

DIGITAL PITOT PROBE SYSTEM

User Manual



WARNING

Read this document before using the product.

This probe is an experimental prototype, for measurement purposes only.

This system is not certified for use on aircraft.

Contents

1	INTRODUCTION.....	3
2	PHYSICAL CONNECTIONS.....	8
3	CARE AND HANDLING.....	10
4	SOFTWARE AND DRIVERS.....	11
5	TECHNICAL SUPPORT.....	16
	Appendix 1: Additional communications details.....	17
	Appendix 2: Hardware triggering.....	26

Version Control

Version	Date	Summary of changes	Software ver.
1.0	06-2026	New document	1.0

1 INTRODUCTION

Principle of operation

This probe system includes a Pitot-type pressure-based directional velocity probe. The probe system also contains a pressure altimeter, a six-component inertial measurement unit, a fluid temperature sensor (which may be built into the probe sting), a case temperature sensor and a humidity sensor. The probe system can be configured to stream data over a serial data line once powered on, but may also be used together with a computer using the on-board USB terminal.

System description

Integrated modular digital Pitot-static probe system.

System components

1x Modular probe system assembly	Modular system comprised of 1 (or more) sting assembly, sensor package, retainer and gasket.
1x USB cable	A low-profile USB micro->A connector is normally bundled with the system.

Please ensure that all the system components listed above have been supplied, and that there is no apparent damage from shipping.

Note that the probe tip is covered in a red damage-evident lacquer.

System requirements

To interface with a computer, the probe system requires 64-bit Windows 10 (or newer) operating system (not included). Compatibility with earlier Windows versions is possible, but may require additional drivers to be installed. Note that computer interface is not needed for stand-alone streaming operation. The probe has been pre-loaded with firmware; the computer software drivers and data logging software are available online. Additional drivers may be required to run the software on older Windows machines.

DETAILED SPECIFICATION

Standard sensor specifications (custom available on request)	Differential pressure sensor			Static pressure sensor
	Product code	FD2HP-160P	FD2HP-1K0	
Standard pressure ranges	160 Pa FS	1 kPa FS	6.9 kPa FS	1 atm.
Maximum overpressure	33.5 kPa	37.5 kPa	69 kPa	400 kPa
Sensor accuracy ¹	± 0.1 % FS			
Total error band after auto-zero ²	± 0.5 % FS	± 0.25 % FS	± 0.25 % FS	± 0.25 % FS
Compensated temperature range (extended ranges available)	-40° to +65° C	0° to +50° C	0° to +50° C	-40° to +65° C
Operating temperature range	-40° to +65° C non-condensing			
Storage temperature range	-40° to +65° C non-condensing			
Vibration	Sensors rated to 15 g, 10 Hz to 2 Hz			
Maximum relative humidity	95 %			
Relative ambient humidity sensor specification	0 % to 100 % RH, +/- 3%			
Ambient temperature sensor specification ³	0°C - 65°C ± 0.5°C			
Ambient absolute pressure sensor specification	30-110 kPa FS, +/- 0.1kPa			
Fluid temperature sensor	-40°C - 150°C ± 0.5°C			
Absolute pressure limits	0.2 - 1.5 atm			
Total weight	min. 30 g			
Voltage	6-24 Vdc or via USB			
Power	min. 290 mW			
Communications interface	USB2.0, UART or Ethernet (with adaptor)			
Data acquisition rate	1 kHz (equivalent)			
Digital resolution	24-bit pressure, 16-bit environmental and IMU			
System requirements	Windows 7 or later, minimum 3GHz & 4Gb RAM			
IMU specification	3 axis gyro, 125 °/s FS, ± 3.9 x10 ⁻³ °/s 3 axis accelerometer, 2g FS, ± 0.061 mg			

¹ Includes errors due to pressure non-linearity, pressure hysteresis, non-repeatability and calibration uncertainty.

² Total residual error after auto-zero, excluding residual temperature sensitivity.

³ Temperature recorded at the location of the PCB. Waste heat from electronic components may distort temperature readings.

SERIAL COMMUNICATIONS

When the probe system is powered on, it will undergo a brief system diagnostic test; if the test is passed, then a green LED will illuminate at the rear of the probe near the cable connector. Data will then begin streaming serial data on the UART serial lines if streaming has been enabled. The system will stream data continuously while powered on, until commanded to stop.

The sensors are factory-calibrated, and the calibration data is held in on-board memory. Data are converted into SI units by the firmware and streamed as single-precision floats.

Serial stream

Each data packet consists of 51 bytes, beginning with an unsigned-integer frame character ("#") and terminating with a checksum (both inclusive); the UART configuration is the typical 8-N-1. The data order is shown below, and is the same for both UART and USB.

<i>Byte index</i>	<i>Description</i>	<i>Type</i>	<i>Unit</i>
0	Frame character '#'	uint8	-
1	Pressure 0	float32	Pa
2			
3			
4			
5	Pressure 1	float32	Pa
6			
7			
8			
9	External thermistor temperature	float32	°C
10			
11			
12			
13	Atmospheric pressure	float32	Pa
14			
15			
16			
17	Internal probe temperature	float32	°C
18			
19			
20			
21	Relative humidity	float32	%
22			
23			
24			
25	Accelerometer x component	float32	g
26			
27			
28			
29	Accelerometer y component	float32	g
30			
31			
32			

33	Accelerometer z component	float32	g
34			
35			
36			
37	Gyroscope x component	float32	deg/s
38			
39			
40			
41	Gyroscope y component	float32	deg/s
42			
43			
44			
45	Gyroscope z component	float32	deg/s
46			
47			
48			
49	CRC16-CCITT*	uint16	-
50			

*CRC16-CCITT 0x1021 polynomial. Initial value = 0xFFFF

IMPORTANT NOTE: Data are transmitted using the little-endian convention, so that the first byte transmitted for each quantity is the least significant.

