FAST-RESPONSE 7-HOLE
“COBRA” PROBE

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

Version Control

Version	Date	Summary of changes	Software ver.
1.0	10-2024	New document	2.0
1.1	03-2025	Update for new software	3.0

1 INTRODUCTION

Principle of operation

This probe system includes a seven-hole pressure-based directional velocity and turbulence probe. The probe system also contains a pressure altimeter, a six-component inertial measurement unit, one or more fluid temperature sensors (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 multi-function fast-response seven-hole 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)			
Product code	FD7HP-160P	FD7HP-1K0	FD7HP-6K9
Standard pressure ranges	160 Pa FS	1 kPa FS	6.9 kPa FS
Maximum overpressure	33.5 kPa	37.5 kPa	69 kPa
Sensor accuracy[1]	± 0.1 % FS		
Total error band after auto-zero[2]	± 0.5 % FS	± 0.25 % FS	± 0.25 % FS
Operational mode	Absolute and differential configurations available		

Compensated temperature range	-40° to +65° C	0° to +50° C	0° to +50° 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		

Reference pressure	Connection to steady reference static pressure required for operation		
Sting material	Steel		
Sting diameter	3.7 mm		
Standard tip geometry	Hemispherical		
Angular measurement range[4]	± 45°		
Remote temperature probe	-40°C - 150°C ± 0.5°C		
Absolute temperature limits	5° to +65° C non-condensing		
Absolute pressure limits	10x P_{FS}		
Voltage	6-12 VDC or via USB		
Power	min. 290 mW		
Communications interface	USB2.0 & UART RS232 or UDP Ethernet available[5]		
Data acquisition rate	Typ. 0.8 kHz	Max. 0.9	Ext. 1.6 kHz[6]
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 is recorded at the location of the PCB. Waste heat from electronic components may distort temperature readings.
[4] Angular range specified for hemispherical tip; depends on tip geometry.
[5] Additional adaptor system required, sold separately
[6] Extended range. Requires disabling of temperature compensation. May result in nonuniform sampling.

2 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 71 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.

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	Pressure 2	float32	Pa
10			
11			
12			
13	Pressure 3	float32	Pa
14			
15			
16			
17	Pressure 4	float32	Pa
18			
19			
20			
21	Pressure 5	float32	Pa
22			
23			
24			
25	Pressure 6	float32	Pa
26			
27			
28			
29	External thermistor temperature	float32	°C
30			
31			
32			

33	Atmospheric pressure	float32	Pa
34			
35			
36			
37	Internal probe temperature	float32	°C
38			
39			
40			
41	Relative humidity	float32	%
42			
43			
44			
45	Accelerometer *x* component	float32	g
46			
47			
48			
49	Accelerometer *y* component	float32	g
50			
51			
52			
53	Accelerometer *z* component	float32	g
54			
55			
56			
57	Gyroscope *x* component	float32	deg/s
58			
59			
60			
61	Gyroscope *y* component	float32	deg/s
62			
63			
64			
65	Gyroscope *z* component	float32	deg/s
66			
67			
68			
69	CRC16-CCITT*	uint16	-
70			

*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 35 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	Pressure 2	float32	Pa
10			
11			
12			
13	Pressure 3	float32	Pa
14			
15			
16			
17	Pressure 4	float32	Pa
18			
19			
20			
21	Pressure 5	float32	Pa
22			
23			
24			
25	Pressure 6	float32	Pa
26			
27			
28			
29	External thermistor temperature	float32	°C
30			
31			
32			
33	CRC16-CCITT*	uint16	-
34			

*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.

3 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.

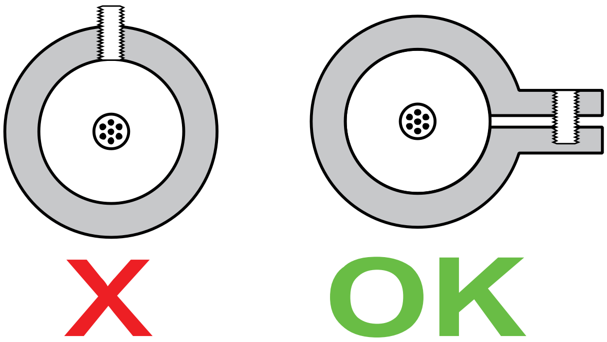


Figure 1: Correct probe mounting

Sting assembly

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

The hole arrangement, when viewed from upstream, is as illustrated in figure 2. Note that the roll alignment is arbitrary. That arrangement may vary for some custom probes.

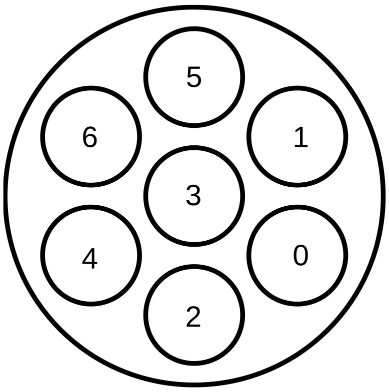


Figure 2: Hole numbering convention, looking at probe tip from upstream, with the probe body located behind.

