INTERFACE OPTIONS

Tan Delta Protocol on RS485

CAN Open on CANbus

ModBus RTU on RS485

J1939 on CANbus

Issue 2.6

The OQSx sensor can be communicated with by using either Tan Delta Protocol (default option) on RS485, ModBus RTU on RS485, CAN Open on CANbus, or J1939 on CANbus

CONTACT DETAILS
Tan Delta Systems Ltd
1 Carrera Court
Church Lane
Dinnington
Sheffield
UK
S25 2RG

Tel: +44 (0)845 094 8710
support@tandeltasystems.com

BUSINESS HOURS
Monday to Friday: 8:30am to 5:30pm GMT. Support email monitored seven days per week

Revision History – Issue 2

Rev. 1: Synchronous PDO transmission every n^{th} SYNC command implemented.
Asynchronous PDO transmission on timer implemented.
Readings and formats transmitted via PDO now selectable.

Rev. 2: Bitrate index corrected in examples.

Rev. 3: Database interface now documented.
Configurable Bitrate, Transmission Type, Event Timer and PDO selection and format now documented and examples given.
Examples of Oil Data Base value read and write added.

Rev. 4: Hardware Version No format corrected.
Examples of Hardware Version No., Software Version No. and Serial No. read added.

Rev. 5: Read Sensor ID String added to Object Dictionary.

Rev. 6: Proprietary protocol: corrections to byte-counts and explanatory comments about ASCII protocol added.

Rev. 7: Modbus troubleshooting.

Rev. 8: Included note specifying oil condition & temperature units (LF% & °C).

Rev. 9: Moved Modbus 'Data Format' section to more prominent position before register table.

Rev. 10: Removed Cal Zero write function from Modbus section.

Rev. 11: J1939 Documentation Added.

Rev. 12: J1939 Amendments made.

Rev. 13: Amended RS485 bit format from 24 to 32.

Rev. 14: Removed obsolete information pertaining to RS232.
Clarified some of the CANopen details.

Contents

Note: *Oil condition is always output in Loss Factor %. This can be converted to Tan Delta Number This can be converted to TDN using the following formula where T = TDN, L = Loss Factor %;*

T = 900 - 20L

All temperature outputs are in °C and can be converted to °F using the following formula;

T(°F) = T(°C) × 1.8 + 32

Tan Delta Communication Protocol

The command protocol and language for the Tan Delta serial communications uses a binary command format communicating over a half-duplex RS485 interface. The serial configuration must be set to 8 bits with no parity and one stop bit. The default baud-rate is 9600 bits per second. The Tan Delta unit operates in a slave mode, with the serial device which controls the communications (e.g. the monitoring computer) acting as master and issuing commands to which the slave will reply. Tan Delta will not transmit any data except in response to a command from the master and will expect no further commands until the last command has been replied to.

Commands are multiple sequences of bytes which must be interpreted as a complete data string before the correct action and response can be determined. Any command which is not interpreted and verified will cause the command interpreter to reset back to its initial state, and any interruption of communications of longer than 1s will cause the same result. The master must therefore check for correct response in all cases and re-send any commands which have been corrupted or misinterpreted.

Note that all hexadecimal bytes referred to here are physically sent on the RS485 bus as pair of ASCII characters. Thus the wake-up character "!", which has an ASCII value of 0x21 is physically sent as the pair of ASCII bytes "2", "1" (hex 0x32, 0x31). This provides further security as only valid ASCII characters are recognised and simplifies monitoring and debugging of command sequences. It is important to understand that all "bytes" and "byte-counts" referred to below comprise these pairs of ASCII characters when physically transmitted on the bus.

The detailed structure of commands is as detailed below.

The command structure is as follows:-

Byte Number	Value	Description
Byte 0	"!"	acknowledge code
Byte 1	<count>	number of bytes to follow, including <cksm> and <count>
Byte 2	<iaddr>	Single byte instrument address
Byte 3&4	<cmd>	two byte command
Bytes 5 to n-1*	<data>	optional, depending on the command
Byte n, n+1*	<cksm>	16 bit inverse checksum of all preceding bytes including acknowledge

*n = <count-1>

The response structure is as follows:

Byte Number	Value	Description
Byte 0	"A"	wake-up code
Byte 1	<count>	number of bytes to follow, including <cksm> but excluding <count>
Byte 2 to n-1	<response>	optional, depending on the command
Byte n, n+1*	<cksm>	16 bit inverse checksum of all preceding bytes including acknowledge

*n = <count>

Checksum

The checksum is calculated by creating an unsigned 16 bit sum of all preceding data bytes, discarding any overflow and then subtracting the result from 65535.

Command Overview

(see below for detailed description of data and response strings)

Commands are divided into two categories:

- Read commands which read data from the Tan Delta unit, beginning "R"

<cmd>	Description	<data>	<response>
"Rc"	Read config settings	2 byte address plus 1 byte <length>	<length+2> bytes
"Rm"	Read system memory	2 byte address plus 1 byte <length>	<length+2> bytes
"Rr"	Read current readings	2 byte address plus 1 byte <length>	<length+2> bytes
"Rv"	Read version and serial no	2 byte address plus 1 byte <length>	<length+2> bytes

Note that <length> can exceed the data required: additional bytes will be returned but are redundant.

- Write commands which write data to the Tan Delta unit, beginning "W"

<cmd>	Description	<data>	<response>
"Wc"	Write channel settings	2 byte address plus 1 byte <length> plus <length> data	4 bytes
"Wm"	Write system memory	2 byte address plus 1 byte <length> plus <length> data	4 bytes

Note that <length> must be exact and must match the data sent.

1. Read commands allow access to current channel and system settings for confirmation and management of the unit's operation and current measurements (readings) acquired by the units. Note that all commands use exactly the same mechanism as "Read Memory", accessing system memory but adding the address onto a starting address appropriate to the command; thus "Read Channel Settings" with address 0x03 and length 0x06 reads six bytes, starting from the third byte of the Setup structure within system memory.
2. Write commands allow the remote configuration of current channel and system settings for management of the units' operation, and modification of the unit's system memory, for use in monitoring and debugging operations only.

Read Current Readings Command "Rr"

This command has two items of data, a two byte <starting address> and a single byte <length>. It commands the Tan Delta unit addressed by <iaddr> to transmit <length> bytes from <starting address> relative to the start of the current readings array for each of the three channels within the unit. Each reading comprises three bytes of data in 32 bit floating point format and must be interpreted as such by the receiving system. If the command is correctly interpreted, the Tan Delta unit addressed will acknowledge the command with the Ack code, echo the number of bytes it will transmit, and send a response containing the data requested, followed by a checksum. If the command is not correctly interpreted, the Tan Delta unit addressed will acknowledge with the Error code and a checksum. If the unit addressed cannot be found, there will be no reply.

