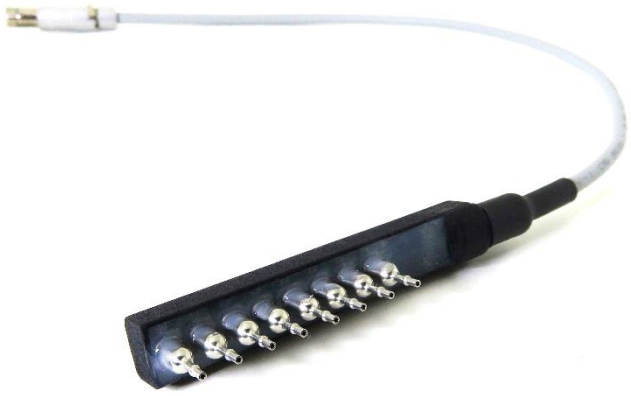
8-CHANNEL MINIATURE PRESSURE SCANNER SYSTEM

User Manual



WARNING

Read this document before using the product.

This system is not certified for use on aircraft.

Contents

Version Control

Version	Date	Summary of changes	Software release
1.0	03-2021	New document	1.0
1.1	04-2021	Product code and imagery updated	1.0
1.2	07-2021	Software update	MUS-8 1.0 CAN-2.0B module 1.1
1.3	08-2022	Addition of USB adapter	MUS-8 1.0 CAN-2.0B module 1.1
1.4	05-2025	Inclusion of additional comms details	MUS-8 1.0 CAN-2.0B module 1.1

1 INTRODUCTION

Principle of operation

The system consists of an 8-channel pressure scanner system and an optional communications module.

The system's power and communications may be via USB or 3.3V UART, with CAN2.0B and Ethernet available via an additional communications module.

System description

Miniature multichannel pressure scanner system.

System components

1x Micro-scanner system	8-channel micro-scanner system with common static reference port
1x communications module/ adapter	CAN2.0B processing module OR USB Adapter
Accessories	Accessories including pressure cables and connectors are available and may have been included in the shipment.

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

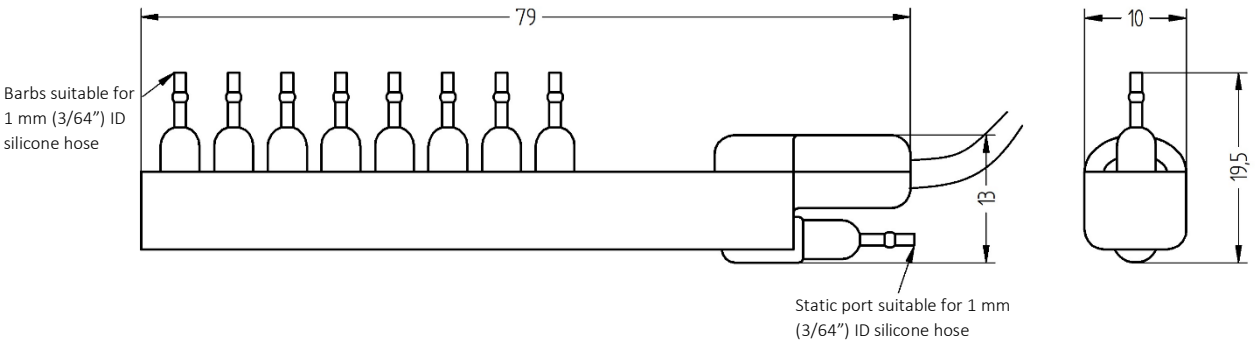
2 DETAILED SPECIFICATION

Specification

Pressure range	± 6.9 kPa FS
Maximum overpressure	34 kPa
Sensor repeatability[1]	± 0.024 % FS
Sensor accuracy[2], inc. calibration	± 0.25 % FS
Total error band after auto-zero[3]	± 1.25 % FS
Operational mode	Differential
Operating temperature range	-40° to +85° C non-condensing
Compensated temperature range	0° to +50° C non-condensing
Storage temperature range	-40° to +85° C
Vibration	Sensors rated to 10 g, 10 Hz to 2 kHz
Shock	50 g, 6 ms duration
Maximum relative humidity	95 % (external surfaces) 100 % (internal surfaces)
Communications interface	USB, 3.3V UART, or CAN2.0B via comms module
Power consumption	324 mW (whole system: scanner and comms module)
Supply voltage	5 Vdc regulated (7-36 Vdc if using CAN module)
Data acquisition rate	200 Hz typ. simultaneous
Digital resolution	24-bit (truncated to 16-bit over CAN)

[1] Includes errors due to pressure non-linearity, pressure hysteresis and non-repeatability.
[2] Includes errors due to pressure non-linearity, pressure hysteresis, non-repeatability and calibration uncertainty.
[3] Total residual error after auto-zero, excluding residual temperature sensitivity.