Sensor package

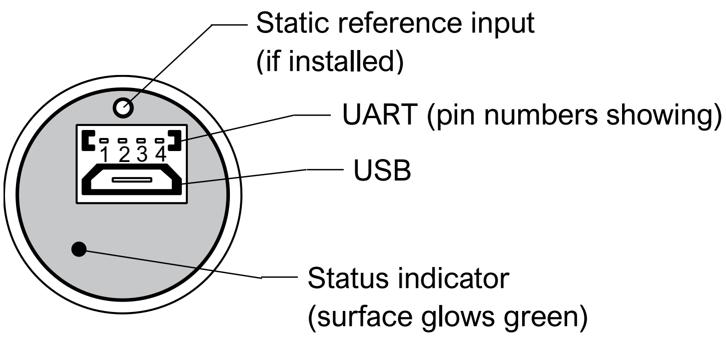
There are two user-accessible ports on the sensor package- a micro-USB port and a serial comms port.

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 3). 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.



UART
Pin 1 = V+
Pin 2 = GND
Pin 3 = Rx
Pin 4 = Tx

Figure 3: 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 4.

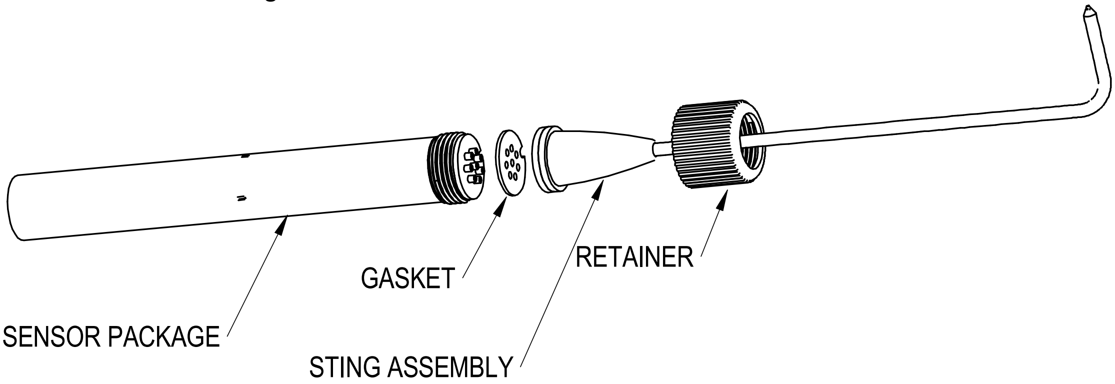


Figure 4: Modular probe system components

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 5. Failure to do this will not prevent assembly and effective sealing of the probe components, but it will result in disagreement between pressure sensor channel numbers and hole numbering.

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, and may be damaged by overtightening.

Figure 5: Correct alignment of modular system components

4 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. If the damage-evident coating is chipped, it is likely that the calibration of the probe has changed.
- 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.

5 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 6 and 7 below.

