DIGITAL 7-HOLE

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

Version Control

Version	Date	Summary of changes	Software vn.
1.0	05-2017	New document	
2.0	09-2019	Updated content and specifications, reformatted	
2.1	01-2020	Expansion of guidance on software and drivers. Addition on modular probes.	
2.2	04-2020	Correction to serial comms pin connections. Updated serial stream and checksum. Addition of appendix containing further comms details.	1.1

1 INTRODUCTION

Principle of operation

This probe system includes a seven-hole pressure-based directional velocity 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 will 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

Bespoke miniature multi-function seven-hole probe system.

System components

1x Probe assembly OR 1x Modular probe system assembly	The probe assembly consists of a seven-hole probe core with a local static sampling sleeve, a sensor package and an enclosure. 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 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.

System requirements

To interface with a computer, the probe system requires 64-bit Windows 7 (or newer) operating system (not included). 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 will be provided electronically. Additional drivers may be required to run the software on older Windows machines.

2 DETAILED SPECIFICATION

7 x differential pressure transducers	ID7HP-160P	ID7HP-1K0	ID7HP-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
Compensated temperature range	-40° to +65° C	0° to +50° C	0° to +50° C
Operating temperature range	-40° to +65° C		
Storage temperature range	-40° to +65° C		
Vibration	Sensors rated to 15 g, 10 Hz to 2 Hz		
Maximum relative humidity	95 %		
Relative 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[4]	30-110 kPa FS, +/- 0.1kPa		
Reference pressure	Integrated with static pressure ring on sting or connectable to laboratory conditions		
Standard sting material	Steel		
Sting diameter	3.7 mm		
Standard tip geometry	Hemispherical		
Angular measurement range[5]	± 45°		
Remote temperature probe	-40°C - 150°C ± 0.5°C		
Absolute temperature limits	5° to +65° C non-condensing		
Absolute pressure limits	0.2 - 1.5 atm		
Weight (approximate)	45 g		
Voltage	6-24 VDC or via USB Reverse-polarity protected ESD protected to ±2 kV		
Power	min. 290 mW		
Communications interface	USB2.0, RS232 or UDP Ethernet available		
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		
Data output	Continuous streaming, 100 Hz Baud rate: 230400 BPS Single-precision floats in SI units TTL 3.3V serial output or via USB TTL serial ESD protected to ±25 kV		
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] Pressure is recorded inside the sensor package enclosure. Please ensure that the vents on the sides of the enclosure are clear to ensure correct readings
[5] Angular range specified for hemispherical tip; depends on tip geometry.

3 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 connections. Data will then begin streaming serial data on the TTL lines. The system will stream data continuously while powered on.

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.

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 case temperature (sensor die)	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	230400 bps
UART config	1 start bit / 8 data bits/ 1 stop bit
Parity	none

The data streams as 3.3V TTL output; this is not 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 70 byte receive buffer to prevent data overrun.

A partial data packet, having only 35 bytes and excluding less critical data, is also available from the probe. The data order for this data packet is shown below.

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 loss or corruption in transmission. Example C++ code and DLL files to compute the CRC-16 checksum are available upon request. 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.

4 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.

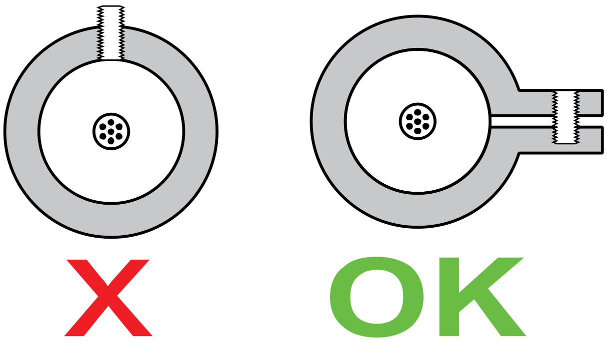

Figure 1: Correct probe mounting

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 below. Note that the roll alignment is arbitrary. Note that this may vary for some probes.

