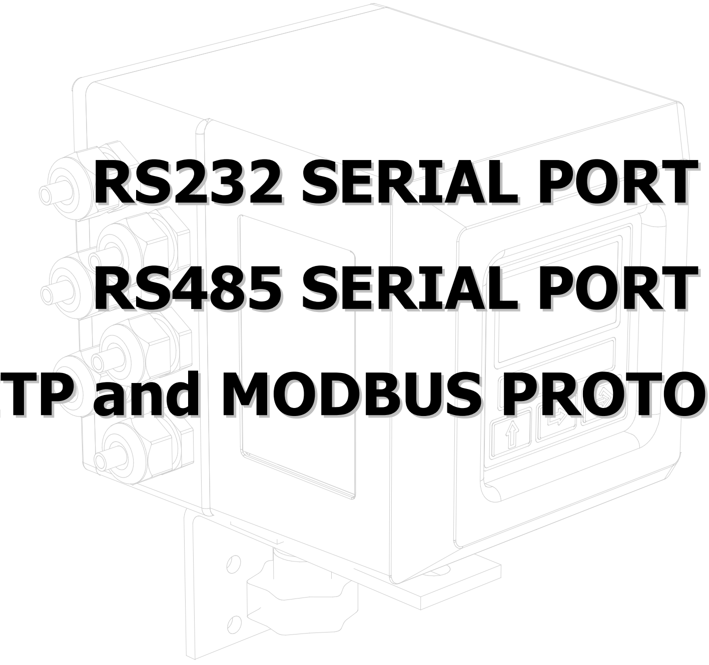
MAN 1000102637 EN Version: B Status: RL (released | freigegeben) printed: 10.01.2018

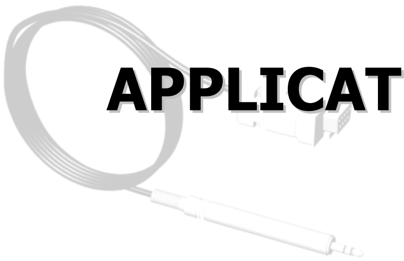
SE 56



RS232 SERIAL PORT

RS485 SERIAL PORT

ETP and MODBUS PROTOCOL



APPLICATION NOTE

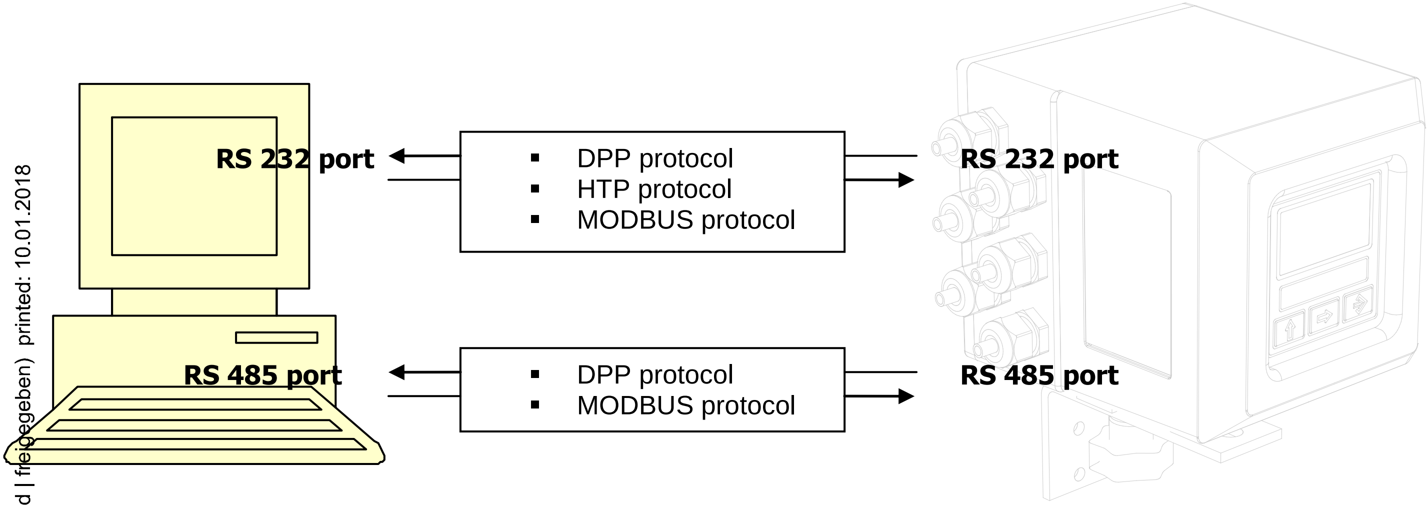
INDEX

MAN 1000102637 EN Version: B Status: RL (released | freigegeben) printed: 10.01.2018

1. HARDWARE CONNECTION WITH THE CONVERTER

See Converter's operation manual

2. AVAILABLE PROTOCOLS



- **DPP = Data Packet Protocol**

Proprietary of Millennium series : it's a network protocol based on data packets; over this protocol it's possible to encapsulate other protocols like BCP (**B**asic **C**ommunication **P**rotocol) or ETP (**E**ncapsulated **T**ransfer **P**rotocol)

- **HTP = Hyper text protocol**

It's a textual protocol used in point to point connections.
ETP commands can be used over HTP

- **MODBUS = field bus**

It's a standard field bus that can be used over point to point or over network connections;
MODBUS can encapsulate ETP commands

3. DPP (DATA PACKET PROTOCOL)

3.1. INTRODUCTION TO DATA PACKET PROTOCOL

With Data Packet Protocol it is possible to communicate with the flow meter through protocols with data packets.
The protocols with data packets available for the flow meter are BCP protocol (Basic Communication protocol) and the ETP protocol (Encapsulated Transfer protocol)

The flow meter have two serial port, the RS 232 port and the RS 485 port.

When the aux module with the RS 232 port is present in the flow meter, the menu «**7-Communication»** have the function «**RS232 pr.**» for the selection of the protocol type.
When the DPP setting is selected in the function «**RS232 pr.**» of the menu «**7-Communication»**, it is possible to send command to RS 232 port of the flow meter with the protocol BCP or with the protocol ETP.

When the flow meter have the option of the MODBUS protocol, the menu «**7-Communication»** have the function «**RS485 pr.**» for the selection of the protocol type.
The possible selections are MODBUS or DPP.
When the DPP setting is selected in the function «**RS485 pr.**» of the menu «**7-Communication»**, it is possible to send command to RS 485 port of the flow meter with the protocol BCP or with the protocol ETP.
If the flow meter don’t have the MODBUS option, the function «**RS485 pr.**» is not present in the menu «**7-Communication»** and the default protocol type for the RS 485 port is DPP.

For the use of the commands and the structure of the BCP protocol see the relative section.

For the use of the commands and the structure of the ETP protocol see the relative section.

3.2. BCP PROTOCOL (BASIC COMMUNICATION PROTOCOL)

3.2.1. INTRODUCTION

The BCP protocol is a set of formatted commands based on an index.

It is possible to read data from the flow meter as Process data or the Data Logger and it is possible to send command to the flow meter as the start stop batch for example.

The protocol is available for the RS 232 and for the RS 485 serial port.

3.2.2. DATA WORD FORMAT

The data bytes travelling in serial form on the communication line are enclosed in ***words*** which have a fixed length of 10 bits:

1 START BIT
8 DATA BITS = 1 BYTE DI DATI
1 STOP BIT

Each word contains one byte of data plus additional bits which serve to synchronise and make the communication safer. These extra bits are added automatically in the transmission phase by the transmitter integrated circuit. In the reception phase, the reverse operation is executed by the receiver integrated circuit: the eight data bits are extracted and the others are eliminated. These operations are executed entirely on a hardware level.
The 8 data bits must be serialised staring from bit 0 (the least significant one).

3.2.3. DATA BLOCK FORMAT

Communication takes place through data blocks or packs of variable length which do not exceed 256 bytes overall. A data block or pack is composed thus::

- **ADDRESS TO**, Containing the address of the device the block is sent to;
- **ADDRESS FROM,** Containing the address of the device which sent the block;
- **COMMAND**, Containing the command code the block refers to;
- **LENGTH**, Containing the length of the data block in bytes
- **DATA BYTES**, The block of data of variable length from zero to 250
- **CRC CHECKSUM BYTE**, Calculated in the following way:

1. CRC initialised to zero
2. CRC rotated to the left by one bit
3. CRC = CRC + block bytes

The process is repeated starting from point 2. for all the block bytes, excluding CRC..

NOTE: The CRC must be subjected to an operation of *rotation to the left* and not just a simple *shift* operation. The difference is shown in the following example:

```
byte value:              10100010
shifting :             1 <=   0100010   <=   0
 rotating:           ← 1 <=   0100010   <=   1 ←
```

A block can contain a maximum of 90 data bytes. A block which does not contain data bytes is formed solely by **ADDRESS TO** + **ADDRESS FROM** + **COMMAND** + **LENGTH** + **CRC CHECKSUM**, with **LENGTH** value equal to zero.

3.2.4. COMMUNICATION SPEED

The millennium series instruments have 4 communication speeds::

- 4800 bps
- 9600 bps
- 19200 bps
- 38400 bps

3.2.5. SERIAL PORT SETTINGS FOR RS 232 AND RS 485

Serial port settings:

- Data bits: 8
- Parity: none (no parity)
- Stop bits: 1
- Flow control: none (no control lines no xon/xoff characters used)

3.2.6. COMMAND FORMATS

A ***command*** is defined as an operating code of 1 byte specifying the type of operation required. The reply to this command is the return of one block with the requested data having the same command code + 128 (80H). The following command codes are available:

codice	Action performed by the measuring device after receiving the command.
0	Send the ***type of instrument*** and ***version*** of the software (all model)
1	Send the ***process data*** (for all models)
2	Send the ***data logger's data*** (ML210/211/212/ML110)
3	Receive ***date and time*** (ML210/211/212/ML110) ***reset the enabled totalizers*** (all model)
4,5,6,7,9, 10,13	RESERVED
8	***Send or receive data for batch*** (ML210 / ML3-F1)
11	Send events data(ML210/211/212/ML110)
12	Send ***min/max*** (ML210/211/212/ML110)
14	***Set set-point*** (ML212)

NOTE: the commands indicated with the term "RESERVED" are used during factory calibration and diagnosis, are protected by a special password and must not be used by the user under any circumstance.

In this description, the command codes in hexadecimal and (decimal) format and the related data format are shown. The numeration of the bits is from 0 (LSB) to 15 (MSB). The numeration of the bytes received is from 0 to N, where 0 is the first byte received and N is the last.

3.2.7. READING THE TYPE OF INSTRUMENT AND THE SOFTWARE VERSION

Command code: **0**
Command format: ADDRESS TO + ADDRESS FROM + **00H** + **00H** + CRC CHECKSUM
Reply data format: bytes 0..5: device name («ML 210»)
byte 6: software version (major number)
byte 7: software version (minor number)
bytes 8-9: (16 bits unsigned integer, byte 8=MSB):
hardware / software enabling flags, bit 0 = LSB, bit 15 = MSB:

bits	**Functions ML 210/211/3f1/110**	**Functions ML212**
0-2	current access level (0..7)	current access level (0..7)
3	channel 1 impulses in use	impulses for volume in use
4	channel 2 impulses in use	duty cycle actuator type in use
5	channel 1 frequency in use	frequency for flow rate in use
6	channel 2 frequency in use	frequency for actuator in use
7	scale range 2 in use	on/off type actuator in use
8	specific weight in use	specific weight in use
9	additional output 3 present	additional output 3 present
10	additional output 4 present	additional output 4 present
11	output 4..20 mA 2 present	output 4..20 mA 2 present
12	RS232 present	input on/off n.2 present
13	batching functions active	analog inputs 1 and 2 present
14	output 4..20 mA 2 present	output 4..20 mA 1 present
15	RS485 present	RS485 present

EXAMPLE: want to send the command 00H to a measuring device located at address 11H (17 decimal).The device that interrogates the network has the address FFH (255 decimal). The block to send will be composed thus

```
address of the measuring device to interrogate (ADDRESS TO)
|
|     address of the device that interrogates the network (ADDRESS FROM)
|     |
|     |       command for reading TYPE of  INSTRUMENT (COMMAND)
|     |       |
|     |       |      data block length  (LENGTH)
|     |       |       |
|     |       |       |        checksum of the entire block (CRC CHECKSUM)
|     |       |       |        |
11H  FFH     00H     00H     84H
```

The reply from the measuring device will be the following:

```
address of the device that interrogated the network and which now must receive the reply  (ADDRESS TO)
|
|         address of the measuring device which sent the message (ADDRESS FROM)
|         |
|         |       command for reading the TYPE of INSTRUMENT + 80H (128 dec.) (COMMAND)
|         |       |
|         |       |       length of the data block (=10 bytes) (LENGTH)
|         |       |       |
|         |       |       |       type of instrument («ML 200»)
|         |       |       |       |
|         |       |       |       |                             versione software (major.minor)
|         |       |       |       |                             |
|         |       |       |       |                             |           hardware / software enabling flags
|         |       |       |       |                             |           |
|         |       |       |       |                             |           |              CRC CHECKSUM)
|         |       |       |       |                             |           |              |
FFH       11H     80H     0AH     4D 4C 20 32 30 30H       01 02H      C0 08H         21H
```

The measuring device has replied that it is an ML200, that its software version is the 1.02, that it has enabled the RS485 serial port, that the output is 4..20mA and that it is using impulse emission on channel 1 (this last information is taken from the hardware / software enabling flags).

3.2.8. READING PROCESS DATA FROM ML210 / 110

The process data is contained in a memory block which the user can request whole or in part, depending on the content of the two command control bytes. Byte 0 (offset) indicates which will be the first data byte of the block to send, whilst byte 1 specifies the number of successive data bytes we want. In this way we can obtain exactly the data we want or several pieces of data at the same time.