Baud	Configurable to 2 Mbps
UART config	1 start bit / 8 data bits / 1 stop bit
Parity	none

The UART Tx has a 3.3V line level output; this may not be compatible with a 5V receiver. A unidirectional level translator (not included) may be used if signals are to be sent to a 5V receiver. The receiving board must also have at least a 71 byte receive buffer to prevent data overrun. The probe may receive 3.3V to 5V inputs.

A partial data packet, having only 15 bytes and excluding less critical data, is also available from the probe.

Byte index	Description	Type	Unit
0	Frame character '#'	uint8	-
1	Pressure 0	float32	Pa
2			
3			
4			
5	Pressure 1	float32	Pa
6			
7			
8			
9	External thermistor temperature	float32	°C
10			
11			
12			
13	CRC16-CCITT*	uint16	-
14			

*CRC16-CCITT 0x1021 polynomial. Initial value = 0xFFFF

Checksum & data corruption warning

A CRC-16 checksum word (uint16) is included at the end of each data packet to provide a warning of data corruption in transmission. Example C++ code and DLL files to compute the CRC-16 checksum are available for download. If the computed and transmitted checksums do not match, the entire data packet should be discarded.

Note that additional details about the probe communications, including a summary of CRC-16-CCITT implementation, are appended to the end of this document.

2 PHYSICAL CONNECTIONS

The probe system is supplied either as a single integrated unit or as a modular system containing 1 or more sting assemblies with a single sensor package. The probe sting should be mounted so that the tip is facing axially in the direction of nominal mean flow. Note that the on-board IMU will enable the roll-alignment of the probe to be matched to calibration conditions.

WARNING: Do not apply local stresses to the probe casing, or the casing may crack. The probe should be held in place with a friction collet, or other circular clamping arrangement.

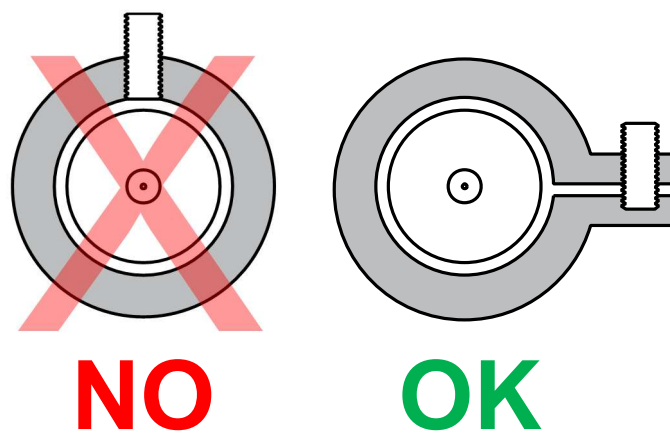


Figure 1: Correct probe mounting

Sting assembly

The sting assembly consists of a total pressure core surrounded by a static pressure sleeve. The tip has been coated with a red varnish witness coating. The static taps, if included, may have been drilled in slightly different axial locations on the sting, for structural reasons.

Sensor package

There are two user-accessible ports on the sensor package- a micro-USB port and a serial comms port.

Micro-USB port: This allows the user to access the sensing and diagnostic functions of the probe system using a PC (with the appropriate drivers and software installed).

Comms port: This is the UART serial communications connection. There are four pins: V+ (1), GND (2), Rx (3) and Tx (4), where pin 1 is on the left when the probe is oriented such that the serial port is above the micro-USB port (see figure 2). Note that pin 1 on the matching plug is marked with a small recessed triangle.

IMPORTANT: UART operates on 3.3V but is 5V tolerant; V+ should be limited to no more than 5VDC for initial testing.

The maximum length of the data cables will depend on local EMI environment and shielding provisions. Powered repeaters should be used for USB cables longer than 3m.

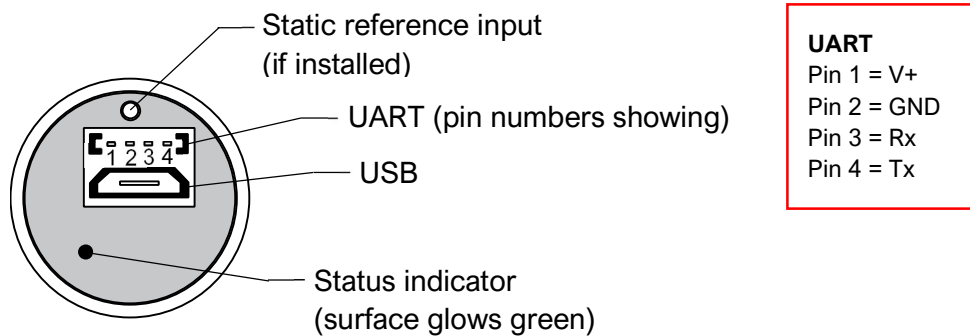


Figure 2: connector interface and pin assignment

IMPORTANT: The Tx terminal is the transmit line for the sensor package. This should be connected to the Rx terminal on the receiving unit.

Note that the pin numbering shown here is for the hardware end. Cable assemblies are not cross-linked: if a mating cable assembly is used, the pin order will be reversed on the cable end.

Modular system components

The modular probe system consists of a sensor package, gasket, sting assembly and retainer as shown in figure 3.

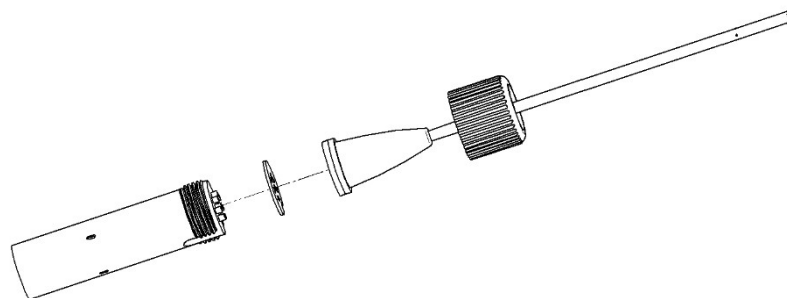


Figure 3: Modular probe system components

To assemble the probe, ensure the gasket and sting assembly are both fitted onto the barbs on the sensor package such that the cut-outs on all 3 components are aligned as shown in figure 4. Failure to do this will not prevent assembly and effective sealing of the probe components, but it will result in incorrect pressure readings.