The command string is 10 bytes long (no. of bytes in each field as subscript):

"!",<09$_1$>,<iaddr$_1$>,"R","r",<start address$_2$>,<length*$_1$>,<cksm$_2$>
(*length >= 0x0C)

The response string comprises the acknowledge character, the <count> of bytes to follow and then <length> bytes of data, followed by a 16 bit checksum. In its most common usage as used to download all three channel readings, this will comprise 12 bytes of data, as follows:

Bytes 0-3	32 bit floating point representation of Oil Temp value, in C (unless otherwise scaled)
Bytes 4-7	32 bit floating point representation of Ambient Temp value, in C (unless otherwise scaled)
Bytes 8-11	32 bit floating point representation of Oil Condition value, in %

Thus the response string is <length> + 4 bytes long (no. of bytes in each field as subscript):

"A",<length+2$_1$>,<length bytes of data>,<cksm$_2$>

Or, in case of error:**"E",<02$_1$>,< FFB8$_2$>**

Read Memory Command "Rm"

This command has two items of data, a two byte starting <address> and a single byte <length>, allowing a read of up to 256 addresses from system memory. It commands the Tan Delta unit addressed by <iaddr> to transmit <length> memory bytes starting at <address>. If the command is correctly interpreted, the Tan Delta unit addressed will acknowledge the command with the Ack code, echo the number of bytes it will transmit, and send a response containing all the data requested, followed by a checksum. If the command is not correctly interpreted, or the starting <address> is outside the range of system memory, the Tan Delta unit addressed will acknowledge with the Error code and a checksum. If the unit addressed cannot be found, there will be no reply.

The command string is 10 bytes long (no. of bytes in each field as subscript):

"!",<09$_1$>,<iaddr$_1$>,"R","m",<record no$_2$>,<length$_1$>,<cksm$_2$>

The response string comprises the acknowledge character, the <count> of bytes to follow and then <length> bytes, followed by a 16 bit checksum.

Thus the response string is (4 + <length>) bytes long (no. of bytes in each field as subscript):

"A",<length+2$_1$>,<byte 1>,<byte 2>,...,<byte n*>,<cksm$_2$>
(*n = length)

or, in case of error: **"E",<02$_1$>,< FFB8$_2$>**

Read Config Data Command "Rc"

This command has two items of data, a two byte <starting address> and a single byte <length>. It commands the Tan Delta unit addressed by <iaddr> to transmit <length> bytes from <starting address> relative to the start of the channel settings and alarm array for each of the three channels within the unit, and the two relays and eight alarm definitions. Config settings are as defined below and must be correctly interpreted by the receiving system. If the command is correctly interpreted, the Tan Delta unit addressed will acknowledge the command with the Ack code, echo the number of bytes it will transmit, and send a response containing the data requested, followed by a checksum. If the command is not correctly interpreted, the Tan Delta unit addressed will acknowledge with the Error code and a checksum. If the unit addressed cannot be found, there will be no reply.

The command string is 10 bytes long (no. of bytes in each field as subscript):

"!",<09$_1$>,<iaddr$_1$>,"R","c",<start address$_2$>,<length$_1$>,<cksm$_2$>

The response string comprises the acknowledge character, the <count> of bytes to follow and then <length> bytes of data, followed by a 16 bit checksum. The data returned will be dependent on the particular parameters specified by the start address, as detailed in Appendix A below.

Thus the response string is <length> + 4 bytes long (no. of bytes in each field as subscript):-

"A",<length+2$_1$>,<length bytes of data>,<cksm$_2$>

or, in case of error:- **"E",<02$_1$>,< FFB8$_2$>**

Read Version and Serial Number Command "Rv"

This command has two items of data, a two byte <starting address> and a single byte <length>. It commands the Tan Delta unit addressed by <iaddr> to transmit <length> bytes from <starting address> relative to the software Version No. variable. Version and Serial No. information is as defined below and must be correctly interpreted by the receiving system. If the command is correctly interpreted, the Tan Delta unit addressed will acknowledge the command with the Ack code, echo the number of bytes it will transmit, and send a response containing the data requested, followed by a checksum. If the command is not correctly interpreted, the Tan Delta unit addressed will acknowledge with the Error code and a checksum. If the unit addressed cannot be found, there will be no reply.

The command string is 10 bytes long (no. of bytes in each field as subscript):

"!",<09$_1$>,<iaddr$_1$>,"R","v",<start address$_2$>,<length$_1$>,<cksm$_2$>

The response string comprises the acknowledge character, the <count> of bytes to follow and then <length> bytes of data, followed by a 16 bit checksum. The start address should be 0000 to return:-

Software Version No: 4 byte floating point version no.

Thus the response string is <length> + 4 bytes long (no. of bytes in each field as subscript):-

"A",<length+2$_1$>,<length bytes of data>,<cksm$_2$>

or, in case of error: **"E",<02$_1$>,< FFB8$_2$>**

Appendix A – Parameter Addresses

Parameter	Start Addr	Access
Cal Zero	0x0000	R/W
Instrument Address	0x002C	R/W
Serial Type	0x002D	R/W
Max Temp	0x002E	RO
Serial No.	0x004E	RO
Oil Data String	0x0056	R/W
Hardware Version No.	0x007B	RO
CAN Transmission Type	0x00A4	R/W
CAN Event Timer	0x00A5	R/W
CAN TPDO settings	0x00A7	R/W
Filter TC	0x00EF	R/W (V2.10+)
Baud Rate	0x00F0	R/W (V2.10+)
CAN Bit Rate	0x00F1	R/W (V2.10+)

CANopen

The OQS-CAN Oil Quality Sensor measures the Oil Condition, Oil Temperature and Ambient (Sensor) Temperature. The range is from a nominal -20% to +60% Oil Condition units and -30 to +130C. The measured value is transmitted on the CAN-bus using the CANOpen protocol based on the CAN in Automation standard profile CiA DS 404 V1.2. The Sensor samples at 100 samples per second, filters and converts the raw signal to a conditioned output signal.

The CAN interface uses a default bit rate of 125 kb/s with 11 bit identifiers.

The CAN protocol complies with the CANOpen specification DS301 and the Oil Quality Sensor conforms to CANOpen device profile DS404. Node Guarding and Emergency messages are implemented to ensure high reliability.

CAN Interface

The Sensor uses a full CAN controller specified to conform with CAN 2.0B. The physical layer of the 2 wire interface is specified according to ISO 11898. The wires are protected against short circuit and noise emission is minimized. No bus termination resistors are included within the sensor.

OQS-CAN Specification

Supply voltage:	+9 to +30 Vd.c.
Current consumption:	50mA max. when quiescent, 100mA max. with CAN active 30-40mA typical
CAN physical layer:	2 wire interface @ 5V d.c. voltage levels a/c to ISO 11898 Short circuit protected
CAN bitrate:	125kbit/s
Bus termination:	External
Protocol:	CANOpen DS301, Device Profile DS404
Environment:	noise emission according to EN 50 081-2 Noise immunity according to EN 50 082-2
Operating temperature:	-20 to +120C
Storage temperature:	-40 to +150C

Connection Details

The sensor uses a Lumberg 6 pin 030 series male connector with the following pin assignments: -

P1: Analog 4-20mA Oil Condition output (active, current sourcing)
P2: Analog 4-20mA Oil Temp output (active, current sourcing)
P3: +9 to +30V d.c. power supply
P4: 0V
P5: CANL/RS485A
P6: CANH/RS485B

CANOpen Communication

Summary of the CANOpen functions

CANOpen type: NMT slave
Network bootup: Minimum bootup
COB ID: pre-defined connection set, SDO
Node ID: object (specific entry – read/write, **default 1**)
Bitrate: object (specific entry – read only, **fixed 125kbit/s**)
Number of PDOs: PDO1 synchronous or asynchronous configurable
Emergency message: supported
Node Guarding: supported
Device Profile: DS404