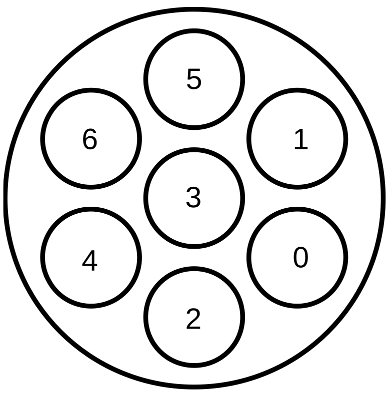

Figure 2: Hole numbering convention, looking at probe tip from upstream, with the probe body located behind.

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 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.

IMPORTANT: UART configuration should always be tested with 5V initially: pins are reverse-polarity protected up to 5V only.

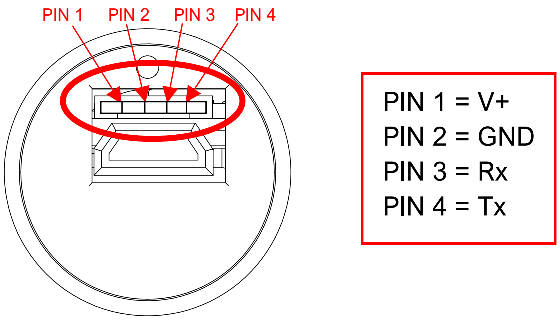

Figure 3: Serial comms pin assignment

The length of the data cables should be limited to 1.8 m or less, and should be routed away from the board and any cables carrying electric motor currents.

IMPORTANT: The Tx line described is the transmit bus for the sensor package. This should be connected to the Rx of the unit receiving the data.

Modular system components

The modular probe system consists of a sensor package, gasket, sting assembly and retainer as shown in Figure 4.

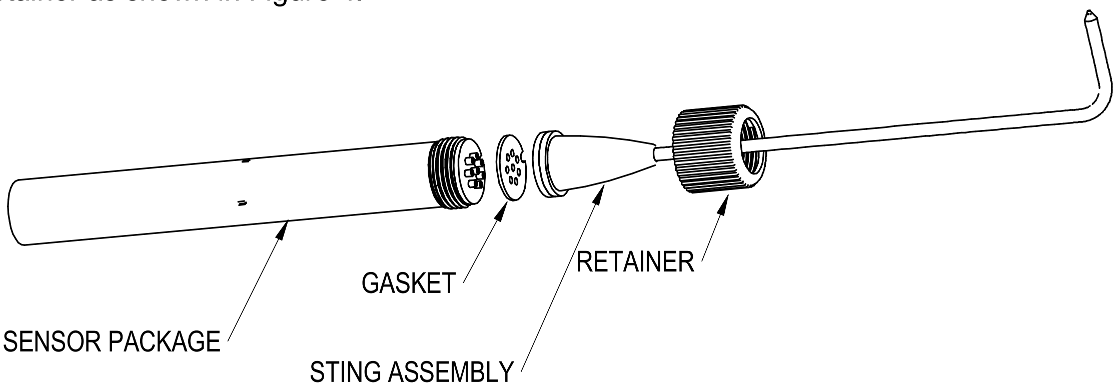

Figure 4: Modular probe system components

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

5 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 probe enclosure is made from sintered Nylon, and is susceptible to solvents. The probe sting is not commoned 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.
- 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.

6 SOFTWARE AND DRIVERS

Although this probe system was designed to function independently of a PC by continuously streaming data, it is possible to interface with a PC via the included USB port for data logging, 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 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 Figure 6 and 7 below.