Command code: **01**
Command format: ADDRESS TO
+ ADDRESS FROM
+ **01H**
+ **02H**
+ byte 0: offset – start of block to transmit
+ byte 1: length of block to transmit
+ CRC CHECKSUM

Reply data format when the sending of the entire block is requested (byte 0 = **0** and byte 1 = **46**):

bytes 0-3: (32 bit single precision IEEE floating point, MSB first) flow rate in %
bytes 4-7: (32 bit single precision IEEE floating point, MSB first) flow rate scale range in t.u.
bytes 8-11: (32 bit single precision IEEE floating point, MSB first) flow rate in t.u.
bytes 12-16: (5 bytes ASCII) flow rate measurement unit
bytes 17-19: (3 bytes ASCII) measurement unit of the counters
byte 20: (8 bits integer) number of decimals after the point for totalizers display
byte 21: (8 bits integer) number of decimals after the point for flow rate display
bytes 22-25: (32 bit long integer, MSB first) totalizer for TOTAL volume +
bytes 26-29: (32 bit long integer, MSB first) totalizer for PARTIAL volume -
bytes 30-33: (32 bit long integer, MSB first) totalizer for TOTAL volume - (or ML210 dosage quantity)
bytes 34-37: (32 bit long integer, MSB first) totalizer for PARTIAL volume - (or ML210 dosed quantity)
bytes 38-41: (32 bit long integer, MSB first) clock expressed in minutes starting from 01-01-1992
bytes 42-43: (16 bit unsigned integer, MSB first) process flags:

bit 0 =1 if the excitation is too fast for the sensor connected
bit 1 =1 if the maximum alarm is active
bit 2 =1 if the minimum alarm is active
bit 3 =1 if the flow rate exceeds the scale range value (overflow)
bit 4 =1 if one or more output impulses are saturated (too many impulses to emit)
bit 5 =1 if the measurement signal is highly disturbed or if the sensor is disconnected
bit 6 =1 if the measurement tube is empty
bit 7 =1 if the circuit powering the coils is not working or the sensor is disconnected
bit 8 =1 if the second measurement scale is active
bit 9 =1 if the flow rate is lower than the cut-off threshold
bit10=1 if the flow rate is negative
bit11=1 if a new measurement value calculated for the display is available
bit12=1 if the counter block signal is active
bit13=1 if dosing is in progress (only for ML210)
bit14=1 if a calibration cycle is in progress
bit15=1 if a flow rate simulation is in progress

byte 44: (8 bits integer) measurement samples per second (Hz)
byte 45: (8 bits integer) measurement dynamic variation as a %

To request the transmission of a single parameter, assign the values of byte 0 and byte 1 in the request as follows, bearing in mind that the data in the memory block has fixed positions:

PARAMETER TO READ	Byte 0	Byte 1
Flow rate in %	0	4
Scale range of the flow rate in technical units	4	4
Flow rate in technical units	8	4
Flow rate measurement unit	12	5
Totalizers measurement unit	17	3
Decimal figures for the totalizers	20	1
Decimal figures for flow rate display	21	1
Totalizer for TOTAL volume +	22	4
Totalizer for PARTIAL volume +	26	4
Totalizer for TOTAL volume - (or dosage quantity)	30	4
Totalizer for PARTIAL volume -(or dosed quantity)	34	4
Date and time in minutes	38	4
Process flags	42	2
Measurement samples per second in Hz	44	1
Measurement dynamic variation in %	45	1

NOTES: The values in the "32 bits single precision IEEE floating point" format are floating point numbers which can be represented during writing by any decimal digits. To keep the same numerical format visible on the instrument display however, it is necessary to calculate the decimal figures with a rather complex algorithm which takes account of instrument precision, flow rate measurement unit, etc. For this purpose and to avoid useless calculations, the number of decimals to use to represent the flow rate values is supplied separately (byte position 21).

The counters are expressed with a 32 bit integer. The «counter decimal figures» parameter, indicates the point position starting from the right: 0 = no decimal, 1=1 decimal figure, and so on.
The date and time are expressed with a 32 bit integer containing the number of minutes elapsed since 01-01-1992. To calculate the date starting from this number, see the programming examples at the end of this manual.

3.2.9. READING PROCESS DATA FROM ML211

This command is similar to the preceding one and differs from it only for the quantity of the data given.

Command code: **01**
Command format: ADDRESS TO
+ ADDRESS FROM
+ **01H**
+ **02H**
+ byte 0: offset – start of block to transmit
+ byte 1: length of block to transmit
+ CRC CHECKSUM

Reply data format when the sending of the entire block is requested (byte 0 = **0** and byte 1 = **78**):

bytes 0-3: (32 bit single precision IEEE floating point, MSB first) flow rate in %
bytes 4-7: (32 bit single precision IEEE floating point, MSB first) flow rate scale range in t.u.
bytes 8-11: (32 bit single precision IEEE floating point, MSB first) flow rate in t.u.
bytes 12-16: (5 bytes ASCII) flow rate measurement unit
bytes 17-19: (3 bytes ASCII) measurement unit of the counters
byte 20: (8 bits integer) number of decimals after the point for volume totalizers display
byte 21: (8 bits integer) number of decimals after the point for flow rate display
bytes 22-25: (32 bits long integer, MSB first) totalizer for volume +
bytes 26-29: (32 bits long integer, MSB first) totalizer for volume -
bytes 30-33: (32 bits long integer, MSB first) totalizer for energy +
bytes 34-37: (32 bits long integer, MSB first) totalizer for energy -
bytes 38-41: (32 bit long integer, MSB first) clock expressed in minutes starting from 01-01-1992
bytes 42-44: 3 x (8 bits unsigned char) 3 bytes of process flags:

byte 0:

bit 0 =0 (not used)
bit 1 =1 if the flow rate is lower than the cut-off threshold
bit 2 =1 if the flow rate is negative
bit 3 =1 if a new measurement value calculated for the display is available
bit 4 =1 if the counter block signal is active
bit 5 =0 (non used)
bit 6 =1 if a calibration cycle is in progress
bit 7 =1 if a flow rate simulation is in progress

byte 1:

bit 0 =1 if the temperature sensors are disconnected or broken
bit 1 =1 if the maximum alarm for the flow rate is active
bit 2 =1 if the minimum alarm for the flow rate is active
bit 3 =1 if a measure exceeds the scale range value (overflow)
bit 4 =1 if one or more output impulses are saturated (too many impulses to emit)
bit 5 =1 if the measurement signal is highly disturbed or if the sensor is disconnected
bit 6 =1 if the measurement tube is empty
bit 7 =1 if the circuit powering the coils is not working or the sensor is disconnected

byte 2:

bit 0 =1 if the maximum alarm for the thermal power is active
bit 1 =1 if the minimum alarm for the thermal power is active
bit 2 =1 if the maximum alarm for the delta T is active
bit 3 =1 if the minimum alarm for the delta T is active
bit 4 =1 if the maximum alarm for the temperature T1 is active
bit 5 =1 if the minimum alarm for the temperature T1 is active
bit 6 =1 if the maximum alarm for the temperature T2 is active
bit 7 =1 if the minimum alarm for the temperature T2 is active

byte 45: (8 bits integer) number of measure samples per second (hertz)
bytes 46-49: (32 bits single precision IEEE floating point, MSB first) thermal power in %
bytes 50-53: (32 bits single precision IEEE floating point, MSB first) full scale thermal power in t.u.
bytes 54-57: (32 bits single precision IEEE floating point, MSB first) thermal power in t.u.
bytes 58-62: (5 bytes ASCII) measure unit for the thermal power
bytes 63-66: (4 bytes ASCII) measure unit for the thermal energy
byte 67: (8 bits integer) number of decimals after the point for energy totalizers display
byte 68: (8 bits integer) number of decimals after the point for thermal power display
byte 69: (8 bits integer) measure unit for the temperature (C or F degrees)
bytes 70-73: (32 bits single precision IEEE floating point, MSB first) delta T value in t.u.
bytes 74-77: (32 bits single precision IEEE floating point, MSB first) T1 temperature in t.u.
bytes 78-81: (32 bits single precision IEEE floating point, MSB first) T2 temperature in t.u.

To request the transmission of a single parameter, assign the values of byte 0 and byte 1 in the request as follows, bearing in mind that the data in the memory block has fixed positions:

PARAMETRO DA LEGGERE	Byte 0	Byte 1
Flow rate in %	0	4
Full scale flow rate	4	4
Flow rate in technical units	8	4
Measure unit for the flow rate	12	5
Measure unit for the volume totalizers	17	3
Number of decimal digit after the point for volume totalizers	20	1
Number of decimal digit after the point for the flow rate	21	1
Totalizer for volume +	22	4
Totalizer for volume -	26	4
Totalizer for thermal energy +	30	4
Totalizer for thermal energy -	34	4
Time and date expressed in minutes	38	4
Process flags	42	3
Number of samples per second in Hz	45	1
Thermal power in %	46	4
Full scale thermal power in t.u.	50	4
Thermal power in t.u.	54	4
Measure unit for the thermal power	58	5
Measure unit for the thermal energy	60	3
Number of decimal digit after the point for energy totalizers	63	1
Number of decimal digit after the point for the th. power	64	1
T in technical units	65	1
Delta T in t.u.	66	4
T1 value in t.u.	70	4
T2 value t.u.	74	4

NOTE: For the numeric representations and the number of decimal digits, see the preceding point.

3.2.10. READING PROCESS DATA FROM ML212

This command is similar to the preceding one and differs from it only for the quantity of the data given.

Command code: **01**
Command format: ADDRESS TO
+ ADDRESS FROM
+ **01H**
+ **02H**
+ byte 0: offset – start of block to transmit
+ byte 1: length of block to transmit
+ CRC CHECKSUM

Reply data format when the sending of the entire block is requested (byte 0 = **0** and byte 1 = **59**):

bytes 0-3: (32 bit single precision IEEE floating point, MSB first) flow rate in %
bytes 4-7: (32 bit single precision IEEE floating point, MSB first) flow rate scale range in t.u.
bytes 8-11: (32 bit single precision IEEE floating point, MSB first) flow rate in t.u.
bytes 12-16: (5 bytes ASCII) flow rate measurement unit
bytes 17-19: (3 bytes ASCII) measurement unit of the counters
byte 20: (8 bits integer) number of decimals after the point for volume totalizers display
byte 21: (8 bits integer) number of decimals after the point for flow rate display
bytes 22-25: (32 bits long integer, MSB first) totalizer for volume +
bytes 26-29: (32 bits long integer, MSB first) totalizer for volume -
bytes 30-33: (32 bits long integer, MSB first) totalizer for actuator closing pulses
bytes 34-37: (32 bits long integer, MSB first) totalizer for actuator opening pulses
bytes 38-41: (32 bit long integer, MSB first) clock expressed in minutes starting from 01-01-1992
bytes 42-43: (16 bit unsigned integer, MSB first) process flags:

bit 0 =1 if the deviation alarm is active
bit 1 =1 if the maximum alarm is active
bit 2 =1 if the minimum alarm is active
bit 3 =1 if the flow rate exceeds the scale range value (overflow)
bit 4 =1 if one or more output impulses are saturated (too many impulses to emit)
bit 5 =1 if the measurement signal is highly disturbed or if the sensor is disconnected
bit 6 =1 if the measurement tube is empty
bit 7 =1 if the circuit powering the coils is not working or the sensor is disconnected
bit 8 =1 if the measure sample rate is too high for the sensor
bit 9 =1 if the flow rate is lower than the cut-off threshold
bit10=1 if the flow rate is negative
bit11=1 if a new measurement value calculated for the display is available
bit12=1 if the counter block signal is active
bit13=1 if the actuator positioning alarm is active
bit14=1 if a calibration cycle is in progress
bit15=1 if a flow rate simulation is in progress

byte 44: (8 bits integer) measurement samples per second (Hz)
byte 45: (8 bits integer) measurement dynamic variation as a %
bytes 46-49: (32 bits single precision IEEE floating point, MSB first) set-point value in %
bytes 50-53: (32 bits single precision IEEE floating point, MSB first) regulator output value in %
bytes 54-57: (32 bits single precision IEEE floating point, MSB first) deviation value in %
byte 58: (1 byte ASCII) regulator status flags:

bit 0 =1 if the regulator is in "manual" mode
bit 1 =1 if the regulator output is inverted
bit 2 =1 if the regulator output is in "safety" state
bit 3 =1 if the deviation alarm is active
bit 4 =1 if the actuator alarm is active

bit 5 =1 if the measure at the AIN1 input is out of range
bit 6 =1 if the measure at the AIN1 input is out of range

To request the transmission of a single parameter, assign the values of byte 0 and byte 1 in the request as follows, bearing in mind that the data in the memory block has fixed positions:

PARAMETER TO READ	Byte 0	Byte 1
Flow rate in %	0	4
Full scale flow rate	4	4
Flow rate in technical units	8	4
Measure unit for the flow rate	12	5
Measure unit for the volume totalizers	17	3
Number of decimal digit after the point for volume totalizers	20	1
Number of decimal digit after the point for the flow rate	21	1
Totalizer for volume +	22	4
Totalizer for volume -	26	4
Totalizer for actuator closing impulses	30	4
Totalizer for actuator opening impulses	34	4
Date and time in minutes	38	4
Process flags	42	2
Measurement samples per second in Hz	44	1
Measurement dynamic variation in %	45	1
Set-point value in %	46	4
Regulator output value in %	50	4
Deviation value in %	54	4
Regulator status flags	58	1

NOTE: For the numeric representations and the number of decimal digits, see the preceding point.

3.2.11. READING PROCESS DATA FROM ML3F1

The process data is contained in a memory block which the user can request whole or in part, depending on the content of the two command control bytes. Byte 0 (offset) indicates which will be the first data byte of the block to send, whilst byte 1 specifies the number of successive data bytes we want. In this way we can obtain exactly the data we want or several pieces of data at the same time..