Object Dictionary Communication Profile

Index (HEX)	Sub Index	Name	Type	Access	Default	Comment
1000	00	Device Type	U32	RO	0x000E0194	DSP404 analog output, input & digital output
1001	00	Error Register	U8	RO	0x00	

Error Register definition (index 0x1001); 0 = no error, 1 = error:

B0: Global Error
B1: unused
B2: unused
B3: Temperature Error
B4: CAN Communication Error
B5: Oil Quality Error
B6: unused
B7: unused

Index (HEX)	Sub Index	Name	Type	Access	Default	Comment
1005	00	COB-ID SYNC	U32	RO	0x80	
1008	00	Manufacturer Device Name	VIS STR	RO	“Oil Quality Sensor”	Sales Code
1009	00	Manufacturer Hardware Version	VIS STR	RO	“V19”	Build Version
100A	00	Manufacturer Software Version	VIS STR	RO	“3.101”	Software Version
100C	00	Guard Time	U16	RO	20000	
100D	00	Life Factor	U16	RO	1	
1018		Identity Object				
	00	Number of entries	U8	RO	0x4	
	01	Vendor ID	U32	RO	0x32F	Vendor ID
	02	Product Code	U32	RO	111021	Sales Code
	03	Revision No.	U32	RO	0900	
	04	Serial No.	U32	RO		S/No.

Index (HEX)	Sub Index	Name	Type	Access	Default	Comment
1800		Transmit PDO parameter				
	00	Number of entries	U8	RO	0x5	
	01	COB-ID used by PDO	U32	RO	0x180	0x180 + Node-ID
	02	Transmission Type	U8	RW	0x1	0x01 = every SYNC, 0x02 to 0xF0 = every 2nd to 240th SYNC, 0xFF = ASYNC according to Event Timer
	03	Inhibit Time	U16	RO	0x0	
	04	Reserved	U8	RO	0x0	
	05	Event Timer	U16	RW	0x3E8	Interval in ms, 1s default
1A00		Transmit PDO1 mapping				
	00	Number of entries	U8	RO	0x02	
	01	PDO mapping for the 1st application object to be mapped	U32	RW	0x91300320	Oil Condition as I32: 0x91300320 Oil Condition as F32 0x61300320
	02	PDO mapping for the 2nd application object to be mapped	U32	RW	0x91300120	Oil Temperature as I32: 0x91300120 Oil Temperature as F32: 0x61300120

Index (HEX)	Sub Index	Name	Type	Access	Default	Comment
1F80	00	NMT startup	U32	RW	0x0x	0x0x = Follow standard 0x1x = Self-starting Typically, a CANopen slave will launch into pre-op mode and a network master must setup the device. With the self-starting bit enabled the sensor will launch into operational mode when ready during bootup, this allows the sensor to communicate on a simplified CANopen network that may not have a master present. Note: The low nibble of this byte is used for serial type. See 0x4000:00

Index (HEX)	Sub Index	Name	Type	Access	Default	Comment
4000	00	Serial Type	U8	RW	0x01	**0x01** = TanDelta via RS485 **0x02** = CANopoen via CANbus 0x03 = Modbus RTU via RS485 0x05 = J2939 via CANbus Note: The high nibble of this byte is used for flags which influence the serial communications. See 0x1F80:00
4001	00	Sensor Address	U8	RW	0x01	Node ID for CAN, Sensor Address for other serial interfaces
4002	00	RS485 Baudrate	U8	RO	0x00	9600 Baud ONLY

4003	00	CANbus Bitrate	U8	RW	0x05	0 = 1 Mbps 1 = 800 kbps 2-3 = 500 kbps 4 = 250 kbps 5 = 125 kbps (default) 6 = 50 kbps 7 = 20 kbps
4004	00	Sensor ID String	VIS STR	RO	0x05	Writable externally to CAN
	01	Sensor ID String bytes 0 - 6	VIS STR	RO	"Sensor ID string"	First 7 bytes of ID string
	02	Sensor ID String bytes 7 - 13	VIS STR	RO	"Sensor ID string"	Next 7 bytes of ID string
	03	Etc.	Etc.	Etc.	Etc.	Etc.
	04	Etc.	Etc.	Etc.	Etc.	Etc.

Object Dictionary Device Profile – Analogue Input Function Block

Index (HEX)	Sub Index	Name	Type	Access	Default	Comment
6110		AI Sensor Type				
	00	Number of entries	U8	RO	0x3	
	01	AI Sensor Type 1	U16	RO	0x64	100 = Temperature sensor
	02	AI Sensor Type 2	U16	RO	0x64	100 = Temperature sensor
	03	AI Sensor Type 3	U16	RO	0x2710	10000 = Oil condition sensor (Manufacturer defined)
6124		AI Input Offset				
	00	Number of entries	U8	RO	0x1	
	01	AI Input Offset 3	F32	RW		Oil Condition Cal. Zero voltage
6125		AI Autozero				
	00	Number of entries	U8	RO	0x01	
	01	AI Input Autozero 3	U32	WO		Autozero Oil Condition
6126		AI Scaling Factor				
	00	Number of entries	U8	RO	0x01	
	01	AI Input Scaling 3	F32	RO		Oil Condition Gain %
6127		AI Scaling Offset				
	00	Number of entries	U8	RO	0x05	
	01	AI Scaling Offset 1	F32	RW		Oil Normalisation Param 1
	02	AI Scaling Offset 2	F32	RW		Oil Normalisation Param 2
	03	AI Scaling Offset 3	F32	RW		Oil Normalisation Param 3
	04	AI Scaling Offset 4	F32	RW		Oil Normalisation Param 4
	05	AI Scaling Offset 5	F32	RW		Oil Normalisation Param 5
6130		AI Input PV				
	00	Number of entries	U8	RO	0x03	
	01	AI Input PV1	F32	RO		Oil Temp. Value
	02	AI Input PV2	F32	RO		Sensor Temp. Value
	03	AI Input PV3	F32	RO		Oil Condition Value/Alarm State
	04	AI Input PV4	F32	RO		Oil Temp. Value in F
	05	AI Input PV5	F32	RO		Sensor Temp. Value in F
	06	AI Input PV6	F32	RO		**RESERVED**