To secure the probe assembly, slide the retainer over the sting and screw by hand onto the sensor package. Note that the system does not require significant force to tighten and seal (max 1 Nm), and may be damaged by overtightening.



Figure 4: Correct alignment of modular system components

3 CARE AND HANDLING

WARNING: Do not allow any liquids to come into contact with the sensor package, or the system may be permanently damaged.

- Place protective covers over the probe when not in use, and handle the probe with care.
- Protect the sensor package from moisture and dust, and store in an ESD-safe sleeve when not in service.
- The sensor package enclosure includes components made from acrylic, and may be susceptible to solvents. The probe sting is not electrically connected to the system ground. Sensible ESD precautions should be taken before handling.
- Ensure that no dust, dirt or liquid enters the probe. This will alter the system performance. Any foreign material introduced into the sensors themselves may permanently damage the sensors.
- Do not use the probe in wet or condensing conditions. In the event of clean water ingress, the probe can be dried by heating to 40°C.
- The probe sting can be removed and cleared with compressed air if necessary. Ensure that the sting is dry before reassembling.
- Care must be taken to avoid damage to the probe sting or the probe tip.
- Probes may be cleaned using a mild detergent solution and a fibre-free cloth. Do not use solvents or alcohol on the probe, and take care not to get any cleaning solution in the holes.
- Do not apply any bending moments on the probe sting, as the junction between the probe and the sensor package may be damaged.
- Connect cables to the probe with care, as the socket mountings are fragile. Ensure that appropriate strain relief is used: cable strain may cause erroneous sensor readings. Excessive cable bending will damage the probe.

4 SOFTWARE AND DRIVERS

Although this probe system can function independently of a PC by continuously streaming data (when configured to do so), it is possible to interface with a PC via the included USB port for data logging, conversion, diagnostics and visualization.

Drivers

There are two external drivers which must be downloaded and installed on the computer in order for the PC to be able to interface with the probe system, in addition to the specific system driver for your probe.

- [National Instruments LabView Run Time Engine \(LVRTE\)](#)
- [National Instruments VISA Run-Time Engine \(NIVISA\)](#)

These drivers are freely available for download from National Instruments. Ensure that the 64-bit version of the NI LabVIEW Run Time Engine is selected (note this is not the default option), and restart the computer following each installation. Correct download settings for each driver are shown in figures 5 and 6 below.

Home > Support > Software and Driver Downloads > NI Software Product Downloads > Download Detail Page

LabVIEW

LabVIEW is systems engineering software for applications that require test, measurement, and control.

[+ Read More](#)

DOWNLOADS

Supported OS [?] Windows [View Readme](#)

Version [?] Use latest available

Application Bitness [?] 64-bit

Note: Unless you require the additional memory provided by the 64-bit option, NI recommends that you download the 32-bit version. [Learn More](#)

Included Editions [?] Base, Full, Professional Runtime

Language [?] English, French, German, Japanese, Korean, Simplified Chinese

Driver Software Included [?] No

Figure 5: Download settings for NI LabVIEW Runtime Engine

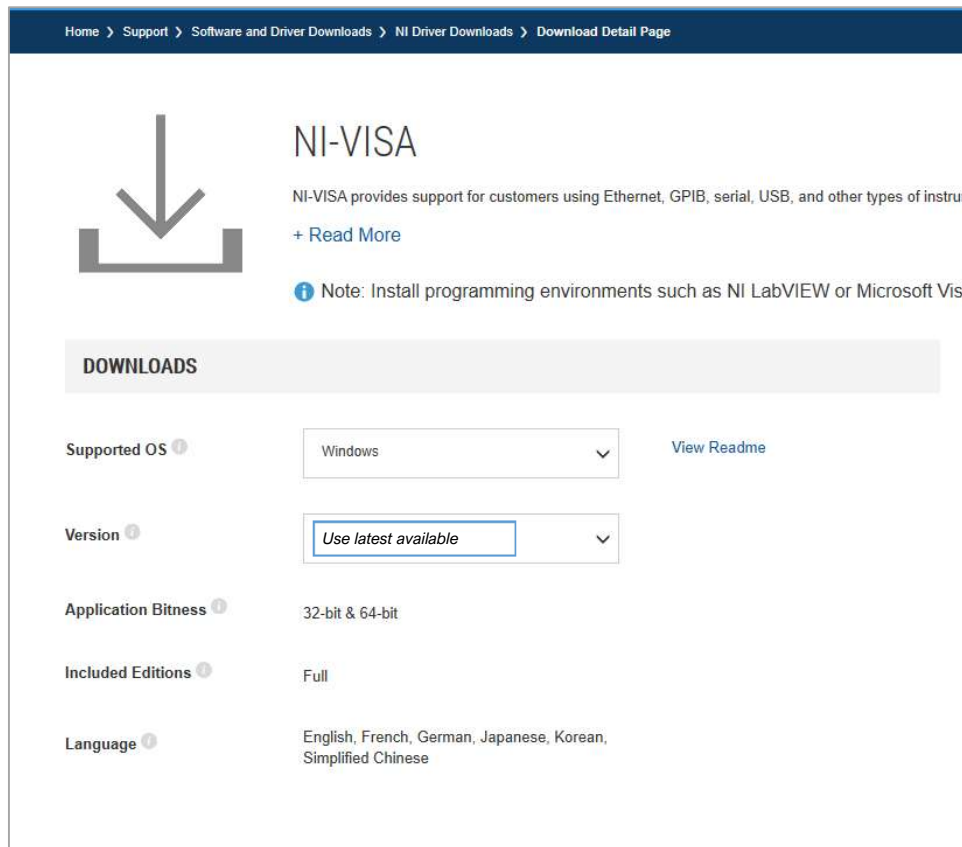


Figure 6: Download settings for NI VISA Runtime Engine

Additionally, a further USB driver install may be required for some older versions of Windows (not required for Windows 10).