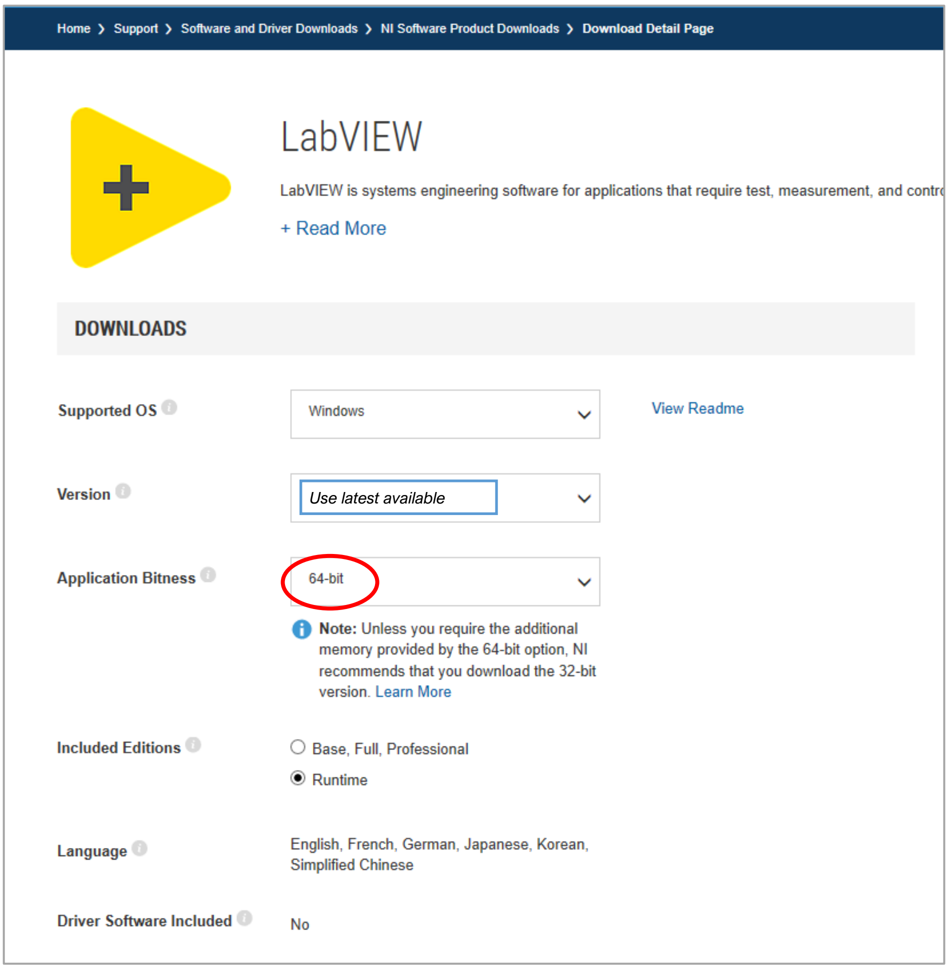


Figure 6: Download settings for NI LabVIEW Runtime Engine

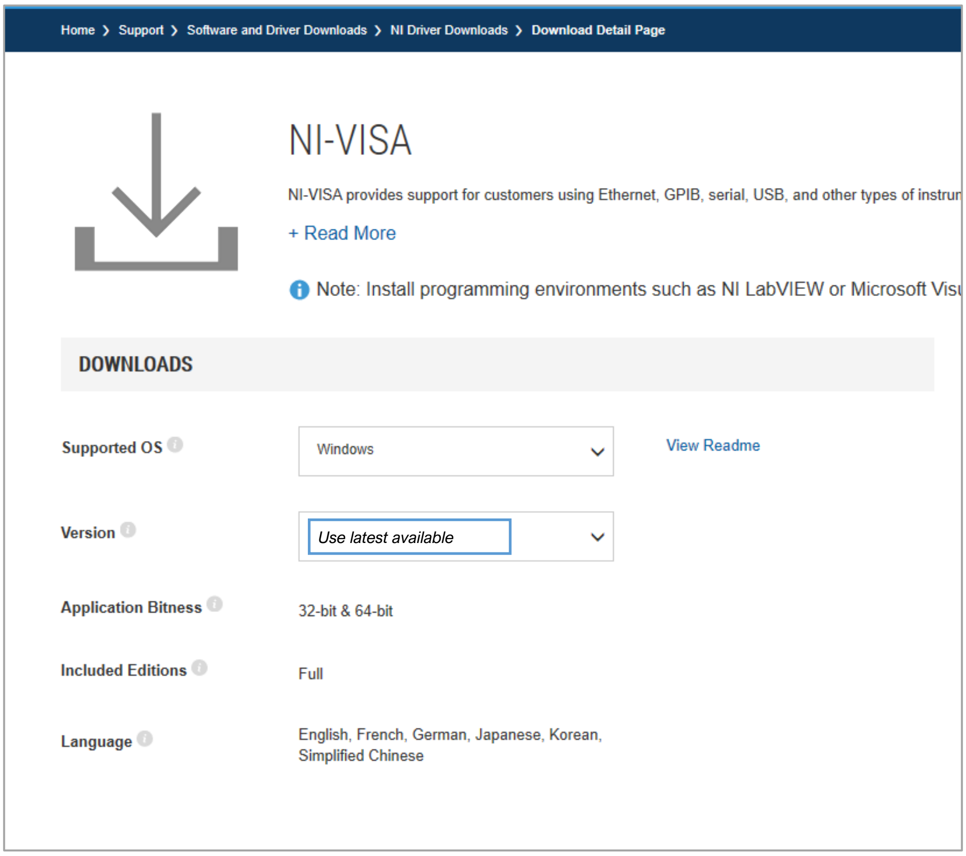


Figure 7: Download settings for NI VISA Runtime Engine

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 [7HP 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.