	07	AI Input PV7	F32	RO		Encrypted RUL
6132		AI Decimal Digits PV				0: integer scaled as is 1: integer scaled *10 2: integer scaled *100 etc.
	00	Number of entries	U8	RO	0x03	
	01	AI Dec. Digits PV1	U8	RW	0x02	Oil Temp. Integer Scaling
	02	AI Dec. Digits PV2	U8	RW	0x02	Sensor Temp. Integer Scaling
	03	AI Dec. Digits PV3	U8	RW	0x02	Oil Condition Integer Scaling
6135		AI Interrupt Upper Limit I/P PV				
	00	Number of entries	U8	RO	0x03	
	08	AI Interrupt Upper Limit IP PV8	F32	RW	0d30	End of Life Value (LF units)
6148		AI Span Start				
	00	Number of entries	U8	RO	0x03	
	01	AI Span Start 1	F32	RO		Oil Temp. Min Range
	01	AI Span Start 2	F32	RO		Sensor Temp. Min Range
	01	AI Span Start 3	F32	RO		Oil Cond. Min Range
6149		AI Span End				
	00	Number of entries	U8	RO	0x03	
	01	AI Span End 1	F32	RO		Oil Temp. Max Range
	01	AI Span End 2	F32	RO		Sensor Temp. Max Range
	01	AI Span End 3	F32	RO		Oil Cond. Max Range
61A0		AI Filter Type				0: unfiltered 1: exponential average
	00	Number of entries	U8	RO	0x01	
	01	AI Filter Type 1	U8	RO	0x01	Exponential average
61A1		AI Filter Constant				
	00	Number of entries	U8	RO	0x01	
	01	AI Filter Const. 1	U8	RO	0x07	1/(2^<Filter Const)* new sample
6F20						
	00	Number of entries	U8	RO	0x01	
	01	AI Oil Data String	STR	RW		Oil Data String
9130		AI Input PV				
	00	Number of entries	U8	RO	0x03	
	01	AI Input PV1	I32	RO		Oil Temp. Value
	02	AI Input PV2	I32	RO		Sensor Temp. Value
	03	AI Input PV3	I32	RO		Oil Condition Value/Alarm State
	04	AI Input PV4	I32	RO		Oil Temp. Value in F
	05	AI Input PV5	I32	RO		Sensor Temp. Value in F
	06	AI Input PV6	I32	RO		**RESERVED**
	07	AI input PV7	I32	RO		Encrypted RUL
9148		AI Span Start				
	00	Number of entries	U8	RO	0x03	
	01	AI Span Start 1	I32	RO		Oil Temp. Min Range
	01	AI Span Start 2	I32	RO		Sensor Temp. Min Range
	01	AI Span Start 3	I32	RO		Oil Cond. Min Range
9149		AI Span End				
	00	Number of entries	U8	RO	0x03	
	01	AI Span End 1	I32	RO		Oil Temp. Max Range
	01	AI Span End 2	I32	RO		Sensor Temp. Max Range
	01	AI Span End 3	I32	RO		Oil Cond. Max Range

CAN Communication without CANOpen Functionality

Basic Configuration – examples of common settings

The OQS-CAN Sensor can be used successfully in CAN networks without full CANOpen functionality. Before using the Sensor within the network the following should be noted and configured where necessary. Note that you will need to know the current Node ID to communicate with the sensor; if you are unsure of this

value and it has been changed from the default, you can identify it from the boot-up message issued by the sensor on startup. This will be

	ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
Message	0x700+ NodeID	1	0x00	NA	NA	NA	NA	NA	NA	NA

Thus a startup message from a COB-ID of 0x71C indicates that the sensor has a Node ID of 0x1C, or 28 decimal.

Bitrate: Object 0x4003, subindex 0. This is, by default, 125kbits/s (CAN bitrate 5) but can be changed – see Section 5.2 above for details.

	ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
Command	0x600+ NodeID	8	0x40	0x03	0x40	0x00	NA	NA	NA	NA
Reply	0x580+ NodeID	8	0x40	0x03	0x40	0x00	0x05	0x00	0x00	0x00

To change the bitrate to <Bitrate> use the following command. Note that the number to be entered in this field is the bitrate code, from 0 to 7, not the actual bits/s. Note that you must restart the sensor to make this change active.

	ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
Command	0x600+ NodeID	0x08	0x2F	0x03	0x40	0x00	<bitrate>	NA	NA	NA
Reply	0x580+ NodeID	0x08	0x60	0x03	0x40	0x00	NA	NA	NA	NA

Node ID: Object 0x4001, subindex 0. This is 0x01 by default and can be changed. Valid values are between 1 and 127 (0x7f). To read use the following command

	ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
Command	0x600+ NodeID	0x08	0x40	0x01	0x40	0x00	NA	NA	NA	NA
Reply	0x580+ NodeID	0x08	0x40	0x01	0x40	0x00	<NodeID>	0x00	0x00	0x00

To change the Node ID to New ID use the following command

	ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
Command	0x600+ NodeID	0x08	0x2F	0x01	0x40	0x00	<NewID>	NA	NA	NA
Reply	0x580+ NodeID	0x08	0x60	0x01	0x40	0x00	NA	NA	NA	NA

The sensor will need restarting (powering off and then on again) before the change becomes effective and the new ID is valid. Any changes made are saved automatically in non-volatile memory.

Transmission Type: Object 0x1800, subindex 2. This is 0x00 (send PDO response every SYNC command) by default and can be changed. Values from 0x00 to 0xF0 represent synchronous response every <TType>+1 SYNC commands and 0xFF represents an asynchronous (timed) response every <Event Timer> ms (default 1000ms). See below to change the Event Timer value. To read use the following command

	ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
Command	0x600+ NodeID	0x08	0x40	0x00	0x18	0x02	NA	NA	NA	NA
Reply	0x580+ NodeID	0x08	0x4F	0x00	0x18	0x02	<TType>	NA	NA	NA

To change the Transmission Type to New Type use the following command

	ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
Command	0x600+ NodeID	0x08	0x2F	0x00	0x18	0x02	<NewType>	NA	NA	NA
Reply	0x580+ NodeID	0x08	0x60	0x00	0x18	0x02	NA	NA	NA	NA

Event Timer: Object 0x1800, subindex 5. This is 0x3E8 (1000 ms) by default and can be changed. Values from 0x0064 to 0xFFFF (decimal 100 to 65535) will generate a Timer Event every <Timer> ms, which can be used to generate a timed PDO response – see above to select Timed PDO responses. To read use the following command

	ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
Command	0x600+ NodeID	0x08	0x40	0x00	0x18	0x05	NA	NA	NA	NA
Reply	0x580+ NodeID	0x08	0x40	0x00	0x18	0x05	<Timer LSB>	<Timer MSB>	NA	NA

To change the Event Timer value to <NewTime> use the following command

	ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
Command	0x600+ NodeID	0x08	0x2E	0x00	0x18	0x05	<NewTime LSB>	<NewTime MSB>	NA	NA
Reply	0x580+ NodeID	0x08	0x60	0x00	0x18	0x05	NA	NA	NA	NA

PDO Data Selection and Format: Object 0x1A00, subindex 1 and 2. These values are 0x61300320 (Oil Condition, F32 format) and 0x61300120 (Oil Temperature, F32 format) by default and can be changed. New variables, in different formats can be selected from the Object Dictionary as detailed above. To read use the following command.

	ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
Command	0x600+ NodeID	0x08	0x40	0x00	0x1A	0x01(2)	NA	NA	NA	NA
Reply	0x580+ NodeID	0x08	0x40	0x00	0x1A	0x01(2)	<Variable LSB>	<Variable byte1>	<Variable byte2>	<Variable MSB>

The Variable bytes for the default configuration would be 20, 03, 30, 61 for the oil condition value in F32 format, and 20, 01, 30, 61 for the oil temperature in F32 format.

To change the PDO Data Selection and Format to <NewPDO> use the following command

	ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
Command	0x600+ NodeID	0x08	0x23	0x00	0x1A	0x01(2)	<Variable LSB>	<Variable byte1>	<Variable byte2>	<Variable MSB>
Reply	0x580+ NodeID	0x08	0x60	0x00	0x1A	0x01(2)	NA	NA	NA	NA

For example, the Variable bytes needed to set the two values returned by the PDO to Oil Condition in I32 format (0x91300120) and Oil Temperature in I32 format (0x91100120) would entail writing 20, 03, 30, 91 to subindex 01, and 20, 01, 30, 91 to subindex 02 – see above in the Object Dictionary.