Executable

An executable software package can be downloaded for your probe system, which facilitates direct communication between your PC and your probe using the probe's micro-USB connector. After launching the software, you should see the window reproduced in figure 8.

Starting procedure

- 1) Connect the computer to the probe's micro-USB socket.
- 2) The probe will perform a power-on self-test, lasting a couple of seconds. If all tests pass, the rear of the probe will illuminate solid green.
- 3) Load the program [2HP USER INTERFACE]. It will start in the [ACTION] tab.
- 4) Using the [COM PORT] drop down menu, select the probe COM port.
- 5) Enter the desired setup options (see below)
- 6) Press the white arrow near the top left corner of the window to run the application.
- 7) Ensure that there are no errors in the [ERROR] dialogue box. If an error has occurred at this stage, it is most likely an invalid COM port selection. If using an older Windows version this may alternatively be a compatibility error requiring the additional installation described above.

Setup options

Within the [ACTION] tab, the user can select some options for data collection and processing.

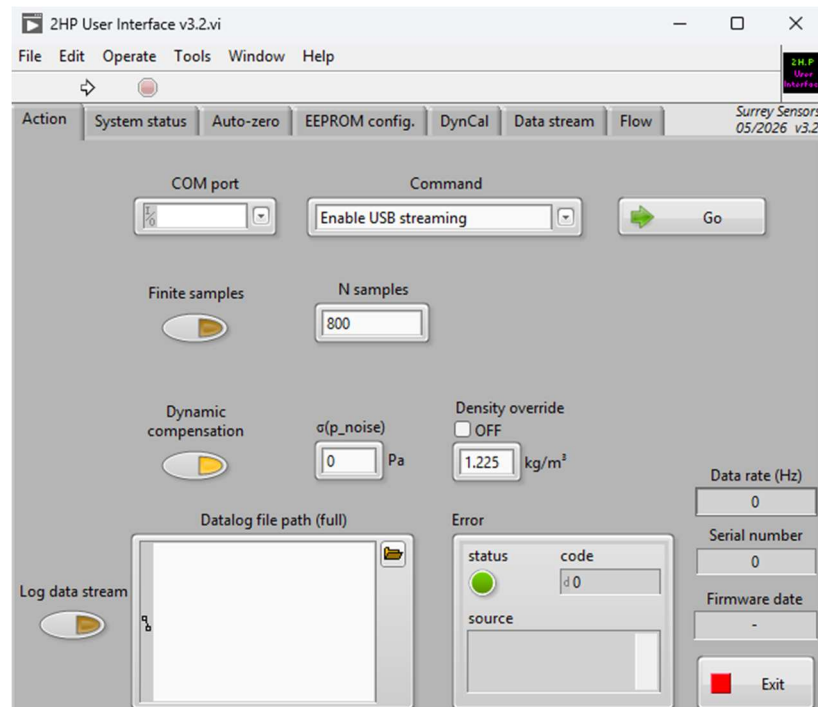


Figure7: User interface window

- Finite samples:** The probe can be instructed to collect a specified number N samples only. If this option is disabled, the probe will collect samples until manually stopped using the [DISABLE USB STREAMING] command.
- Dynamic compensation:** If enabled and the probe has been dynamically calibrated, the system will apply the gain and phase corrections from the dynamic calibration to produce time-accurate outputs. **IMPORTANT NOTE:** *If there is significant spectral energy at frequencies higher than the Nyquist, this can lead to aliasing errors.*
- $\sigma(p_noise)$:** If the probe is used in low-turbulence environments, the electrical/signal noise may dominate and yield excess energy at high frequencies. By entering the dimensional zero-flow signal noise RMS level here, this can be filtered from later measurements. For raw outputs, enter [0]. See [GET SENSOR NOISE VALUE].
- Density override:** The probe requires estimates of the fluid density in order to convert the measured pressures into velocities. Normally, the probe uses its internal temperature and pressure sensors and the ideal gas approximation to obtain estimates of density. If the probe tip is in a different environment than the body, if the working fluid is not air, or if the user has independent, high-quality measurements of the fluid density, this can be entered manually and will be treated as constant. The entered value will only be used if the [DENSITY OVERRIDE] option is [ON].
- Log data stream:** A tab-delimited ASCII text output of a time history of all the sensor values can be saved to disk by enabling the [LOG DATA STREAM] switch. If the [DATALOG FILE PATH] box is left empty, the user will be prompted where to save the file. It is helpful to put a known extension on the file such as ".txt". Data is written to disk every second and stopped upon disabling USB streaming. If streaming is

resumed and the filename is not changed, the user will be prompted to either replace the file or cancel. If cancel is chosen, a prompt will appear to allow a new filename to be chosen. If this is also cancelled, data streaming will begin but no log file will be created and the [LOG DATA STREAM] switch will be disabled.

Available commands

The [COMMAND] drop-down menu in the [ACTION] tab contains the following commands. A command is run by pressing the [GO] button to the right of the [COMMAND] menu. The application will automatically switch to the relevant tab. Once finished, return to the [ACTION] tab to select another command and run as desired.

