**SIRIUS**

**THREE-PHASE VOLTAGE STABILISER**

# REMOTE COMMUNICATION CARD (E-stab)

## Description

The communication card manages the remote connection to the voltage stabiliser. The information concerning the status of the stabiliser can be transferred via Ethernet connection, GPRS modem or locally via a USB drive. Data can be analysed from a remote location, thus allowing for actively monitoring the functioning mode and, if necessary, altering the setting of some parameters. Downloading and uploading software, data and functioning parameters can be carried out either locally or remotely.

In addition, a MODBUS® protocol allows for the communication of information to and from a user-controlled device. The card uses the tools available on the user interface: local display (showing alarms, setting parameters and connection data); keypad for surfing the menu into which said information is organised and USB drives for data exchange and software updating operations. Below is a sketch of the remote connection.

server

data analysis

remote parameter control assistance

user management

STABILISER

RS232 local communication

USB

GPRS

Ethernet

with instruments

(stabimon)

MODBUSR data communication

The card (indicated also as e-stab) is assembled on the LED signalling board and consists of two parts:

support

LED & USB

interface

remote communication

data transmission

* ‘Coldfire’ card (upper card). The card hosts the CPU, the USB line transferred onto the user interface panel, the Ethernet line and the flash memory for storing data.
* Support card. The card supplies the coldfire card, hosts the RS485 connections to the input/output instruments, interfaces the local display and the GPRS modem, receives the data input from the keypad and hosts a Real Time Clock (RTC).

The remote communication board is connected to the base board via a CAN line and is provided with protective fuses. A reset button (S2) is located near P12 terminal block.

M ODEM ( RJ 11)



1

P 4

DL3

1

+ 5 V

2

+ 3 . 3 V

V M T P 2

DL1 DL 2

1 2

D L 3

D L 6

1

D L 9

P 1 5

2 5

2 6

1

2

S 2

F 1

1

D L 1 0

P 14

+

- 2

P 15

P 17

1

3 2 1

T P 1

T P 4

T P 3

G N D

RS 232 RS 485

P 1 1

I N P U T 2 4 V D C

SUPPLY

RS 232

ETHERNET ( RJ 145)

D L 7D L 5 D L 2 D L 8 D L 4 D L 1

## LED meaning

#### Support card

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ref. | Colour | Parametre | ON | OFF | Blinking |
| DL1 | green | +3,3Vdc logic supply | available | absent | - |
| DL2 | green | +5Vdc logic supply | available | absent | - |
| DL3 | green | Modem supply | available | absent | - |

#### Upper card

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ref. | Colour | Parametre | ON | OFF | Blinking |
| DL1 | green | Ethernet connection | - | No connection | Connection active |
| DL2 | red | Configuration index | *Not related to the working mode* | | |
| DL3 | red | enables I/O | I/O disabled | I/O enabled | - |
| DL4 | green | Ethernet connection | - | No connection | Connection active |
| DL5 | red | Configuration index | *Not related to the working mode* | | |
| DL6 | yellow | *No function related whatever the status* | | | |
| DL7 | red | Configuration index | *Not related to the working mode* | | |
| DL8 | green | Ethernet connection | - | No connection | Connection active |
| DL9 | red | interrupt Ethernet | interrupt request | No interrupt request | - |
| DL10 | green | reset | card in reset mode | card not in reset mode | - |

## Terminali

#### Support card

|  |  |  |
| --- | --- | --- |
| Ref. | Type | Description |
| P4 | RJ11 6P/6C | Modem supply and command |
| P5 | 8-pole AMP male | PLD programming |
| P14 | 2-pole male | Card supply |
| P15 | 4-pole AMP male | RS232 modem serial |
| P17 | 3-pole male | RS485 |

#### Upper card

|  |  |  |
| --- | --- | --- |
| Ref. | Type | Description |
| P15 | 2-pole AMP male | Second RS232 modem serial |
| P13 | 10-pole AMP male | CAN BUS |
| P12 | 20-pole Flat male | 2 RS232 serial (Debug Coldfire) |
| P11 | RJ45 8P/8C | Ethernet |

## Input/output instruments connection sketch

TR A B SG

RS485

1

2

A B

P17A (JP3 OPEN)