Dimensions



3 COMMUNICATIONS

When the scanner is powered on, it will display a single green LED, then undergo a brief system diagnostic test; once this is complete a second green LED will illuminate. If there is a boot error, the second green LED will not illuminate.

The CAN2.0B communications module will display a single green LED if the system diagnostic test is passed, and a red error light if a CAN bus state error is present. The system will stream data immediately on power-up following successful completion of these tests (~10ms).

CAN2.0B Data Packet Formats

The CAN2.0B communications module comes pre-configured with a CAN Standard base ID of 0x0001, bus length of 1 m, and baud rate of 1 Mbps. Ensure that baseID, baseID+1 and baseID+2 are not occupied on the same CAN network.

Note these settings are configurable with access to the USB port on the CAN communications module, as detailed in Section 6. User-configurable options are: base ID (0 to 0x07FF) (or to use an Extended base ID (0 to 0x1FFFFFFF)), baud rate (only 125, 250, 500, 800, 1000 kbps), CAN bus length, and address filtering and masking.

Byte no.	Value	Type	Description	Converted Unit	Conversion Info.
0	P0	int16	Sensor 0	Pa	P0 = int16_val * 6894.7573 / 32767.0
1					
2	P1	int16	Sensor 1	Pa	P1 = int16_val * 6894.7573 / 32767.0
3					
4	P2	int16	Sensor 2	Pa	P2 = int16_val * 6894.7573 / 32767.0
5					
6	P3	int16	Sensor 3	Pa	P3 = int16_val * 6894.7573 / 32767.0
7					

Byte no.	Value	Type	Description	Converted Unit	Conversion Info.
0	P4	int16	Sensor 4	Pa	P4 = int16_val * 6894.7573 / 32767.0
1					
2	P5	int16	Sensor 5	Pa	P5 = int16_val * 6894.7573 / 32767.0
3					
4	P6	int16	Sensor 6	Pa	P6 = int16_val * 6894.7573 / 32767.0
5					
6	P7	int16	Sensor 7	Pa	P7 = int16_val * 6894.7573 / 32767.0
7					

Byte no.	Value	Type	Description	Converted Unit	Conversion Info.
0	T_board	int16	Board temp.	°C	T_board = int16_val * 0.01
1					
2	P_status	uint8	Status byte	-	1 pass, 0 fail (each sensor bit)
3	CRC_ok	uint8	CRC-16 flag	-	1 pass, 0 fail

IMPORTANT NOTE: there comms module has no termination resistor included. It is up to the user to terminate the CAN BUS as appropriate.

Serial stream

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	Pressure 7	float32	Pa
30			
31			
32			

33	T_board	float32	°C
34			
35			
36			
37	Status 0	uint8	-
38	Status 1	uint8	-
39	Status 2	uint8	-
40	Status 3	uint8	-
41	Status 4	uint8	-
42	Status 5	uint8	-
43	Status 6	uint8	-
44	Status 7	uint8	-
45	CRC16-CCITT*	uint16	-
46			

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

USB Command list

Cmd. Byte	Description	Return	Additional information
s	Retrieve status byte	3x uint8	P_val_ok, P_status_ok, EEPROM_CRC16_pass
S	Perform self-test and retrieve status bytes	3x uint8	P_val_ok, P_status_ok, EEPROM_CRC16_pass
z	Perform temporary auto-zero	8x float32	P_offset(0 to 7)[32]
Z	Perform permanent auto-zero (write values to EEPROM)	8x float32	P_offset(0 to 7)[32]
e	Read all EEPROM values	49x uint8	P_offset(0 to 7)[32], T_board_offset[4], SerialNum[2], Powerup_serial_data_period[4], UART_on_powerup[1], UART_baud[4], CRC16[2]
E	Write all EEPROM values (including new checksum)	-	
D	Enable USB streaming (returns at [datarate] until stream disabled)	47x uint8	'#'[1], P0[4], P1[4], P2[4], P3[4], P4[4], P5[4], P6[4], P7[4], T_board[4], S0[1], S1[1], S2[1], S3[1], S4[1], S5[1], S6[1], S7[1], CRC16[2]
d	Disable USB streaming	-	
H	Enable hardware trigger	-	
h	Disable hardware trigger	-	
F	Set data period	-	Send 1x uint32, Data_Period[4], containing the data period in microseconds
f	Get data period	1x uint32	Data period (us)
N	Get serial number	1x uint16	Serial_num[2] (EEPROM index 36…37)

