Modbus TCP/IP Protocol for CEMB Cube

1. General Information

This document describes the Modbus TCP/IP Interface for CEMB Cube device. Before you begin programming, you should refer to the documentation available at site http://www.modbus.org to familiarize yourself with the basics of the Modbus protocol. In particular, refer to the "Modbus messaging on TCP/IP implementation guide v1.0b".

This document does not tell you how to program your DCS computer to access the CEMB Cube device nor how to configure the interface database. You must refer to the manuals for the DCS computer or controller for that information.

The Modbus protocol was developed in 1979 by Modicon, for industrial automation systems and Modicon programmable controllers. It has since become an industry standard method for the transfer of discrete/ analog I/O information and register data between industrial control and monitoring devices. Modbus is now a widely-accepted, open, public-domain protocol. Modbus devices communicate using a client-server (client-server) technique in which only one device (the client) can initiate transactions (called queries). The other devices (servers) respond by supplying the requested data to the client, or by taking the action requested in the query. A server is any peripheral device (I/O transducer, valve, network drive, or other measuring device) which processes information and sends its output to the client using Modbus. The CEMB Cube acts as server device, while a typical client device is a PLC/DCS running appropriate application software.

It's important to make the distinction that Modbus itself is an *application protocol*, as it defines rules for organizing and interpreting data, but remains simply a messaging structure, independent of the underlying physical layer.

Modbus TCP/IP uses TCP/IP and Ethernet to carry the data of the Modbus message structure between compatible devices. That is, Modbus TCP/IP combines a physical network (Ethernet), with a networking standard (TCP/IP), and a standard method of representing data (Modbus as the application protocol). Essentially, the Modbus TCP/IP message is simply a Modbus communication encapsulated in an Ethernet TCP/IP wrapper. In practice, Modbus TCP embeds a standard Modbus data frame into a TCP frame, without the Modbus checksum, as shown in the following figure.

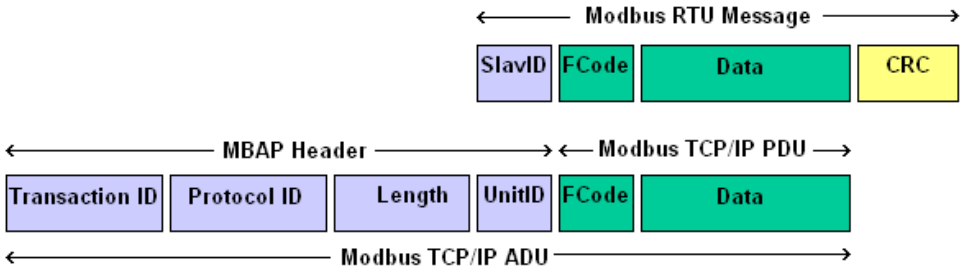


Figure 1

The Modbus commands and user data are themselves encapsulated into the data container of a TCP/IP telegram without being modified in any way. However, the Modbus error checking field (checksum) is not used, as the standard Ethernet TCP/IP link layer checksum methods are instead used to guaranty data integrity. Further, the Modbus slave

address field is supplanted by the unit identifier in Modbus TCP/IP, and becomes part of the Modbus Application Protocol (MBAP) header (more on this later).

Thus, a Modbus TCP/IP Application Data Unit (ADU) takes the form of a 7 byte header (transaction identifier + protocol identifier + length field + unit identifier), and the protocol data unit (function code + data). The MBAP header is 7 bytes long and includes the following fields:

- **Transaction/invocation Identifier (2 Bytes)**: This identification field is used for transaction pairing when multiple messages are sent along the same TCP connection by a client without waiting for a prior response.
- **Protocol Identifier (2 bytes)**: This field is always 0 for Modbus services and other values are reserved for future extensions.
- **Length (2 bytes):** This field is a byte count of the remaining fields and includes the unit identifier byte, function code byte, and the data fields.
- **Unit Identifier (1 byte)**[1]: This field is used for routing purpose when addressing a device on a MODBUS+ or MODBUS serial line sub-network. In that case, the "Unit Identifier" carries the MODBUS slave address of the remote devicehis field is used to identify a remote server located on a non TCP/IP network (for serial bridging).
  …
  On TCP/IP, the MODBUS server is addressed using its IP address; therefore, the MODBUS Unit Identifier is useless. The value 0xFF has to be used.
  When addressing a MODBUS server connected directly to a TCP/IP network, it's recommended not using a significant MODBUS slave address in the "Unit Identifier" field. In the event of a re-allocation of the IP addresses within an automated system and if a IP address previously assigned to a MODBUS server is then assigned to a gateway, using a significant slave address may cause trouble because of a bad routing by the gateway. Using a non-significant slave address, the gateway will simply discard the MODBUS PDU with no trouble. 0xFF is recommended for the "Unit Identifier" as non-significant value.
  Remark : The value 0 is also accepted to communicate directly to a MODBUS/TCP device.
  In CEMB Cube device, **the unit ID is set to 0xFF**, and the client must use this value.