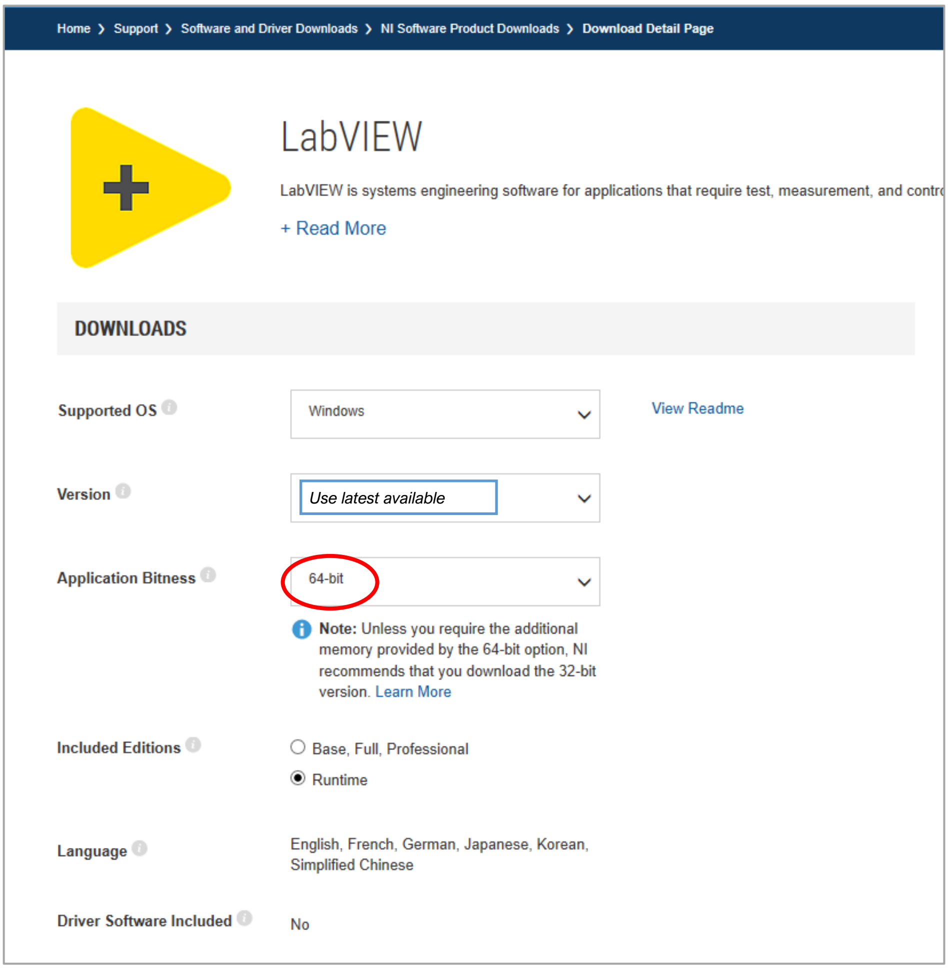

Figure 6: Download settings for NI Labview Runtime Engine

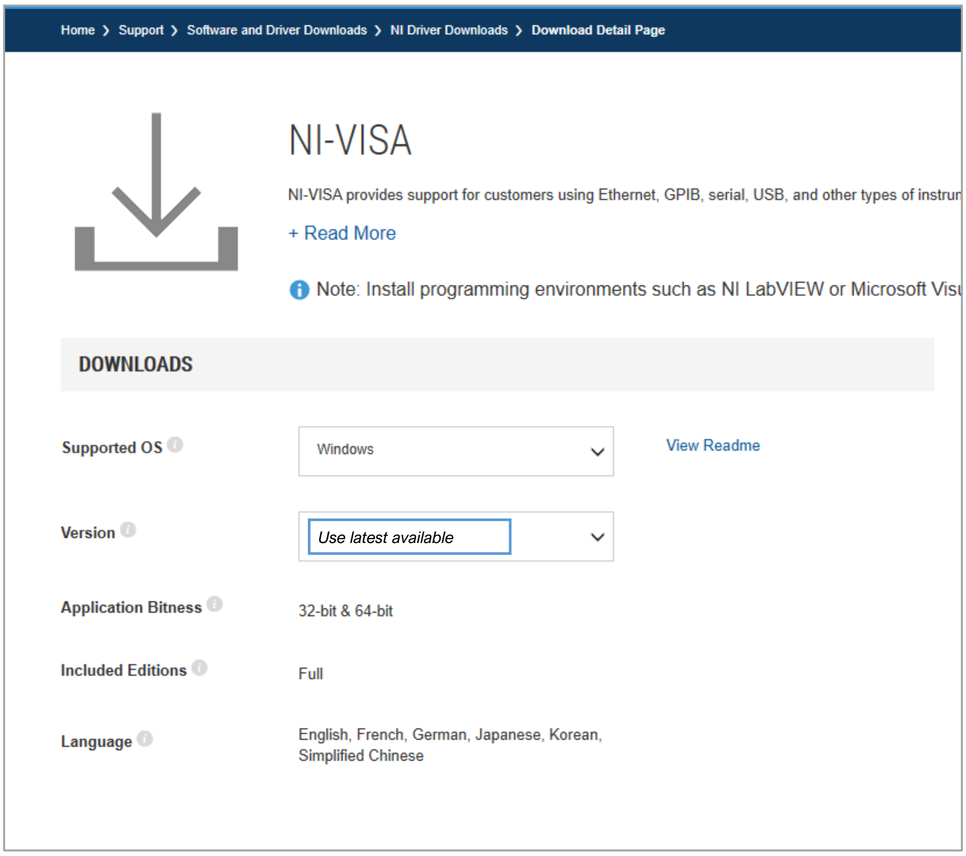

Figure 7: Download settings for NI VISA Runtime Engine

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

Executable

An executable software package is provided with 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 below (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 with a green LED.