3

GND

TR A B SG

RS485

INPUT INSTRUMENT OUTPUT INSTRUMENT

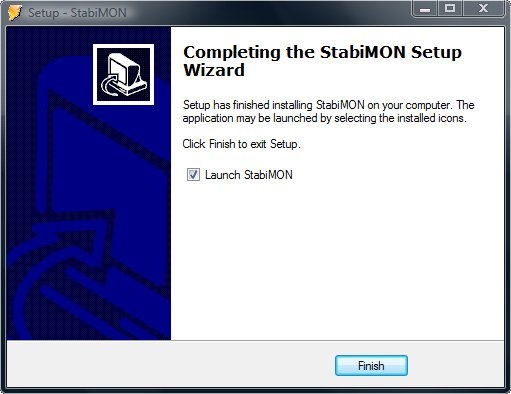
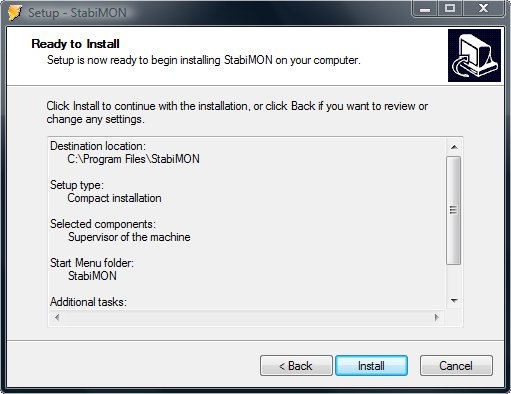
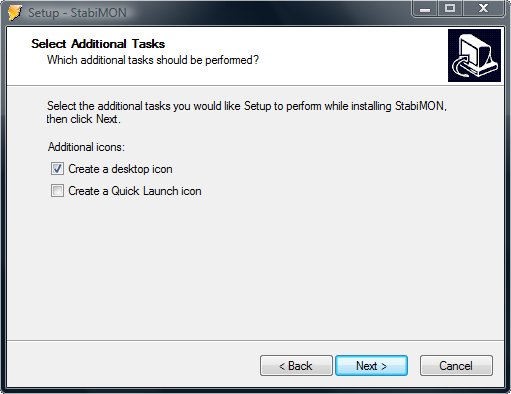
# REMOTE COMMUNICATION

## Stabimon

STABIMON is a software that manaqges the communication with the voltage stabiliser. The program can be run when the user wishes to communicate with the stabiliser or simply read the collected information.

#### Installation

|  |  |
| --- | --- |
| *Language choice* |  |
| *Introductory page* |  |
| *Choice of the installation folder* |  |
| *Type of installation*  Two different installations are available:   * server of communication creates the database and the service that communicates to the different machines. In this case the user can control the machines installed in the factory without connecting to the server located at HQ, but the server and the machine must have the same network address * supervisor of the communication installs only the user interface that allows for the machines available on the server at HQ to be shown. |  |
| *Name of the menu* |  |



*Create desktop icons*

*Summary*

*End of installation*

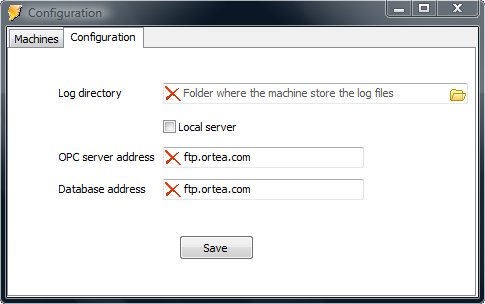
#### Network configuration

* + - 1. Port configuration

The communication of the machine with HQ server uses a TCP/IP channel on the socket 12346. If in the factory there is a firewall or router, this socket must be open for outgoing messages.

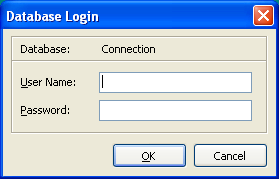
The communication of the StabiMON with HQ server uses a TCP/IP channel on the socket 12348. If in the building where is running the program there is a firewall or router, this socket must be open for outgoing messages.

* + - 1. Address configuration

When the program cannot connect to HQ server (for example the first time the user start the application) the program displays the following page where the user can insert the Database address and the OPC server address.

#### Running the program

At the start, an access form to the Database asking for the user’s username and password is displayed. Said data is necessary to identify the user and grant access to the program services.



Once the credentials have been typed in, the plant synoptic is displayed.