Command code: **01**
Command format: ADDRESS TO
+ ADDRESS FROM
+ **01H**
+ **02H**
+ byte 0: offset – start of block to transmit
+ byte 1: length of block to transmit
+ CRC CHECKSUM

Reply data format when the sending of the entire block is requested (byte 0 = **0** and byte 1 = **46**):

bytes 0-3: (32 bit single precision IEEE floating point, MSB first) flow rate in %
bytes 4-7: (32 bit single precision IEEE floating point, MSB first) flow rate scale range in t.u.
bytes 8-11: (32 bit single precision IEEE floating point, MSB first) flow rate in t.u.
bytes 12-16: (5 bytes ASCII) flow rate measurement unit
bytes 17-19: (3 bytes ASCII) measurement unit of the counters
byte 20: (8 bits integer) number of decimals after the point for totalizers display
byte 21: (8 bits integer) number of decimals after the point for flow rate display
bytes 22-25: (32 bit long integer, MSB first) totalizer 1
bytes 26-29: (32 bit long integer, MSB first) totalizer 2
bytes 30-33: (32 bit long integer, MSB first) everytime to 0
bytes 34-37: (32 bit long integer, MSB first) everytime to 0

bytes 38-41: (32 bit long integer, MSB first) everytime to 0
bytes 42-43: (16 bit unsigned integer, MSB first) process flags

bit 0 =1 if the excitation is too fast for the sensor connected
bit 1 =1 if the maximum alarm is active
bit 2 =1 if the minimum alarm is active
bit 3 =1 if the flow rate exceeds the scale range value (overflow)
bit 4 =1 if one or more output impulses are saturated (too many impulses to emit)
bit 5 =1 if the measurement signal is highly disturbed or if the sensor is disconnected
bit 6 =1 if the measurement tube is empty
bit 7 =1 if the circuit powering the coils is not working or the sensor is disconnected

bit 8 =1 if the second measurement scale is active
bit 9 =1 if the flow rate is lower than the cut-off threshold
bit10=1 if the flow rate is negative
bit11=1 if a new measurement value calculated for the display is available
bit12=1 if the counter block signal is active
bit13=1 if dosing is in progress
bit14=1 if a calibration cycle is in progress
bit15=1 if a flow rate simulation is in progress

byte 44: (8 bits integer) measurement samples per second (Hz)
byte 45: (8 bits integer) measurement dynamic variation as a %

To request the transmission of a single parameter, assign the values of byte 0 and byte 1 in the request as follows, bearing in mind that the data in the memory block has fixed positions:

PARAMETRO DA LEGGERE	Byte 0	Byte 1
Flow rate in %	0	4
Scale range of the flow rate in technical units	4	4
Flow rate in technical units	8	4
Flow rate measurement unit	12	5
Totalizers measurement unit	17	3
Decimal figures for the totalizers	20	1
Decimal figures for flow rate display	21	1
Totalizer for T1	22	4
Totalizer for T2	26	4
Process flags	42	2
Measurement samples per second in Hz	44	1
Measurement dynamic variation in %	45	1

NOTES: The values in the "32 bits single precision IEEE floating point" format are floating point numbers which can be represented during writing by any decimal digits. To keep the same numerical format visible on the instrument display however, it is necessary to calculate the decimal figures with a rather complex algorithm which takes account of instrument precision, flow rate measurement unit, etc. For this purpose and to avoid useless calculations, the number of decimals to use to represent the flow rate values is supplied separately (byte position 21).
The counters are expressed with a 32 bit integer. The «counter decimal figures» parameter, indicates the point position starting from the right: 0 = no decimal, 1=1 decimal figure, and so on.
The date and time are expressed with a 32 bit integer containing the number of minutes elapsed since 01-01-1992. To calculate the date starting from this number, see the programming examples at the end of this manual.

3.2.12. SETTING THE SET-POINT OF THE ML212

With this command it is possible to send the set-point value in % format to the ML202. If the value is positive, this is intended as REMOTE set-point value and must be refreshed continuously. The maximum time between two refreshes it is regulated by the same timer used to signal the deviation alarm condition and thus it is freely changeable. If the value sent is negative, it is considered as LOCAL set-point value and in this case it is not necessary to refresh it continuously. In any cases the regulator receives the absolute value of the number sent.

Command code: **14**
Command format: ADDRESS TO
+ ADDRESS FROM
+ **0EH**
+ **04H**
+ 32 bits IEEE floating point containing the set-point value in %
+ CRC CHECKSUM

The same number is returned as answer by the ML212

3.2.13. READING THE INTERNAL DATA-LOGGER DATA OF ML210 / 211 / 212 / 110

The data logger can be read one record at a time with this command. If we assign the value AAH to the byte specifying the address (byte 0), the data logger is cancelled. If we request the address of a record not present in the memory, the returned data has no sense. For this purpose we recommend starting from the request for record 0 and to check the number of records present in the memory which is returned by the measuring device.

Command code : **02**
Command format: ADDRESS TO
+ ADDRESS FROM
+ **02H**
+ **01H**
+ byte 0: index of the data logger data to read, AAH to cancel the entire content
+ CRC CHECKSUM

Reply data format in the case where there are registrations present:

byte 0: (8 bits integer) number of record requested
byte 1: (8 bits integer) total number of records present in the memory
bytes 2-5: (32 bits long integer, MSB first) data saving time and date expressed in minutes starting from 01-01-1992
bytes 6-9: (32 bits long integer, MSB first) data counted +
bytes 10-13: (32 bits long integer, MSB first) data counted -
bytes 14-17: (32 bits single precision IEEE floating point, MSB first) portata in u.t.
bytes 18-20: (3 bytes ASCII) (3 bytes ASCII) counter measurement unit
byte 21: (8 bits integer) number of decimal figures after the point for counter display
bytes 22-26: (5 bytes ASCII) flow rate t.u.
byte 27: (8 bits integer) number of decimal figures after the point for flow rate

Reply data format in the case where registrations are NOT present or the data logger is disabled

byte 0: (8 bits integer) requested record number
byte 1: (8 bits integer) total number of records present in the memory

Reply data format in the case where the AAH cancellation code is sent:

byte 0: (8 bits integer) AAH code.

NOTE: The time and date of sample collection is expressed in minutes starting from 01/01/1992. For conversion, see the programming examples at the end of this manual.

3.2.14. READING THE EVENTS ON INTERNAL DATA-LOGGER DATA OF ML210 / 211 / 212 / 110

The data logger can be read one record at a time with this command. If we assign the value AAH to the byte specifying the address (byte 0), the data logger is cancelled. If we request the address of a record not present in the memory, the returned data has no sense. For this purpose we recommend starting from the request for record 0 and to check the number of records present in the memory which is returned by the measuring device..

Command code : **11**
Command format: ADDRESS TO
+ ADDRESS FROM
+ **0BH**
+ **01H**
+ byte 0: index of the data logger data to read, AAH to cancel the entire content
+ CRC CHECKSUM

Reply data format in the case where there are registrations present:

byte 0: (8 bits integer) number of record requested
byte 1: (8 bits integer) total number of records present in the memory
bytes 2-5: (32 bits long integer, MSB first) data saving time and date expressed in minutes starting from 01-01-1992
bytes 6-9: (32 bits long integer, MSB first) flags events

bit 00: error sensor RTD / batch alarm
bit 01: alarm max flow
bit 02: alarm min flow
bit 03: alarm overflow
bit 04: alarm overflow pulse
bit 05: input error
bit 06: pipe empty
bit 07: coils interrupt
bit 08: alarm max t. power
bit 09: alarm min t. power
bit 10: alarm max delta T
bit 11: alarm min delta T
bit 12: alarm max T1
bit 13: alarm min T1
bit 14: alarm max T2
bit 15: alarm min T2
bit 16: current loop open
bit 17: power supply error

Code 000300FFH or 0003FFFFH = means converter switch on

Reply data format in the case where registrations are NOT present or the data logger is disabled

byte 0: (8 bits integer) requested record number
byte 1: (8 bits integer) total number of records present in the memory

Reply data format in the case where the AAH cancellation code is sent:

byte 0: (8 bits integer) AAH code.

NOTE: The time and date of sample collection is expressed in minutes starting from 01/01/1992. For conversion, see the programming examples at the end of this manual.

3.2.15. READING THE MAX-MIN ON INTERNAL DATA-LOGGER DATA OF ML210 / 110

The data logger can be read one record at a time with this command. If we assign the value AAH to the byte specifying the address (byte 0), the data logger is cancelled. If we request the address of a record not present in the memory, the returned data has no sense. For this purpose we recommend starting from the request for record 0 and to check the number of records present in the memory which is returned by the measuring device..

Command code: **12**
Command format: ADDRESS TO
+ ADDRESS FROM
+ **0CH**
+ **01H**
+ byte 0: index of the data logger data to read, AAH to cancel the entire content
+ CRC CHECKSUM

Reply data format in the case where there are registrations present:

bytes 0-3:(32 bits IEEE floating point, MSB first) MAX FLOW RATE IN T.U.
bytes 4-7:(32 bits IEEE floating point, MSB first) MIN FLOW RATE IN T.U.

3.2.16. LETTURA VALORI MIN-MAX DEL LOGGER INTERNO DI ML211

The data logger can be read one record at a time with this command. If we assign the value AAH to the byte specifying the address (byte 0), the data logger is cancelled. If we request the address of a record not present in the memory, the returned data has no sense. For this purpose we recommend starting from the request for record 0 and to check the number of records present in the memory which is returned by the measuring device..

Command code: **12**
Command format: ADDRESS TO
+ ADDRESS FROM
+ **0CH**
+ **01H**
+ byte 0: index of the data logger data to read, AAH to cancel the entire content
+ CRC CHECKSUM

Reply data format in the case where there are registrations present :

bytes 0-3:(32 bits IEEE floating point, MSB first) MAX FLOW RATE IN T.U.
bytes 4-7:(32 bits IEEE floating point, MSB first) MIN FLOW RATE IN T.U.
bytes 8-11:(32 bits IEEE floating point, MSB first) MAX T. POWER IN T.U.
bytes 12-15:(32 bits IEEE floating point, MSB first) MIN T. POWER IN T.U.
bytes 16-19:(32 bits IEEE floating point, MSB first) deltaT MAX IN T.U.
bytes 20-23:(32 bits IEEE floating point, MSB first) deltaT MIN IN T.U.
bytes 24-27:(32 bits IEEE floating point, MSB first) T1 MAX IN T.U.
bytes 28-31:(32 bits IEEE floating point, MSB first) T1 MIN IN T.U.
bytes 32-35:(32 bits IEEE floating point, MSB first) T1 MAX IN T.U.
bytes 36-39:(32 bits IEEE floating point, MSB first) T1 MIN IN T.U.

3.2.17. WRITING THE DATE AND TIME OR RESET THE ENABLED TOTALIZERS

Codice comando: **03**
Formato comando: ADDRESS TO
+ ADDRESS FROM
+ **03H**
+ **04H**
+ 32 bits long integer containing the date and time expressed in minutes starting from 01-01-1992, or the value **FFFFFFFFH** to **reset the totalizers**
the most significant byte must be sent first.
+ CRC CHECKSUM

Reply data format: the same block is returned containing the current clock value in minutes. If the value sent is outside the limits, (date beyond 31-12-2091), the clock restarts from 01-01-1992. In case of totalizers reset, the same value FFFFFFFFH is returned.

3.2.18. SEND / RECEVE BATCH PROCESS DATA ML210 /ML 3F1

The batch data can be read or written with the command described below. Th command contains an OPCODE byte specifying the operation type to be done on the selected batch data. With these commands one of the 16 batch memories can be written, read or it can be set as active batch process. Once set a selected batch memory, it will be possible start, stop or reset the batch process. To set a batch memory as active batch process, a write or read command can be used indifferently, but the bit 6 of the OPCODE must be set to 1

– Command for reading a batch data memory:

Command code: **08**
Command format: ADDRESS TO
+ ADDRESS FROM
+ **08H**
+ **01H** (1 DECIMALE)
+ OPCODE byte, operation code:
bits 0..4 = number of the batch memory to be read (0..15)
bit 5 = must be set to **zero**
bit 6 = **1** if this batch memory must be set as active batch process, **0** if this batch memory is to be read only.
bit 7 = must be set to **zero**
+ CRC CHECKSUM

Format of the returned data: 16 bytes containing the batch data for the selected memory:

bytes 0-7: (8 bytes ASCII) memory batch name (allowed characters: 0..9, a..z, A..Z, space (32 DEC, 20 HEX))
bytes 8-9: (2 bytes 16 bit unsigned integer, MSB first) number of batch processes done for this batch memory.
bytes 10-11: (2 bytes 16 bit unsigned integer, MSB first) value of the safety batch timer expressed in tenths of seconds.
bytes 12-15: (4 bytes 32 bits unsigned long integer, MSB first) batch quantity value expressed in the same units and with the same decimal digits of the volume counters.