The complete Modbus TCP/IP Application Data Unit is embedded into the data field of a standard TCP frame and sent via TCP to well-known system port 502, which is specifically reserved for Modbus applications. Modbus TCP/IP clients and servers listen and receive Modbus data via port 502.

We can see that the operation of Modbus over Ethernet is nearly transparent to the Modbus register/command structure. Thus, if you are already familiar with the operation of traditional Modbus, then you are already very with the operation of Modbus TCP/IP.

2. Modbus Data Types and Function Codes

The Modbus standard defines storage in:

- Bits. Each bit has its own address.
- Registers (16-bit). Each register has its own address and it can hold integers in the range 0 to 65535 (dec), which is 0 to 0xFFFF (hex).

[1] From "Modbus messaging on TCP/IP implementation guide v1.0b"

Modbus defines “table” names dependent on whether the storage is in a single bit or in a 16-bit register, and whether it is possible to write to the storage.

Table 1: Modbus Data Types

Storage in	Access	Modbus “table”	Example use
Bits	Read only	Discrete inputs	A digital input
Bits	Read and write	Coils	A digital output
16-bit register	Read only	Input registers	Several digital inputs, or an analog input
16-bit register	Read and write	Holding registers	Several digital outputs, or a setting parameter

Function codes are used to describe the read or write operations.

Table 2: Modbus Function Codes

Code	Meaning	CEMB Cube device Data	Access
01	Read Coils	Alarm Limits Enabling (Read)	Read
02	Read Input Status	Device Statuses Channel Statuses Measurements Statuses	Read
03	Read Holding Registers	Device configuration parameters Measurements configuration parameters Alarm Thresholds	Read
04	Read Input Registers	Real-Time Measurement Values Analog/Digital Outputs Values	Read
05	Write Single Coil	Alarm Limits Enabling (Write)	Write
06	Write Single Register	Device configuration parameters Measurements configuration parameters Alarm Thresholds	Write
15	Write Multiple Coils	Alarm Limits Enabling	Write
16	Write Multiple Registers	Device configuration parameters Measurements configuration parameters Alarm Thresholds Write real-time shaft speed from external source	Write

The Modbus protocol exposes two different Function Codes to write registers:

- 06 Write Single Register can be used to write one register only (no more)
- 16 Write Multiple Register can be used to write any number of consecutive registers, starting from 1

As detailed in the following paragraph “Data encoding”, some of the data are stored in a set of consecutive registers, and for this reason they must be written all together.

The most easy and effective way is using Function Code 16 (WRITE MULTIPLE REGISTERS) to write any number of registers (even if the number is 1).

Similar considerations are valid for coils too, that can be written through Function Code 05 (Write Single Coil) and 15 (Write Multiple Coils). The data model for CEMB Cube doesn’t split any data over multiple coils and so it’s up to the user using Function Code 05 or 15.

3. Modbus Addressing model

As described in the Modbus application protocol, two different concepts are involved in Modbus communication:

- Modbus registers (or modbus data): they must be numbered from 1 to n
- Modbus addresses in the PDU (Protocola Data Unit) messages: they starts from 0 to 65565

Table 3 shows how the several Modbus registers are addressed in the PDU Modbus messages.

Table 3: Modbus register numbers and Modbus addresses in PDU messages

Data Type	Modbus Register number (1-based)		Modbus PDU Message Address (0-based)	
	Format (decimal)	Range	Format (decimal)	Range
Coils	0XXXX	00001-09999	XXXX	0000-9998
Discrete Inputs	1XXXX	10001-19999	XXXX	0000-9998
Input Registers	3XXXX	30001-39999	XXXX	0000-9998
Holding Registers	4XXXX	40001-49999	XXXX	0000-9998

As an example:

- The coil known as 'Coil 1' in the device is addressed as coil 0000 in the data address field of a Modbus PDU message. Coil 127 decimal is addressed as coil 007E hex (126 decimal).
- The discrete input known as 'Discrete Input 10001' in the device is addressed as discrete input 0000 in the data address field of a Modbus PDU message. The function code field already specifies a 'discrete input' operation. Therefore the '1XXXX' reference is implicit.
- The input register input known as 'Input register 30001' in the device is addressed as input register 0000 in the data address field of a Modbus PDU message. The function code field already specifies a 'input register' operation. Therefore the '3XXXX' reference is implicit.
- The holding register input known as 'Holding register 40001' in the device is addressed as holding register 0000 in the data address field of a Modbus PDU message. The function code field already specifies a 'holding register' operation. Therefore the '4XXXX' reference is implicit.

NOTE

This document lists both the Modbus Register numbers (1-based) and the Modbus PDU Message Address (0-based). Use the format that is appropriated for your host.