* + - 1. Synoptic

The stabilisers synoptic page can be visualised in different ways, but each one exploits coloured LED to represent the stabiliser status. The colour meaning is as follows:

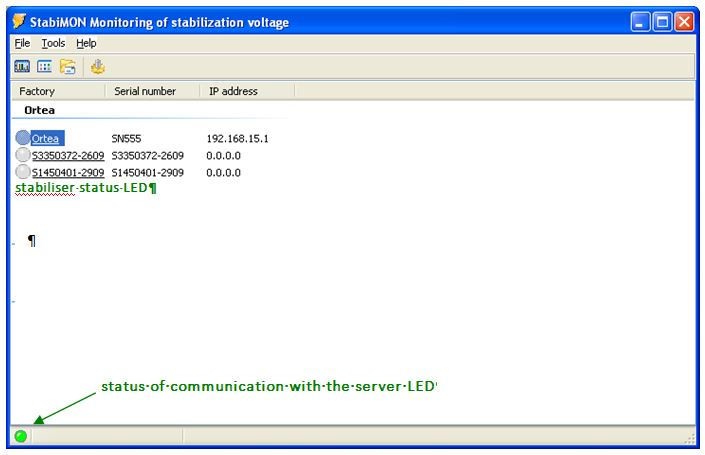
(○) Trasparent no communication with the stabiliser. The communicating card is off or not connected to the

server.

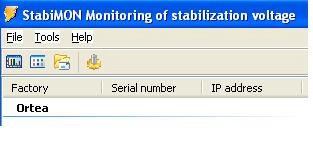
(●) Green the stabiliser is communicating without any errors.

(●) Red the stabiliser is communicating, but there are anomalies that require the operator’s attention. (●) Yellow the communication card cannot connect to the base board.

In addition to the LED associated to the stabilisers, there is another LED at the bottom of the page indicating the communication with the server. If the LED is not green, the program cannot communicate with the server. Should this occur during the setup operation, please check the local network configuration.



By clicking on the icon in the synoptic page, it is possible to move to the graph page.



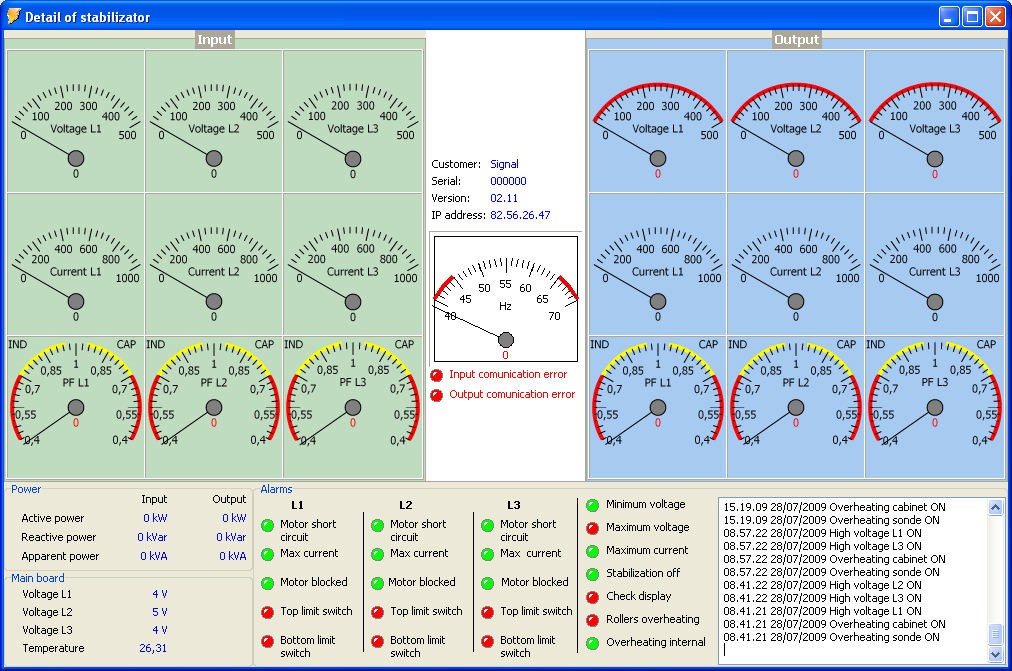
synoptic display change

settings

graph page selection

By clicking on the stabiliser name, the relevant detailed page is displayed.

