with the I2C interface configuration, the part enters Idle state after power-up, and waits for a command from the SPI master. To start a measurement cycle, one of the 3- byte Measurement Commands (see Table 1) must be issued by the master. To start a memory read operation, the memory read request (see Table 3) must be sent. The data returned by the sensor during this command request consists of the Status Byte followed by two undefined data bytes.

On successful decode of a measurement command, for the Absolute channel only, the EOC-A pin is set Low as the core goes into Active state for measurement and calculation. When complete, updated sensor data is written to the output registers, and the core goes back to the Idle state. For the Absolute channel, the EOC-A pin is set to a High level at this point, and the Busy status bit is set to 0. For the Differential channel, the EOC-D pin provides a 5-microsecond High pulse at this point, and the Busy status bit is set to 0. At any point during the Active or Idle periods, the SPI master can request the Status Byte by sending a Status Read command (a single byte with value 0xF0).

As with the I2C configuration, a Busy bit of value 0 in the Status Byte (or a high level on the EOC-A pin for the Absolute channel) indicates that a valid data set may be read from the sensor. The Data Read command must be sent from the SPI master (The first byte of value 0xF0 followed by 6 bytes of 0x00). For memory read operations, see Table 4 for reading back the result.

Initialization Instructions

To ensure reliable operation over all operating conditions, the following system design items are recommended:

1. After power-on, for SPI communications, the following signal sequence ensures an interface lock to SPI (this is not necessary for I2C applications):
    a. Set /SSA to LOW state, delay 1 msec, set /SSA to HIGH state.
    b. Set /SSD to LOW state, delay 1 msec, set /SSD to HIGH state.
2. Perform a single reading on each channel:
    a. Send the Start-Single command to each channel (See Table 1).
    b. Delay for completion > 10 msec;
    c. Read and discard the result (See Table 1).
3. Proceed with EEPROM readings to obtain the Extended Compensation coefficients.
    a. At elevated temperatures, observe the maximum EEPROM reading access times, which will be greater than the Typical times.
    b. If a BUSY status is returned with the EEPROM result, retry with a longer delay between the Read Request and readback of the result.

Extended Compensation Instructions

AUAV Series sensors have internal memory locations containing extended compensation coefficients. For optimal accuracy of pressure readings, system designers can use these values to apply an additional 3rd-order error-correction adjustment to data delivered from the sensor, as well as additional temperature compensation.

Theory of Extended Compensation:

The four linearity coefficients are programmed for each channel of the sensor at the factory as a 3rd order minimization solution to

(1) Error = Pref - (POut + f(POut)),

where

Pref is the true pressure applied;

POut is the sensor output;

f(POut) is a cubic correction function, Ax3+Bx2+Cx+D.

Then

(2) Pcorr = Pout + f(Pout) as the linearity-corrected pressure value.

For improved accuracy over temperature, residual temperature dependent errors are minimized by the term:

(3) TCadj = (1 - (Es * 2.5 * | 0.5 - Pcorr |)) * (T - Tref) * TC50

where:

TC50 = TC50H/TC50Scale for T > Tref

TC50 = TC50L/TC50Scale for T < Tref

and

TC50Scale = 100 * 100 * 167772.

This represents the possible range of temperature-dependent error, scaled to temperature counts.

Then

(4) Pcomp = Pcorr – Tcadj

for the final optimized pressure value. This is used in the Pressure Output Transfer Function on Page 4 to obtain pressure in appropriate units.

Implementation: How to use:

On system startup:

Read the seven coefficients (A, B, C, D, Es, TC50H, & TC50L) from sensor EEPROM for each channel, using the command sequence described in the datasheet section 'Digital Interface Command Formats'.

A, B, C & D are 32-bit signed integers, representing a scaled magnitude from -1.0 to +1.0.

Es, TC50H, & TC50L are 8-bit signed integers, representing a scaled magnitude from -1.0 to +1.0.