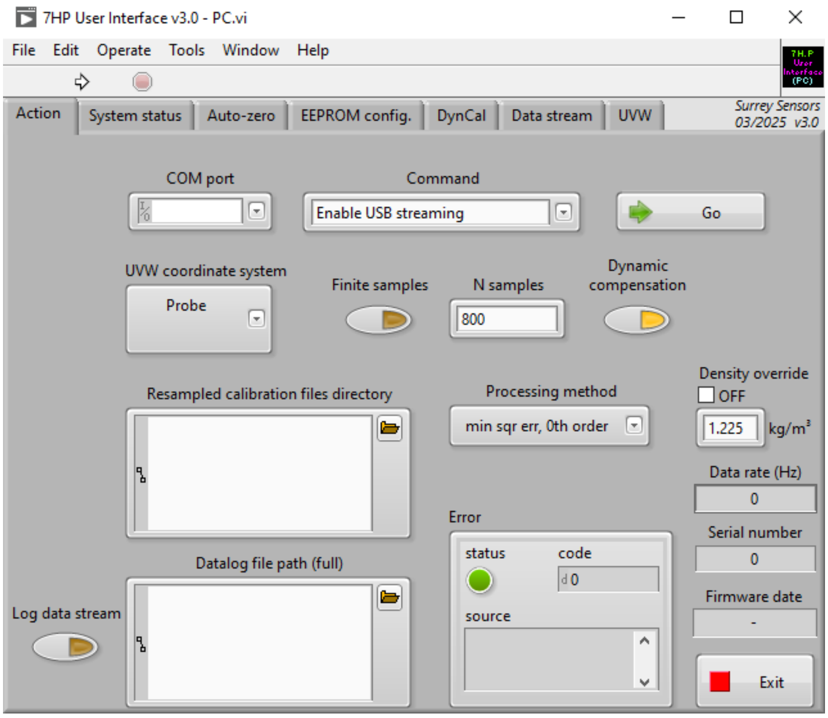


Figure 8: Probe system companion software screen-shot

- **UVW coordinate system:** Select between probe, wind tunnel and rotated wind tunnel coordinate systems for reporting the velocity components (if calibration data files are available). A description of these coordinate systems is appended to this manual.

- **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, 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*.

- **Calibration:** If a probe directional calibration data file is available, navigate to the directory containing these files in the [RESAMPLED CALIBRATION FILES DIRECTORY] window. If no calibration file directory is selected, raw pressures will be recorded only. More details are included in Appendix 3. The interpolation technique can also be specified using the [PROCESSING METHOD] drop-down menu.

- **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 of 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 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 actives 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 my invalidate your calibration.*

- **Get DynCal data:** This option retrieves the dynamic calibration data from the probe EEPROM. The gains and phase shifts from the calibration are plotted, and can be exported as an ASCII data file from the [DYNCAL] tab.

- **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 components are obtained using the generalized (sectorless) technique of Shaw-Ward *et al.* (2015). Estimates of turbulence intensity are also provided, although it should be noted that these are based on direct pressure measurements only. The software will convert pressures to velocities if appropriately-formatted calibration files are found. The system requires eleven independent ASCII text format calibration files: one each for the seven pressures, pitch angle, yaw angle, fluid speed and density. These files can be generated from unstructured calibration data files using a separate utility (see Appendix 3).

6 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 Table

Byte num.	Bit	Description	Info		Byte default value
0	0	Pressure sensor 0 checksum okay	1 yes, 0 no	0	128
	1	Pressure sensor 1 checksum okay	1 yes, 0 no	0	
	2	Pressure sensor 2 checksum okay	1 yes, 0 no	0	
	3	Pressure sensor 3 checksum okay	1 yes, 0 no	0	
	4	Pressure sensor 4 checksum okay	1 yes, 0 no	0	
	5	Pressure sensor 5 checksum okay	1 yes, 0 no	0	
	6	Pressure sensor 6 checksum okay	1 yes, 0 no	0	
	7	1	-	128	
1	0	Pressure sensor 0 temperature in range	1 yes, 0 no	0	128
	1	Pressure sensor 1 temperature in range	1 yes, 0 no	0	
	2	Pressure sensor 2 temperature in range	1 yes, 0 no	0	
	3	Pressure sensor 3 temperature in range	1 yes, 0 no	0	
	4	Pressure sensor 4 temperature in range	1 yes, 0 no	0	
	5	Pressure sensor 5 temperature in range	1 yes, 0 no	0	
	6	Pressure sensor 6 temperature in range	1 yes, 0 no	0	
	7	1	-	128	
2	0	Pressure sensor 0 value in range	1 yes, 0 no	0	128
	1	Pressure sensor 1 value in range	1 yes, 0 no	0	
	2	Pressure sensor 2 value in range	1 yes, 0 no	0	
	3	Pressure sensor 3 value in range	1 yes, 0 no	0	
	4	Pressure sensor 4 value in range	1 yes, 0 no	0	
	5	Pressure sensor 5 value in range	1 yes, 0 no	0	
	6	Pressure sensor 6 value in range	1 yes, 0 no	0	
	7	1	-	128	
3	0	Environmental sensors ident pass	1 yes, 0 no	0	128
	1	IMU ident pass	1 yes, 0 no	0	
	2	IMU acc self test pass	1 yes, 0 no	0	
	3	IMU gyr self test pass	1 yes, 0 no	0	
	4	External thermistor value in range	1 yes, 0 no	0	
	5	MCU EEPROM checksum okay	1 yes, 0 no	0	
	6	DynCal present and checkum okay	1 yes, 0 no	0	
	7	1	-	128	

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, Env_ident, IMU_ident, IMU_acc_pass, IMU_gyr_pass, T0_val_ok, EEPROM_CRC16_pass	
S	Perform self-test and retrieve status bytes	4x uint8	P_chk_ok, P_tmp_ok, P_val_ok, Env_ident, IMU_ident, IMU_acc_pass, IMU_gyr_pass, T0_val_ok, EEPROM_CRC16_pass	
z	Perform temporary auto-zero	7x float32	P_raw_offset(0 to 6)[28]	
Z	Perform permanent auto-zero (write values to EEPROM)	7x float32	P_raw_offset(0 to 6)[28]	
R	Read all EEPROM values	70x uint8	Refer to "EEPROM map.xls" for description of returned bytes	
W	Write all EEPROM values	-		
D	Enable USB/UART streaming	71x uint8	'#'[1], P0[4], P1[4], P2[4], P3[4], P4[4], P5[4], P6[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.xls" for description of mode bytes
N	Get serial number	1x float32	Serial_num[4] (EEPROM index 44…47)	