* + - 1. Stabiliser dashboard



Input multimeter

Output multimeter

space for general info on

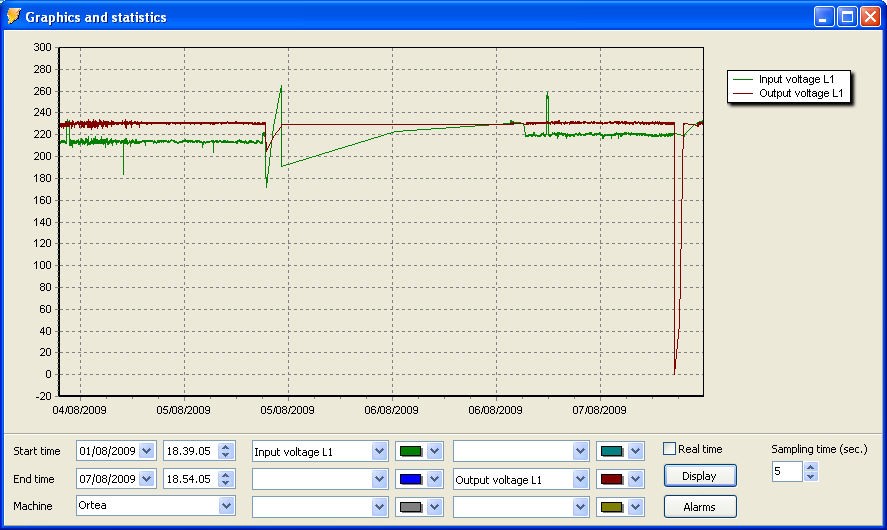
the board

Communication errors between the

communication card and the input/output multimeters and the base board

In a single page, the dashboard provide the main information concerning voltage, current, power and alarm status.

* + - * + *Top left-hand side* of the page: each phase input voltage, current and cos.
        + *Top right-hand side*: corresponding output parameters.
        + *Area between the input and output paramet*ers: mains frequency and general information for the stabiliser identification. Below said data, the communication errors (if any) are listed.
        + *Lower part* of the page: input and output active, reactive and apparent powers, voltage and temperature measured on the base board and reproduction of the LED status as available on the stabiliser control panel. The LED are red in case of error. Beside, a list indicating start (ON) and stop (OFF) of the alarm events is displayed.
      1. Graphs

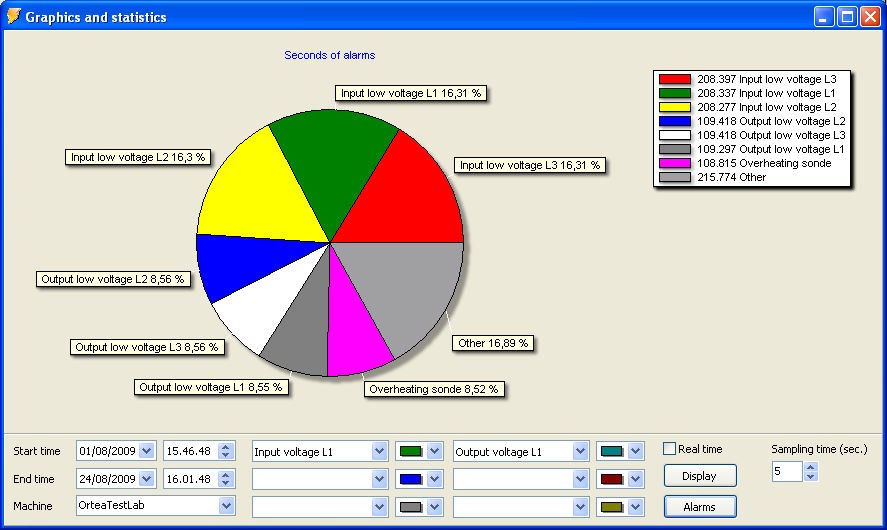


Displaying the information in graphical format requires the selection of:

* stabiliser
* period of time of interest
* parameters to be shown.

After the selection, press ‘display’ (allow for some time: if the amount of data is important, this time could extend to a few minutes).

If the flag ‘real time’ is selected, the graph is continuously updated with the data read in the sampling range specified. It is also possible to display (for the selected stabiliser and time period) a pie graph showing the alarm distribution for statistical reasons.