G	Get current data packet	47x uint8	'#'[1], P0[4], P1[4], P2[4], P3[4], P4[4], P5[4], P6[4], P7[4], T_board[4], S0[1], S1[1], S2[1], S3[1], S4[1], S5[1], S6[1], S7[1], CRC16[2]
J	Set power-up default data period	-	Send 1x uint32, Data_Period[4], containing the data period in microseconds
B	Set power-up default UART baud rate	-	Send 1x uint32, UART_Baud[4]
b	Get UART baud rate	1x uint32	UART_baud[4]
Q	Set power-up default UART streaming enabled	-	
q	Get power-up default UART streaming enabled	1x uint8	1 = yes, 0 = no
R	System soft reset	-	

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

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 system communications, including a summary of CRC-16-CCITT implementation, are appended to the end of this document.

4 PHYSICAL CONNECTIONS

Scanner

Pressure interface

The 8 pressure barbs plus 1 common reference barb are suitable for 1 mm/ 3/64” ID silicone air lines.

Signal interface

Power and communications to the scanner are via the comms module (CAN2.0B) or USB adapter.

USB Adapter

This unit enables a USB Micro-B cable to be connected to the system. The round connector should be connected to that on the scanner, with red markings aligned, and USB cable plugged into the port on the adapter.

CAN2.0B communications module

This processor unit converts data into a CAN2.0B-compatible format. This includes truncation of the data from 24 to 16 bit, to optimise throughput and improve compatibility with other systems.

User Interface

The CAN2.0B module user interface is via a 4-way Molex Mizu-P25, 2.50 mm pitch waterproof wire-to-wire connector.

The system power supply V+ must be connected to a regulated DC supply (7 - 36 V, 12 V nominal). V+ is reverse-polarity protected.

PIN 1 is labelled on the connector and the pins are in order 1-4. The pin assignment is as follows:

Table 1: CAN2.0B comms module connector conductor assignment

PIN	Description
1	CANH
2	CANL
3	GND
4	V+

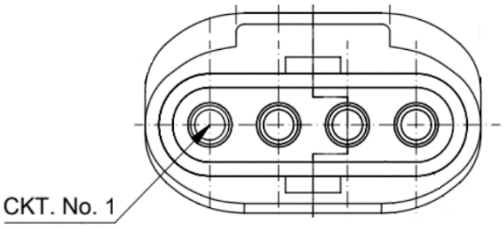


Figure 1: CAN2.0B comms module connector pin locations

5 CARE AND HANDLING

WARNING: Do not allow any conductive materials to come into contact with the system, or it may be permanently damaged.

- During fitting of silicone tubing to scanner and pressure sensor ports, ensure that no dust, dirt or liquid enters the lines. This will alter the system performance. Any foreign material introduced into the sensors themselves may permanently damage the sensors.
- Protect the sensors from moisture and dust, and store in an ESD-safe sleeve when not in service.
- Ensure that all electronic and pressure connections are appropriately strain-relieved.
- Do not use the pressure scanner system in wet or condensing conditions. Store in dry environment, or with desiccant pouch.

6 SOFTWARE AND DRIVERS

To interface with a computer, the system requires 64-bit Windows 7 (or newer) operating system (not included).

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 system, in addition to the specific system driver.

- 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.

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

USB Data Stream Executable

An executable is available operating the device via USB. Connect to the computer via the USB Micro-B port on the adapter unit.

After launching the software, you should see the window reproduced below (Figure 2).

1) Using the [COM PORT] drop down menu, select the appropriate COM port.
2) Press the white run arrow to run software.
3) 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.
4) Choose desired action under [COMMAND] drop-down.
5) Press [GO].