4. Data encoding

The Modbus protocol itself is declared as a 'big-Endian' protocol, as per the Modbus Application Protocol Specification, V1.1.b:

> "Modbus uses a "big-Endian" representation for addresses and data items. This means that when a numerical quantity larger than a single byte is transmitted, the most significant byte is sent first."

Although Modbus.org standard documents provide some guidance for implementing the Modbus protocol, they do not address the question of word order beyond the 16-bit register level.

32-bit numeric values

When the need for transferring 32-bit (i.e. 4 byte) values with the Modbus protocol later came about in the 1980's, Little-Endian Intel processors dominated the PC market so most vendors chose to map the least significant word onto the lower address of the register pair.

This lack of standardization for values larger than 16 bits has resulted in a situation where Modbus implementers have to make an arbitrary choice as to which address of the register pair contains the most significant word of 32-bit values such as IEEE 754-1985 single-precision floats and signed or unsigned 32-bit integers.

Since the most common default word order today is Little-Endian, that is the word order that is used in CEMB Cube device.

For example, when a 32-bit integer value of decimal 16909060 (or 0x01020304 hexadecimal) is mapped onto two holding registers, 4 1001 and 41002, then register 41001 contains the least significant word (0x0304) and register 41002 contains the most significant word (0x0102).

Even if CEMB Cube device use this representation, a good client SHOULD be configurable to accept both orders for words. It's up to the customer to configure the client in the proper way.

Floating point numbers

When a floating point is necessary to represent a given value, CEMB Cube device uses IEEE 754-1985 Single Precision representation, which requires 32 bits. These kind of numbers are named Float32 in this document, and they are contained into two consecutive registers as described in the previous paragraph.

IEEE 754-1985 allows different representations for "Not a Number" values. CEMB Cube uses the qNaN (Quiet NaN) representation, corresponding to the value 0x7FC00000, which means:

- Sign bit 0
- Exponent 0xFF
- Fraction bit[22] == 1, bits[21:0] == 0

In this document, this special value is referred as NaN.

16-bit integer numbers

A 16-bit integer number can be represented by a single modbus register, which is 16-bit large. This document refers to 16-bit numbers in the following way:

- U16 or UInt16 unsigned integer in the range [0 – 65535]
- I16 or Int16 signed integer in the range [-32768 – 32767]

8-bit numeric values

Two 8-bit numeric values can be packed in a single 16-bit modbus register. The two numbers are packed into the Least Significative Byte (LSB) and Most Significative Byte (MSB). MSB is also called High Byte (H) and LSB is also called Low Byte (L).

Strings

The strings are encoded in UTF-8 (see here) and must be terminated by a 'null' byte (value 0x00). Since UTF-8 represents every character using one to four bytes (8-bit), all the registers (16 bits) that compose the string must be appended before decoding the string itself.

Every string can hold a predefined maximum number of bytes, because a given number of Modbus registers has been reserved. Since every register contains two bytes, the maximum number of bytes is always even.

However, the maximum number of characters for the string is not fixed, because it depends on how many bytes are necessary to encode the several characters (from 1 to 4). If all the characters can be encoded using a single byte, considering that the string must be zero-terminated, its length can be maximum $(2 * n_{registers}) - 1$.

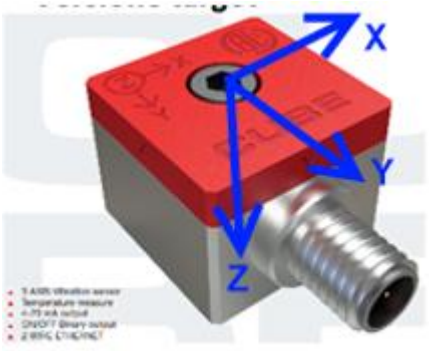
For example, when the string "Hello world!" is mapped onto 8 registers, from 41011 to 41018, they contains the following values:

	41011	41012	41013	41014	41015	41016	41017	41018
Hex	0x4865	0x6C6C	0x6FA0	0x776F	0x726C	0x6421	0x0000	0x0000
Char	'H' 'e'	'l' 'l'	'o' ' '	'w' 'o'	'r' 'l'	'd' '!'	'\0' '\0'	'\0' '\0'

5. Naming conventions

CEMB Cube device has four measurement channels:

- 3-axis vibration channels, named X, Y and Z (capital letters) in this document.
  The direction are clearly marked on the case of CEMB cube device, as shown in the following picture.



- One temperature channel, named T (capital letter) in this document.

6. General Register Layout

The Modbus TCP server uses fixed protocol addresses for the starting locations of data in the device. The data addresses are not the physical addresses in the CEMB Cube device. Protocol messages use the data addresses to access data that is available from the module. Table 4 summarizes both the Modbus PDU Message addresses and the Modbus register numbers.