## 11.2 MODBUS® TCP/IP communication protocol

*WARNING.* The MODBUS® communication protocol is available if the unit is fitted with:

* Remote communication card software version not older than 01.05.00
* Control CPU software version not older than 04.00

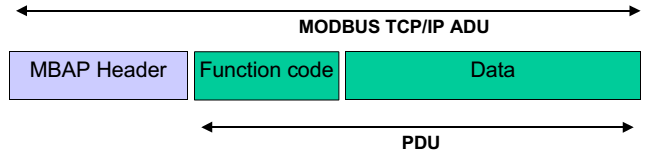
No interface is provided to manage the data communication. The basic MODBUS® server functions are:

* waiting for a MODBUS® request on 502 TCP port;
* treating this request;
* providing with a MODBUS® response on thebasis of the device context. The supported MODBUS® data is:
* Input register (read only information): word 16 bit size
* Holding register (read information): word 16 bit

The communication is based on a simple package called Protocol Data Unit (PDU). There are three types of PDU:

* *Request PDU*. consisting of a code specifying a function (Function Code, 1 byte) and function specific data (Function Data, varying number of bytes)
* *Response PDU*, consisting of:the function code corresponding to the request (Function Code, 1 byte) and response specific data (Response Data, varying number of bytes)
* *Exception Response PDU*, consisting of the function code corresponding to the request + 0x80 (128), (Error Code, 1 byte) and a code specifying the exception (Exception Code, 1 byte)

This package is enclosed in a message called ADU (Application Data Unit):



A MODBUS® TCP/IP ADU message is therefore a PDU package with a dedicated header called MBAP (MODBUS® Application Protocol header).

* + - 1. MODBUS®® TCP protocol

The communication message has the following structure:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Transaction ID  (16 bit) | Protocol ID  (16 bit) | Length  (16 bit) | Address  (8 bit) | Function  (8 bit) | Data  (N x 8 bit) |

1. *Transaction ID* it is an identifier replied from the server and it is used to trace the server response (it could be a random or progressive 16 bit value)

|  |  |  |
| --- | --- | --- |
| 2. | *Protocol ID* | fixed to 0 |
| 3. | *Length* | number of following bytes |
| 4. | *Address* | board adderss (1 to 254). Not used. |
| 5. | *Function* | code of the function that must be executed by the server. |
| 6. | *Data* | field containing the data sent to the server or received from the server in response to a query |

* + - 1. MODBUS® functions

The available functions are:

|  |  |
| --- | --- |
| 03h = Read holding register | Allows to read the board setup information (see table below) |
| 04h = Read input register | Allows to read board and measure instrument information |

For instance, in order to read the value of the Target voltage, which resides at board address 24 (18 Hex), the message is the following:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 00h | 00h | 00h | 00h | 00h | 06h | 01h | 04h | 00h | 18h | 00h | 01h |

where:

00 00 = Transaction ID

00 00 = Protocol ID

00 06 = Length

01 = server address (unused)

04 = MODBUS® function ‘Read input register’

00 18 = Address of the required register (Target voltage)

00 01 = Number of registers to be read starting from the required one The board answer is the following:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 00h | 00h | 00h | 00h | 00h | 05h | 01h | 04h | 02h | 00h | C8h |

where:

00 00 = Transaction ID (identical to the request Transaction ID)

00 00 = Protocol ID

00 05 = Length

01 = board address (unused; it is only an echo of the request) 04 = Function requested by the client

02 = Number of bytes of the response sent

00 C8 = Hex value of the Target voltage (200 V)

* + - 1. Function 03h : Read Holding Register

The MODBUS® function 03 allows for one or more consecutive setup information from the server memory to be read. Up to 9 consecutive addresses can be read. The address of each measure is listed in a tables further on.

If the measure address is not included in the table or the number of requested measures exceeds 9, the board will return an error code (see error table)

*client query example:*

|  |  |
| --- | --- |
| MSB Transaction ID | 00h |
| LSB Transaction ID | 00h |
| MSB Protocol ID | 00h |
| LSB Protocol ID | 00h |
| MSB Length | 00h |
| LSB Length | 06h |
| 1B server address | 01h |
| 1B Function | 03h |
| MSB address | 00h |
| LSB address | 00h |
| MSB register number | 00h |
| LSB register number | 08h |

In the above example, the server is requested for 8 consecutive registers beginning with address 00h. Registers from 0 to 7 will then be returned.