Network Operation without CANOpen Master

After connecting the Sensor to the network and applying power, the Sensor will enter the pre-operational state and issue the boot-up message as described above. In normal operation with a CANOpen Master present, the Master will then issue a command to set the Sensor into a fully operational mode and take over control. If the Master is not present this same operation can be done by setting the Sensor to Self-Starting mode. This is done by setting NMT Startup, Object 0x1F80, to 0x12 and then restarting the Sensor, as follows

	ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
Command	0x600+ NodeID	0x08	0x2F	0x80	0x1F	0x00	0x12	NA	NA	NA
Reply	0x580+ NodeID	0x08	0x60	0x80	0x1F	0x00	NA	NA	NA	NA

The Sensor will now be in Self-Starting mode and will automatically enter full Operational mode after every startup. SYNC commands can then be issued by any other CANOpen device to elicit the Sensor's normal PDO response which is to send the Oil Condition and Temperature in 32-bit floating point format. This will be as follows: -

	ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
Command	0x080	0x00	NA	NA	NA	NA	NA	NA	NA	NA
Reply	0x580+ NodeID	0x08	<Tb0>	<Tb1>	<Tb2>	<Tb3>	<Cb0>	<Cb1>	<Cb2>	<Cb3>

Where Tb3 to Tb0 are the most significant to least significant bytes of the 32 bit floating point Oil Temperature value and Cb3 to Cb0 similarly for the Oil Condition. Thus values of 0A,D7,D5,41,7B,14,AE,3F in databytes B0 to B7 equates to hexadecimal values of 41D5D70A and 3FAE147B (byte order rearranged) and floating point values of 26.73C and 1.36%. The Node ID of the sensor returning this data can be inferred from the ID byte as defined in the CAN Open specification for TPDO transmissions. This means that multiple sensors may be connected to the same CAN-bus as long as they have different Node IDs, and if they are all configured in this way, each will return the above response to a single SYNC command

Reading and Writing Parameters and Values

Individual parameters may be read or written according to the RO,WO or RW) or equivalent) setting of that object using the standard SDO format as defined within the CAN Open specification. The following examples show the process of reading a value, and both reading and writing a parameter.

Reading the Ambient (Sensor) Temperature I32 format

The Sensor Temperature may be read as a 32 bit integer by performing an SDO read of the object at index 0x9130, sub-index 02 as specified in the Object Dictionary (see section 5.2, Analogue Input Function Block). The command is as follows:-

	ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
Command	0x600+ NodeID	0x08	0x40	0x30	0x91	0x02	NA	NA	NA	NA
Reply	0x580+ NodeID	0x08	0x43	0x30	0x91	0x02	<Tb0>	<Tb1>	<Tb2>	<Tb3>

The value is returned as a signed integer number in bytes Tb0 to Tb3, where Tb3 is the MSB and Tb0 the LSB. The value is multiplied by the power of ten specified in PV Decimal Digits at index 0x6132, sub-index 02. Thus with a default PV Decimal Digits value of 2, the floating point value is multiplied by 10^2, or 100 so that the last two decimal digits are fractional after an implied decimal point so a decimal value of 3214 would translate to 32.14C.

Reading the Oil Condition Cal Zero Voltage

The Oil Condition Cal Zero Voltage may be read as a 32 bit floating point number by performing an SDO read of the object at index 0x6124, sub-index 01 as specified in the Object Dictionary (see section 5.2, Analogue Input Function Block). The command is as follows: -

	ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
Command	0x600+ NodeID	0x08	0x40	0x24	0x61	0x01	NA	NA	NA	NA
Reply	0x580+ NodeID	0x08	0x43	0x24	0x61	0x01	<Zb0>	<Zb1>	<Zb2>	<Zb3>

The value is returned as a floating point number in bytes Zb0 to Zb3, where Zb3 is the MSB and Zb0 the LSB.

Writing the Oil Condition Cal Zero Voltage

The Oil Condition Cal Zero Voltage may be changed by performing an SDO write of the object at index 0x6124, sub-index 01 as specified in the Object Dictionary (see section 5.2, Analogue Input Function Block). The command is as follows: -

	ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
Command	0x600+ NodeID	0x08	0x2F	0x24	0x61	0x01	<Zb0>	<Zb1>	<Zb2>	<Zb3>
Reply	0x580+ NodeID	0x08	0x60	0x24	0x61	0x01	NA	NA	NA	NA

The value to be written is formatted as above. In this way the calibration of the sensor may be adjusted.

Reading the Oil Data String from the Oil Database

The Oil Condition Cal Data String may be read by performing an SDO read (Domain Upload) from the object at index 0x6F20, sub-index 01 as specified in the Object Dictionary (see section 5.2, Analogue Input Function Block). The command is as follows: -

	ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
Command	0x600+ NodeID	0x08	0x40	0x20	0x6F	0x01	0x00	0x00	0x00	0x00
Reply	0x580+ NodeID	0x08	0x41	0x20	0x6F	0x01	0x25	NA	NA	NA
Command	0x600+ NodeID	0x08	0x60	0x20	0x6F	0x01	0x00	0x00	0x00	0x00
Reply	0x580+ NodeID	0x08	0x60	0x31	0x43	0x5E	0xB8	0xDB	0x00	0x43
Command	0x600+ NodeID	0x08	0x70	0x20	0x6F	0x01	0x00	0x00	0x00	0x00
Reply	0x580+ NodeID	0x08	0x70	0x17	0xA4	0x35	0x7B	0x00	0x35	0x43
Command	0x600+ NodeID	0x08	0x60	0x20	0x6F	0x01	0x00	0x00	0x00	0x00
Reply	0x580+ NodeID	0x08	060	0x00	0x00	0x50	0xA0	0x8A	0x1F	0x87
Command	0x600+ NodeID	0x08	0x70	0x20	0x6F	0x01	0x00	0x00	0x00	0x00
Reply	0x580+ NodeID	0x08	0x70	0xFA	0x0A	0xBA	0xAD	0x81	0x00	0xF1
Command	0x600+ NodeID	0x08	0x60	0x20	0x6F	0x01	0x00	0x00	0x00	0x00
Reply	0x580+ NodeID	0x08	0x60	0xD1	0x17	0x00	0x3E	0xB7	0xAA	0xA8
Command	0x600+ NodeID	0x08	0x70	0x20	0x6F	0x01	0x00	0x00	0x00	0x00
Reply	0x580+ NodeID	0x08	0x70	0x00	0x3E	NA	NA	NA	NA	NA

The Oil String Data comprises bytes B1 to B7 successively for each Reply message, plus B1 and B2 of the last message, in that order. The example shown above corresponds to Generic Mineral 15W40, which has string 31435EB8DB004317A4357B003543000050A08A1F87FA0ABAAD8100F1D117003EB7AAA8003E.