Table 4: Modbus PDU Message addresses and Modbus register numbers

Data Type	Function Code	Address Ranges	
		Modbus PDU Message Address (0-based)	Modbus Register number (1-based)
Alarm Limits Enabling	1 (R) , 5 or 15 (W)	0 - 31	0001 - 0032
Trigger Command Execution	1 (R) , 5 or 15 (W)	9800 - 9802	9800 - 9803
Device Statuses	2	0 – 15	10001 – 10016
Channel Statuses	2	16 – 79	10017 – 10080
Measurements Statuses	2	80 – 95	10081 – 10096
Real-Time Measurement Values	4	0 - 63	30001 - 30064
Analog/Digital Outputs Values	4	64 - 68	30065 - 30069
Device configuration parameters	3 (R), 6 or 16 (W)	0 - 79	40001 – 40080
Measurements configuration parameters	3 (R), 6 or 16 (W)	80 - 199	40081 – 40200
Alarm Thresholds	3 (R), 6 or 16 (W)	200 - 229	40201 – 40230
Analog/Digital Ouputs configuration	3 (R), 6 or 16 (W)	230 - 249	40231 – 40250
Repeated Device Statuses *	3 (R)	1000	41001
Repeated Channel Statuses *	3 (R)	1001 - 1004	41002 – 41005
Repeated Measurements Statuses *	3 (R)	1005	41006
Repeated **Real-Time Measurement Values ***	3 (R)	1006 - 1069	41007 - 41070
Repeated Analog/Digital Outputs Values *	3 (R)	1070 - 1074	41071 - 41075
Write real-time shaft speed from external source	3 (R), 6 or 16 (W)	2000 - 2001	42001 - 42002

The repeated data registers that are marked with an asterisk (*) in Table 4 contain duplicated data in different registers for some Modbus devices that only support the 4XXXX series registers. Coils and inputs registers are bit-packed in the 4XXXX registers so that the first coil/input corresponds to the least significant bit (bit 0) of the 4XXXX register.

7. Modbus Register Data Model

This document describes each data type in detail and describes the purpose of each register. For more information about these data types, refer to the references in Table 5.

Table 5: Data Model for CEMB Cube

Data Type	Page Number

8. Real-Time Measurement Values

The real-time measurement values include all the measurements calculated over vibration values:

Use Function Code 4 (READ INPUT REGISTERS) to read the real-time measurement values.

NOTE
These values are repeated in the 4XXXX address range. Refer to Paragraph 6 for register layout.

Table 6: Real-Time Measurement Values

Measurements	PDU Message Address (0-based)	Modbus Register number (1-based)	Data Type	Default / Example
***accX* Overall**	0 - 1	30001 - 30002	Float32	0.0
***accY* Overall**	2 – 3	30003 – 30004	Float32	0.0
***accZ* Overall**	4 – 5	30005 – 30006	Float32	0.0
accMAX Overall(X,Y,Z)	6 - 7	30007 - 30008	Float32	0.0
***Not used*[2]**	8 - 9	30009 – 30010		
***velX* Overall**	10 – 11	30011 – 30012	Float32	0.0
***velY* Overall**	12 – 13	30013 – 30014	Float32	0.0
***velZ* Overall**	14 - 15	30015 - 30016	Float32	0.0
velMAX Overall(X,Y,Z)	16 – 17	30017 – 30018	Float32	0.0
***Not used*[3]**	18 – 19	30019 – 30020		
T	20 – 21	30021 – 30022	Float32	0.0
Shaft speed	22 – 23	30023 – 30024	Float32	0.0
velDA*[4] *Unbalance	24 - 25	30025 - 30026	Float32	0.0
velDA Misalignment	26 – 27	30027 – 30028	Float32	0.0
velDA Looseness	28 – 29	30029 – 30030	Float32	0.0
velDA Other	30 – 31	30031 – 30032	Float32	0.0
***Not used*[5]**	32 – 63	30033 - 30064		

[2] All these registers are read as 0
[3] All these registers are read as 0
[4] DA means Diagnostic Axis (i.e. the axis to be used for calculation of diagnostic vibration components)
[5] All these registers are read as 0

9. Analog/Digital Outputs Values

Some models of CEMB Cube family have analog and/or digital outputs, identified by the names U1 and U2.

When they're available, the following registers contains the values/states set to the ouputs. Otherwise, reading from these registers returns 0.

Use Function Code 4 (READ INPUT REGISTERS) to read the Analog/Digital Outputs values.

NOTE
These values are repeated in the 4XXXX address range. Refer to Paragraph 6 for register layout.

Table 7: Analog/Digital Outputs Values

Measurements	PDU Message Address (0-based)	Modbus Register number (1-based)	Data Type	Default / Example
U1 Analog Output mA	64 – 65	30065 – 30066	Float32	0.0
U2 Analog Output mA	66 – 67	30067 – 30068	Float32	0.0
U2 Digital Output State	68	30069	Uint16 – valid values: 0x0 – inactive 0xFFFF – active 0x00FF – High-Z	0x0

10. Device Statuses

The Device Statuses contain all the information on operational condition of the device. They have a value of "1" for true and "0" for false.