- **Enable USB streaming:** this command activates the USB data stream. The [DATA STREAM] tab is activated and a running plot of all seven pressure sensor values is displayed, along with values for all the other on-board sensors. This will continue indefinitely until the user presses the [STOP] button or selects another tab, or exits the program by closing the window. Note that the data stream from the UART serial output is not affected by enabling USB streaming.
- **Get data rate:** this command allows the user to retrieve the system sampling rate. Note that the sampling rate is factory-set for best dynamic performance and is not normally user-selectable.
- **Get serial number:** each probe is encoded with its own unique serial number, which is displayed in the window below the [COM PORT] drop-down menu. This function will retrieve the probe's serial number and display it in the [ACTION] tab. Note that the serial number is also retrieved as part of the normal startup operations. The serial numbers are decimal values; the probe having serial number "1234" will display as [d1234].
- **Get status info:** running this command will activate the [SYSTEM STATUS] tab. The system will be queried for the status of the last self-test. Green indicators indicate the test was passed, while red indicates a fault. Descriptive text beside each indicator is provided.
- **Run self-test:** the system will be made to perform the power-up self-test again, and then query for the result. This allows the user to easily check any changes made without having to disconnect and reconnect the USB cable.
- **Perform temporary auto-zero:** this command activates the [AUTO ZERO] tab and begins collecting samples from each pressure sensor to estimate the zero pressure reading. An offset is then temporarily stored in volatile memory that will remain until the system is powered down. During this procedure, ensure that the sensors are subjected to zero differential pressure.
- **Disable USB streaming:** this command stops data transmission over USB.
- **Set HW Trig:** This command enables hardware triggering. Hardware triggering allows the probe's UART port to be used for synchronization, with specific EEPROM settings. See appendix for details.
- **Disable HW Trig:** This command disables hardware triggering.

- **Get EEPROM values:** this function displays the [EEPROM CONFIG] tab and read off all the EEPROM values from the system.
- **Set EEPROM values:** The EEPROM is the probe's on-board memory, and is factory-set for normal operation. To make changes to the EEPROM values, first run [GET EEPROM VALUES] to populate the table with the current values. These values should be recorded separately to ensure that the probe can be returned to factory settings. To make changes, alter the values as desired; return to the [ACTION] tab and run the [SET EEPROM VALUES] command. Finally, return to the [ACTION] tab and run the [GET EEPROM VALUES] command again to check the values are as desired and, importantly, that the EEPROM checksum indicator is green. A red indicator means the data has become corrupt in transit and should be re-transmitted. Note that probes typically ship with UART data streaming disabled; to enable this, the [UART STREAMING] setting in the EEPROM needs to be changed to [1] (for 'ON').
- **Perform permanent auto-zero:** this function should not be used under normal operation. This command is similar to temporary auto-zero, but the zero values are written permanently to the EEPROM. **WARNING** – *this will overwrite the existing offsets that were used during probe calibration. Carrying out this procedure may invalidate your calibration.*
- **Get DynCal data:** This option retrieves the dynamic calibration data from the probe EEPROM if the probe has been dynamically calibrated. The gains and phase shifts from the calibration are plotted, and can be exported as an ASCII data file from the [DYNCAL] tab.
- **Get sensor noise value:** This function measures the zero-flow signal noise for filtering purposes. To activate this function, ensure the [FINITE SAMPLES] switch is in the [ON] position, and enter the number of samples for statistical convergence in the [N SAMPLES] field (normally, 10 seconds or 8,000 samples is sufficient). Click [GO] to run the function, and ensure that the probe is kept stationary in still air and that all other possible ambient EMI sources are running. When complete, the [σ (P_NOISE)] field will be populated with the RMS signal noise in Pa and this will be used to de-noise further measurements. A data file "sn???? sigma(p_noise)(pa).txt" will also be generated for future reference, where ???? is the probe serial number and the file contains the value. This can be entered directly into the [σ (P_NOISE)] field during future tests if the EMI environment remains unchanged. The value of [σ (P_NOISE)] will default back to [0] when the executable is closed and re-launched.
- **Get firmware timestamp:** this command returns the firmware version number, to verify compatibility with software versions.
- **Exit:** this command stops the program from running, but leaves the window open so as to preserve settings for later use. This has the same functionality as pressing the [EXIT] button.

Notes on calibration and data reduction

The velocity is computed under the assumption of unity recovery factor and incompressible flow, using Bernoulli's equation. The density is approximated using the values obtained by the probe's on-board sensors, or using the override value entered in the [ACTION] tab. Should compressible-flow conversions be desired, these can be implemented in post-processing using the raw data provided.

5 TECHNICAL SUPPORT

Full technical support is available for this product and its associated software.

If you experience any difficulty in installation or use, or if you need additional support in the operation of the system, please contact your Surrey Sensors Ltd. account manager or technical representative.

Appendix 1: Additional communications details

Status bytes map

Byte num.	Bit	Description	Info	Byte default value
0	0	Pressure sensor 0 checksum okay	1 yes, 0 no	128
	1	Pressure sensor 1 checksum okay	1 yes, 0 no	
	2	1 (not used)	-	
	3	1 (not used)	-	
	4	1 (not used)	-	
	5	1 (not used)	-	
	6	1 (not used)	-	
	7	1 (not used)	-	
1	0	Pressure sensor 0 temperature in range	1 yes, 0 no	128
	1	Pressure sensor 1 temperature in range	1 yes, 0 no	
	2	1 (not used)	-	
	3	1 (not used)	-	
	4	1 (not used)	-	
	5	1 (not used)	-	
	6	1 (not used)	-	
	7	1 (not used)	-	
2	0	Pressure sensor 0 value in range	1 yes, 0 no	128
	1	Pressure sensor 1 value in range	1 yes, 0 no	
	2	1 (not used)	-	
	3	1 (not used)	-	
	4	1 (not used)	-	
	5	1 (not used)	-	
	6	1 (not used)	-	
	7	1 (not used)	-	
3	0	Environmental sensors ident pass	1 yes, 0 no	128
	1	IMU ident pass	1 yes, 0 no	
	2	IMU accelerometer self test pass	1 yes, 0 no	
	3	IMU gyro self test pass	1 yes, 0 no	
	4	External thermistor value in range	1 yes, 0 no	
	5	MCU EEPROM checksum okay	1 yes, 0 no	
	6	Dynamic calibration checksum OK	1 yes, 0 no	
	7	1 (not used)	-	