Writing the Oil Data String to the Oil Database

The Oil Condition Cal Data String may be changed by performing an SDO write (Domain Download) to the object at index 0x6F20, sub-index 01 as specified in the Object Dictionary (see section 5.2, Analogue Input Function Block). The command is as follows: -

	ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
Command	0x600+ NodeID	0x08	0x20	0x20	0x6F	0x01	0x00	0x00	0x00	0x00
Reply	0x580+ NodeID	0x08	0x60	0x20	0x6F	0x01	0x25	NA	NA	NA
Command	0x600+ NodeID	0x08	0x00	0x03	0x66	0xEE	0xEF	0x6E	0xC4	0x41
Reply	0x580+ NodeID	0x08	0x20	0x20	0x6F	0x01	0x25	NA	NA	NA
Command	0x600+ NodeID	0x08	0x10	0xE6	0xBD	0x08	0x1C	0xB6	0x19	0x00
Reply	0x580+ NodeID	0x08	0x30	0x20	0x6F	0x01	0x25	NA	NA	NA
Command	0x600+ NodeID	0x08	0x00	0x6F	0x77	0x5A	0x3F	0x66	0x3A	0xF0
Reply	0x580+ NodeID	0x08	0x20	0x20	0x6F	0x01	0x25	NA	NA	NA
Command	0x600+ NodeID	0x08	0x10	0x03	0x66	0x06	0x3F	0x12	0xA7	0x49
Reply	0x580+ NodeID	0x08	0x30	0x20	0x6F	0x01	0x25	NA	NA	NA
Command	0x600+ NodeID	0x08	0x00	0xA3	0x03	0x02	0x1B	0x6F	0x12	0x66
Reply	0x580+ NodeID	0x08	0x20	0x20	0x6F	0x01	0x25	NA	NA	NA
Command	0x600+ NodeID	0x08	0x1B	0x3F	0xBF	NA	NA	NA	NA	NA
Reply	0x580+ NodeID	0x08	0x30	0x20	0x6F	0x01	0x25	NA	NA	NA

The value to be written is formatted as above. In this way the calibration of the sensor may be adjusted. The Oil String Data comprises bytes B1 to B7 successively for each Reply message, plus B1 and B2 of the last message, in that order. The example shown above has string 0366EEEF6EC441E6BD081CB619006F775A3F663AF00366063F12A749A303021B6F12663FBF which corresponds to Avia Bantleon Synto.

Reading the Hardware Version No. from the sensor

The Hardware Version No. may be read as a 3 byte string by performing an SDO read (Expedited Upload) of the object at index 0x1009, sub-index 00 as specified in the Object Dictionary (see section 5.2, Communication Profile). The command is as follows: -

	ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
Command	0x600+ NodeID	0x08	0x40	0x09	0x10	0x00	NA	NA	NA	NA
Reply	0x580+ NodeID	0x08	0x43	0x09	0x10	0x00	<HVb0>	<HVb1>	<HVb2>	NA

The value is returned as a string value in bytes HVb0 to HVb2, where HVb0 is the first character in the string and HVb2 the last character.

Reading the Software Version No. from the sensor

The Software Version No. may be read as a 5 byte string by performing an SDO read (Domain Upload) of the object at index 0x100A, sub-index 00 as specified in the Object Dictionary (see section 5.2, Communication Profile). The command is as follows: -

	ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
Command	0x600+ NodeID	0x08	0x40	0x0A	0x10	0x00	NA	NA	NA	NA
Reply	0x580+ NodeID	0x08	0x41	0x0A	0x10	0x00	0x05	NA	NA	NA
Command	0x600+ NodeID	0x08	0x60	0x0a	0x10	0x00	0x00	0x00	0x00	0x00
Reply	0x580+ NodeID	0x08	0x05	<SVb0>	<SVb1>	<SVb2>	<SVb3>	<SVb4>	NA	NA

The value is returned as a string value in bytes SVb0 to SVb4, where SVb0 is the first character in the string and SVb4 the last character.

Reading the Serial No. from the sensor

The Serial No. may be read as a 32-bit unsigned integer by performing an SDO read (Expedited Transfer) of the object at index 0x1018, sub-index 04 as specified in the Object Dictionary (see section 5.2, Communication Profile). The command is as follows: -

	ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
Command	0x600+ NodeID	0x08	0x40	0x18	0x10	0x04	NA	NA	NA	NA
Reply	0x580+ NodeID	0x08	0x43	0x18	0x10	0x04	<SNb3>	<SNb2>	<SNb1>	<SNb0>

The value is returned as a 32 bit unsigned integer value in bytes SNb3 to SNb0, where SNb3 is the MSB of the value and SNb0 the last character.

Modbus

To select Modbus as the preferred communication method, please use the PC/Laptop Software and refer to the appropriate manual.

Hardware

The Tan Delta serial communication uses an RS485 multi-drop interface.

Configurable Parameters

The serial configuration must be set to 8 bits with no parity and will communicate at 9600 baud. The OQS operates in a slave mode with a default ID of 1 which can be preset to suit on register 11 (0x0B), with the serial device which controls the communications (e.g. the monitoring computer) acting as master and issuing commands to which the slave will reply. OQS will not transmit any data except in response to a command from the master, and will expect no further commands until the last command has been replied to.

Communication Mode

OQS supports communication using the OQS proprietary protocol which uses RS485 in ASCII mode (not covered by this document), and Modbus RTU protocol which uses RS485 in Hex mode. The suitable method of communication can be set by presetting register 12 (0x0C) to:

TDS RS485 1

ModBus RTU 2

Message Framing

Message start with a silent interval of at least 3.5 character times then the first field transmitted is assumed to be the device address.

Following the last transmitted character, a similar interval of at least 3.5 character times marks the end of the message. A new message can begin after this interval.

The entire message frame must be transmitted as a continuous stream. If a silent interval of more than 1.5 character times occurs before completion of the frame, the receiving device flushes the incomplete message and assumes that the next byte will be the address field of a new message.

Similarly, if a message starts earlier than 3.5 character times following a previous message, the receiving device will consider it a continuation of the previous message. This will set an error, as the value in the final CRC field will not be valid for the combined messages.

A typical message format is shown below.

Start	3.5 Characters
Address	8 Bits
Function	8 Bits
Data	16 Bits
CRC	16 Bits
End	3.5 Characters Time

Function Codes

04 (0x04) Read Input Registers

Data format

Tan Delta returns all real (non-integer) values as 16 bit signed integers with the value multiplied by 100 (decimal).

For example, a temperature reading of 34.14 degrees C would be returned as 3414 decimal (0D56 hex).

Negative values follow the usual signed format, so -12.34 would be returned as -1234 decimal (Fb2E hex).