Use Function Code 2 (READ INPUT STATUS) to read the Device Statuses.

NOTE
These values are repeated (as Read Only) in the 4XXXX address range. Refer to Paragraph 6 for register layout.

Table 8: Device Status Addresses for CEMB Cube

Device Status	PDU Message Address (0-based)	Modbus register number (1-based)
Good Operating Mode	0	10001
ANY Hardware Fault	1	10002
MEMS Hardware Fault	2	10003
EEPROM Hardware Fault	3	10004
Configuration Fault (Corrupted Setup)	4	10005
Boot in progress	5	10006
Digital Output Driver Overcurrent/Fault	6	10007
Invalid Setup[6]	7	10008
Not used	8-15	10009 – 10016

11. Channel Statuses

These digital inputs contain a true or false value for every status of each channel of the device. A "0" indicates false and a "1" indicates true. If a device doesn't have all the channels enabled (ON), then the Channel Status bits for the disabled (unused) channels will be 0. Use Function Code 2 (READ INPUT STATUS) to access the Channel Status values for the device.

NOTE
These values are repeated (as Read Only) in the 4XXXX address range. Refer to Paragraph 6 for register layout.

Table 9: Channel Status Bits for CEMB Cube

Device Status	PDU Message Address (0-based)	Modbus register number (1-based)
X ON	16	10017
X Invalid (for ANY reason)	17	10018
X Fault	18	10019
X Saturation	19	10020

[6] One or more parameters have been set to values not allowed. The whole setup is not accepted and the Cube device is NOT in Good Operating Mode. The Cube Manager software can be used to download a working set of parameters to the device.

Not used	20-31	10021-10032
Y ON	32	10033
Y Invalid (for ANY reason)	33	10034
Y Fault	34	10035
Y Saturation	35	10036
Not used	36-47	10037-10048
Z ON	48	10049
Z Invalid (for ANY reason)	49	10050
Z Fault	50	10051
Z Saturation	51	10052
Not used	52-63	10053-10064
T ON	64	10065
T Invalid (for ANY reason)	65	10066
T Fault	66	10067
T Saturation	67	10068
Not used	68-79	10069-10080

12. Measurements Statuses

These digital inputs contain a true or false value for every status of each measurement done by the device. A "0" indicates false and a "1" indicates true. Use Function Code 2 (READ INPUT STATUS) to access the Measurement Status values.

NOTE
These values are repeated (as Read Only) in the 4XXXX address range. Refer to Paragraph 6 for register layout.

Table 10: Measurement Status Bits for CEMB Cube

Measurement Status	PDU Message Address (0-based)	Modbus register number (1-based)
ANY Alert (Vibration OR T)	80	10081
ANY Vibration Alert (ANY acc OR ANY vel)	81	10082
ANY acc Alert (accX OR accY OR accZ)	82	10083
ANY vel Alert (vibX OR vibY OR vibZ)	83	10084
ANY X Alert (accX OR velX)	84	10085
ANY Y Alert (accY OR velY)	85	10086
ANY Z Alert (accZ OR velZ)	86	10087
accX Alert	87	10088
accY Alert	88	10089
accZ Alert	89	10090
velX Alert	90	10091
velY Alert	91	10092
velZ Alert	92	10093
accMAX(X,Y,Z) Alert	93	10094
velMAX(X,Y,Z) Alert	94	10095
T Alert	95	10096

13. Alarm Limits Enabling

Every alarm limit can be enabled or disabled: a true or false value indicates if the specific alarm limit is enabled (1 = true) or not (0 = false).

You can both read and write alarm enabling statuses.

For a CEMB Cube device the list of alarm limits enabling statuses is listed in Table 11.

NOTE
These values are repeated (as Read Only) in the 4XXXX address range, together with the Alarm Limit Thresholds. Refer to Paragraph 16 for register layout.

Table 11: CEMB Cube device - Alarm Enabling

Alarm enabling	PDU Message Address (0-based)	Modbus Register number (1-based)	Read or Write
***accX* Overall Alert Enabled**	0	1	Read / Write
***accY* Overall Alert Enabled**	1	2	Read / Write
***accZ* Overall Alert Enabled**	2	3	Read / Write
***accMAX(X,Y,Z)* Overall Alert Enabled**	3	4	Read / Write
***velX* Overall Alert Enabled**	4	5	Read / Write
***velY* Overall Alert Enabled**	5	6	Read / Write
***velZ* Overall Alert Enabled**	6	7	Read / Write
***velMAX(X,Y,Z)* Overall Alert Enabled**	7	8	Read / Write
T Alert Enabled	8	9	Read / Write
Not Used	9 – 31	10 - 32	

14. Device configuration parameters

For a CEMB Cube the list of configuration parameters is shown in the following Table 12.