*server response:*

|  |  |
| --- | --- |
| MSB Transaction ID | 00h |
| LSB Transaction ID | 00h |
| MSB Protocol ID | 00h |
| LSB Protocol ID | 00h |
| MSB Length | 00h |
| LSB Length | 13h |
| 1B server address | 01h |
| 1B Function | 03h |
| 1B Byte number | 10h |
| MSB register 00h | 00h |
| LSB register 00h | 00h |
| ------------------------ | ---- |
| MSB register 07h | 00h |
| LSB register 07h | 00h |

The response always includes the server address, the function code requested by the client and the contents of the requested registers.

* + - 1. Function 04h : Read Input Register

The MODBUS® function 04 allows one or more consecutive measures from the server memory to be read. Up to 125 consecutive addresses can be read. The address of each measure is listed in a tables further on.

If the measure address is not included in the table or the number of requested measures exceeds 125, the board will return an error code (see error table)

*client query example:*

|  |  |
| --- | --- |
| MSB Transaction ID | 00h |
| LSB Transaction ID | 00h |
| MSB Protocol ID | 00h |
| LSB Protocol ID | 00h |
| MSB Length | 00h |
| LSB Length | 06h |
| server address | 01h |
| Function | 04h |
| MSB address | 00h |
| LSB address | 18h |
| MSB register number | 00h |
| LSB register number | 08h |

In the above example the server is requested for 8 consecutive registers beginning with address 24 (18h). Registers from 24 to 31 will then be returned.

*server response:*

|  |  |
| --- | --- |
| MSB Transaction ID | 00h |
| LSB Transaction ID | 00h |
| MSB Protocol ID | 00h |
| LSB Protocol ID | 00h |
| MSB Length | 00h |
| LSB Length | 13h |
| server address | 01h |
| Function | 04h |
| Byte number | 10h |
| MSB register 18h | 00h |
| LSB register 18h | 00h |
| --------------- | ------- |
| MSB register 1Fh | 00h |
| LSB register 1Fh | 00h |

The response always includes the server address, the function code requested by the client and the contents of the requested registers.

* + - 1. Error table

If an error occurs, the server responds with an error code. Below an example with the response to a read of multiple registers.

|  |  |
| --- | --- |
| MSB Transaction ID | 00h |
| LSB Transaction ID | 00h |
| MSB Protocol ID | 00h |
| LSB Protocol ID | 00h |
| MSB Length | 00h |
| LSB Length | 03h |
| 1B server address | 01h |
| 1B Function | 90h |
| 1B Exception Code | 02h |

The function is alway 80h + code of the function requested by the client (10h in this case). The following table shows the exception codes returned in case of invalid queries.

|  |  |
| --- | --- |
| 01h | Invalid function |
| 02h | Invalid register address |
| 03h | Parameter value out of bounds |

* + - 1. Limits:

This server accepts only 2 clients connection at the same time.

* + - 1. List of the exported informations

*READ: Holding registers (03H)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Add. | Words | Description | Format | Unit |
| 0 | 1 | Output Target | Int16 | Volt (ex. 230v) |
| 1 | 1 | Reset alarms command | Int16 | Any value to reset |
| 2 | 1 | Reset service LED command | Int16 | Any value to reset |
| 3 | 1 | Fan start-up temperature setup (step 1) | Int16 | C°/100 |
| 4 | 1 | Fan start-up temperature setup (step 2) | Int16 | (ex.4000 if 40°) |
| 5 | 1 | Fan start-up temperature setup (step 3) | Int16 | C°/100 |
| 6 | 1 | Fan start-up temperature setup (step 4) | Int16 | (ex.4000 if 40°) |
| 7 | 1 | Overheating alarm temperature setup | Int16 | C°/100 |
| 8 | 1 | Phase rotation setup | Int16 | 0/1 |

