STAUFF®

Local Solutions For Individual Customers Worldwide

STAUFF Protocol on RS485
CAN Open on CANbus
ModBus RTU on RS485

Manual

CONTENT

1. STAUFF Communication Protocol

The command protocol and language for the STAUFF serial communications uses a binary command format communicating over a half duplex RS232 or RS485 interface. The serial configuration must be set to 8 bits with no parity and will initially communicate at 9600 baud. The STAUFF 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. STAUFF 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 mis-interpreted.
The detailed structure of commands is as detailed below.

The command structure is as follows:-

BYTE 0	"!"	WAKE-UP 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-2>

The response structure is as follows:

BYTE 0	"!"	WAKE-UP CODE
Byte 1	<count>	number of bytes to follow, including <cksm> and <count>
Byte 2 to n-1	<response>	Optional, depending on the command
Byte n, n+1*	<cmd>	16 bit inverse checksum of all preceding bytes including acknowledge

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.

1.1. 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 STAUFF 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

▶ Write commands which write data to the STAUFF 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

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.

1.2. 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 STAUFF 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 24 bit floating point format and must be interpreted as such by the receiving system. If the command is correctly interpreted, the STAUFF 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 STAUFF 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 9 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$_2$>**

1.3. 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 STAUFF unit addressed by <iaddr> to transmit <length> memory bytes starting at <address>. If the command is correctly interpreted, the STAUFF 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 STAUFF 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>,< $FFB8_2$>**

1.4. 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 STAUFF 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 STAUFF 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 STAUFF 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):

"!",<091>,<iaddr1>,"R","c",<start address2>,<length1>,<cksm2>

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>,< $FFB8_2$>**

1.5. 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 STAUFF 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 STAUFF 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 STAUFF 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$_2$>**

1.6. 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+)

2. CanBus

The OCS-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.

2.1. 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.

2.2. OCS-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 +130C
Storage temperature:	-40 to +150C

2.3. 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

2.4. CANOpen Communication

2.4.1. 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

2.4.2. 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	"V12"	Build Version
100A	00	Manufacturer Software Version	VIS STR	RO	"V1.12"	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	0x61300320	Oil Condition as I32: 0x91300320 Oil Condition as F32 0x61300320
	02	PDO mapping for the 2nd application object to be mapped	U32	RW	0x61300120	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: NMT master must start NMT slave. 0x1x: NMT slave will then enter Pre-operational state (initialization) followed by Operational state automatically on successful initialization. Note: x = 0 for RS232, 1 for RS485, 2 for CAN

INDEX (HEX)	SUB INDEX	NAME	TYPE	ACCESS	DEFAULT	COMMENT
4000	00	Serial Type	U8	RW	0x01	0 for RS232, 1 for RS485, 2 for CAN. Note: NMT startup is stored in the same variable; bit 0x10 forces self startup.
4001	00	Sensor Address	U8	RW	0x01	Node ID for CAN, Sensor Address for other serial interfaces
4002	00	Baudrate	U8	RO	0x00	9600 Baud ONLY
4003	00	CAN Bitrate	U8	RW	0x05	0 = 1Mb/s, 1 = 800 kb/s, 2 = 500kbit, 3 = 500kb/s (duplicated), 4 = 250kb/s, 5 = 125kb/s (default), 6 = 50kb/s, 7 = 20kb/s
4004	00	Sensor ID String	VIS STR	RO	"Sensor ID string"	Writable externally to CAN

2.4.3. 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 (mfr 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
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
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

INDEX (HEX)	SUB INDEX	NAME	TYPE	ACCESS	DEFAULT	COMMENT
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
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

2.5. CAN Communication without CANOpen Functionality

2.5.1. Basic Configuration – examples of common settings

The OCS-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 Chapter 2.4.2 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.

2.5.2. 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

2.5.3. 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.

2.5.4. 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 chapter 2.4.3 , 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.

2.5.5. 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 chapter 2.4.3 , 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.

2.5.6. 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 chapter 2.4.3 , 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.

2.5.7. 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 chapter 2.4.3 , 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

	ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
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.

2.5.8. 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 chapter 2.4.3 , 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.

2.5.9. 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 chapter 2.4.2 , 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.

2.5.10. 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 chapter 2.4.2, 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.

2.5.11. 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 chapter 2.4.2, 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.

2.6. Revision History

Rev 2: Synchronous PDO transmission every nth SYNC command implemented
Asynchronous PDO transmission on timer implemented
Readings and formats transmitted via PDO now selectable

Rev 3: Bitrate index corrected in examples

Rev 4: 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 5: Hardware Version No format corrected.
Examples of Hardware Version No., Software Version No. and Serial No. read added.

Rev. 6: Read Sensor ID String added to Object Dictionary.

3. Modbus

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

3.1. Hardware

The STAUFF serial communication uses an RS485 multi-drop interface.

3.2. Configurable Parameters

The serial configuration must be set to 8 bits with no parity and will communicate at 9600 baud. The OCS 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. OCS 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.