G	Get current data packet (full)	71x uint8	'#'[1], P0[4], P1[4], P2[4], P3[4], P4[4], P5[4], P6[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)	35x uint8	'#'[1], P0[4], P1[4], P2[4], P3[4], P4[4], P5[4], P6[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	1969x 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	Pressure 2 raw offset	float32	-
9			
10			
11			
12	Pressure 3 raw offset	float32	-
13			
14			
15			
16	Pressure 4 raw offset	float32	-
17			
18			
19			
20	Pressure 5 raw offset	float32	-
21			
22			
23			
24	Pressure 6 raw offset	float32	-
25			
26			
27			
28	Atmospheric pressure offset	float32	Pa
29			
30			
31			
32	External thermistor temperature offset	float32	°C
33			
34			
35			
36	Serial number	float32	-
37			
38			
39			
40	UART baud rate (Set to 0 to use the Rx line as the hardware trigger line. Ensure UART streaming on power-up is set to 0 in this case)	float32	bps
41			
42			
43			
44	Accelerometer xyz scale factor	float32	-
45			
46			
47			
48	Gyroscope x component offset	float32	deg/s
49			
50			
51			
52	Gyroscope y component offset	float32	deg/s
53			
54			
55			
56	Gyroscope z component offset	float32	deg/s
57			
58			

59			
60	Default data rate on power-up	uint16	Hz
61			
62	UART streaming enabled on power-up	unit8	-
63	USB streaming enabled on power-up	unit8	-
64	UART streaming full data packet	unit8	-
65	Accelerometer range mode	unit8	-
66	Gyroscope range mode	unit8	-
67	IMU update rate mode	unit8	-
68	HW trigger edge polarity (0 for falling)	unit8	0 or 1
69	MCU EEPROM CRC16-CCITT	uint16	-
70			
71	DynCal N points per channel	uint8	-
72	Data rate of dynamic calibration	uint16	Hz
73			
74	Frequency step size	float32	Hz
75			
76			
77			
78	Ch 0 gain scale value	float32	-
79			
80			
81			
82	Ch 0 gain correction, frequency 0	uint16	-
83			
84	Ch 0 gain correction, frequency 1	uint16	-
85			
	...		
216	Ch 0 gain correction, frequency 67	uint16	-
217			
218	Ch 0 phase scale value	float32	rad
219			
220			
221			
222	Ch 0 phase correction, frequency 0	uint16	-
223			
224	Ch 0 phase correction, frequency 1	uint16	-
225			
	...		
356	Ch 0 phase correction, frequency 67	uint16	-
357			
358	Ch 1 gain scale value	float32	-
359			
360			
361			
362	Ch 1 gain correction, frequency 0	uint16	-
363			
364	Ch 1 gain correction, frequency 1	uint16	-
365			
	...		
496	Ch 1 gain correction, frequency 67	uint16	-
497			
498	Ch 1 phase scale value	float32	rad
499			
500			

501			
502 503	Ch 1 phase correction, frequency 0	uint16	-
504 505	Ch 1 phase correction, frequency 1	uint16	-
	...		
636 637	Ch 1 phase correction, frequency 67	uint16	-
638 639 640 641	Ch 2 gain scale value	float32	-
642 643	Ch 2 gain correction, frequency 0	uint16	-
644 645	Ch 2 gain correction, frequency 1	uint16	-
	...		
776 777	Ch 2 gain correction, frequency 67	uint16	-
778 779 780 781	Ch 2 phase scale value	float32	rad
782 783	Ch 2 phase correction, frequency 0	uint16	-
784 785	Ch 2 phase correction, frequency 1	uint16	--
	...		
916 917	Ch 2 phase correction, frequency 67	uint16	-
918 919 920 921	Ch 3 gain scale value	float32	-
922 923	Ch 3 gain correction, frequency 0	uint16	-
924 925	Ch 3 gain correction, frequency 1	uint16	-
	...		
1056 1057	Ch 3 gain correction, frequency 67	uint16	-
1058 1059 1060 1061	Ch 3 phase scale value	float32	rad
1062 1063	Ch 3 phase correction, frequency 0	uint16	-
1064 1065	Ch 3 phase correction, frequency 1	uint16	-
	...		
1196 1197	Ch 3 phase correction, frequency 67	uint16	-
1198 1199 1200 1201	Ch 4 gain scale value	float32	-
1202	Ch 4 gain correction, frequency 0	uint16	-