3) Load the program. It will start in the [ACTION] tab.
4) Using the [COM PORT] drop down menu, select the appropriate COM port.
5) Press the white arrow near the top left corner of the window to run the application.
6) 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.

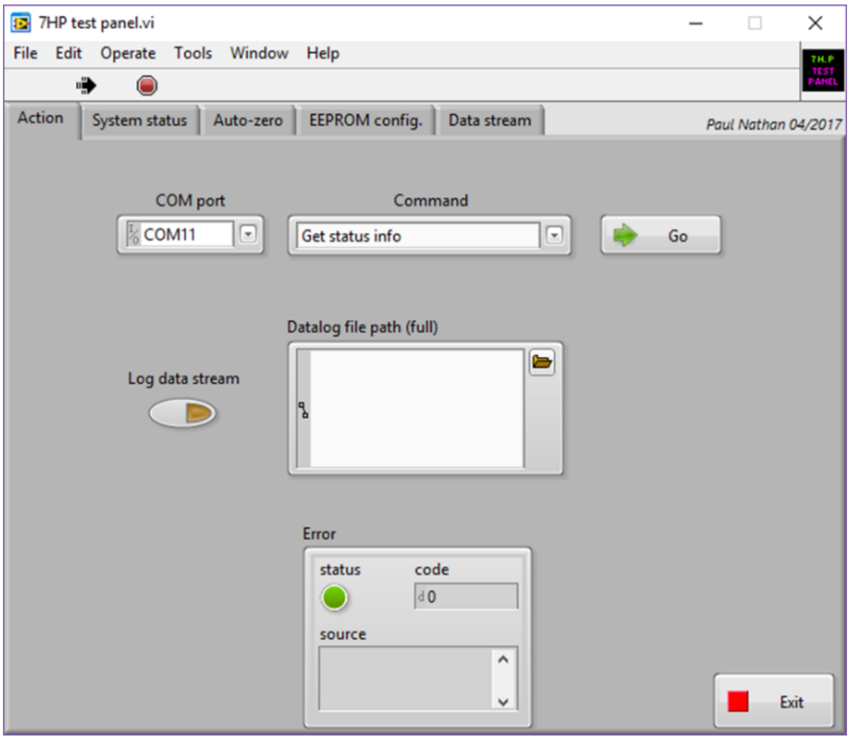

Figure 8: Probe system companion software screen-shot

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.