– Comando per scrivere i dati su una memoria di dosaggio:

Command code: **08**
Command format: ADDRESS TO
+ ADDRESS FROM
+ **08H**
+ **11H** (17 DECIMAL)
+ OPCODE byte, operation code:
bits 0..4 = number of the batch memory to be written (0..15)
bit 5 = must be set to **one**
bit 6 = **1** if this batch memory must be set as active batch process, **0** if this batch memory is to be written only.
bit 7 = must be set to **zero**
+ 16 bytes of data to be written on the batch memory:
bytes 0-7: (8 bytes ASCII) batch memory name (allowed characters: 0..9, a..z, A..Z, space (32 DEC, 20 HEX))
bytes 8-9: (2 bytes 16 bit unsigned integer, MSB first) number of batch processes done for this batch memory.
bytes 10-11: (2 bytes 16 bit unsigned integer, MSB first) value of the batch safety timer expressed in tenths of seconds.
bytes 12-15: (4 bytes 32 bits unsigned long integer, MSB first) batch quantity value expressed in the same units and with the same decimal digits of the volume counters.
+ CRC CHECKSUM

Format of the returned data: 16 bytes containing the same batch data for the selected memory sent (if there were any characters or values not allowed, they are corrected to the right values). The data format are the same of the precedent function.

c) - Command for reading the current batch process state:

Command code: **08**
Command format: ADDRESS TO
+ ADDRESS FROM
+ **08H**
+ **01H** (1 DECIMAL)
+ OPCODE byte = **80H** (128 DECIMAL)
+ CRC CHECKSUM

Format of the returned data: 1 byte containing the batch process state:
0 = batch process is correctly terminated (pre-set quantity reached).
1 = batch process is running (the valve is opened and the counters are running).
2 = batch process is suspended (the valve is closed before the pre-set quantity is reached).

d) - Command for starting or suspending the current batch process:

Command code: **08**
Command format: ADDRESS TO
+ ADDRESS FROM
+ **08H**
+ **01H** (1 DECIMAL)
+ OPCODE byte = **81H** (129 DECIMAL)
+ CRC CHECKSUM

Format of the returned data: 1 byte containing the batch process state as described above.

e) - Command for resetting the current batch process:

Command code: **08**
Command format: ADDRESS TO
+ ADDRESS FROM
+ **08H**
+ **01H** (1 DECIMAL)
+ OPCODE byte = **82H** (130 DECIMAL)
+ CRC CHECKSUM

Format of the returned data: 1 byte containing the batch process state as described above. This command resets the dosing quantity and the safety timer counters.

3.3. ETP PROTOCOL (ENCAPSULATED TRANSFER PROTOCOL)

3.3.1. INTRODUCTION

The ETP is a protocol owner of the converter that has the function of the read and set the parameters of the converter through formatted strings of characters.

3.3.3. DATA WORD FORMAT

The data bytes travelling in serial form on the communication line are enclosed in ***words*** which have a fixed length of 10 bits:

1 START BIT
8 DATA BITS = 1 BYTE DI DATI
1 STOP BIT

Each word contains one byte of data plus additional bits which serve to synchronise and make the communication safer. These extra bits are added automatically in the transmission phase by the transmitter integrated circuit. In the reception phase, the reverse operation is executed by the receiver integrated circuit: the eight data bits are extracted and the others are eliminated. These operations are executed entirely on a hardware level.
The 8 data bits must be serialised staring from bit 0 (the least significant one).

3.3.6. COMMUNICATION SPEED

The millennium series instruments have 4 communication speeds::

- 4800 bps
- 9600 bps
- 19200 bps
- 38400 bps

3.3.10. SERIAL PORT SETTINGS FOR RS 232 AND RS 485 PORT

Serial port settings:

- Data bits: 8
- Parity: none (no parity)
- Stop bits: 1
- Flow control: none (no control lines no xon/xoff characters used)

3.3.11. GENERAL INPUT SYSNTAX

The information are entered as text line strings, with one or more command-sequences terminated by the <CR> character. The command-sequences are executed in the same order as they are found in the string. The execution of the commands contained in the input string does not start until the <CR> character is received. The optional <LF> character that may follows the <CR> is not considered but it is accepted because usually the Hyper-Terminal or similar application sends also this extra character when the <CR> is sent.
As a rule, an input string is composed of one or more command-sequences, terminated by the <CR> character and an optional <LF> character.

The command-sequence is composed by the following elements, exactly in this order:

- A five-letter mnemonic command, always present
- An operator, always present
- An optional value, present only when requested by the operator type
- An optional comment-separator, may be present if it is also present the value
- A comment, present only if it is also present the comment-separator
- An optional command-separator, present only if another command-sequence follows it

With the exception of the comment element, no other extra characters or spaces are allowed in the command-sequence.

Command: The commands are always represented by a five-letters mnemonic code and are case insensitive, so for example the command MODSV can be written "MODSV", "Modsv", "modsv", "mOdSv" and in any combination of upper / lower case letters.

Operator: The operators permit to choose one of the three possible functions associated to the command at which they are attached and they are:

- READ, indicated by the ? symbol. It is used to read values.
- SET, indicated by the = symbol. It is used to set values.
- HELP, indicated by the =? sequence of symbols. It is used to display a set of options or a range of permissible values related to the command.

Value: The values can be numbers, strings or special formatted fields like the date / time or the IP addresses, depending on what it is expected by the command.
Numeric values are always checked for validity range and strings are checked for length.
IP addresses and date / time fields are checked only for the correct syntax but not for the values, so please be careful, because in case of misspelled characters or wrong numeric values the result may be different from what it is expected.
In case of floating-point numbers, the decimal point symbol to be used is the dot (**.**), not the comma (**,**).

Comment-separator: This is an optional element and it is indicated by the ":" symbol.

Comment: This element may be present only when the value to input belongs to a list of options, composed by numbers and descriptions.
Normally the user doesn't have to supply both, but in case of copying and pasting some values coming from a previously listed configuration, this ensures the full compatibility between the output and the input formats.

Command-separator: This element is required when more than one command-sequence is submitted in an input string and it is indicated with the "," symbol.
For each command-sequence recognized and executed, the converter returns one of the following output types, depending on it:

···a result code, when a function execution was requested
···an expression, when a parameter or a process data value was requested
···a list of options or a range of values, when an help on a parameter was requested

Each answer is separated from the other by the comma symbol (the same used as command-separator).
The complete output string is terminated by a **<CR><LF>** sequence.
Unrecognized command-sequences are silently discarded without response and without halting the execution of the next sequence, if it is present.
Illegal parameter's values and operations performed in wrong contexts are reported and identified by error codes.

Result-code: The format of the results is the following: code-number:description, without any blank spaces separating the number from the description.

There are six possible result-codes:

···0:OK, the execution was correct
···1:CMD ERR, wrong context, execution was not possible due to a configuration limit or wrong working conditions
···2:PARAM ERR, the expected parameter was out of the allowed range
···3:EXEC ERR, the execution of the command was not successful due to an internal error condition
···4:RANGE ADJ, the entered parameter caused an internal automatic adjustment on other ranges
···5:ACCESS ERR, the execution of the command was not possible due to an insufficient privilege level
···6:BUFFER FULL, the input or the output strings exceed the maximum allowable space.

3.3.12. SPECIAL CHARACTERS

The following characters have special meaning in the protocol and thus they can't be used for other purposes:

···<CR> carriage-return character, value 13 decimal, 0D hexadecimal, terminates the input string and starts the elaboration
···<LF> line-feed character, value 10 decimal, 0A hexadecimal, may follows the <CR> but it is never considered
···? question mark character, value 63 decimal, 3F hexadecimal, it is an operator character
···= equal sign character, value 61 decimal, 3D hexadecimal, it is another operator character
···: colon character, value 58 decimal, 3A hexadecimal, it is the comment-separator character
···, comma character, value 44 decimal, 2C hexadecimal, it is the command-separator character

3.3.13. ACCESS CODE FOR FUNCTION WITH PRIVILEGE LEVEL

The access and the changing of some parameters can made only after the inserting of a level code.
This can made sending an access code corresponding to the level L2 as first command before inserting the command.
The syntax for inserting the code is: "ACODE=n" with n=access code L2
The lifecycle of the code is limited at the execution of the string command that following the code.
After the acquisition of the command the access level return at default status.
Is necessary to insert the code level every time the command or the list of the command request the insertion of the code access greater than the default level.
If the level L2 of the converter is set to 0, then is not necessary to insert the code L2.
When the access code is not sufficient for the command, the converter return the following error code:

5:ACCESS ERR

Example of string with command for code level L2=12345 and command for reading model a software version of the converter:

“ACODE=12345,MODSV?” + chr(13)

3.3.14. DATA BLOCK FORMAT

Communication takes place through data blocks or packs of variable length which do not exceed 256 bytes overall. A data block or pack is composed thus::

- **ADDRESS TO**, Containing the address of the device the block is sent to;
- **ADDRESS FROM,** Containing the address of the device which sent the block;
- **BLOCK CODE**, Containing the code that it indicates if after this block given there is a successive block;
- **BLOCK LENGTH**, Containing the length of the data block in bytes
- **BLOCK DATA**, The block of data of variable length from zero to 250 bytes
- **CRC CHECKSUM BYTE**, Calculated in the following way:

4. CRC initialised to zero
5. CRC rotated to the left by one bit
6. CRC = CRC + block bytes

The process is repeated starting from point 2. for all the block bytes, excluding CRC..

NOTE: The CRC must be subjected to an operation of *rotation to the left* and not just a simple *shift* operation. The difference is shown in the following example:

```
byte value:          10100010
shifting :         1 <=   0100010   <=   0
rotating:        ← 1 <=   0100010   <=   1←
```

NOTE: The code to insert in **BLOCK CODE** can have the following values:

90 decimal (5A hex)

91 decimal (5B hex)

Insert 90 decimal when **BLOCK DATA** length is smaller or equal to 250 characters

If the length of the block data is greater than 250 characters, split the block data in packets of 250 characters.

Every packets with **BLOCK DATA** of 250 characters have BLOCK CODE equal to 91 dec.

The last block of data with numbers of byte smaller or equal to 250 have BLOCK CODE with 90 dec.

3.3.15. EXAMPLE OF ETP COMAND

Reading model and software version of the converter:

mnemonic: MODSV

syntax of the command: “MODSV?”

list of the bytes and corresponding values of the command:

byte 0: address to =	chr(0)	
byte 1: address from =	chr(170)	
byte 2: block code =	chr(90)	(length of the next data block smaller than 250 characters)
byte 3: length =	chr(7)	(length of the next data block)
byte 4: block data =	chr(77) = ‘M’	
byte 5: block data =	chr(79) = ‘O’	
byte 6: block data =	chr(68) = ‘D’	
byte 7: block data =	chr(83) = ‘S’	
byte 8: block data =	chr(86) = ‘V’	
byte 9: block data =	chr(63) = ‘?’	
byte 10: carriage-return <CR> =	chr(13)	
byte 11: checksum =	chr(239)	

List of the bytes to send to the converter in hexadecimal format:

chr(0x00) + chr(0xAA) + chr(0x5A) + chr(0x08) + chr(0x4D) + chr(0x4F) + chr(0x44) + chr(0x53) + chr(0x56) + chr(0x3F) + chr(0x0D) + chr(0xEF)

3.3.16. EXAMPLE OF RESPONSE FOR PRECECDENT ETP COMAND

Header string response:

byte 0 = chr(170)	
byte 1 = chr(0)	
byte 2 = chr(218)	218 dec = 0xDA hex (code 218 = length of the next data block smaller than 250 characters)
byte 3 = chr(29)	number of characters of the data block = 29 bytes

Body string response:

byte 4 = 77	chr(77) =	‘M’	char 1
byte 5 = 76	chr(76) =	‘L’	char 2
byte 6 = 32	chr(32) =	‘ ’	char 3
byte 7 = 50	chr(50) =	‘2’	char 4
byte 8 = 49	chr(49) =	‘1’	char 5
byte 9 = 48	chr(48) =	‘0’	char 6
byte 10 = 32	chr(32) =	‘ ’	char 7
byte 11 = 86	chr(86) =	‘V’	char 8
byte 12 = 69	chr(69) =	‘E’	char 9
byte 13 = 82	chr(82) =	‘R’	char 10
byte 14 = 46	chr(46) =	‘.’	char 11
byte 15 = 51	chr(51) =	‘3’	char 12
byte 16 = 46	chr(46) =	‘.’	char 13
byte 17 = 54	chr(54) =	‘6’	char 14
byte 18 = 48	chr(48) =	‘0’	char 15
byte 19 = 32	chr(32) =	‘ ’	char 16
byte 20 = 77	chr(77) =	‘M’	char 17
byte 21 = 97	chr(97) =	‘a’	char 18
byte 22 = 121	chr(121) =	‘y’	char 19
byte 23 = 32	chr(32) =	‘ ’	char 20
byte 24 = 49	chr(49) =	‘1’	char 21
byte 25 = 53	chr(53) =	‘5’	char 22
byte 26 = 32	chr(32) =	‘ ’	char 23
byte 27 = 50	chr(50) =	‘2’	char 24
byte 28 = 48	chr(48) =	‘0’	char 25
byte 29 = 48	chr(48) =	‘0’	char 26
byte 30 = 55	chr(55) =	‘7’	char 27
byte 31 = 13	chr(13) =		char 28
byte 32 = 10	chr(10) =		char 29

byte 33 = 247   checksum

3.3.17. COMUNICATION

The above mentioned communication protocol is used to exchange data between one unit defined as the MASTER and another unit defined as the SLAVE.

The term MASTER is used here to indicate a device capable of initiating data transmission on its own initiative.

The term SLAVE however, is used to indicate a device capable of transmitting data only in reply to a command received.

Neither of the two types of devices can receive and transmit contemporaneously: these operations take place in two different time phases.

Two MASTER type units cannot be connected together because both of the devices would begin to transmit autonomously and would never succeed in synchronizing themselves.

Analogously, two SLAVE type devices could never exchange information because neither of the two is capable of initiating communication on its own initiative.

The flow of information in both directions therefore takes place alternately using blocks containing a maximum of 95 words each. A word is composed of 10 bits in all: 1 start bit + 8 data bits + 1 stop bit.

These data blocks are received contemporaneously by all the devices connected along the communication line, but only the device with the address equal to the one contained in the data block will be the real destined.

The words comprising the block must be sent on the serial line in a continuous mode, without dead time between one word and the next. The maximum dead time permitted between two words must not exceed the duration of one word.

SLAVE type devices always generate a reply to every message sent to them and correctly received. In the case of reception errors, there is no reply. In the case of command interpretation errors (incorrect format, missing parameters, etc.) either a block without data or no reply, may be the reply depending on the case.