1203			
1204	Ch 4 gain correction, frequency 1	uint16	-
1205			
	...		
1336	Ch 4 gain correction, frequency 67	uint16	-
1337			
1338	Ch 4 phase scale value	float32	rad
1339			
1340			
1341			
1342	Ch 4 phase correction, frequency 0	uint16	-
1343			
1344	Ch 4 phase correction, frequency 1	uint16	-
1345			
	...		
1476	Ch 4 phase correction, frequency 67	uint16	-
1477			
1478	Ch 5 gain scale value	float32	-
1479			
1480			
1481			
1482	Ch 5 gain correction, frequency 0	uint16	-
1483			
1484	Ch 5 gain correction, frequency 1	uint16	-
1485			
	...		
1616	Ch 5 gain correction, frequency 67	uint16	-
1617			
1618	Ch 5 phase scale value	float32	rad
1619			
1620			
1621			
1622	Ch 5 phase correction, frequency 0	uint16	-
1623			
1624	Ch 5 phase correction, frequency 1	uint16	-
1625			
	...		
1756	Ch 5 phase correction, frequency 67	uint16	-
1757			
1758	Ch 6 gain scale value	float32	-
1759			
1760			
1761			
1762	Ch 6 gain correction, frequency 0	uint16	-
1763			
1764	Ch 6 gain correction, frequency 1	uint16	-
1765			
	...		
1896	Ch 6 gain correction, frequency 67	uint16	-
1897			
1898	Ch 6 phase scale value	float32	rad
1899			
1900			
1901			
1902	Ch 6 phase correction, frequency 0	uint16	-
1903			
1904	Ch 6 phase correction, frequency 1	uint16	-

1905			
	...		
2036	Ch 6 phase correction, frequency 67	uint16	-
2037			
2038	DynCal CRC-16 CCITT	uint16	-
2039			

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, 0x64E6, 0x74C7, 0x44A4, 0x5485,
  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, 0x6657, 0x7676, 0x4615, 0x5634,
  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: Reference coordinate systems

The spherical coordinate system commonly used in physics (ISO 80000-2:2019 convention) is adopted here and used throughout. By convention, the pitch angle α increases with rotation toward the *z*-axis, and the yaw angle β is positive rotating anti-clockwise about the *z*-axis (see figure 9). Here, *u*, *v* and *w* are the velocity components along *x*, *y* and *z*, respectively.

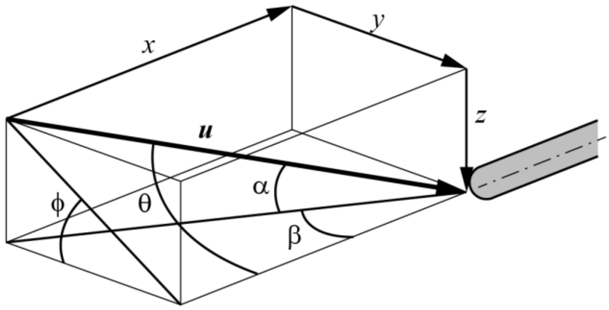


Figure 9: Coordinate conventions

Probe-based coordinate system:

This convention is intended for use on a moving platform, such as an aircraft or vehicle.

$$u = |U| \cos(\beta) \cos(\alpha)$$
$$v = |U| \sin(\beta) \cos(\alpha)$$
$$w = |U| \sin(\alpha)$$

Wind tunnel-based coordinate system:

This convention is intended for use in stationary-probe wind tunnel applications with a right-handed coordinate system with the probe pointing in the upstream *x* direction, such that the z-axis is vertical (environmental flow convention).

$$u = |U| \cos(\beta) \cos(\alpha)$$
$$v = -|U| \sin(\beta) \cos(\alpha)$$
$$w = |U| \sin(\alpha)$$

Rotated wind tunnel-based coordinate system:

This convention is intended for use in stationary-probe wind tunnel applications with a right-handed coordinate system with the probe pointing in the upstream *x* direction, such that the y-axis is vertical (aerospace flow convention).

$$u = |U| \cos(\beta) \cos(\alpha)$$
$$v = |U| \sin(\alpha)$$
$$w = |U| \sin(\beta) \cos(\alpha)$$

Appendix 3: Calibration process and files

Calibration algorithm

The calibration algorithm used is the generalized, sectorless technique of Shaw-Ward *et al*[1]. The pressure at each hole i is reduced to a nondimensional coefficient C_i as

$$C_i = \frac{P_{max} - P_i}{P_{max} - P_{min}}$$

where P_i is the differential pressure recorded at hole i, and P_{max} and P_{min} are the maximum and minimum differential pressures recorded over all holes, respectively. These coefficients are reasonably independent of velocity magnitude, but calibration should still be carried out near the expected operating speeds or at the centre of the expected range. During calibration, empirical functions $C_i(\alpha, \beta)$ are obtained (where α and β are the pitch and yaw angles) as well as a stagnation differential pressure coefficient C_0.

During measurement, then, P_i is converted into C_i and an interpolation process is used to return the closest matching flow angles. With the matching flow angles then known, C_0 can be interpolated from the calibration data at those angles, returning the local stagnation pressure; the velocity magnitude can then be obtained from the stagnation pressure.