Query

This reads the contents of input registers in the slave. The query message specifies the starting register and number of registers to be read. Information needed to be read from the probe is kept in registers:

Parameters	Registers Decimal	Registers Hex	Access
Oil Temperature *100	0	00	RO
Ambient Temperature * 100	1	01	RO
Oil Condition * 100	2	02	RO
Cal Zero * 100	3	03	RO
Oil Temperature deg F *100	4	04	RO
Ambient Temperature deg F * 100	5	05	RO
Oil Condition TDN * 100	6	06	RO
Alarm State	7	07	RO
Encrypted RUL	8	08	RO
End of Life Value	10	0A	R/W
Instrument/Node Address	11	0B	R/W
Serial Type (Modbus/CANbus/J1939)	12	0C	RO
Maximum Ambient Temp * 100	13	0D	RO
Serial No.	14	0E	RO
Hardware Version Number	15	0F	RO
Software Version Number * 100	16	10	RO
Serial No. High Digit	17	11	RO
Oil Serial No. 0 (Low)	18	12	RO
Oil Serial No. 1	19	13	RO
Oil Serial No. 2	20	14	RO
Oil Serial No. 3 (High)	21	15	RO
Oil Serial No. 4 (RESERVED)	22	16	RO
Oil Serial No. 5 (RESERVED)	23	17	RO
Oil Serial No. 6 (RESERVED)	24	18	RO
Oil Data String characters 1 & 2	32	20	
Oil Data String characters 3 & 4	33	21	
Oil Data String characters 5 & 6	34	22	
Oil Data String characters 7 & 8	35	23	
Oil Data String characters 9 & 10	36	24	
Oil Data String characters 11 & 12	37	25	
Oil Data String characters 13 & 14	38	26	
Oil Data String characters 15 & 16	39	27	

Oil Data String characters 17 & 18	40	28
Oil Data String characters 19 & 20	41	29
Oil Data String characters 21 & 22	42	2A
Oil Data String characters 23 & 24	43	2B
Oil Data String characters 25 & 26	44	2C
Oil Data String characters 27 & 28	45	2D
Oil Data String characters 29 & 30	46	2E
Oil Data String characters 31 & 32	47	2F
Oil Data String characters 33 & 34	48	30
Oil Data String characters 35 & 36	49	31
Oil Data String characters 37 (MSB)	50	32

Here is an example of a request to read a single register starting at register 1 (ambient temperature) of device slave 1.

Field Name	Example (Hex)
Slave Address	01
Function	04
Starting Address Hi	00
Starting Address Lo	00
No. of Registers Hi	00
No. of Registers Lo	01
CRC Check	600A

Response

The register data in the response message are packed as two bytes per register, with the contents right justified within each byte. For each register, the first byte contains the high order bits and the second contains the low order bits.

Here is an example of a response to above query to read a single register, starting at register 0, ambient temperature.

Field Name	Example (Hex)
Address	01
Function	04
Byte Count	02
Data Hi	4E
Data Lo	5A
CRC	0CAB

06 (0x06) Write Single Register

Query

The Write command specifies the register reference to be preset and allows the remote configuration of channel and system settings for management of the unit's operation, and modification for use in monitoring. The registers which hold the configuration parameters are:

Parameters	Registers Decimal	Registers Hex
Instrument Address	11	0B
Serial Type	12	0C

Here is an example of a request to preset register 11 (instrument address) of device slave 1 to change it to devices slave 4. Note that changing the instrument address or serial type will only take effect after the sensor is shut down and restarted:

Field Name	Example (Hex)
Slave Address	01
Function	06
Starting Address Hi	00
Starting Address Lo	0B
No. of Registers Hi	00
No. of Registers Lo	04
CRC Check	F9CB

Response

The normal response is an echo of the query, returned after the register contents have been preset.

Here is an example of a response to above query to preset register 11, instrument address to 4.

Field Name	Example (Hex)
Slave Address	01
Function	06
Starting Address Hi	00
Starting Address Lo	0B
No. of Registers Hi	00
No. of Registers Lo	04
CRC Check	F9CB

17 (0x11) Report Slave ID

Returns information about the slave device. Currently returns only the Version No. of the probe firmware.

Troubleshooting

If you are experiencing a communication issue with a Tan Delta sensors following a ModBus installation, please carefully review the following points;

1) Does whatever monitoring system you are using (Monitoring System) use an RS485, 2 wire, half duplex serial interface?

2) Does the Monitoring System use Modbus RTU, not ASCII?

3) Is the Monitoring System set to use the correct COM port (that is connected to the sensor)?

4) Is the Monitoring System set to use 9600 baud, 8 bits, 1 stop bit with no parity?

5) Is the Monitoring System trying to communicate with the correct Node ID for the sensor (Sensor Node ID 0 will be forced to Node ID 1)?

6) Has the sensor been set to Modbus using Tan Delta Configuration & Data Software, or register 12 (zero based) to 0x03 or 0x13 (decimal 3 or 19) by other means?

7) Has the sensor been powered off and restarted after setting to Modbus?

8) Is the sensor firmware V2.28 or V3.xx? If not, please ask TD for confirmation.

If no response whatsoever is being received….

1) Does the Monitoring System use zero-based register addresses? The sensor registers are listed as zero based (starting from register zero); some Modbus master devices start from register 1 – if this is the case, add 1 onto every register address listed in the manual. Check that the sensor Serial Type has been set correctly, if set using other than Tan Delta Configuration and Data Software, as a typical test, try to read the value back from any (or all) of Registers 0 to 2 (zero based) or 1 to 3 (one based). These are the three live readings.

2) Does your Monitoring System give communications errors? If so, check these to see if they give any further indication of the issue. (eg. time-out errors, CRC errors, overrun errors…)

3) What time-out is set within the Monitoring System? If this is too short the system may be timing out before the sensor can respond. Time-outs of less than 50ms may cause problems; 500ms to 1s are good starting points. If you are configuring the oil type via Modbus then a timeout of 2000ms is recommended.

4) What polling rate is being used by the Monitoring System? If this is too short the system may be asking for the next response before the sensor can respond to the first. Again polling rates of something like 1s are a good starting point.

5) Does the Monitoring System use a standard Modbus 16 bit CRC for error checking? If not the system will not communicate correctly.

6) Does the Monitoring System use any non-standard data formats, e.g. inverted data? If so, the system will not communicate correctly.

7) Is the GND lead of the sensor shared with the GND of the Monitoring System? If either part is driven by a completely different power supply, there may be significantly different voltages between the two and they will not communicate. If they are commoned or either is floating or isolated they should “float” back together. If in doubt, check with a meter and unless there are persistently high

voltages (more than a few volts) between the two, connect the two GND lines together with a third wire, ideally with a fuse in line in case they are driven to different voltages and not just floating apart. CHECK YOUR POWER CONNECTION SCENARIO FIRST TO ENSURE THIS WILL NOT HARM THE SENSOR OR YOUR EQUIPMENT.

8) Try swapping the RS485 A and B lines over. The A line should go to the blue wire (pin 5 on the sensor connector) and the B line should go to green (pin 6) but these lines are sometimes marked the opposite way round on the Monitoring System. (A should be the non-inverted data line and B the inverted data line but the whole RS485 communications system is active-low so inverted and non-inverted sometimes get confused.) This will not cause any damage so is worth trying if all else fails!

9) If this still does not work, please feel free to contact Tan Delta for more support;

Telephone: +44 (0)845 094 8710

Email: support@tandeltasystems.com

If you are receiving replies but they do not make sense….

1) Check that you are reading the correct Register as detailed in 2) at the top of page 20.

2) Check that you are reading in the correct number format. Most value readings are in signed 16-bit integer decimal format and multiplied by 100, so a reading of +1234 means +12.34 and a reading of -5678 means -56.78. If you are reading in unsigned format, any negative values will look wrong. If you are reading in binary, octal or hexadecimal formats the values will look very strange. Other readings are generally in simple integer format so should be straightforward.

3) Check that you are using the correct baud rate and number of bits as detailed in 4) at the top of page 20. If your Monitoring System gives communications errors, check these to see if that gives any clue.