Code Example:

Declarations:

```
extern unsigned char inbuf[], outbuf[];  // sensor
float ALIN_A , ALIN_B, ALIN_C, ALIN_D, A_Es, A_TC50H, A_TC50L;  // Abs coeffs
float DLIN_A , DLIN_B, DLIN_C, DLIN_D, D_Es, D_TC50H, D_TC50L;  // Diff coeffs

// Platform-dependent SPI fns, which perform EEPROM read of specified
// address and following address. These 16b values are merged as [HW:LW]
// for an int32 return.
extern int32_t Read_ACfg_SPI(uint8_t regadr);
extern int32_t Read_DCfg_SPI(uint8_t regadr);

void Init_Aconfig(void);
void Init_Dconfig(void);
```

Extended Compensation Instructions (Cont'd)

After sensor power-on, read EEPROM coefficient values, convert signed 32-bit integers to Float:

For Absolute channel:

```
void Init_Aconfig(void)
{
    int32_t i32A = 0, i32B =0, i32C =0, i32D=0,  i32TC50HLE=0;
    int8_t i8TC50H = 0, i8TC50L = 0, i8Es = 0;

       // These i32 Reads return 2 register values merged as int32
       // i32 then normalized to +/- 1.0

    i32A = Read_ACfg_SPI(47);     // returns 47 | 48 as int32
    ALIN_A = ((float)(i32A))/((float)(0x7FFFFFFF));

    i32B = Read_ACfg_SPI(49);       // returns 49 | 50 as int32
    ALIN_B = (float)(i32B)/(float)(0x7FFFFFFF);

    i32C = Read_ACfg_SPI(51);       // returns 51 | 52 as int32
    ALIN_C = (float)(i32C)/(float)(0x7FFFFFFF);

    i32D = Read_ACfg_SPI(53);       // returns 53 | 54 as int32
    ALIN_D = (float)(i32D)/(float)(0x7FFFFFFF);

    i32TC50HLE = Read_ACfg_SPI(55);  // 55:  TC50H | TC50L
    i8TC50H = (i32TC50HLE >> 24)  & 0xFF;                          // 55 H
    i8TC50L = (i32TC50HLE >> 16)  & 0xFF;                          // 55 L
    i8Es =     (i32TC50HLE )  & 0xFF;                              // 56 L

    A_Es     = (float)(i8Es)/(float)(0x7F);
    A_TC50H = (float)(i8TC50H)/(float)(0x7F);
    A_TC50L = (float)(i8TC50L)/(float)(0x7F);
}
```

Differential channel function *Init_Dconfig()* would be similar, but would use EEPROM addresses 43 – 52 in lower- level function *Read_DCfg_SPI()* and initialize DLIN_*, D_Es, D_TC50H and D_TC50L.

These floating-point constant values for A, B, C, D, Es, TC50H and TC50L are used in adjustments applied to each reading, described as follows:

Correction applied to each reading:

For each pressure value read from the sensor (POut), calculate

PComp = POut + A*POut^3 +B*POut^2 + C*POut +D - TCadj.

If the application operates in an environment consistently near 25C, TCadj is an optional term.

Example: Differential Channel Reading, using I2C:

Start Differential sensor reading:

```
outbuf[0] = 0xAD; ui8Address = 38; // Request 4x oversampled reading
// Platform-dependent I2C write function, for example
I2C_Write(ui8Address, outbuf, 1); // send 1-byte request
```

After conversion delay (or on EOC-D pulse received), read result:

```
I2C_Read(ui8Address, inbuf, 7); // read 7 bytes: Status, P, T
```

Extended Compensation Instructions (Cont'd)

The sensor 24-bit pressure and temperature values are now in the I2C input buffer, preceded by the status byte. That is, inbuf[0] = Status Byte, inbuf[1], [2], [3] contain the pressure value, and inbuf[4], [5],[6] contain temperature.