Generating calibration data files

To facilitate (and accelerate) the interpolation process, the calibration data files used by the testing executable need to be in the form of structured grids in (α, β). These are saved as independent files in user-readable ASCII format, for verification purposes. Appropriately-formatted calibration files may be generated from a single unstructured table of data points using the calibration file generation tool, CALIBRATION_DATA_RESAMPLER.EXE.

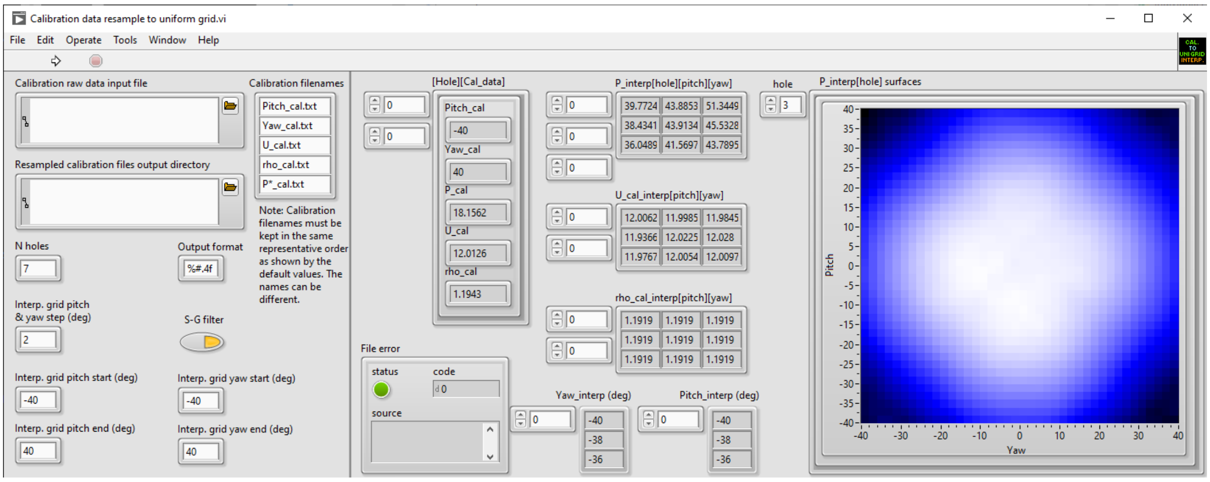


Figure 10: Calibration file generator user interface

---

[1] Shaw-Ward, S., Titchmarsh, A. and Birch, D. M. (2015) "Calibration and use of *n*-hole velocity probes." AIAA J. 53(2), 336-346.

Operating instructions:

Step 1: Identify the unstructured (raw) data input file. This must be a tab-delimited ASCII *.TXT data file; the first two rows are reserved for column headers, and the columns must contain data organized as follows:

Yaw angle (deg)	Pitch angle (deg)	P0 (Pa)	P1 (Pa)	P2 (Pa)	...	PN (Pa)	U (m/s)	ρ (kg/m^3)

where *N* is the number of holes in the probe (*N* = 7 here), *U* is the calibration free-stream velocity and ρ is the fluid density estimate.

Once the file is identified, click on the folder icon in the [CALIBRATION RAW DATA INPUT FILE] field and navigate to the file location.

Step 2: Locate directory for output files. The output files must all be in the same directory; this is the directory you need to identify in the [RESAMPLED CALIBRATION FILES DIRECTORY] window under the [ACTION] tab of the data acquisition executable, as part of the setup process.

Step 3: Enter desired parameters for resampling. This software will interpolate your data over a regular grid, as required for the data conversion process. The user can tailor how this is done. The editable parameters are as follows:

N Holes: This allows the user to modify the number of holes in the probe. This should be set to '7' for a seven-hole probe.

Output format: This allows the user to modify the precision of the output data. Normally, this should be left to six decimal places, or [`%#.6f`].

Interp grid pitch & yaw step (deg): This allows the user to specify the grid spacing over which the data will be resampled. It is normally recommended that this be set to the raw data grid spacing (or smallest spacing, if a nonuniform measurement grid was used). Data will be resampled to the specified grid spacing using a piecewise nonlinear interpolation.

S-G filter: This option is available to minimize the influence of noise in the data fields, and reduce the uncertainty if the individual data points were not well converged. When enabled, a Savitzky-Golay[2] filter is used to smooth the fields.

Interp. grid. pitch start (deg): Minimum pitch angle at which calibration data will be required. This should normally be the minimum angle at which calibration data were obtained, although the option is available to use only a subset of the available raw data.

[2] Schafer, R. W. (2011) "What is a Savitzky-Golay Filter?" IEEE Sig. Proc. 28(4).

Interp. grid. yaw start (deg): Minimum yaw angle at which calibration data will be required. This should normally be the minimum angle at which calibration data were obtained, although the option is available to use only a subset of the available raw data.