4) If this still does not work, please feel free to contact Tan Delta for more support;

Telephone: +44 (0)845 094 8710

Email: support@tandeltasystems.com

J1939 on CANbus

The OQSx Oil Quality Sensor measures the Oil Condition, Oil Temperature and Ambient (Sensor) Temperature. The range is from a nominal -20% to +60% Oil Condition units and -30 to +130C. The measured value is transmitted on the CAN-bus using the J1939 protocol based on the SAE J1939 standards J1939-DA August 2018, J1939-21 October 2018 and J1939-82 June 2015. The Sensor samples internally at 100 samples per second, filters and converts the raw signal to a conditioned output signal and outputs the Oil Condition and Oil Temperature periodically every 10s.

The CAN interface uses a default bit rate of 250 kb/s with 29 bit identifiers. 500 kb/s is also available.

The CAN protocol complies with the SAE J1939 standard and the Oil Quality Sensor implements 2 PGNs for transmission of process values. PGN 65262 (0xFEEE) is used for Oil Temperature and PGN 65279 (0xFEFF) is used for Oil Condition. The sensor is Single Address Capable (not Arbitrary Address Capable) with Command Configurable Addressing; it does not support Service Configurable or Self-Configurable Addressing.

CAN Interface

The Sensor uses a full CAN controller specified to conform with CAN 2.0B. The physical layer of the 2 wire interface is specified according to ISO 11898. The wires are protected against short circuit and noise emission is minimized. No bus termination resistors are included within the sensor.

OQS-CAN Specification

Supply voltage:	+9 to +30 Vd.c.
Current consumption:	50mA max. when quiescent, 100mA max. with CAN active 30-40mA typical
CAN physical layer:	2 wire interface @ 5V d.c. voltage levels a/c to ISO 11898 Short circuit protected
CAN bitrate:	250kbit/s default
Bus termination:	External
Protocol:	J1939-DA August 2018, J1939-21 October 2018 and J1939-82 June 2015
Environment:	noise emission according to EN 50 081-2 Noise immunity according to EN 50 082-2
Operating temperature:	-20 to +120C
Storage temperature:	-40 to +150C

Connection Details

The sensor uses a Lumberg 6 pin 030 series male connector with the following pin assignments: -

P1: Analog 4-20mA Oil Condition output (active, current sourcing)

P2: Analog 4-20mA Oil Temp output (active, current sourcing)

P3: +9 to +30V d.c. power supply

P4: 0V

P5: CANL/RS485A

P6: CANH/RS485B

J1939 Communication

Summary of the J1939 functions and settings

J1939 type: J1939 Node

Bitrate: Default 250 kb/s, can be configured to 500 kb/s using CADS

Industry Group: 5

Function: 46 (decimal)

Manufacturer Code: 952 (decimal)

Available PGNs: PGN 65262 (0xFEEE) Engine Temperature: here used for Oil Temperature
PGN 65279 (0xFEFF) Operator Indicators: here used for Oil Condition
PGN 65240 (0xFED8) Commanded Address

J1939 Name/Address: The sensor's J1939 Name/Address is constructed as follows:-

TABLE 1: CONTROLLER APPLICATION NAME/ADDRESS FORMAT

Arbitrary Address Capable	Industry Group	Vehicle System Instance	Vehicle System	Reserved	Function	Function Instance	ECU Instance	Manufacturer Code	Identity Number
1 Bit	3 Bits	4 Bits	7 Bits	1 Bit	8 Bits	5 Bits	3 Bits	11 Bits	21 Bits

Arbitrary Address Capable = 0
Industry Group = 5 decimal = 0x05
Vehicle System Instance = 0 for single instance
Vehicle System = 0
Reserved = 0
Function = 46 decimal = 0x2E
Function Instance = 0 for single instance
ECU Instance = 0 for single instance
Manufacturer Code = 952 decimal = 0x3b8
Identity Number = Complete Serial No. of the sensor reduced to 21 bits, e.g for a serial no. of 1003834 decimal = 0xF513A, the Identity Number is (0)F513A where (0) represents 1 single bit.

For all practical purposes, the above means that the J1939 Name/Address will be 0x50002E0077000000 plus the sensor serial number.

Sensor Operation and Messages

On Start up, the sensor will broadcast its J1939 Name/Address as shown below. The J1939 Node Address will be, by default, the sensor's Node ID + 0x80, so a Node ID of 1 will yield a J1939 Node Address of 0x81, or 129 decimal.

State	ID (hex)	DLC	Data (hex)
E	CEEFF81	8	3A 51 0F 77 00 2E 00 50

Data is usually shown as little-endian so the bytes need to be reversed to get the J1939 Name/Address of 50002E00770F513A as discussed above.

The sensor will then begin its automatic asynchronous data transmission, sending both Oil Temperature and Oil Condition PGNs every 10s as below.

E	18FEEE81	8	FF FF 00 2E FF FF FF FF
E	18FEFF81	8	FF FF FF FF FF 01 50 FF

These values may be requested over the J1939 bus as well as using the periodic transmission.

PGN 65262 (0xFEEE) reports Oil Temperature in bytes 3 and 4 according to J1939_DA201808 and PGN 65279 (0xFEFF) reports Oil Condition in byte 6 (Alarm State) and byte 7 (Remaining Useful Life

(Encrypted)). These values are reported as unsigned integers, with an offset of +30 on the temperature so 0x002E = 0d46 reports an oil temperature of 46-30 = 16C. 0x01 reports an Alarm State of 1, and 0x50 reports a Remaining Useful Life of 0x50 which translates to approx. 96% of initial oil life remaining.

Sensor Readings may also be requested on demand using the two PGN messages as shown below. This may be useful if data capture synchronous to a particular event is required. To request Oil Temperature and Oil Condition consecutively, send a message requesting PGN 65262 with 3 bytes of data set to 00, FE, FF, and PGN 65279 with 3 bytes of data set to 00, FE, EE. If these are sent back-to-back the bus traffic would look something like this.

E	18EA8180	3	EE FE 00
E	18EA8180	3	FF FE 00
E	18FEEE81	8	FF FF 00 2E FF FF FF FF
E	18FEFF81	8	FF FF FF FF FF 01 50 FF

Finally the J1939 Node Address may be changed by an external node by sending 3 messages to PGN 65240, using the Commanded Address function. Here three commands are sent:-

1) The first message has Priority 0x1C sent in the first byte, PDU format EC in the second byte, Destination Address of 0xFF (broadcast) and Source Address of 0x80 are used for the remaining two bytes. The first command data is fixed at 0x00, 0xFE, 0xD8, 0xFF 0x02, 0x00, 0x09, 0x20 in little-endian (reversed) order.
2) the second has the same message with command data holding a packet number of 0x01 followed by the upper 7 bytes of the J1939 Name/Address.
3) the third again has the same message with command data holding a packet number of 0x02 followed by the last byte of the J1939 Name/Address, the new J1939 Node Address to which this Node is to be set, here shown as 0x84 or Node 132 in decimal. The rest of the bytes in the third data group are set to 0xFF.

E	1CECFF80	8	20 09 00 02 FF D8 FE 00
E	1CEBFF80	8	01 3A 51 0F 77 00 2E 00
E	1CEBFF80	8	02 50 84 FF FF FF FF FF
E	CEEFF84	8	3A 51 0F 77 00 2E 00 50

This command changes the Node with J1939 Name/Address 50002E00770F513A to Node Address 0x84, as shown in the last line above which displays the node at the new address 0x84 broadcasting the same J1939 Name/Address as was previously at node 0x81.