*READ : Input registers for information on the control board (04H)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Add. | Words | Description | Format | Unit |
| 0 | 15 | Serial number | (\*) String char [30] | Hex → Ascii |
| 15 | 5 | Current date | String char [10] – MM/DD/YY | Hex → Ascii |
| 20 | 4 | Current time | String char [8] – HH:MM:SS | Hex → Ascii |
| 24 | 1 | Target voltage | Int16 | Volt (ex.230) |
| 25 | 1 | Temperature | Int16 | C°/100 (ex.4000 if 40°) |
| 26 | 2 | Phase U motor service hours | (\*\*)Int32 | Sec. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Add. | Words | Description | Format | Unit |
| 28 | 2 | Phase V motor service hours | Int32 | Sec. |
| 30 | 2 | Phase W motor service hours | Int32 | Sec. |
| 32 | 2 | Phases service hours | Int32 | Sec. |
| 34 | 2 | Phase U motor service partial ours | (\*\*)Int32 | Sec. |
| 36 | 2 | Phase V motor service partial ours | Int32 | Sec. |
| 38 | 2 | Phase W motor service partial ours | Int32 | Sec. |
| 40 | 2 | Phases service partial hours | Int32 | Sec. |
| 42 | 2 | Alarms | (\*\*\*)Int32 – bit field |  |
| 44 | 1 | Phase U (ph-n) input voltage | Int16 | Volt (ex. 230V) |
| 45 | 1 | Phase V (ph-n) input voltage | Int16 | Volt (ex. 230V) |
| 46 | 1 | Phase W (ph-n) input voltage | Int16 | Volt (ex. 230V) |
| 47 | 1 | Phase U (ph-n) input current | Int16 | A (ex 35A) |
| 48 | 1 | Phase V (ph-n) input current | Int16 | A (ex 35A) |
| 49 | 1 | Phase W (ph-n) input current | Int16 | A (ex 35A) |
| 50 | 1 | Phase U (ph-n) output voltage | Int16 | Volt (ex. 230V) |
| 51 | 1 | Phase V (ph-n) output voltage | Int16 | Volt (ex. 230V) |
| 52 | 1 | Phase W (ph-n) output voltage | Int16 | Volt (ex. 230V) |
| 53 | 1 | Phase U (ph-n) output current | Int16 | A (ex 35A) |
| 54 | 1 | Phase V (ph-n) output current | Int16 | A (ex 35A) |
| 55 | 1 | Phase W (ph-n) output current | Int16 | A (ex 35A) |
| 56 | 1 | Phase U roller current | Int16 | A (ex 7A) |
| 57 | 1 | Phase V roller current | Int16 | A (ex 7A) |
| 58 | 1 | Phase W roller current | Int16 | A (ex 7A) |

(\*) String format: the string value is formatted with 2 chars for every word: the first into the high address and the second into the low. For example, the string “CATS” coded with 0x43 0x41 0x54 0x53 will be formatted into 2 words as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Word 1 | Add 0 – high value | ‘C’ | Hex 43 |
|  | Add 0 – low value | ‘A’ | Hex 41 |
| Word 2 | Add 1 – high value | ‘T’ | Hex 54 |
|  | Add 1 – low value | ‘S’ | Hex 53 |

(\*\*) Int32 format: double word formatted with 2 words, being the high value into the first word. For example, the double value 12345678 Hex will be formatted into 2 words as follows:

|  |  |
| --- | --- |
| Word 1 | Hex 1234 |
| Word 2 | Hex 5678 |

(\*\*\*) Int32 bit field special case. It is formatted as described in (\*\*), but the bit value is as follows:

|  |  |
| --- | --- |
| Bit0 | "MAX Current" |
| Bit1 | "By PASS” – Stabilisation OFF |
| Bit2 | "Overheating" |
| Bit3 | "Overheating roller" |
| Bit4 | "Phase U short curr." – Regulator Motor |
| Bit5 | "Phase U Vout min" |
| Bit6 | "Phase U Vout max" |
| Bit7 | "Phase U locked" – Regulator Motor |
| Bit8 | "Phase V short curr." – Regulator Motor |
| Bit9 | "Phase V Vout min" |
| Bit10 | "Phase V Vout max" |
| Bit11 | "Phase V locked" – Regulator Motor |
| Bit12 | "Phase W short curr." – Regulator Motor |
| Bit13 | "Phase W Vout min" |
| Bit14 | "Phase W Vout max" |
| Bit15 | "Phase W locked" – Regulator Motor |
| Bit16 | "Overheating internal” |
| Bit17 | “Phase U MAX curr.” |
| Bit18 | "Phase V MAX curr.” |
| Bit19 | "Phase W MAX curr.” |
| Bit20 | "CPU MSTE949" |
| Bit21 | "CPU BODY949" |
| Bit22 | "PHASE LOST" |

|  |  |
| --- | --- |
| Bit23 | "PHASE SEQUENCE" |
| Bit24 | "WRONG FACT.CALIB." |
| Bit25 | "WRONG USER CALIB." |
| Bit26 | "INPUT PHASE LOST" |
| Bit27..31 | Reserved for future use |