The that elapses between the sending of the last data word by the MASTER and the sending of the first reply word by the SLAVE can be schematized as follows:

```
       last word transmitted by the MASTER
       |
       |        reply processing: time variable from  0 to 25 milliseconds at the most
       |        |
       |        |        time equal to 3 complete words
       |        |        |
       |        |        |          first word transmitted by the SLAVE as reply
       |        |        |          |
.....xxxxx|   t0  |   t1  |xxxxxx......
```

Time t0 depends on the type of command sent and the type of converter it is sent to. Time t1 depends on the communication speed used. A further time t2 - equal to the word length - must be added to these two times. The integrated circuit receiver needs this time to acquire and extract the data bytes of the first word received.

For the system to work correctly, a MASTER type device must respect the following protocol:

1. block transmission with the code-command;
2. wait for the first reply word for a period of time slightly longer than t0 max. + t1 + t2;
3. reception of the rest of the reply block if arrived or the re-transmission of the block with the code - command. After a certain number of re-transmissions without reply, an error message may be generated.

The SLAVE reply must always be waited for within a certain time limit (timeout), once this time has elapsed a new block can be transmitted. If the wait time is too short, a fresh MASTER transmission may take place during the same time the SLAVE is replying, with the consequent loss of both data blocks.

If the wait time is excessive, the system could become too slow in signalling faults, especially if there are many instruments connected in the network.

In any case, a timer must be used to indicate the time limit because reply from the SLAVE cannot be trusted with certainty: in fact, this latter cannot reply in the case of line errors.

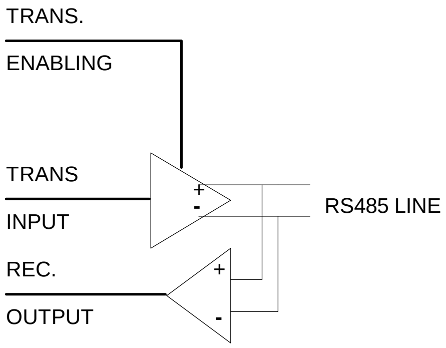
If the communication speed is 9600 bps and the format is 10 bits, the length of a word is equal to 1.042 ms. A reasonable time limit could be calculated thus:

$$T_{lim} = t0 \text{ max.} + t1 + t2 + 1 \text{ ms} = 25 + 3 \text{ x } 1.042 + 1 \text{ x } 1{,}042 + 1 = 30.17 \text{ ms.}$$

IMPORTANT NOTES: There must be a «silence» interval, equal to at least 3 complete words (30 bits), between each data block and the successive one in transit on the serial line.Before starting transmission, to the network, activate the transmitter for a time of no less than 3 words (30 bits) and then begin to send the output characters. This is to give the devices in the network time to eliminate any false characters received when the network was not in use status.

3.3.18. SUGGESTIONS FOR RS485 INTERFACE USE

When programs are developed which use this type of interface, it is as well to make a few considerations. Firstly, we should remember that data transits both in reception and in transmission on the RS485 and that when no transmitter is enabled, the line is in a condition of relatively high impedance.
Physically the RS485 interface is created by a transmitter with a differential output and by a receiver, also with a differential input. The output of the first is physically connected to the input of the second, according to the scheme shown below:



From this scheme we can see how the transmission and reception of different data simultaneously is not possible. Each function must be activated separately with a specific software command which changes the logical status of an interface pin (the transmitter enabling signal). This is the main difference with respect to the RS232 interface normally present on PCs.

In fact, whilst with the RS232 we do not have to worry about the data traffic because it travels on two separate lines (RXD and TXD), with the RS485 there is only one line with the traffic suitably managed.

The first operation we must do therefore is to activate the transmitter and then send all the data to the communication port. Next, we have to wait until the last piece of data has gone from the port to the line and only at this point can we disables the transmitter.

The management software must also absolutely prevent two different devices on the same line from being activated for transmission at the same time: this would mean that we would have the outputs of the drivers of the two transmitters connected in parallel, with a consequent overheating of the components and a loss of information. This eventuality would not break the devices however as they are thermally protected.

On this matter it is necessary to pay extreme attention to the delay between the transmission of the last data byte (the checksum) and the effective disabling of the transmitter. This delay must be lower than the time equivalent for transmitting a 10 bit word, because after this time has elapsed, the device addressed will begin its transmission phase.

For the converter with software version 3.50 or greater, in the menu Communication is present the function A.delay.

This setting value in milliseconds, have the function to insert a delay interval for the response from the converter, so that the transmitter device have enough time to switch in receiver status.

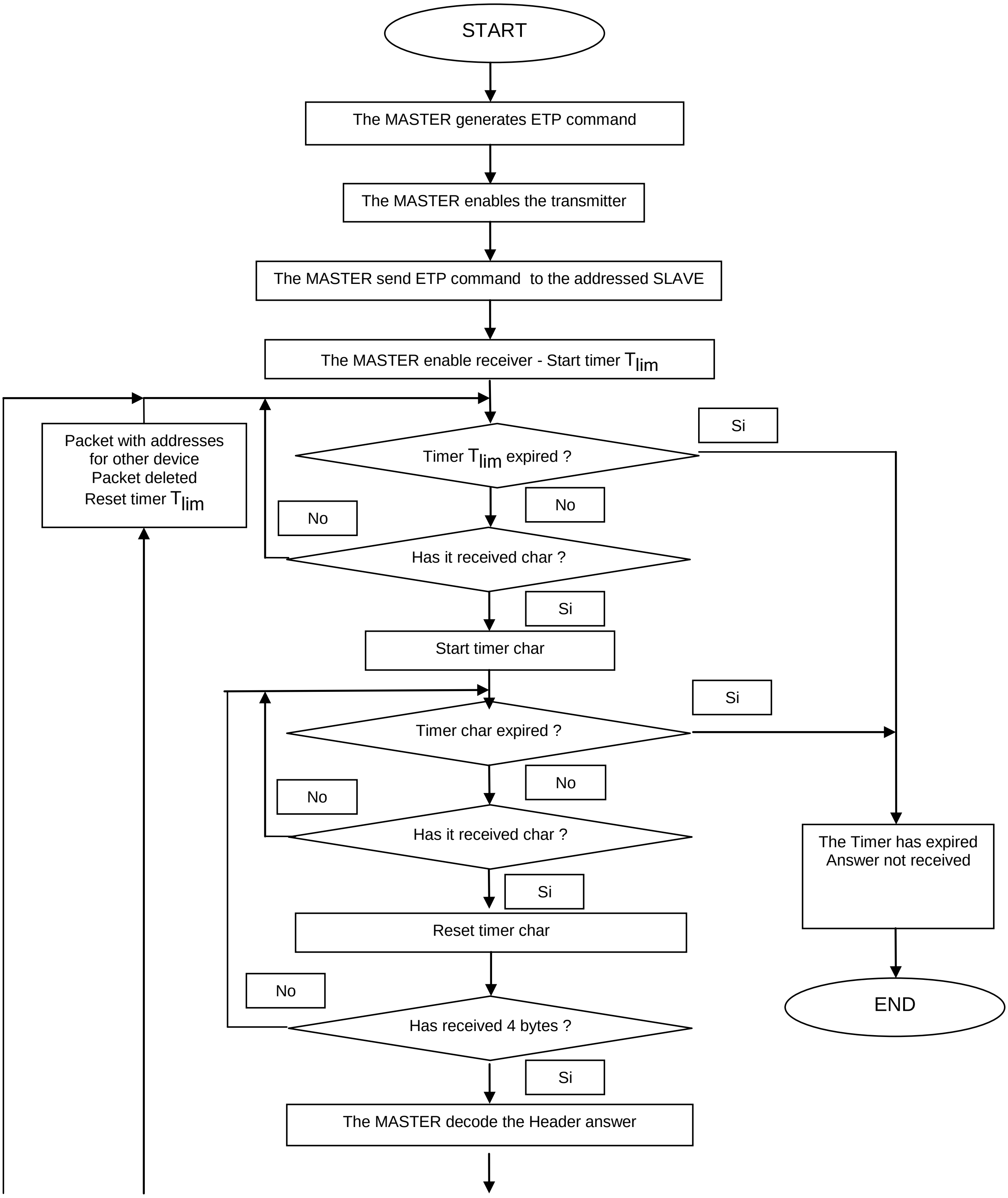
It is also necessary to note that when the line is not piloted by any transmitter, it is in a condition of relatively high impedance and this may give rise to the reception of characters without the right meaning and possible disturbance.
On this question, the communication management software must activate the transmitter a few seconds before transmission starts (typically a time of no less than the length of 3 words at the speed used) so as not to give the line a defined logical status (MARK) and to permit the receivers connected to eliminate any false characters. In the same way, during reception, it must take any such characters possibly present during line inactivity and rapidly eliminate them to prevent saturation of the reception buffers, especially so at high speeds

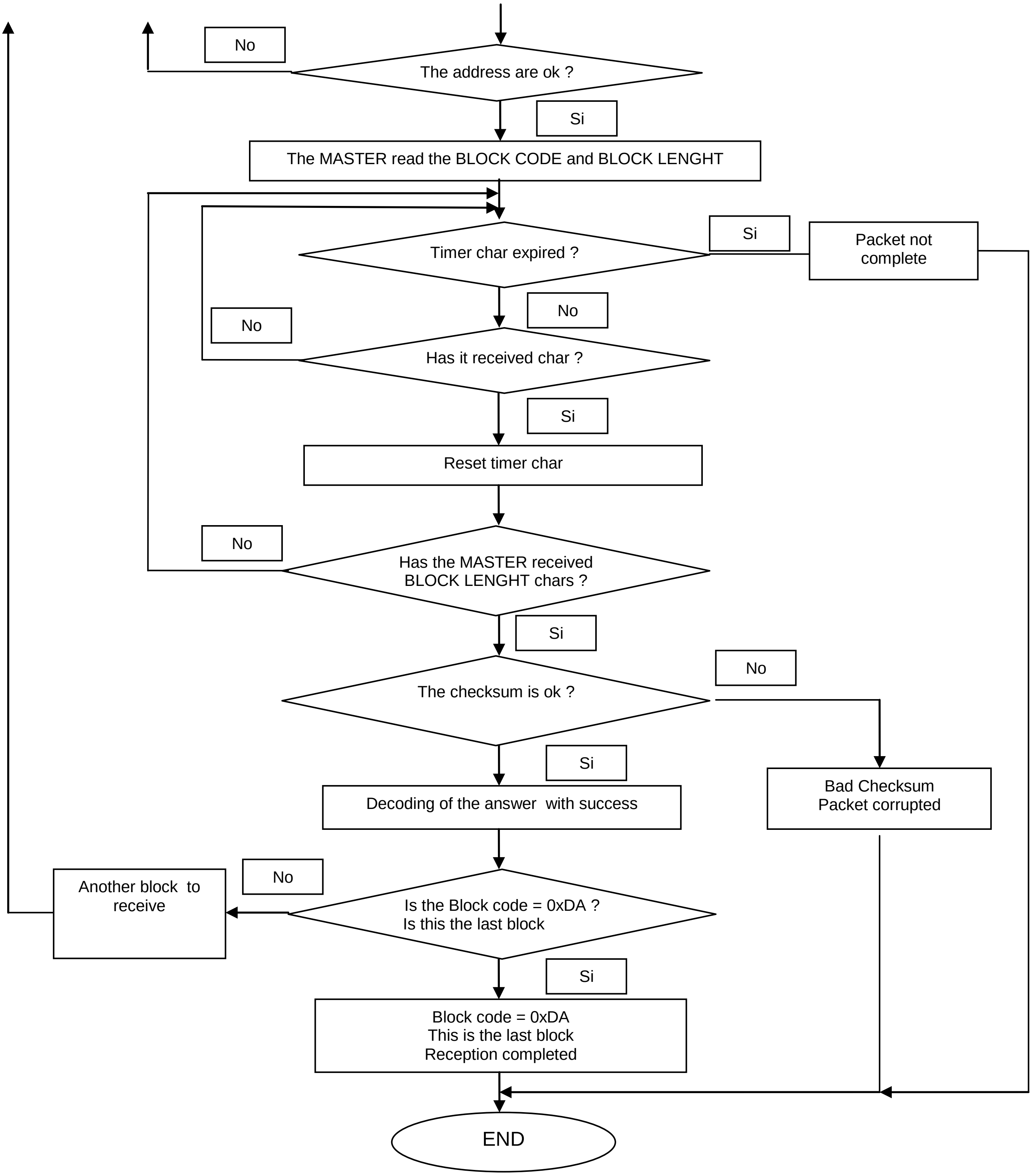
Furthermore, given that the receiver is anyway connected to the transmitter output, in some types of interface it is necessary to eliminate any characters possibly present, received during the transmission phase, because they are copies of what has just been sent. See the next section for some programming examples.

MAN 1000102637 EN Version: B Status: RL (released | freigegeben) printed: 10.01.2018

3.3.19. FLOW CHART FOR SENDING ETP COMMAND AND RECEPTION OF THE ANSWER

Flow chart with description of the sending of an ETP command to a converter through the RS 485 port and wait cycle for the reception of the answer


START
The MASTER generates ETP command
The MASTER enables the transmitter
The MASTER send ETP command to the addressed SLAVE
The MASTER enable receiver - Start timer T_{lim}
Packet with addresses for other device Packet deleted Reset timer T_{lim}
Timer T_{lim} expired ?
Si
No
No
Has it received char ?
Si
Start timer char
Si
Timer char expired ?
No
No
Has it received char ?
Si
Reset timer char
No
Has received 4 bytes ?
Si
The MASTER decode the Header answer
The Timer has expired Answer not received
END



Note:
The timer char is set to 2.5 time the time necessary for to receive a char: 2.5 * tchar.
Every time there is a char in the receive buffer the timer char is restart.
If the receive buffer doesn’t receive a char in the next 2.5 * tchar, the timer char expire and the reception of data is terminated.