Now, apply the corrective adjustments, using coefficients read from the Differential channel EEPROM:

Declarations & definitions:

```
// constants:
const int32_t Tref_Counts = 7576807; // temperature counts at 25C
const float TC50Scale = 100.0 * 100.0 * 167772.2 // scale TC50 to 1.0% FS0

// local variables:
float AP3, BP2, CP, Corr, Pcorr, Pdiff, TCadj, TC50, Pnfso, Tcorr, Pcorrt;
int32_t iPraw, Tdiff, iTemp, iPCorrected;
uint32_t PComp;
```

First, correct for linearity, as in Equation (2):

```
// Convert unsigned 24-bit pressure value to signed +/- 23-bit:
iPraw = (inbuf[1]<<16) + (inbuf[2]<<8) + inbuf[3] - 0x800000;
// Convert signed 23-bit value to float, normalized to +/- 1.0:
Pnorm = (float)iPraw;       // cast to float
Pnorm /= (float) 0x7FFFFF;

AP3 = DLIN_A * Pnorm * Pnorm * Pnorm;    // A*Pout^3
BP2 = DLIN_B * Pnorm * Pnorm;            // B*Pout^2
CP = DLIN_C * Pnorm;                     // C*POut
Corr = AP3 + BP2 + CP + DLIN_D;          // Linearity correction term
Pcorr = Pnorm + Corr;                    // Corrected P, range +/-1.0.
```

At this point, Pcorr represents the linearity-corrected pressure, optimized at 25C temperature.
For room-temperature applications, this may be sufficient compensation.

Converting back to the sensor native format of unsigned 24-bit integer:

```
iPcorr = (int32_t)(Pcorr * (float)0x7FFFFF);    // Convert to signed 23-bit
iPcorr += 0x800000;                             // Back to unsigned 24-bit
```

The Pressure Output Transfer Function on Page 4 can then be applied, using iPcorr as the value for Pdig.

For systems exposed to temperatures significantly above or below 25C, additional temperature compensation may be applied, as shown below:

```
// Compute difference from reference temperature, in sensor counts:
iTemp = (inbuf[4]<<16) + (inbuf[5]<<8) + inbuf[6];   // 24-bit temperature
Tdiff = iTemp - Tref_Counts; // see constant defined above.
```

Re-normalize the linearity-corrected pressure from +/- 1.0 to [0 – 1.0):

```
Pnfso = (Pcorr + 1.0)/2.0;

//TC50: Select High/Low, based on current temp above/below 25C:
if (Tdiff > 0)
      TC50 = D_TC50H;
else
      TC50 = D_TC50L;
// Find absolute difference between midrange and reading (abs(Pnfso-0.5)):
if (Pnfso > 0.5)
      Pdiff = Pnfso - 0.5;
else
      Pdiff = 0.5 - Pnfso;
```

Extended Compensation Instructions (Cont'd)

Now, the temperature-dependent adjustment as a function of pressure and temperature difference from reference values as in Equation (3):

```
Tcorr = (1.0 - (D_Es * 2.5 * Pdiff)) * Tdiff * TC50 / TC50Scale;
```

this adjustment is applied to the linearity-corrected value, as in Equation(4):

```
PCorrt = Pnfso - Tcorr; // corrected P: float, [0 to +1.0)
```

Re-normalize back to unsigned 24-bit value, finishing Equation (4):

```
Pcomp = (uint32_t) (PCorrt * (float)0xFFFFFF);
```

Start next sensor reading:

```
outbuf[0] = 0xAD;                          // Request 4x oversampled reading
success = I2C_Write(ui8Address, outbuf, 1) // send 1-byte request
```

Convert to pressure units:

The P_{comp} result represents the corrected output of the sensor, to be used in the Pressure Output Transfer Function as P_{dig} when computing pressure in calibrated measurement units.

For illustration, continuing with the Differential channel example:
For a sensor with a differential calibrated range of +/- 10 inH2O,

$$P_{inH2O} = 1.25 * ((P_{dig} - 2^{23})/ 2^{24}) * (2 * 10) inH_2O$$

If a reading from this sensor results in P_{comp} = 9145890 counts,

$$P_{inH2O} = 1.25 * ((9145890 – 8388608)/ 16777216) * 20\ inH_2O$$
$$= 1.1284\ inH2O$$