Interp. grid. pitch end (deg): Maximum pitch angle at which calibration data will be required. This should normally be the maximum angle at which calibration data were obtained, although the option is available to use only a subset of the available raw data.

Interp. grid. yaw end (deg): Maximum yaw angle at which calibration data will be required. This should normally be the maximum angle at which calibration data were obtained, although the option is available to use only a subset of the available raw data.

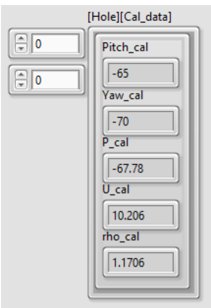
Calibration filenames: The user is free to rename the files generated by this software utility. The fields are not labelled individually, but the order is shown below with default file names.

Pitch angles `Pitch_cal.txt`
Yaw angles `yaw_cal.txt`
Velocity `U_cal.txt`
Density `rho_cal.txt`
Pressures `P*_cal.txt`

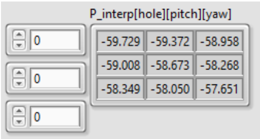
Extensions will not automatically be added. For the pressure file name, an asterisk (*) character will be replaced by the hole index number in the output files. ***IMPORTANT:* Although file names are editable, default file names must be used for the files to be recognized by the data acquisition executable.**

Step 4: Run the executable. This is done by clicking on the arrow icon under the menu bar.

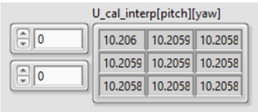
If the executable ran without error, a green indicator will appear under [STATUS] in the [FILE ERROR] field. The user may then inspect the outputs.



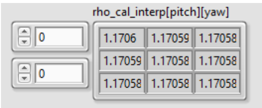
HOLE/CAL_DATA: This field allows the user to inspect each of the entries in the raw data file. The first incrementing window allows the user to select the hole number (affecting the pressure [P_CAL] only); the second incrementing window allows the user to select the row number in the input data set. The output entries are in the same order as the columns in the input data file.



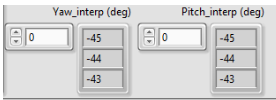
P_INTERP/HOLE/PITCH/YAW: This field allows the user to inspect the interpolated output pressure data arrays. The first incrementing window is the hole number; the second is the pitch angle index and the third is the yaw angle index. The output shows a 3 x 3 window of the data written to the output file, centred on the point selected (the specified pitch angle/yaw angle indices).



U_CAL_INTERP/HOLE/PITCH/YAW: This field allows the user to inspect the interpolated output velocity data array. The first incrementing window is the pitch angle index and the second is the yaw angle index. The output shows a 3 x 3 window of the data written to the output file, centred on the point selected (the specified pitch angle/yaw angle indices).



RHO_CAL_INTERP/HOLE/PITCH/YAW: This field allows the user to inspect the interpolated output density data array. The first incrementing window is the pitch angle index and the second is the yaw angle index. The output shows a 3 x 3 window of the data written to the output file, centred on the point selected (the specified pitch angle/yaw angle indices).



YAW_INTERP (DEG) & PITCH_INTERP (DEG): This field shows the angles at which data were interpolated. The incrementing windows are the respective indices.

When the resolution is sufficiently high, the software also produces a filled contour map of C_i over the pitch and yaw angle space for each hole. The user can select the hole number from the incrementing window. Note that this is for identifying irregular response surfaces only; contour levels are self-scaling, but may be adjusted using the right-click menu.

Step 5: Load the resultant files in the data acquisition [ACTION] pane. When running the [7HP USER INTERFACE] executable, the directory path of the resampled calibration files should be entered into the [RESAMPLED CALIBRATION FILES DIRECTORY] field. The data acquisition executable will then be able to stream velocity components and process data in real-time.

File management
The generation of properly-formatted structured calibration files need only be done once per calibration, and the resultant files can be loaded any time the data acquisition executable is launched. Normally, the formatted data files would be supplied with the probe if the probe was factory-calibrated.

Appendix 4: 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 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:

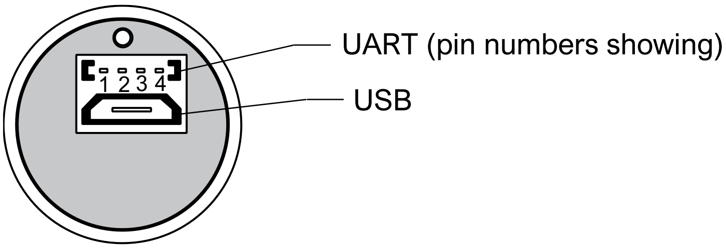
- Set [UART BAUD RATE] to [0] (zero)
- Set [UART STREAMING] to [0] (zero)
- Set [HW TRIGGER POLARITY] to either [0] (zero) or [1] (one). A value of zero will trigger USB data streaming on a falling edge, while a value of one will trigger on a rising edge.

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.



UART
Pin 1 = V+
Pin 2 = GND
Pin 3 = Rx
Pin 4 = Tx

Figure 11: 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.