3.3.20. LIST OF SUPPORTED COMMANDS GROUPED SIMILARLY TO THE INTERNAL MENU SYSTEM

NOTE (VALID FOR ALL FUNCTIONS) : returned value 5 = ACCESS ERR (insufficient access level)

COMMANDS AND FUNCTIONS EQUIVALENT TO THE “1-SENSOR” MENU		
Name and description	Modes	Returned values or codes
PDIMV (**P**ipe **DI**a**M**eter **V**alue) Reads or sets the nominal diameter of the sensor.	Read command: **PDIMV?**	value of nominal diameter if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **PDIMV=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **PDIMV=?**	***min <> max (unit)*** if accepted **5:ACCESS ERR** if insufficient access level
CFFKA (**C**oe**FF**icient **KA**) Reads or sets the value of the gain coefficient KA	Read command: **CFFKA?**	Value of the coefficient if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **CFFKA=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **CFFKA=?**	***min <> max*** if accepted **5:ACCESS ERR** if insufficient access level
SMODL (**S**ensor **MOD**e**L**) Reads or sets the value of the sensor model	Read command: **SMODL?**	Value of the sensor model if accepted **5:ACCESS ERR** if insufficient access level **5:ACCESS ERR** if insufficient access level
	Set command: **SMODL=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **SMODL=?**	***min <> max*** if accepted **5:ACCESS ERR** if insufficient access level
SIPOS (**S**ensor **I**nsertion POSition) Reads or sets the value that identifies the insertion position for that type of sensor	Read command: **SIPOS?**	Value of the insertion position if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **SIPOS=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **SIPOS=?**	***min <> max*** if accepted **5:ACCESS ERR** if insufficient access level
CKLP0, CKLP1, CKLP2, CKLP3, CKLP4, CKLP5 (**C**oefficient **KL P**ositive **0**, **1**, **2**, **3**, **4** an **5**) Reads	Read command: **CKLP0?**	Value of the coefficient if accepted **5:ACCESS ERR** if insufficient access level

or sets the value for the six coefficients for the linearization of the positive flow rate range. **NOTE:** the examples refer to the command **CKLP0**, but the others are the same	Set command: **CKLP0=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **CKLP0=?**	***min <> max*** if accepted **5:ACCESS ERR** if insufficient access level
CKLN0, CKLN1, CKLN2, CKLN3, CKLN4, CKLN5 (**C**oefficient **KL** **N**egative **0**, **1**, **2**, **3**, **4** an **5**) Reads or sets the value for the six coefficients for the linearization of the negative flow rate range. **NOTE:** the examples refer to the command **CKLN0**, but the others are the same	Read command: **CKLN0?**	Value of the coefficient if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **CKLN0=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **CKLN0=?**	***min <> max*** if accepted **5:ACCESS ERR** if insufficient access level
SFREQ (**S**ampling **FREQ**uency) Reads or sets the measure sampling frequency of the instrument.	Read command: **SFREQ?**	Value of the measure sampling frequency **5:ACCESS ERR** if insufficient access level
CRVRF (**C**oil **R**egulator **V**oltage **R**e**F**erence) Reads or sets the value of the reference for the coil current regulator.	Read command: **CRVRF?**	Value of the coil regulator voltage reference **5:ACCESS ERR** if insufficient access level
	Help command: **CRVRF=?**	***min <> max*** if accepted **5:ACCESS ERR** if insufficient access level
CRRMA (**C**oil **R**egulator Regulation **MA**rgin) Reads or sets the value of the regulation margin for the coil current regulator.	Read command: **CRRMA?**	Value of the coil regulator margin **5:ACCESS ERR** if insufficient access level
	Help command: **CRRMA=?**	***min <> max*** if accepted **5:ACCESS ERR** if insufficient access level
EPDEN (**E**mpty **P**ipe **D**etection **EN**able) Enables or disables the empty pipe detection circuit.	Read command: **EPDEN?**	State of the pipe detection circuit **5:ACCESS ERR** if insufficient access level
	Set command: **EPDEN=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **EPDEN=?**	**0:OFF,1:ON** if accepted **5:ACCESS ERR** if insufficient access level
ELCLN (**Electrodes CLeaNing**) Reads or sets the value of the electrodes cleaning level	Read command: **ELCLN?**	Value of the electrodes cleaning **5:ACCESS ERR** if insufficient access level
	Help command: **ELCLN=?**	***min <> max*** if accepted **5:ACCESS ERR** if insufficient access level

EPDTH (**E**mpty **P**ipe **D**etection **TH**reshold) Level of signal to detect the empty pipe condition	Read command: **EPDTH?**	Value of the EP detection **5:ACCESS ERR** if insufficient access level
	Help command: **EPDTH=?**	***min <> max*** if accepted **5:ACCESS ERR** if insufficient access level
SZPCC (**S**ensor **Z**ero **P**oint **C**alibration **C**ommand) Executes the calibration of the sensor zero point.	Read command: **SZPCC?**	**0** if the calibration is terminated **1** if the calibration is in progress **5:ACCESS ERR** if insufficient access level
	Set command: **SZPCC=1**	**0:OK** if execution ok **2:PARAM ERR** if parameter not equal to 1 **5:ACCESS ERR** if insufficient access level
	Help command: **SZPCC=?**	**1:EXECUTE** if accepted **5:ACCESS ERR** if insufficient access level
SZPCR (**S**ensor **Z**ero **P**oint **C**alibration Reset) Resets the zero point calibration value evaluated with the preceding instruction SZPCC.	Read command: NOT SUPPORTED	**1:CMD ERR** if read command is issued **5:ACCESS ERR** if insufficient access level
	Set command: **SZPCR=1**	**0:OK** if execution ok **2:PARAM ERR** if parameter not equal to 1 **5:ACCESS ERR** if insufficient access level
	Help command: **SZPCR=?**	**1:EXECUTE** if accepted **5:ACCESS ERR** if insufficient access level

COMMANDS AND FUNCTIONS EQUIVALENT TO THE "2-SCALES" MENU

Name and description	Modes	Returned values or codes
FRMUT (**F**low **R**ate **M**easure **U**nit **T**ype) Reads or sets the type of measure units relative to the flow rate. Possible values are: **0:VM** = Volume, Metric **1:WM** = Weight, Metric **2:VI** = Volume, Imperial or American **3:WI** = Weight, Imperial or American	Read command: **FRMUT?**	Value of measure unit if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **FRMUT=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **FRMUT=?**	List of ***num:description*** if accepted **5:ACCESS ERR** if insufficient access level
FRMUV (**F**low **R**ate **M**easure **U**nit **V**alue) Reads or sets the value of measure units relative to the flow rate.	Read command: **FRMUV?**	Value of measure unit if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **FRMUV=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **FRMUV=?**	List of ***num:description*** if accepted **5:ACCESS ERR** if insufficient access level
FRFS1 (**F**low **R**ate **F**ull **S**cale **1**) Reads or sets the value of flow rate full scale 1.	Read command: **FRFS1?**	Value of flow rate full scale 1 if accepted **5:ACCESS ERR** if insufficient access level

	Set command: **FRFS1=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **FRFS1=?**	***min <> max (units)*** if accepted **5:ACCESS ERR** if insufficient access level
FRFS2 (**Fl**ow **R**ate **F**ull **S**cale **2**) Reads or sets the value of flow rate full scale 2.	Read command: **FRFS2?**	Value of flow rate full scale 2 if accepted **1:CMD ERR** if full scale 2 not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **FRFS2=*n***	**0:OK** if parameters accepted **1:CMD ERR** if full scale 2 not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **FRFS2=?**	***min <> max (units)*** if accepted **1:CMD ERR** if full scale 2 not enabled **5:ACCESS ERR** if insufficient access level
VTMUT (**V**olume **T**otalizer **M**easure **U**nit **T**ype) Reads or sets the type of measure units relative to the totalizers. Possible values are: **0:VM** = Volume, Metric **1:WM** = Weight, Metric **2:VI** = Volume, Imperial or American **3:WI** = Weight, Imperial or American	Read command: **VTMUT?**	Type of measure unit if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **VTMUT=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **VTMUT=?**	List of ***num:description*** if accepted **5:ACCESS ERR** if insufficient access level
VTMUV (**V**olume **T**otalizer **M**easure **U**nit **V**alue) Reads or sets the value of measure units relative to the totalizers.	Read command: **VTMUV?**	Value of measure unit if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **VTMUV=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of rang **5:ACCESS ERR** if insufficient access level
	Help command: **VTMUV=?**	List of ***num:description*** if accepted **5:ACCESS ERR** if insufficient access level
VTDPP (**V**olume **T**otalizers **D**ecimal **P**oint **P**osition) Reads or sets the value representing the number of decimal digits for representing the volume totalizers.	Read command: **VTDPP?**	Value of decimal digits if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **VTDPP=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **VTDPP=?**	***min <> max*** if accepted **5:ACCESS ERR** if insufficient access level
CH1PV (**CH**annel **1** **P**ulse **V**alue) Reads or sets the value representing the volume of one totalization pulse for the channel 1.	Read command: **CH1PV?**	Value of volume pulse for channel 1 **1:CMD ERR** if channel 1 not enabled **5:ACCESS ERR** if insufficient access level

	Set command: **CH1PV=*n***	**0:OK** if parameters accepted **1:CMD ERR** if channel 1 not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **CH1PV=?**	***min <> max (units)*** if accepted **1:CMD ERR** if channel 1 not enabled **5:ACCESS ERR** if insufficient access level
CH2PV (**CH**annel **2** **P**ulse **V**alue) Reads or sets the value representing the volume of one totalization pulse for the channel 2.	Read command: **CH2PV?**	Value of volume pulse for channel 2 **1:CMD ERR** if channel 2 not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **CH2PV=*n***	**0:OK** if parameters accepted **1:CMD ERR** if channel 2 not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **CH2PV=?**	***min <> max (units)*** if accepted **1:CMD ERR** if channel 2 not enabled **5:ACCESS ERR** if insufficient access level
CH1PT (**CH**annel **1** **P**ulse Time) Reads or sets the value representing the time duration pulse for the channel 1.	Read command: **CH1PT?**	Value of pulse time for channel 1 **1:CMD ERR** if channel 1 not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **CH1PT=*n***	**0:OK** if parameters accepted **1:CMD ERR** if channel 1 not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **CH1PT=?**	***min <> max (units)*** if accepted **1:CMD ERR** if channel 1 not enabled **5:ACCESS ERR** if insufficient access level
CH2PT (**CH**annel **2** **P**ulse Time) Reads or sets the value representing the time duration pulse for the channel 2.	Read command: **CH2PT?**	Value of pulse time for channel 2 **1:CMD ERR** if channel 2 not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **CH2PT=*n***	**0:OK** if parameters accepted **1:CMD ERR** if channel 2 not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **CH2PT =?**	***min <> max (units)*** if accepted **1:CMD ERR** if channel 2 not enabled **5:ACCESS ERR** if insufficient access level
VMSGC (**V**olume to **M**ass **S**pecific **G**ravity **C**oefficient) Reads or sets the value representing the transformation coefficient between volume and mass.	Read command: **VMSGK?**	Value of volume-to-mass coefficient **1:CMD ERR** if weight units not enabled **5:ACCESS ERR** if insufficient access level

	Set command: **VMSGK=*n***	**0:OK** if parameters accepted **1:CMD ERR** if weight units not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **VMSGK=?**	***min <> max (units)*** if accepted **1:CMD ERR** if weight units not enabled **5:ACCESS ERR** if insufficient access level
A1CSY (**A**nalog input **1** **C**ustom **SY**mbol) Reads or sets the value representing the custom measure unit used for the analog input 1. If the values is to be set, a fixed 3-characters length string must be supplied.	Read command: **A1CSY?**	Value of analog input 1 measure unit symbol **1:CMD ERR** if analog input 1 not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **1CSY=*string***	**0:OK** if parameters accepted **1:CMD ERR** if analog input 1 not enabled **2:PARAM ERR** if *string* length is not 3 chars **5:ACCESS ERR** if insufficient access level
	Help command: **A1CSY=?**	***3 CHR STRING*** if accepted **1:CMD ERR** if analog input 1 not enabled **5:ACCESS ERR** if insufficient access level
A1MUV (**A**nalog input **1** **M**easure **U**nit **V**alue) Reads or sets the value of measure units relative to the totalizers.	Read command: **A1MUV?**	Value of measure unit if accepted **1:CMD ERR** if analog input 1 not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **A1MUV=*n***	**0:OK** if parameters accepted **1:CMD ERR** if analog input 1 not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **A1MUV=?**	List of ***num:description*** if accepted **1:CMD ERR** if analog input 1 not enabled **5:ACCESS ERR** if insufficient access level
A1SSV (**A**nalog input **1** **S**tart **S**cale **V**alue) Reads or sets the value of the start scale point for the analog input 1.	Read command: **A1SSV?**	Start scale value of analog in.1 if accepted **1:CMD ERR** if analog input 1 not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **A1SSV=*n***	**0:OK** if parameters accepted **1:CMD ERR** if analog input 1 not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level

	Help command: **A1SSV=?**	***min <> max (units)*** if accepted **1:CMD ERR** if analog input 1 not enabled **5:ACCESS ERR** if insufficient access level
A1FSV (**A**nalog input **1** **F**ull **S**cale **V**alue) Reads or sets the value of the full scale point for the analog input 1.	Read command: **A1FSV?**	Full scale value of analog in.1 if accepted **1:CMD ERR** if analog input 1 not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **A1FSV=*n***	**0:OK** if parameters accepted **1:CMD ERR** if analog input 1 not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **A1FSV=?**	***min <> max (units)*** if accepted **1:CMD ERR** if analog input 1 not enabled **5:ACCESS ERR** if insufficient access level