For a compensated reading of P_{comp} = 3000000 counts,

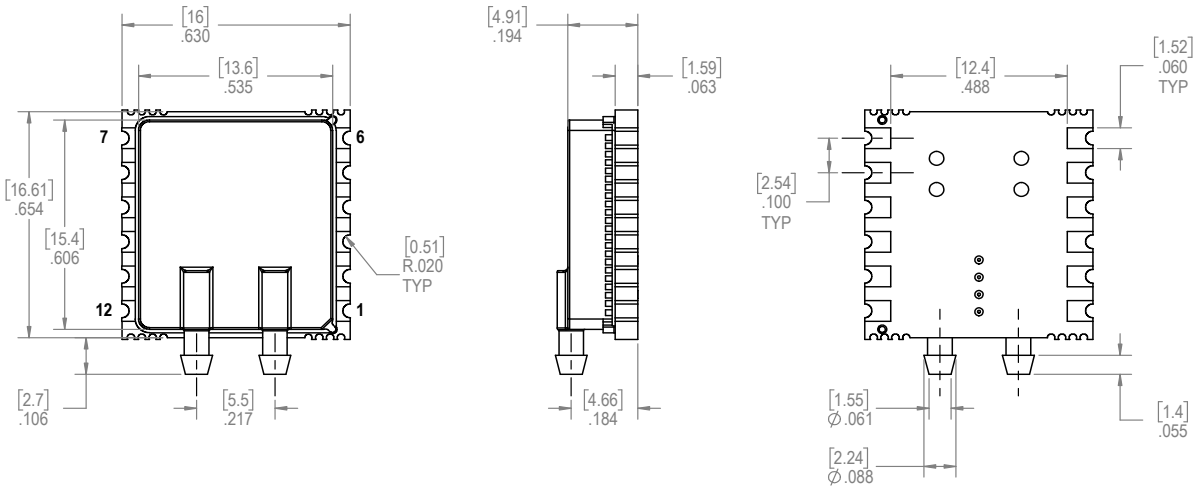
$$P_{inH2O} = 1.25 * ((3000000 – 8388608)/ 16777216) * 20\ inH_2O$$
$$= -8.0297\ inH2O$$

For the Absolute channel, the computations are identical to obtain P_{comp}, but using the set of coefficients ALIN_A , ALIN_B, ALIN_C, ALIN_D, A_Es, A_TC50H, and A_TC50L.

The transfer function differs in offset value from that of the Differential channel,
so (for example) a P_{comp} value of 9145890 counts would represent an absolute pressure of:

$$P_{mbarA} = 250\ mbarA + 1.25 * (P_{dig} - (0.1 * 2^{24})/ 2^{24}) * 1000\ mbarA$$
$$P_{mbarA} = 250\ mbarA + 1.25 * ((9145890 – 1677722)/ 16777216) * 1000\ mbarA$$
$$= 806.42\ mbarA$$

Package Drawing

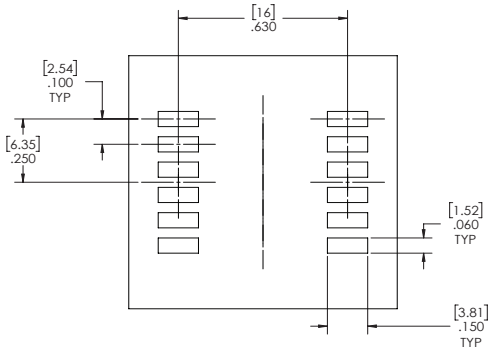


Notes (Unless Otherwise Specified)
1) Dimensions are in inches [mm].
2) TEST Pin: Pull up to V_s for EMI immunity.
3) Product Mass: 1.36g ± 15%.
4) Measurement [16.61] / .654 is measured from the middle flat milled surface edge.