*READ : Input registers for information on the measure instrument (04H)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Address | Words | Description | Format | Unit |
| 100 | 2 | phase U (ph-n) input voltage | Int32 | V (ex.230v) |
| 102 | 2 | phase V (ph-n) input voltage | Int32 | V (ex.230v) |
| 104 | 2 | phase W (ph-n) input voltage | Int32 | V (ex.230v) |
| 106 | 2 | phase U (ph-n) input LOW voltage | Int32 | V (ex.230v) |
| 108 | 2 | phase V (ph-n) input LOW voltage | Int32 | V (ex.230v) |
| 110 | 2 | phase W (ph-n) input LOW voltage | Int32 | V (ex.230v) |
| 112 | 2 | phase U (ph-n) input HIGH voltage | Int32 | V (ex.230v) |
| 114 | 2 | phase V (ph-n) input HIGH voltage | Int32 | V (ex.230v) |
| 116 | 2 | phase W (ph-n) input HIGH voltage | Int32 | V (ex.230v) |
| 118 | 2 | phase U Input current | Int32 | A/100 (ex 3500 if 35A) |
| 120 | 2 | phase V Input current | Int32 | A/100 (ex 3500 if 35A) |
| 122 | 2 | phase W Input current | Int32 | A/100 (ex 3500 if 35A) |
| 124 | 2 | Input total active power | Int32 | W |
| 126 | 2 | Input total reactive power | Int32 | var |
| 128 | 2 | Input total apparent power | Int32 | VA |
| 130 | 2 | phase U (ph-n) output voltage | Int32 | V (ex.230v) |
| 132 | 2 | phase V (ph-n) output voltage | Int32 | V (ex.230v) |
| 134 | 2 | phase W (ph-n) output voltage | Int32 | V (ex.230v) |
| 136 | 2 | phase U (ph-n) output LOW voltage | Int32 | V (ex.230v) |
| 138 | 2 | phase V (ph-n) output LOW voltage | Int32 | V (ex.230v) |
| 140 | 2 | phase W (ph-n) output LOW voltage | Int32 | V (ex.230v) |
| 142 | 2 | phase U (ph-n) output HIGH voltage | Int32 | V (ex.230v) |
| 144 | 2 | phase V (ph-n) output HIGH voltage | Int32 | V (ex.230v) |
| 146 | 2 | phase W (ph-n) output HIGH voltage | Int32 | V (ex.230v) |
| 148 | 2 | phase U output current | Int32 | A/100 (ex 3500 if 35A) |
| 150 | 2 | phase V output current | Int32 | A/100 (ex 3500 if 35A) |
| 152 | 2 | phase W output current | Int32 | A/100 (ex 3500 if 35A) |
| 154 | 2 | phase U output high current | Int32 | A/100 (ex 3500 if 35A) |
| 156 | 2 | phase V output high current | Int32 | A/100 (ex 3500 if 35A) |
| 158 | 2 | phase W output high current | Int32 | A/100 (ex 3500 if 35A) |
| 160 | 2 | Output total active power | Int32 | W |
| 162 | 2 | Output total reactive power | Int32 | var |
| 164 | 2 | Output total apparent power | Int32 | VA |
| 166 | 2 | phase U output active power | Int32 | W |
| 168 | 2 | phase U output reactive power | Int32 | var |
| 170 | 2 | phase U output apparent power | Int32 | VA |
| 172 | 2 | phase V output active power | Int32 | W |
| 174 | 2 | phase V output reactive power | Int32 | var |
| 176 | 2 | phase V output apparent power | Int32 | VA |
| 178 | 2 | phase W output active power | Int32 | W |
| 180 | 2 | phase W output reactive power | Int32 | var |
| 182 | 2 | phase W output apparent power | Int32 | VA |
| 184 | 2 | phase U output power factor | Int32 | (ex. 99 if cos ɸ 0,99) |
| 186 | 2 | phase V output power factor | Int32 | (ex. 99 if cos ɸ 0,99) |
| 188 | 2 | phase W output power factor | Int32 | (ex. 99 if cos ɸ 0,99) |
| 190 | 2 | Output frequency | Int32 | Hz/10 (ex. 500 if 50 Hz) |