- **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 should not be used under normal operating circumstances, and is included for development purposes only. It 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.

- **Perform permanent auto-zero**: this command repeats the above, but this time writing the offsets permanently to non-volatile memory. ***WARNING*** *– this will overwrite the existing offsets that were used during probe calibration. Carrying out this procedure will invalidate your calibration!*

- **Get EEPROM values:** displays the [EEPROM CONFIG] tab and read off all the EEPROM values from the system.

- **Set EEPROM values:** this command should not be used under normal operating circumstances, and is included for development purposes only. To make changes to the EEPROM values, first run [GET EEPROM VALUES] to populate the table with the current values. Then, change 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. ***WARNING*** *– this will overwrite the existing EEPROM values that were used during probe calibration. Carrying out this procedure will invalidate your calibration!*

- **Set data rate*:*** This command allows the user to set the system sampling rate using a drop-down menu.

- **Get data rate*:*** This command allows the user to retrieve the current setting of the system sampling rate.

- **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 TTL serial output is not affected by enabling USB streaming.

- **Disable USB streaming:** this command stops data transmission over USB.

- **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].

- **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.

Data logging option:

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 on the [ACTION] tab prior to running the [ENABLE USB STREAMING] command. 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.

Notes on data reduction

The probe outputs seven pressures and a static pressure for use in resolving the three components of velocity. There are a number of open-source techniques for data conversion, offering various levels of trade-off between accuracy and speed. Examples and advice on implementing this process for your application are available from your Surrey Sensors Ltd. representative. A directional calibration service is also available on request.

7 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: 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	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	BME280 ident pass	1 yes, 0 no	0	192
	1	BMI160 ident pass	1 yes, 0 no	0	
	2	BMI160 acc self test pass	1 yes, 0 no	0	
	3	BMI160 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	1	-	64	
	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, BME280_ident, BMI160_ident, BMI160_acc_pass, BMI160_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, BME280_ident, BMI160_ident, BMI160_acc_pass, BMI160_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/TTL 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 TTL enabled according to physical layer on which the command was sent
d	Disable USB/TTL streaming	-		USB or TTL 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 TTL streaming on power-up	-		
y	Disable TTL streaming on power-up	-		
P	Set TTL stream full data packet mode (power-up default)	-	Send 1x uint8. 1 = full data, 0 = partial data	
p	Get TTL stream full data packet mode	1x uint8	1 = full data, 0 = partial data	
F	Set data rate	-	Send 1x uint16, DataRate[2] containing one of the allowable integer data rates.	
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 TTL baud rate (bps) (power-up default)	-	Send 1 float32 (as 4 bytes), Baud[4]	
b	Get TTL baud rate (bps)	1x float32	Baud[4]	

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 Map

byte index	Description	Type	Unit
0, 1, 2, 3	Pressure 0 raw offset	float32	-
4, 5, 6, 7	Pressure 1 raw offset	float32	-
8, 9, 10, 11	Pressure 2 raw offset	float32	-
12, 13, 14, 15	Pressure 3 raw offset	float32	-
16, 17, 18, 19	Pressure 4 raw offset	float32	-

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	deg C
33			
34			
35			
36	Serial number	float32	-
37			
38			
39			
40	TTL baud rate	float32	bps
41			
42			
43			
44	Accelerometer xyz scale factor	float32	-
45			
46			
47			
48	Default data rate on power-up	uint16	(Hz)
49			
50	TTL streaming enabled on power-up	unit8	-
51	USB streaming enabled on power-up	unit8	-
52	TTL streaming full data packet	unit8	-
53	Accelerometer range mode	unit8	-
54	Gyroscope range mode	unit8	-
55	IMU update rate mode	unit8	-
56	Gyroscope x component offset	float32	deg / s
57			
58			
59			
60	Gyroscope y component offset	float32	deg / s
61			
62			
63			
64	Gyroscope z component offset	float32	deg / s
65			
66			
67			
68	CRC16-CCITT	uint16	-
69			

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	