Some of them are Read Only and so cannot be written, some others are Read/Write and can be read using Function Code 16 (recommended) or 06 (non recommended) – see paragraph “Modbus Data Types and Function Codes”.

Table 12: CEMB Cube configuration parameters

Parameter	PDU Message Address (0-based)	Modbus Register number (1-based)	Read/Write	Data Type	Default / Example

Product HW Code	0 - 5	40001 - 40006	Read Only	string – max 12 chars	88SC064052
Model Name ID	6	40007	Read Only	U16 – valid values: 0x0 – CUBE/0 0x3 – CUBE/1 0x5 – CUBE/2 0x6 – CUBE/3	0
Model Name	7 – 16	40008 – 40017	Read Only	string – max 20 chars	CUBE/3
Device SN	17 – 21	40018 – 40022	Read Only	string – max 10 chars	S0030991
MAC address	22 – 30	40023 – 40031	Read Only	string – 18 chars	00-08-79-EF-07-53
Firmware number	31 – 33	40032 – 40034	Read Only	string – max 6 chars	M4041
Firmware release	34 – 38	40035 – 40039	Read Only	string – max 10 chars	2.3.1
Not used	39	40040	Read Only		
Tag Name	40 - 49	40041 – 40050	Read / Write	string – max 20 chars	-----
IP address (IPv4)	50 – 57	40051 – 40058	Read / Write	string – max 16 chars	192.168.20.23
Subnet mask (IPv4)	58 – 65	40059 – 40066	Read / Write	string – max 16 chars	255.255.255.0
IP Gateway (IPv4)	66 – 73	40067 – 40074	Read / Write	string – max 16 chars	0.0.0.0
Not used	74 – 79	40075 – 40080	Read / Write		

15. Measurements configuration parameters

For a CEMB Cube the list of parameters to configure the several measurements is shown in the following Table 12.

Some of them are Read Only and so cannot be written, some others are Read/Write and can be read using Function Code 16 (recommended) or 06 (non recommended) – see paragraph “Modbus Data Types and Function Codes”.

Table 13: CEMB Cube Measurements configuration parameters

Parameter	PDU Message Address (0-based)	Modbus Register number (1-based)	Read/Write	Data Type	Default / Example
Acc unit	80 - 84	40081 – 40085	Read Only	string – max 10 chars	G
Acc LB (Lower Boundary)	85 - 86	40086 – 40087	Read Only	Float32	0.0
Vel unit	87 - 91	40088 – 40092	Read Only	string – max 10 chars	mm/s
Vel LB (Lower Boundary)	92 - 93	40093 – 40094	Read Only	Float32	0.0
Not used	94	40095	Read Only		
Waveform nr. of samples	95	40096	Read Only	UInt16	4096
Frequency unit	96 - 98	40097 – 40099	Read Only	string – max 6 chars	Hz
T unit	99 - 102	40100 – 40103	Read Only	string – max 8 chars	°C
Shaft speed unit	103 - 105	40104 - 40106	Read Only	string – max 6 chars	rpm
Frequency Range LB (Lower Boundary)	106 - 107	40107 – 40108	Read Only	Float32	0.0

Frequency Range UB (Upper Boundary)	108 - 109	40109 – 40110	Read Only	Float32	5000.0
Not used	110 - 119	40111 - 40120	Read Only		
Acc unit ID	120	40121	Read / Write	U16 – valid values: 0x0 – g 0xF – m/s^2	0x0
Acc Overall Mode	121	40122	Read / Write	U16 – see *Table 14: Overall Measurement Mode*	12
Acc HB (Higher Boundary)	122 - 123	40123 – 40124	Read / Write	Float32	25.0
Acc HighPass filter	124	40125	Read / Write	UInt16 – as [Hz]	1000.0
Acc LowPass filter	125	40126	Read / Write	UInt16 – as [Hz]	5000.0
Vel unit ID	126	40127	Read / Write	U16 – valid values: 0x0F00 – mm/s 0xF000 – in/s	0x0F00
Vel Overall Mode	127	40128	Read / Write	U16 – see *Table 14: Overall Measurement Mode*	0
Vel HB (Higher Boundary)	128 – 129	40129 – 40130	Read / Write	Float32	20.0
Vel HighPass filter	130	40131	Read / Write	UInt16 – as [Hz]	2.0
Vel LowPass filter	131	40132	Read / Write	UInt16 – as [Hz]	2000.0
Diagnostic Axis	132	40133	Read / Write	U16 – valid values: 0x00 – Z 0x0F – X 0xF0 – Y	0x00
Reaction time	133	40134	Read / Write	U16 – valid values: 0x7 – fast 0xF – medium 0x1F - slow	0xF
Frequency unit ID	134	40135	Read / Write	U16 – valid values: 0x0 – cpm 0xF – Hz	0xF
T unit ID	135	40136	Read / Write	U16 – valid values: 0x0 – °C 0xF - °F	0x0
T LB (Lower Boundary)	136 – 137	40137 – 40138	Read / Write	Float32	20.0
T HB (Higher Boundary)	138 – 139	40139 – 40140	Read / Write	Float32	120.0
Shaft speed source	140	40141	Read / Write	U16 – valid values: 0x0 – nominal value 0x3 – estimated 0x5 – not available 0x6 – from external source	0
Nominal Shaft speed	141 – 142	40142 – 40143	Read / Write	Float32	1500.0
Shaft speed HB (Higher Boundary)	143 – 144	40144 – 40145	Read / Write	Float32	1800.0
Not used	145 – 146	40146 – 40147	Read Only		
Shaft speed estimation band HB (High Boundary)	147 – 148	40148 – 40149	Read / Write	Float32	1600.0
Shaft speed unit ID	149	40150	Read / Write	U16 – valid values: 0x0 – rpm	0x0
Raw Data Signal	150	40151	Read / Write	U16 – valid values:	0xF