COMMANDS AND FUNCTIONS EQUIVALENT TO THE “3-MEASURES” MENU

Name and description	Modes	Returned values or codes
MFDMP (**M**easure **F**ilter **D**a**MP**ing) Reads or sets the value of the measure filter	Read command: **MFTCV?**	Value of measure time constant if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **MFTCV=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **MFTCV=?**	***min <> max (units)*** if accepted **5:ACCESS ERR** if insufficient access level
	Help command: **MFMXT=?**	***min <> max (units)*** if accepted **5:ACCESS ERR** if insufficient access level
MFCUT (**M**easure **F**ilter **Cu**t-off **T**hreshold) Reads or sets the value of the measure filter cut-off threshold.	Read command: **MFCUT?**	Value of measure cut-off thr. if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **MFCUT=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **MFCUT=?**	***min <> max (units)*** if accepted **5:ACCESS ERR** if insufficient access level
ACALE (**A**uto-**CAL**ibration **E**nable) Enables or disables the auto-calibration feature of the	Read command: **ACALE?**	Enable/disable status if accepted **5:ACCESS ERR** if insufficient access level

MAN 1000102637 EN Version: B Status: RL (released | freigegeben) printed: 10.01.2018

instrument.	Set command: **ACALE=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **ACALE=?**	List of ***num:description*** if accepted **5:ACCESS ERR** if insufficient access level
ARNGE (**A**uto-**R**a**NG**e **E**nable) Enables or disables the auto-range feature of the instrument.	Read command: **ARNGE?**	Enable/disable status if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **ARNGE=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **ARNGE=?**	List of ***num:description*** if accepted **5:ACCESS ERR** if insufficient access level

COMMANDS AND FUNCTIONS EQUIVALENT TO THE “4-ALARMS” MENU

Name and description	Modes	Returned values or codes
FRAXP (**F**low **R**ate **A**larm ma**X** **P**ositive threshold) Reads or sets the value of the maximum positive flow rate alarm threshold.	Read command: **FRAXP?**	Value of positive max threshold if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **FRAXP=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **FRAXP=?**	***min <> max (units)*** if accepted **5:ACCESS ERR** if insufficient access level
FRAXN (**F**low **R**ate **A**larm ma**X** **N**egative threshold) Reads or sets the value of the maximum positive flow rate alarm threshold.	Read command: **FRAXN?**	Value of negative max threshold if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **FRAXN=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **FRAXN=?**	***min <> max (units)*** if accepted **5:ACCESS ERR** if insufficient access level
FRANP (**F**low **R**ate **A**larm mi**N** **P**ositive threshold) Reads or sets the value of the minimum positive flow rate alarm threshold.	Read command: **FRANP?**	Value of positive min threshold if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **FRAXP=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **FRAXP=?**	***min <> max (units)*** if accepted **5:ACCESS ERR** if insufficient access level

FRANN (**F**low **R**ate **A**larm mi**N** **N**egative threshold) Reads or sets the value of the minimum positive flow rate alarm threshold.	Read command: **FRANN?**	Value of negative min threshold if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **FRANN=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **FRANN=?**	***min <> max (units)*** if accepted **5:ACCESS ERR** if insufficient access level
ATHYS (**A**larm **T**hresholds **HYS**teresis) Reads or sets the value of the alarm threshold hysteresis.	Read command: **ATHYS?**	Value of hysteresis if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **ATHYS=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **ATHYS=?**	***min <> max (units)*** if accepted **5:ACCESS ERR** if insufficient access level
OCACV (**O**utput **C**urrent **A**larm **C**ondition **V**alue) Reads or sets the value of the alarm value at which the current output will be set in case of error conditions. Valid only if the current output is installed and enabled.	Read command: **OCACV?**	Value of alarm current output if accepted **1:CMD ERR** if current output not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **OCACV=*n***	**0:OK** if parameters accepted **1:CMD ERR** if current output not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **OCACV=?**	***min <> max (units)*** if accepted **1:CMD ERR** if current output not enabled **.** **5:ACCESS ERR** if insufficient access level

COMMANDS AND FUNCTIONS EQUIVALENT TO THE "5-INPUTS" MENU		
Name and description	Modes	Returned values or codes
VTTPE (**V**olume **T**otalizer **T**otal **P**ositive reset **E**nable) Enables or disables the consent for resetting the total positive volume totalizer when receiving an external signal on the digital input. Valid only if the digital input is installed and enabled.	Read command: **VTTPE?**	Status of reset consent if accepted **1:CMD ERR** if digital input not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **VTTPE=*n***	**0:OK** if parameters accepted **1:CMD ERR** if digital input not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level

	Help command: **VTTPE=?**	List of ***num:description*** if accepted **1:CMD ERR** if digital input not enabled **5:ACCESS ERR** if insufficient access level
VTPPE (**V**olume **T**otalizer **P**artial **P**ositive reset **E**nable) Enables or disables the consent for resetting the partial positive volume totalizer when receiving an external signal on the digital input. Valid only if the digital input is installed and enabled.	Read command: **VTPPE?**	Status of reset consent if accepted **1:CMD ERR** if digital input not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **VTPPE=*n***	**0:OK** if parameters accepted **1:CMD ERR** if digital input not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **VTPPE=?**	List of ***num:description*** if accepted **1:CMD ERR** if digital input not enabled **5:ACCESS ERR** if insufficient access level
VTTNE (**V**olume **T**otalizer **T**otal **N**egative reset **E**nable) Enables or disables the consent for resetting the total negative volume totalizer when receiving an external signal on the digital input. Valid only if the digital input is installed and enabled.	Read command: **VTTNE?**	Status of reset consent if accepted **1:CMD ERR** if digital input not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **VTTNE=*n***	**0:OK** if parameters accepted **1:CMD ERR** if digital input not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **VTTNE=?**	List of ***num:description*** if accepted **1:CMD ERR** if digital input not enabled **5:ACCESS ERR** if insufficient access level
VTPNE (**V**olume **T**otalizer **P**artial **N**egative reset **E**nable) Enables or disables the consent for resetting the partial negative volume totalizer when receiving an external signal on the digital input. Valid only if the digital input is installed and enabled.	Read command: **VTPNE?**	Status of reset consent if accepted **1:CMD ERR** if digital input not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **VTPNE=*n***	**0:OK** if parameters accepted **1:CMD ERR** if digital input not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **VTPNE=?**	List of ***num:description*** if accepted **1:CMD ERR** if digital input not enabled **5:ACCESS ERR** if insufficient access level

TCLIE (**T**otalizers **C**ount **L**ock **I**nput **E**nable) Enables or disables the consent for locking the totalizers when receiving an external signal on the digital input. Valid only if the digital input is installed and enabled.	Read command: **TCLIE?**	Status of locking consent if accepted **1:CMD ERR** if digital input not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **TCLIE=*n***	**0:OK** if parameters accepted **1:CMD ERR** if digital input not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **TCLIE=?**	List of ***num:description*** if accepted **1:CMD ERR** if digital input not enabled **5:ACCESS ERR** if insufficient access level
CALIE (**CAL**ibration **I**nput **E**nable) Enables or disables the consent for calibrating when receiving an external signal on the digital input. Valid only if the digital input is installed and enabled.	Read command: **CALIE?**	Status of calibration consent if accepted **1:CMD ERR** if digital input not enabled
	Set command: **CALIE=*n***	**0:OK** if parameters accepted **1:CMD ERR** if digital input not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **CALIE=?**	List of ***num:description*** if accepted **1:CMD ERR** if digital input not enabled **5:ACCESS ERR** if insufficient access level
SRCIE (**S**cale **R**ange **C**hange **I**nput **E**nable) Enables or disables the consent for changing the scale range when receiving an external signal on the digital input. Valid only if the digital input is installed and enabled.	Read command: **SRCIE?**	Status of range change consent if accepted **1:CMD ERR** if digital input not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **SRCIE=*n***	**0:OK** if parameters accepted **1:CMD ERR** if digital input not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **SRCIE=?**	List of ***num:description*** if accepted **1:CMD ERR** if digital input not enabled **5:ACCESS ERR** if insufficient access level
WKUIE (**W**a**K**e-**U**p **I**nput **E**nable) Enables or disables the consent for waking-up from the energy saving standby mode when receiving an external signal on the digital input.	Read command: **WKUIE?**	Status of wake-up consent if accepted **1:CMD ERR** if digital input not enabled **5:ACCESS ERR** if insufficient access level

Valid only if the digital input is installed and enabled.	Set command: **WKUIE =*n***	**0:OK** if parameters accepted **1:CMD ERR** if digital input not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **WKUIE =?**	List of ***num:description*** if accepted **1:CMD ERR** if digital input not enabled **5:ACCESS ERR** if insufficient access level

COMMANDS AND FUNCTIONS EQUIVALENT TO THE "6-OUTPUTS" MENU		
Name and description	Modes	Returned values or codes
OUT1F (**OUT**put **1** **F**unction) Sets the function type related to the digital output 1. Valid only if the digital outputs are installed and enabled.	Read command: **OUT1F?**	Type of function if accepted **1:CMD ERR** if digital outputs not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **OUT1F=*n***	**0:OK** if parameters accepted **1:CMD ERR** if digital outputs not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **OUT1F=?**	List of ***num:description*** if accepted **1:CMD ERR** if digital outputs not enabled **5:ACCESS ERR** if insufficient access level
OUT2F (**OUT**put **2** **F**unction) Sets the function type related to the digital output 2. Valid only if the digital outputs are installed and enabled.	Read command: **OUT2F?**	Type of function if accepted **1:CMD ERR** if digital outputs not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **OUT2F=*n***	**0:OK** if parameters accepted **1:CMD ERR** if digital outputs not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **OUT2F=?**	List of ***num:description*** if accepted **1:CMD ERR** if digital outputs not enabled **5:ACCESS ERR** if insufficient access level
OUTIS (**OUT** **I**mmediate **S**tate) function for set in direct way the value of the digital outputs	Read command: **OUTIS?**	Type of function if accepted **1:CMD ERR** if digital outputs not enabled **5:ACCESS ERR** if insufficient access level

	Set command: **OUTIS=*n***	**0:OK** if parameters accepted **1:CMD ERR** if digital outputs not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **OUTIS=?**	List of ***num:description*** if accepted **1:CMD ERR** if digital outputs not enabled **5:ACCESS ERR** if insufficient access level
CO1FS (**C**urrent **O**utput **1** **F**ull **S**cale) Sets the full scale value for the current output 1: it can be 20 or 22 mA. Valid only if the current output1 is installed and enabled.	Read command: **CO1FS?**	Full scale value if accepted **1:CMD ERR** if current output not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **CO1FS=*n***	**0:OK** if parameters accepted **1:CMD ERR** if current output not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **CO1FS=?**	List of ***num:description*** if accepted **1:CMD ERR** if current output not enabled **5:ACCESS ERR** if insufficient access level
CO1SS (**C**urrent **O**utput **1** **S**tart **S**cale) Sets the start scale value for the current output 1: it can be 0 or 4 mA. Valid only if the current output1 is installed and enabled.	Read command: **CO1SS?**	Start scale value if accepted **1:CMD ERR** if current output not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **CO1SS=*n***	**0:OK** if parameters accepted **1:CMD ERR** if current output not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **CO1SS=?**	List of ***num:description*** if accepted **1:CMD ERR** if current output not enabled **5:ACCESS ERR** if insufficient access level
CO1FM (**C**urrent **O**utput **1** **F**ield **M**ode) Sets the field mode value for the current output 1: it can be +, -, +/-, –0+. Valid only if the current output1 is installed and enabled.	Read command: **CO1FM?**	Field mode for current output 1 if accepted **1:CMD ERR** if current output not enabled
	Set command: **CO1FM=*n***	**0:OK** if parameters accepted **1:CMD ERR** if current output not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level

	Help command: **CO1FM=?**	List of ***num:description*** if accepted **1:CMD ERR** if current output not enabled **5:ACCESS ERR** if insufficient access level

COMMANDS AND FUNCTIONS EQUIVALENT TO THE "7-COMMUNICATION" MENU

Name and description	Modes	Returned values or codes
IF2PT (**IF2** **P**rotocol **T**ype) Reads or sets the protocol type for the IF2 port: it can be DPP or HTP.	Read command: **IF2PT?**	Type of protocol if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **IF2PT=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **IF2PT=?**	List of ***num:description*** if accepted **5:ACCESS ERR** if insufficient access level
DVADR (**D**e**V**ice **AD**d**R**ess) Reads or sets the device address. Valid only if the RS232 port is installed and enabled.	Read command: **DVADR?**	Device address if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **DVADR=*n***	**0:OK** if parameters accepted **1:CMD ERR** if RS232 port not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **DVADR=?**	***min <> max*** if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level
232SP (rs**232** **SP**eed) Reads or sets the RS232 speed in bps. Valid only if the RS232 port is installed and enabled.	Read command: **232SP?**	RS232 speed if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **232SP=*n***	**0:OK** if parameters accepted **1:CMD ERR** if RS232 port not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **232SP=?**	List of ***num:description*** if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level
232PT (rs**232** **P**rotocol **T**ype) Reads or sets the protocol type for the RS232 port: it can be DPP or HTP. Valid only if the RS232 port is installed and enabled.	Read command: **232PT?**	RS232 protocol type if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **232PT=*n***	**0:OK** if parameters accepted **1:CMD ERR** if RS232 port not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level