USB Command List

After @

Cmd Byte	Description	Return	Additional information	Comments
s	Retrieve status bytes	4x uint8	P_chk_ok, P_tmp_ok, P_val_ok, BME280_ident, BMI160_ident, BMI160_acc_pass, BMI160_gyr_pass, T0_val_ok, EEPROM_CRC16_pass, DynCal_CRC16_pass	
S	Perform self-test and retrieve status bytes	4x uint8	P_chk_ok, P_tmp_ok, P_val_ok, BME280_ident, BMI160_ident, BMI160_acc_pass, BMI160_gyr_pass, T0_val_ok, EEPROM_CRC16_pass	
z	Perform temporary auto- zero	2x float32	P_raw_offset(0 to 1)[8]	
Z	Perform permanent auto- zero (write values to EEPROM)	2x float32	P_raw_offset(0 to 1)[8]	
R	Read all EEPROM values	51x uint8	Refer to EEPROM map for description of returned bytes	
W	Write all EEPROM values	-		
D	Enable USB / UART streaming	51x uint8	#[1], P0[4], P1[4], T_ext[4], P_atm[4], Tint[4], RH[4], ax[4], ay[4], az[4], wx[4], wy[4], wz[4], CRC16[2]	Returns at [DataRate] until stream is disabled. USB or UART enabled according to physical layer on which the command was sent
d	Disable USB / UART streaming	-		USB or UART disabled according to physical layer on which the command was sent
A	Enable USB streaming on power-up	-		
a	Disable USB streaming on power-up	-		
Y	Enable UART streaming on power-up	-		
y	Disable UART streaming on power-up	-		
P	Set UART stream full data packet mode (power-up default)	-	Send 1x uint8. 1 = full data, 0 = partial data	
p	Get UART stream full data packet mode	1x uint8	1 = full data, 0 = partial data	
f	Get data rate	1x uint16	Data rate (Hz)	
J	Set data rate (power-up default)	-	Send 1x uint16, DataRate[2] containing one of the allowable integer data rates.	
X	Set inertial sensor modes (power-up defaults)	-	Send 3x bytes, {acc_Range_Mode, gyr_Range_Mode, IMU_Rate_Mode}	

x	Get inertial sensor modes	3x uint8	{acc_Range_Mode, gyr_Range_Mode, IMU_Rate_Mode}	Refer to IMU modes table for description of mode bytes
N	Get serial number	1x float32	Serial_num[4] (EEPROM index 36...39)	
K	Get sensors range	2x float32	Range0[4]...Range1[4] in Pa	
I	Get Firmware timestamp	1x uint32	Unix timestamp	
G	Get current data packet (full)	51x uint8	'#[1], P0[4], P1[4], T_ext[4], P_atm[4], Tint[4], RH[4], ax[4], ay[4], az[4], wx[4], wy[4], wz[4], CRC16[2]	
g	Get current data packet (partial)	15x uint8	'#[1], P0[4], P1[4], T_ext[4], CRC16[2]	
B	Set UART baud rate (bps) (power-up default)	-	Send 1 float32 (as 4 bytes), Baud[4]	
b	Get UART baud rate (bps)	1x float32	Baud[4]	
C	Overwrite DynCal EEPROM tables	-		
c	Get DynCal data tables	569x uint8	Parse and post-process according to EEPROM format	

IMU Modes

Accelerometer range		
Mode (byte)	Eng. Value	Unit
0	±2	g
1	±4	g
2	±8	g
3	±16	g

Gyroscope range		
Mode (byte)	Eng. Value	Unit
0	±125	°/s
1	±250	°/s
2	±500	°/s
3	±1000	°/s
4	±2000	°/s

IMU refresh rate		
Mode (byte)	Eng. Value	Unit
0	6.25	Hz
1	12.5	Hz
2	25	Hz
3	50	Hz
4	100	Hz
5	200	Hz
6	400	Hz
7	800	Hz
8	1600	Hz

EEPROM Table

Byte index	Description	Type	Unit
0	Pressure 0 raw offset	float32	-
1			
2			

3			
4	Pressure 1 raw offset	float32	-
5			
6			
7			
8	Atmospheric pressure offset	float32	Pa
9			
10			
11			
12	External thermistor temperature offset	float32	°C
13			
14			
15			
16	Serial number	float32	-
17			
18			
19			
20	UART baud rate*	float32	bps
21			
22			
23			
24	Accelerometer xyz scale factor	float32	-
25			
26			
27			
28	Gyroscope x component offset	float32	deg / s
29			
30			
31			
32	Gyroscope y component offset	float32	deg / s
33			
34			
35			
36	Gyroscope z component offset	float32	deg / s
37			
38			
39			
40	Default data rate on power-up	uint16	Hz
41			
42	UART streaming enabled on power-up	unit8	-
43	USB streaming enabled on power-up	unit8	-
44	UART streaming full data packet	unit8	-
45	Accelerometer range mode	unit8	-

46	Gyroscope range mode	uint8	-
47	IMU update rate mode	uint8	-
48	HW trigger edge polarity (0 for falling, 1 for rising)	uint8	0 or 1
49	MCU EEPROM CRC16-CCITT	uint16	-
50			
51	DynCal N pts per channel	uint8	-
52	Data rate of dynamic calibration	uint16	Hz
53			
54	Frequency step size (Evenly spaced data from 0 Hz, except last value which is always Nyquist)	float32	Hz
55			
56			
57			
58	Ch 0 gain scale value	float32	-
59	Float32 value = (uint16 value/65535) x scale value		
60			
61			
62	Ch 0 gain correction, f0	uint16	-
63			
64	Ch 0 gain correction, f1	uint16	-
65			
66	Ch 0 gain correction, f2	uint16	-
67			
...
196	Ch 0 gain correction, f67	uint16	-
197			
198	Ch 0 phase scale value	float32	-
199			
200			
201			
202	Ch 0 phase correction, f0	uint16	-
203			
204	Ch 0 phase correction, f1	uint16	-
205			
206	Ch 0 phase correction, f2	uint16	-
207			
...
336	Ch 0 phase correction, f67	uint16	-
337			
338	Ch 1 gain scale value	float32	-
339			
340			
341			

342	Ch 1 gain correction, f0	uint16	-
343			
344	Ch 1 gain correction, f1	uint16	-
345			
346	Ch 1 gain correction, f2	uint16	-
347			
...
476	Ch 1 gain correction, f67	uint16	-
477			
478	Ch 1 phase scale value	float32	-
479			
480			
481			
482	Ch 1 phase correction, f0	uint16	-
483			
484	Ch 1 phase correction, f1	uint16	-
485			
486	Ch 1 phase correction, f2	uint16	-
487			
...
616	Ch 1 phase correction, f67	uint16	-
617			
618	DynaCal CRC-16 CCITT	uint16	-
619			

*Set to zero or negative value to use the UART Rx line as the hardware trigger line. Ensure UART streaming on power-up is set to zero in this case.