				0x0 – vel 0xF – acc	
Frequency Range ID	151	40152	Read / Write	U16 – valid values: 0x0 – standard 0x3 – low speed	0x0
Shaft Speed Estimation Harmonic	152	40153	Read / Write	U16	1
Not used	153 - 199	40154 - 40200	Read / Write		

Overall Measurement Mode

The Overall Measurement Mode parameter can have one of the following values.

Table 14: Overall Measurement Mode

Value	Measurement Mode
0x0 (0)	RMS
0x3 (3)	Pk
0xC (12)	PkPk

16. Alarm Thresholds

Every alarm can be enabled or disabled, and has a threshold level. If an alarm is enabled and the corresponding measurement value exceeds the threshold, the alert condition(s) is asserted (see paragraph “Measurements Statuses”).

You can both enable and disable alarms and read and write alarm thresholds.

Table 15: Alarm Thresholds

Parameter	PDU Message Address (0-based)	Modbus Register number (1-based)	Read or Write	Data Type	Default / Example
Repeated Alarm Limits Enabling *	200 – 201	40201 - 40202	Read / Write	Bits packed (see Alarm Limits Enabling)	
***accX* Overall Alert Threshold**	202 – 203	40203 – 40204	Read / Write	Float32	5.0
***accY* Overall Alert Threshold**	204 – 205	40205 – 40206	Read / Write	Float32	5.0
***accZ* Overall Alert Threshold**	206 – 207	40207 – 40208	Read / Write	Float32	5.0
***accMAX(X,Y,Z)* Overall Alert Threshold**	208 – 209	40209 – 40210	Read / Write	Float32	5.0
***velX* Overall Alert Threshold**	210 – 211	40211 – 40212	Read / Write	Float32	10.0
***velY* Overall Alert Threshold**	212 – 213	40213 – 40214	Read / Write	Float32	10.0
***velZ* Overall Alert Threshold**	214 – 215	40215 – 40216	Read / Write	Float32	10.0
***velMAX(X,Y,Z)* Overall Alert Threshold**	216 – 217	40217 – 40218	Read / Write	Float32	10.0
***T* Alert Threshold**	218 – 219	40219 – 40220	Read / Write	Float32	80.0
Debouncing Time [s]	220 – 221	40221 – 40222	Read / Write	Float32	3.0
Not Used	222 – 229	40223 – 40230	Read / Write		

17. Analog/Digital Ouputs configuration

Some models of CEMB Cube family have analog and digital outputs, identified by the names U1 and U2.

When they're available, they can be configured through the following registers. Otherwise, writing to these registers has no effect.

Table 16: CEMB Cube Analog/Digital Ouputs configuration

Parameter	PDU Message Address (0-based)	Modbus Register number (1-based)	Read/Write	Data Type	Default / Example
U1 Analog Output Meas	230	40231	Read / Write	U16 – see *Table 17* for valid values	18
U2 Analog Output Meas	231	40232	Read / Write	U16 – see *Table 17* for valid values	9
U2 Digital Output Logic type	232	40233	Read / Write	U16 – valid values: 0x0000 – PNP 0x000F – NPN 0x00F0 – Push-Pull	0x0000
Not Used	233	40234	Read / Write		
U2 Digital Output Condition	234	40235	Read / Write	U16 – see *Table 18* for valid values	5
Not Used	235 – 249	40236 – 40250	Read / Write		

Analog Output binding

When U1 and, optionally, U2 are Analog Output, they can be bound to one measurement, based on the following table.