3.3. Communication Mode

OCS supports communication using the OCS 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

3.4. 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

3.5. Function Codes

04 (0x04) Read Input Registers

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
Oil Temperature *100	0	00
Ambient Temperature * 100	1	01
Oil Condition * 100	2	02
Cal Zero * 100	3	03
Instrument/Node Address	11	0B
Serial Type (RS485/Modbus/CANbus)	12	0C
Maximum Ambient Temp * 100	13	0D
Serial No.	14	0E
Hardware Version Number	15	0F
Software Version Number * 100	16	10
Oil Data String characters 1 & 2	17	11
Oil Data String characters 3 & 4	18	12
Oil Data String characters 5 & 6	19	13
Oil Data String characters 7 & 8	20	14
Oil Data String characters 9 & 10	21	15
Oil Data String characters 11 & 12	22	16
Oil Data String characters 13 & 14	23	17
Oil Data String characters 15 & 16	24	18
Oil Data String characters 17 & 18	25	19
Oil Data String characters 19 & 20	26	1A
Oil Data String characters 21 & 22	27	1B
Oil Data String characters 23 & 24	28	1C
Oil Data String characters 25 & 26	29	1D
Oil Data String characters 27 & 28	30	1E
Oil Data String characters 29 & 30	31	1F
Oil Data String characters 31 & 32	32	20
Oil Data String characters 33 & 34	33	21
Oil Data String characters 35 & 36	34	22
Oil Data String characters 37 (MSB)	35	23

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

3.5.1. 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
Zero Calibration	3	03
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.

3.6. Calibrating the Probe

In order to obtain the best accuracy of oil reading you should zero the sensor on a clean sample of oil before use. Ideally this should be done while the sensor is fully connected to your measuring circuit. Follow the below steps:

If you know that the oil in the application for which you are going to use the sensor is new, clean oil, this can be done while the probe is fitted to its target application, otherwise the sensor should be immersed in a clean container of new oil and tilted to expel any air bubbles which may collect in the sensing gap, and then zeroed using the following procedure.

Preset the following command to the probe after 3.5 character time quite time, using the Write Single Register to write to register 3; the data written is unimportant.

FIELD NAME	Example (Hex)
SLAVE ADDRESS	01
FUNCTION	06
STARTING ADDRESS HI	00
STARTING ADDRESS LO	03
NO. OF REGISTERS HI	00
NO. OF REGISTERS LO	01
CRC CHECK	B80A

Check the response which should be the echo of the query send to the probe.

Repeat stage 2 and 3 so that it has been written 3 times within 10 SECONDS for probe to zero the output in the clean oil sample. If this write command is not done 3 times within 10 seconds it will be ignored, to ensure that it is not reset accidentally.

DATA FORMAT

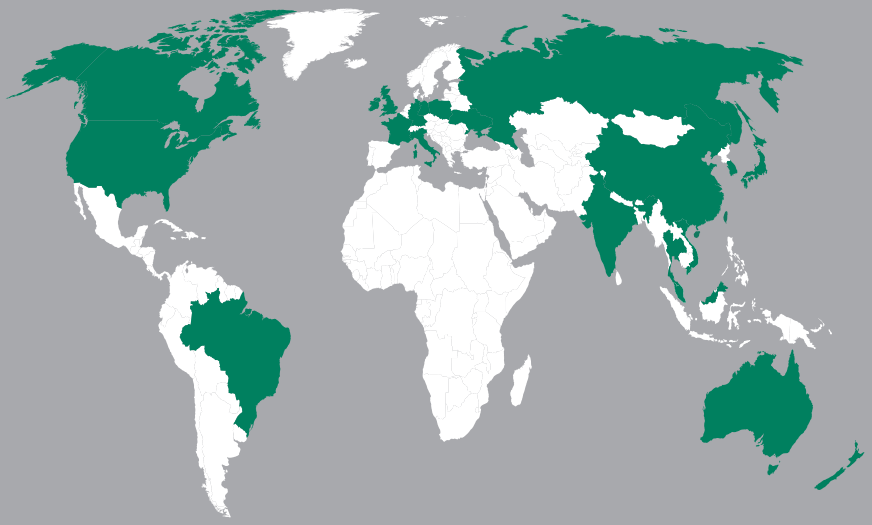
STAUFF returns all real (non-integer) values as 16 bit signed integers with the value multiplied by 100 (decimal).

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).

STAUFF®

Local Solutions For Individual Customers Worldwide



GERMANY / DEUTSCHLAND
Walter Stauffenberg GmbH & Co. KG
Im Ehrenfeld 4 ▪ 58791 Werdohl
Tel.: +49 23 92 916 0
Fax: +49 23 92 916 160
sales@stauff.com

Globally available through wholly-owned branches and distributors in all industrial countries. Full contact details at:

www.stauff.com/contact

Globale Präsenz mit eigenen Niederlassungen und Händlern in sämtlichen Industrieländern. Vollständige Kontaktdaten unter:

www.stauff.com/kontakt