Summary of CRC-16-CCITT Implementation in C++

Global Variables and Constants

```
uint16_t CRC16_LUT[256];  
const uint16_t poly = 0x1021;  
const uint16_t crc_init = 0xFFFF;
```

CRC-16 Lookup Table (LUT) Generation

The following function is called once at the start. The 1D array of length 256 “CRC16_LUT” is then stored in memory for all time and used whenever a CRC is computed.

```
void Generate_CRC16_LUT()  
{  
    for (uint16_t i = 0; i < 256; i++)  
    {  
        uint16_t Byte = i << 8;  
  
        for (uint8_t Bit = 0; Bit < 8; Bit++)  
        {  
            if ((Byte & 0x8000) != 0)  
            {  
                Byte <<= 1;  
                Byte ^= poly;  
            }  
            else  
            {  
                Byte <<= 1;  
            }  
        }  
  
        CRC16_LUT[i] = Byte;  
    }  
}
```

Alternatively, the LUT can be hard-coded as a constant:

```
// CRC-16 lookup table for CCITT polynomial 0x1021

static const uint16_t CRC16_LUT[256] =
{
    0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50A5, 0x60C6, 0x70E7,
    0x8108, 0x9129, 0xA14A, 0xB16B, 0xC18C, 0xD1AD, 0xE1CE, 0xF1EF,
    0x1231, 0x0210, 0x3273, 0x2252, 0x52B5, 0x4294, 0x72F7, 0x62D6,
    0x9339, 0x8318, 0xB37B, 0xA35A, 0xD3BD, 0xC39C, 0xF3FF, 0xE3DE,
    0x2462, 0x3443, 0x0420, 0x1401, 0x44E6, 0x54C7, 0x64A4, 0x7485,
    0xA56A, 0xB54B, 0x8528, 0x9509, 0xE5EE, 0xF5CF, 0xC5AC, 0xD58D,
    0x3653, 0x2672, 0x1611, 0x0630, 0x76D7, 0x66F6, 0x5695, 0x46B4,
    0xB75B, 0xA77A, 0x9719, 0x8738, 0xF7DF, 0xE7FE, 0xD79D, 0xC7BC,
    0x48C4, 0x58E5, 0x6886, 0x78A7, 0x0840, 0x1861, 0x2802, 0x3823,
    0xC9CC, 0xD9ED, 0xE98E, 0xF9AF, 0x8948, 0x9969, 0xA90A, 0xB92B,
    0x5AF5, 0x4AD4, 0x7AB7, 0x6A96, 0x1A71, 0x0A50, 0x3A33, 0x2A12,
    0xDBFD, 0xCBDC, 0xFBBF, 0xEB9E, 0x9B79, 0x8B58, 0xBB3B, 0xAB1A,
    0x6CA6, 0x7C87, 0x4CE4, 0x5CC5, 0x2C22, 0x3C03, 0x0C60, 0x1C41,
    0xEDAE, 0xFD8F, 0xCDEC, 0xDDCD, 0xAD2A, 0xBD0B, 0x8D68, 0x9D49,
    0x7E97, 0x6EB6, 0x5ED5, 0x4EF4, 0x3E13, 0x2E32, 0x1E51, 0x0E70,
    0xFF9F, 0xEFBE, 0xDFDD, 0xCFFC, 0xBF1B, 0xAF3A, 0x9F59, 0x8F78,
    0x9188, 0x81A9, 0xB1CA, 0xA1EB, 0xD10C, 0xC12D, 0xF14E, 0xE16F,
    0x1080, 0x00A1, 0x30C2, 0x20E3, 0x5004, 0x4025, 0x7046, 0x6067,
    0x83B9, 0x9398, 0xA3FB, 0xB3DA, 0xC33D, 0xD31C, 0xE37F, 0xF35E,
    0x02B1, 0x1290, 0x22F3, 0x32D2, 0x4235, 0x5214, 0x6277, 0x7256,
    0xB5EA, 0xA5CB, 0x95A8, 0x8589, 0xF56E, 0xE54F, 0xD52C, 0xC50D,
    0x34E2, 0x24C3, 0x14A0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
    0xA7DB, 0xB7FA, 0x8799, 0x97B8, 0xE75F, 0xF77E, 0xC71D, 0xD73C,
    0x26D3, 0x36F2, 0x0691, 0x16B0, 0x4657, 0x5676, 0x6615, 0x7634,
    0xD94C, 0xC96D, 0xF90E, 0xE92F, 0x99C8, 0x89E9, 0xB98A, 0xA9AB,
    0x5844, 0x4865, 0x7806, 0x6827, 0x18C0, 0x08E1, 0x3882, 0x28A3,
    0xCB7D, 0xDB5C, 0xEB3F, 0xFB1E, 0x8BF9, 0x9BD8, 0xABBB, 0xBB9A,
    0x4A75, 0x5A54, 0x6A37, 0x7A16, 0x0AF1, 0x1AD0, 0x2AB3, 0x3A92,
    0xFD2E, 0xED0F, 0xDD6C, 0xCD4D, 0xBDAA, 0xAD8B, 0x9DE8, 0x8DC9,
    0x7C26, 0x6C07, 0x5C64, 0x4C45, 0x3CA2, 0x2C83, 0x1CE0, 0x0CC1,
    0xEF1F, 0xFF3E, 0xCF5D, 0xDF7C, 0xAF9B, 0xBFBA, 0x8FD9, 0x9FF8,
    0x6E17, 0x7E36, 0x4E55, 0x5E74, 0x2E93, 0x3EB2, 0x0ED1, 0x1EF0
};
```