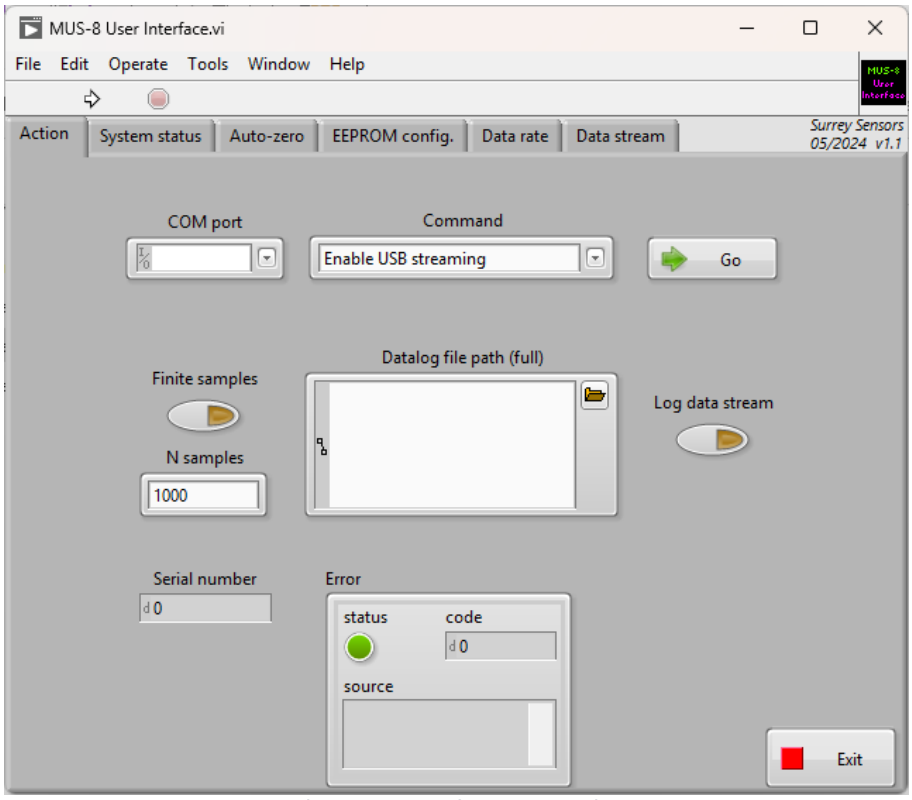


Figure 2: USB software screenshot

CAN2.0B Configuration Executable

An executable is available for configuring the CAN2.0B module settings. Connect to the computer via the outermost USB port on the module.

After launching the software, you should see the window reproduced below (Figure 3).

1) Using the [COM PORT] drop down menu, select the appropriate COM port.
2) Before making changes, select [GET CONFIG] under the [ACTIONS] menu.
3) Press the white run arrow to run software to read the current device configuration.
4) 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.
5) Make desired changes, ensuring that [MUS-8] is selected under the [CONNECTED DEVICE TYPE] menu and select [SET CONFIG] under [ACTIONS].
6) Press run to apply changes.
7) To verify successful application of settings, re-run the [GET CONFIG] command.