Table 17: Analog Output binding

Value	Measurement bound
0x0 (0)	None
0x3 (3)	*accX* Overall
0x5 (5)	*accY* Overall
0x6 (6)	*accZ* Overall
0x9 (9)	*accMAX Overall(X,Y,Z)*
0xA (10)	*velX* Overall
0xC (12)	*velY* Overall
0x11 (17)	*velZ* Overall
0x12 (18)	*velMAX Overall(X,Y,Z)*

U2 Digital output activation condition

When U2 is a digital output, it's set to its 'active' (or tripped) state when the "activation condition" is evaluated to true. This condition is one of the Measurement Statuses listed in paragraph Measurements Statuses.

Table 18: U2 Digital output activation condition

Value	Activation condition
0x0 (0)	None (U2 never activated)
0x3 (3)	ANY Alert (Vibration OR T)
0x5 (5)	ANY Vibration Alert (ANY acc OR ANY vel)

0x6 (6)	ANY acc Alert (accX OR accY OR accZ)
0x9 (9)	ANY vel Alert (vibX OR vibY OR vibZ)
0xA (10)	X Alert (accX OR velX)
0xC (12)	Y Alert (accY OR velY)
0x11 (17)	Z Alert (accZ OR velZ)
0x12 (18)	accX Alert
0x14 (20)	accY Alert
0x17 (23)	accZ Alert
0x18 (24)	velX Alert
0x1B (27)	velY Alert
0x1D (29)	velZ Alert
0x1E (30)	accMAX(X,Y,Z) Alert
0x21 (33)	velMAX(X,Y,Z) Alert
0x22 (34)	T Alert
0x24 (36)	ANY Hardware Fault

18. Write real-time shaft speed from external source

If the shaft speed source is set to “from external device”, the value of the real-time shaft speed can be written by a modbus client, that gets this value from somewhere else. As an example:

- from another device which has a tachometer sensor
- from a motor drive
- from a PLC
- …

In this case, CUBE device uses this value as the real-time shaft speed to calculate the expert analysis vibrations on the Diagnostic Axis (Unbalance, Misalignment, Looseness, Other).

If the shaft speed source is set to a value different from “from external device”, writing to the following modbus registers has no effect.

Table 19: CEMB Cube Real-Time shaft speed from external source

Parameter	PDU Message Address (0-based)	Modbus Register number (1-based)	Read/Write	Data Type	Default / Example
Real-Time shaft speed from EXT	2000 - 2001	42001-42002	Read / Write	Float32	1500.0

19. Trigger Command Execution

CEMB Cube has some commands, used to do some useful operations. The execution of a specific command can be triggered writing a “1” to the specific register, as shown in the table below.

These registers are always read as “0”.

Table 20: CEMB Cube – Trigger Command Execution

Commands	PDU Message Address (0-based)	Modbus Register number (1-based)	Read or Write
Enter Firmware Upgrade Mode	9800	9801	Write 1 to trigger
Enter Configuration Mode	9801	9802	Write 1 to trigger
Apply New Configuration	9802	9803	Write 1 to trigger
Exit Configuration Mode without changes (Abort)	9803	9804	Write 1 to trigger

Enter Firmware Upgrade Mode

This command must be triggered when a firmware upgrade is needed.

The device switches to “firmware upgrade mode” and the new firmware can be loaded in two ways:

- Using application Cube Manager available from CEMB
- Using a browser to access the address http://<IP address of the Cube device>

Enter Configuration Mode

This command must be triggered when the change of one or more configuration parameters is needed. It must be triggered BEFORE writing the new configuration values to the proper modbus registers (see paragraphs 13 to 17).

Apply New Configuration

This command must be triggered AFTER all the new configuration values have been written to the proper modbus registers (see paragraphs 13 to 17).

The device reads all the new configuration values in a single batch, saves them in its non-volatile internal memory, and starts using them.

Exit Configuration Mode without changes

This command must be triggered if the user want to abort the Configuration Mode, WITHOUT applying the new values.

The device discards all the new configuration values written to the modbus registers and reloads the registers with the values from the non-volatile internal memory.

20. Release note

Version	Date	Changelog
0.9.11	November, 15th 2023	- Removed unused register “Shaft speed estimation band LB” - Added register “Shaft speed estimation harmonic”
0.9.9	October, 13th 2023	- Analog output binding to temperature is no more possible
0.9.8	August, 4th 2023	- Added registers related to “shaft speed from external source”
0.9.7	March, 13th 2023	- Hw code is now 10 chars long
0.9.6	March, 10th 2023	- Change Registers for Digital Output configuration - Change string encoding to UTF-8
0.9.5	March, 9th 2023	- Add Registers for shaft speed estimation
0.9.4	March, 6th 2023	- Add Good Operating Mode status
0.9.3	February, 2nd 2023	- Changed digital output state register
0.9.2	January, 11th 2023	- Added registers
0.9.1	November, 29th 2022	- Added register with MAC address - Added Analog/Digital Outputs Values - Changed valid values for enums (tolerant to single-bit flip fault)
0.9	October, 19th 2022	- Changes as requested during internal review meeting
0.8	September, 29th 2022	- First draft