CRC-16 Computation

The following function is called whenever a CRC-16 is required from an array of data.

```
uint16_t Calc_CRC16(uint8_t *Data, uint16_t DataLen, uint16_t crc)
{
    for (uint16_t i = 0; i < DataLen; i++)
    {
        uint8_t index = Data[i] ^ (crc >> 8);

        crc = CRC16_LUT[index] ^ (crc << 8);
    }

    return crc;
}
```

CRC-16 Function Call Example

The data for which the CRC is to be computed is first of all typecast into an array of unsigned char (uint8_t) "DataBytes". This can be done using the `memcpy` function. When generating a CRC value for an array of data the length value "Len" passed to the function is that of the number of bytes in the entire array. However, when checking a CRC value appended to an array of data, the length value passed to the function is two less than that of the entire array so as to exclude the appended CRC word. The CRC value passed to the function is that of the initialiser constant "crc_init", which, for the CCITT specification, is hexadecimal `0xFFFF`.

```
uint16_t CRC_computed = Calc_CRC16(&DataBytes, Len, crc_init);
```

Checksum Test

A checksum test is passed if the computed and transmitted checksum values are equal. With the CRC appended at the end of the transmitted data array the test is carried out as follows

```
uint16_t CRC_appended;  
  
memcpy(&CRC_appended, &DataBytes[Len - 2], 2);  
  
bool CRC_pass = (CRC_appended == CRC_computed);
```

Code implementation can be validated by cross-checking results with a reputable online CRC calculator such as <https://crccalc.com/>

CRC-16-CCITT Algorithm Parameters:

Polynomial divisor:	0x1021	$(x^{16} + x^{12} + x^5 + 1)$
CRC initialiser:	0xFFFF	
Input reflection:	False	
Output reflection:	False	
Output XOR:	0x0000	

Appendix 2: Hardware triggering

The probe includes an internal clock for data acquisition and timing. Occasionally, it may be necessary to begin acquisition on receipt of an external trigger pulse. If so, the probe's UART Rx pin can be reconfigured as a trigger input, and the Tx port as a trigger output. Consequently the probe can only be operated via USB if the external trigger function is used.

If external triggering was specified as a requirement at the time of order, then the probe would have shipped with the required settings. If not, the settings can be modified by the user.

Instructions

Step 1: Launch the user interface application, select the appropriate COM port and run the application by clicking the white arrow in the menu bar.

Step 2: Under the [ACTION] tab, select [GET EEPROM VALUES] from the [COMMAND] drop-down menu and click [GO]. This will populate the [EEPROM CONFIG] tab.

Step 3: It is strongly recommended that a copy of the factory-set EEPROM values are saved in case of accidental corruption.

Step 4: Make the following changes to the EEPROM configuration (if required), following instructions in the manual:

- Set [UART BAUD RATE] to [0] (zero)
- Set [UART STREAMING] to [0] (zero)
- Set [HW TRIGGER POLARITY] to either [0] (zero) or [1] (one). A value of zero will trigger USB data streaming on a falling edge, while a value of one will trigger on a rising edge.

Note that the serial digital lines may be labelled as either TTL or UART in the user interface, depending on the version used.

Step 5: Connect the input trigger signal to the Rx pin (pin 3) on the probe's UART connector (see Figure 11) and the trigger system ground to the GND pin (pin 2).

IMPORTANT: Be sure to use the correct cable assembly for the UART connection, and that this is inserted in the correct orientation.

WARNING: The probe logic operates at 3.3V levels but is 5V tolerant. Any voltage values applied outside of this range to the Tx/Rx pins may damage the probe and will void the warranty.

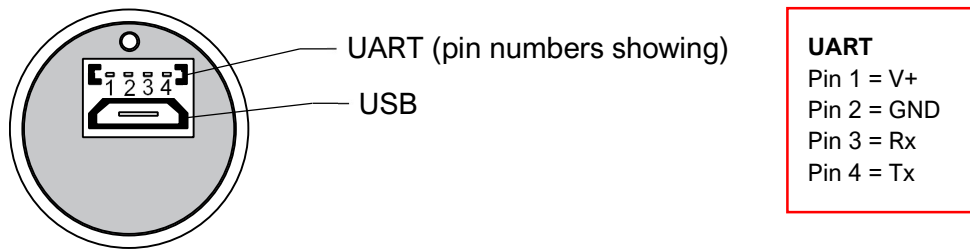


Figure 8: connector interface and pin assignment

Note that the pin numbering shown here is for the hardware end. Cable assemblies are not cross-linked: if a mating cable assembly is used, the pin order will be reversed on the cable end.

Step 6: Enter the desired parameters for acquisition in the user interface software, under the [ACTION] tab.

Step 7: From the [COMMAND] drop-down menu, select [SET HW TRIG] and click [GO].

The probe trigger input is now armed, and acquisition will begin on receipt of the trigger signal. Note that the latency in response is significantly smaller than the smallest sampling time, and can be considered as negligible.

When configured for hardware triggering, the probe's UART Tx line will automatically act as an output trigger. Whenever the probe is streaming data, the polarity of the UART Tx line will switch (following the same convention as the [HW TRIGGER POLARITY] selection). The polarity will switch when the probe is triggered (either by a hardware trigger or software command), and will return to its original state when acquisition stops. The use of the Tx line is optional (it can be left open or unconnected), but care must be taken to ensure that the line is used only to provide trigger pulses when in this mode.

The content of this user manual is for general information only and is subject to change without notice. It may contain inaccuracies or errors and Surrey Sensors Ltd. expressly exclude liability for any such inaccuracies or errors to the fullest extent permitted by law. Your use of any information is entirely at your own risk, for which Surrey Sensors Ltd. shall not be liable.