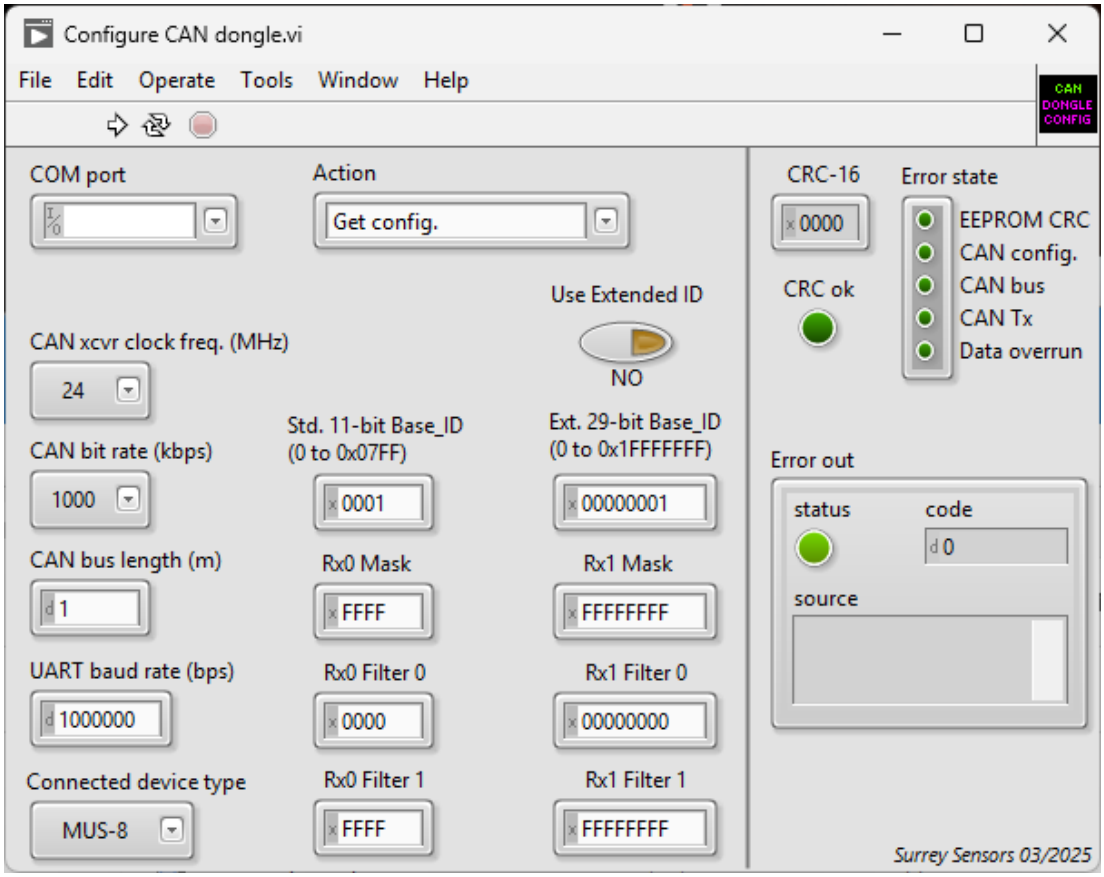


Figure 3: CAN system configuration software screenshot

> ***IMPORTANT NOTE:*** the UART baud rate must match that of the MUS-8.

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

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`	

EEPROM map

Byte index	Description	Type	Unit
0	Pressure 0 offset	float32	Pa
1			
2			
3			
4	Pressure 1 offset	float32	Pa
5			
6			
7			
8	Pressure 2 offset	float32	Pa
9			
10			
11			
12	Pressure 3 offset	float32	Pa
13			
14			
15			
16	Pressure 4 offset	float32	Pa
17			
18			
19			
20	Pressure 5 offset	float32	Pa
21			
22			
23			
24	Pressure 6 offset	float32	Pa
25			
26			
27			
28	Pressure 7 offset	float32	Pa
29			
30			
31			
32	T_board offset	float32	°C
33			
34			
35			
36	Serial number	uint16	-
37			
38	Power-on serial data period	uint32	µs
39			
40			
41			
42	UART data output on power-up	uint8	-
43	UART baud rate	uint32	bps
44			
45			
46			

47 48	CRC16-CCITT	uint16	-

Status byte map

Byte num.	Bit	Description	Info
0	0	Pressure sensor 0 value in range	1 yes, 0 no
	1	Pressure sensor 1 value in range	1 yes, 0 no
	2	Pressure sensor 2 value in range	1 yes, 0 no
	3	Pressure sensor 3 value in range	1 yes, 0 no
	4	Pressure sensor 4 value in range	1 yes, 0 no
	5	Pressure sensor 5 value in range	1 yes, 0 no
	6	Pressure sensor 6 value in range	1 yes, 0 no
	7	Pressure sensor 7 value in range	1 yes, 0 no
1	0	Pressure sensor 0 status good	1 yes, 0 no
	1	Pressure sensor 1 status good	1 yes, 0 no
	2	Pressure sensor 2 status good	1 yes, 0 no
	3	Pressure sensor 3 status good	1 yes, 0 no
	4	Pressure sensor 4 status good	1 yes, 0 no
	5	Pressure sensor 5 status good	1 yes, 0 no
	6	Pressure sensor 6 status good	1 yes, 0 no
	7	Pressure sensor 7 status good	1 yes, 0 no
2	0	On-board temperature sensor okay	1 yes, 0 no
	1	MCU EEPROM checksum okay	1 yes, 0 no
	2	0	-
	3	0	-
	4	0	-
	5	0	-
	6	0	-
	7	0	-