Standard Port Option
Consult Factory for Additional Port Options

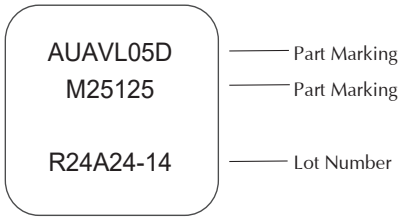
Pin	Symbol	Direction	Description
1	SCL/SCLK	Input	Clock for I2C or SPI usage.
2	SDA/MOSI	I/O, Input	Data I/O for I2C, Data Input for SPI
3	EOC-A	Output	End Of Conversion for Absolute: High level when reading complete.
4	/SSA	Input	For SPI usage, SlaveSelect to enable Absolute channel output. Has weak internal pullups; for noise immunity 10K-47K pull-up to Vs is recommended.
5	GND	Supply	Power supply ground: a low inductance path to supply return plane is recommended.
6	Vs	Supply	Power supply input: external decoupling may be improve noise on output, depending on source impedance.
7	n/c	n/a	Not connected.
8	GND	Supply	Power supply ground: a low inductance path to supply return plane is recommended.
9	TEST	Input	Manufacturing test pin: recommended to pull up to Vs with 10K - 47K for noise immunity.
10	EOC-D	Output	End Of Conversion for Differential: 5 usec positive pulse when reading complete.
11	/SSD	Input	For SPI usage, SlaveSelect to enable Differential channel output. Has weak internal pullups; for noise immunity 10K-47K pull-up to Vs is recommended.
12	MISO	Output	Data output for SPI interface.

Pad Layout



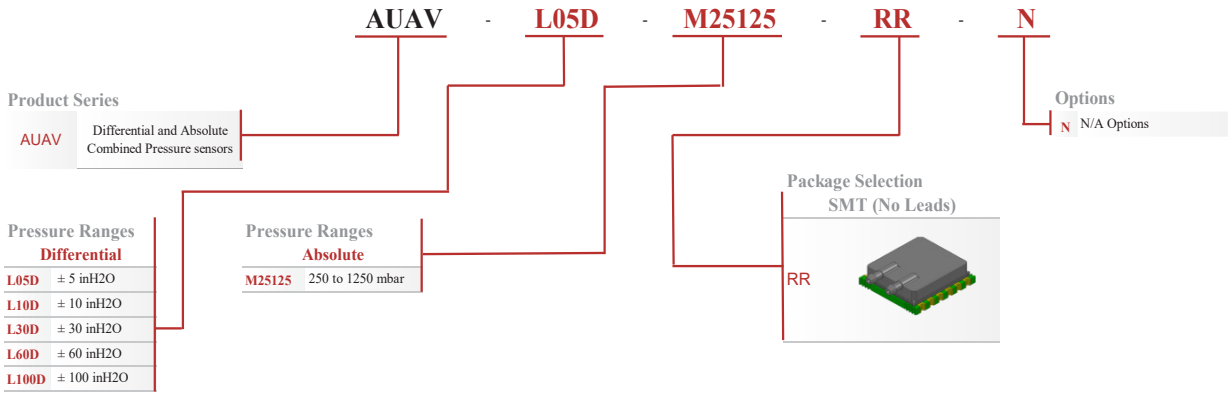
Product Identification

Figure 7: Product Marking Example



How to Order AUAV Series

For example, **AUAV-L05D-M25125-RR-N** defines an All Sensors AUAV Series pressure sensor with 5 inH2O differential pressure and 250 to 1250 mbar absolute pressure range, RR package (SMT with two barbed side ports and no coating options.



Additional Application Information

Assembly Recommendations:

AUAV devices are non-hermetic and should be handled in accordance with IPC/JEDEC standard J-STD-020 classification MSL-4: SMT assembly or other high-temperature processing must be performed within 72 hours of opening factory-sealed packaging, in an environment of < 30C / 60% RH.

Beyond this interval, a bake process at 125C for 24h is required.

If a PCBA cleaning process is required, parts must not be immersed or subject to compressed air. If necessary, manual cleaning is recommended. No foreign matter of any kind may enter the pressure ports.

If secondary system calibration is to be performed, allow 48 hours for settling after SMT exposure.

Evaluation Kit:

All Sensors offers an evaluation kit for AUAV to help design engineers evaluate performance of the sensors early in the development process.

Evaluation Board Part Number: EK-AUAV01

AppNote Reference:

AN-0019 Airspeed and Altitude Calculation