	Help command: **232PT=?**	List of ***num:description*** if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level
SEVSE (Sms Event Send Enable) Enable or disable event send by SMS	Read command: **SEVSE?**	Status of functions if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **SEVSE =*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **SEVSE =?**	List of ***num:description*** if accepted **5:ACCESS ERR** if insufficient access level
SPDSE (Sms Process Data Send Enable) Enable or disable process data send by SMS	Read command: **SPDSE?**	Status of functions if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **SPDSE =*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **SPDSE =?**	List of ***num:description*** if accepted **5:ACCESS ERR** if insufficient access level
SPDSC (Sms Process Data Send Command) send a command for immediately answer with process data	Set command: **SPDSC =*n***	**0:OK** if command accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **SPDSC =?**	List of ***num:description*** if accepted **5:ACCESS ERR** if insufficient access level
ROAME (ROAMing Enable) Enable or disable roaming function	Read command: **ROAME?**	Status of functions if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **ROAME =*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **ROAME =?**	List of ***num:description*** if accepted **5:ACCESS ERR** if insufficient access level
SCAPE (SCADA Protocol Enable) Enable or disable SCADA protocol	Read command: **SCAPE?**	Status of functions if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **SCAPE =*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **SCAPE =?**	List of ***num:description*** if accepted **5:ACCESS ERR** if insufficient access level
GSITM (Gsm Inactivity TiMeout) timeout for inactivity connection	Read command: **GSITM?**	Status of functions if accepted **5:ACCESS ERR** if insufficient access level

	Set command: **GSITM =*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **GSITM =?**	**Time range** if accepted **5:ACCESS ERR** if insufficient access level
DTSDL (DaTa Send DeLay) delay for send data	Read command: **DTSDL?**	Status of functions if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **DTSDL =*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **DTSDL =?**	List of ***num:description*** if accepted **5:ACCESS ERR** if insufficient access level
GSCLN (GSm CaLl Number) telephone number for GSM connection	Read command: **GSCLN?**	Status of functions if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **GSCLN =*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **GSCLN =?**	**MAX 31 CHR STRING** if accepted **5:ACCESS ERR** if insufficient access level
485PT (RS**485** **P**rotocol **T**ype) Reads or sets the protocol type for the RS485 port: it can be DPP, HTP or MODBUS. Valid only if the RS485 port is installed and enabled	Read command: **485PT?**	RS485 protocol type if accepted **1:CMD ERR** if RS485 port not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **485PT =*n***	**0:OK** if parameters accepted **1:CMD ERR** if RS485 port not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **485PT =?**	List of ***num:description*** if accepted **1:CMD ERR** if RS485 port not enabled **5:ACCESS ERR** if insufficient access level
MODBP (**MODB**us **P**arity) parity control in MODBUS protocol	Read command: **MODBP?**	RS485 parity if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **MODBP=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **MODBP=?**	List of ***num:description*** if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level

COMMANDS AND FUNCTIONS EQUIVALENT TO THE "8-DISPLAY" MENU		
Name and description	Modes	Returned values or codes

LLANG (**L**ayout **LANG**uage) Reads or sets the layout language used for all the display messages.	Read command: **LLANG?**	Layout language if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **LLANG=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **LLANG=?**	List of ***num:description*** if accepted **5:ACCESS ERR** if insufficient access level
VTTPR (**V**olume **T**otalizer **T**otal **P**ositive **R**eset) Resets the total positive volume totalizer.	Read command: NOT SUPPORTED	**1:CMD ERR** if read command is issued **5:ACCESS ERR** if insufficient access level
	Set command: **VTTPR=1**	**0:OK** if parameters accepted **2:PARAM ERR** if *n* not equal to 1 **5:ACCESS ERR** if insufficient access level
	Help command: **VTTPR=?**	**1:EXECUTE** if accepted **5:ACCESS ERR** if insufficient access level
VTPPR (**V**olume **T**otalizer **P**artial **P**ositive **R**eset) Resets the partial positive volume totalizer.	Read command: NOT SUPPORTED	**1:CMD ERR** if read command is issued **5:ACCESS ERR** if insufficient access level
	Set command: **VTPPR=1**	**0:OK** if parameters accepted **2:PARAM ERR** if *n* not equal to 1 **5:ACCESS ERR** if insufficient access level
	Help command: **VTPPR=?**	**1:EXECUTE** if accepted **5:ACCESS ERR** if insufficient access level
VTTNR (**V**olume **T**otalizer **T**otal **N**egative **R**eset) Resets the total negative volume totalizer.	Read command: NOT SUPPORTED	**1:CMD ERR** if read command is issued **5:ACCESS ERR** if insufficient access level
	Set command: **VTTNR=1**	**0:OK** if parameters accepted **2:PARAM ERR** if *n* not equal to 1 **5:ACCESS ERR** if insufficient access level
	Help command: **VTTNR=?**	**1:EXECUTE** if accepted **5:ACCESS ERR** if insufficient access level
VTPNR (**V**olume **T**otalizer **P**artial **N**egative **R**eset) Resets the partial negative volume totalizer.	Read command: NOT SUPPORTED	**1:CMD ERR** if read command is issued **5:ACCESS ERR** if insufficient access level
	Set command: **VTPNR=1**	**0:OK** if parameters accepted **2:PARAM ERR** if *n* not equal to 1 **5:ACCESS ERR** if insufficient access level
	Help command: **VTPNR=?**	**1:EXECUTE** if accepted **5:ACCESS ERR** if insufficient access level

ENSDT (**EN**ergy **S**aving **D**isplay **T**ime) Reads or sets the value of the display on time used in the energy saving mode. This time is also used to wait for the incoming of new sms if they are enabled.	Read command: **ENSDT?**	Value of display on time if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **ENSDT=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **ENSDT=?**	***min <> max (units)*** if accepted **5:ACCESS ERR** if insufficient access level
QSTME (Quick STart Menu Enable)	Read command: **QSTME?**	Status of functions if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **QSTME =*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **QSTME =?**	List of ***num:description*** if accepted **5:ACCESS ERR** if insufficient access level
TTNVE (TOTalizers Net Value Enable) enable disable the visualization of Net totalizer	Read command: **TTNVE?**	Status of functions if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **TTNVE =*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **TTNVE =?**	List of ***num:description*** if accepted **5:ACCESS ERR** if insufficient access level
TCMDE (**T**otalizers **C**urrency **M**ode **D**isplay **E**nable) Enables or disables the displaying of the currency values for the totalizers.	Read command: **TCMDE?**	Status of currency mode display if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **TCMDE=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **TCMDE=?**	List of ***num:description*** if accepted **5:ACCESS ERR** if insufficient access level
CUSYT (**CU**rrency **SY**mbol **T**ype) Reads or sets the type of currency symbol used to represent the values converted from the totalizers.	Read command: **CUSYT?**	Type of currency symbol if accepted **1:CMD ERR** if currency not enable **5:ACCESS ERR** if insufficient access level
	Set command: **CUSYT=*n***	**0:OK** if parameters accepted **1:CMD ERR** if currency not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **CUSYT=?**	List of ***num:description*** if accepted **1:CMD ERR** if currency not enabled **5:ACCESS ERR** if insufficient access level

CUCSS (**CU**rrency **C**u**S**tom **S**ymbol) Reads or sets the custom symbol used to represent the currency. If the values is to be set, a fixed 3-characters length string must be supplied.	Read command: **CUCSS?**	Custom currency symbol if accepted **1:CMD ERR** if currency not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **CUCSS=*string***	**0:OK** if parameters accepted **1:CMD ERR** if currency not enabled **2:PARAM ERR** if *string* length is not 3 chars **5:ACCESS ERR** if insufficient access level
	Help command: **CUCSS=?**	***3 CHR STRING*** if accepted **1:CMD ERR** if currency not enabled **5:ACCESS ERR** if insufficient access level
CUDEC (**CU**rrency **DE**cimal **C**iphers) Reads or sets the value of the decimal ciphers for representing the volume totalizers converted to currency.	Read command: **CUDEC?**	Value of decimal ciphers if accepted **1:CMD ERR** if currency not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **CUDEC=*n***	**0:OK** if parameters accepted **1:CMD ERR** if currency not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **CUDEC=?**	***min <> max*** if accepted **1:CMD ERR** if currency not enable **5:ACCESS ERR** if insufficient access level
CUPCF (**CU**rrency **P**ositive **C**onversion **F**actor) Reads or sets the value of the conversion factor coefficient used to convert the partial positive totalizer to currency.	Read command: **CUPCF?**	Value of positive conv. factor if accepted **1:CMD ERR** if currency not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **CUPCF=*n***	**0:OK** if parameters accepted **1:CMD ERR** if currency not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **CUPCF=?**	***min <> max (units)*** if accepted **1:CMD ERR** if currency not enable **5:ACCESS ERR** if insufficient access level
CUNCF (**CU**rrency **N**egative **C**onversion **F**actor) Reads or sets the value of the conversion factor coefficient used to convert the partial negative totalizer to currency.	Read command: **CUNCF?**	Value of negative conv. factor if accepted **1:CMD ERR** if currency not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **CUNCF=*n***	**0:OK** if parameters accepted **1:CMD ERR** if currency not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **CUNCF=?**	***min <> max (units)*** if accepted **1:CMD ERR** if currency not enabled **5:ACCESS ERR** if insufficient access level

COMMANDS AND FUNCTIONS EQUIVALENT TO THE "9-DATALOGGER" MENU		
Name and description	Modes	Returned values or codes
DLOGE (**D**ata **LOG**ger **E**nable) Enables or disables the logging functions for data and events. Events logging are always active.	Read command: **DLOGE?**	Status of data logging functions if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **DLOGE=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **DLOGE=?**	List of ***num:description*** if accepted **5:ACCESS ERR** if insufficient access level
DLGSI (**D**ata **L**o**G**ger **S**ample **I**nterval) Reads or sets the sample time interval at which the data are collected.	Read command: **DLGSI?**	Value of sample time interval if accepted **1:CMD ERR** if logging functions not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **DLGSI=*n***	**0:OK** if parameters accepted **1:CMD ERR** if logging functions not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **DLGSI=?**	List of ***num:description*** if accepted **1:CMD ERR** if logging functions not enabled **5:ACCESS ERR** if insufficient access level
DTIME (**D**ate / **TIME**) Reads or sets the date and time value. **WARNING: when setting the date and time, always check the value returned because malformed input string may lead to incorrect time result.**	Read command: **DTIME?**	Value of date and time if accepted **1:CMD ERR** if logging functions not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **DTIME=*tmstring***	Value of current date and time **1:CMD ERR** if logging functions not enabled **5:ACCESS ERR** if insufficient access level
	Help command: **DTIME=?**	Format of the string to be input **1:CMD ERR** if logging functions not enabled **5:ACCESS ERR** if insufficient access level
DLDRD (**D**ata **L**ogger static **D**ata **R**ea**D**) Reads the specified record, the number of records saved in the static data logger or the maximum number of records that can be saved in memory.	Read number of records: **DLDRD?**	Number of records in memory if accepted **1:CMD ERR** if logging functions not enabled
	Read record: **DLDRD=*n***	Record *n* in CSV format if accepted **1:CMD ERR** if logging functions not enabled **2:PARAM ERR** if *n* out of range

	Read max. capacity: **DLDRD=?**	Maximum range of *n* for the memory in the format ***min <> max*** if accepted **1:CMD ERR** if logging functions not enabled
DLDRE (**D**ata **L**ogger static **D**ata **RE**set) Resets the static data logger and clears all records.	Read command: NOT SUPPORTED	**1:CMD ERR** if command is issued
	Set command: **DLDRE=1**	**0:OK** if parameters accepted **1:CMD ERR** if logging functions not enabled **2:PARAM ERR** if *n* not equal to 1 **5:ACCESS ERR** if insufficient access level
	Help command: **DLDRE=?**	**1:EXECUTE** if accepted **1:CMD ERR** if logging functions not enabled **5:ACCESS ERR** if insufficient access level
DLDSE (**D**ata **L**ogger static **D**ata **S**end **E**nable) Enables or disables the sending of the static data logger's data by email using the GPRS terminal.	Read command: **DLDSE?**	Status of static data send if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **DLDSE=*n***	**0:OK** if parameters accepted **1:CMD ERR** if RS232 port not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **DLDSE=?**	List of ***num:description*** if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level
DLYRD (**D**ata **L**ogger d**Y**namic **R**ea**D**) Reads the specified record, the number of records saved in the dynamic data logger or the maximum number of records that can be saved in memory.	Read number of records: **DLYRD?**	Number of records in memory if accepted **1:CMD ERR** if logging functions not enabled
	Read record: **DLYRD=*n***	Record *n* in CSV format if accepted **1:CMD ERR** if logging functions not enabled **2:PARAM ERR** if *n* out of range
	Read max capacity: **DLYRD=?**	Maximum range of *n* for the memory in the format ***min <> max*** if accepted **1:CMD ERR** if logging functions not enabled
DLYRE (**D**ata **L**ogger d**Y**namic **RE**set) Resets the dynamic data logger and clears all records.	Read command: NOT SUPPORTED	**1:CMD ERR** if command is issued
	Set command: **DLYRE=1**	**0:OK** if parameters accepted **1:CMD ERR** if logging functions not enabled **2:PARAM ERR** if *n* not equal to 1 **5:ACCESS ERR** if insufficient access level

	Help command: **DLYRE=?**	**1:EXECUTE** if accepted **1:CMD ERR** if logging functions not enabled **.** **5:ACCESS ERR** if insufficient access level
DLYSE (**D**ata **L**ogger d**Y**namic **S**end **E**nable) Enables or disables the sending of the dynamic data logger's data by email using the GPRS terminal.	Read command: **DLYSE?**	Status of dynamic data send if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **DLYSE=*n***	**0:OK** if parameters accepted **1:CMD ERR** if RS232 port not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **DLYSE=?**	List of ***num:description*** if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level
DLERD (**D**ata **L**ogger **E**vent **R**ea**D**) Reads the specified record, the number of records saved in the event data logger or the maximum number of records that can be saved in memory**.**	Read number of records: **DLERD?**	Number of records in memory if accepted **1:CMD ERR** if logging functions not enabled
	Read record: **DLERD=*n***	Record *n* in CSV format if accepted **1:CMD ERR** if logging functions not enabled **2:PARAM ERR** if *n* out of range
	Read max capacity: **DLERD=?**	Maximum range of *n* for the memory in the format ***min <> max*** if accepted **1:CMD ERR** if logging functions not enabled
DLERE (**D**ata **L**ogger **E**vent **RE**set) Resets the events logger and clears all records.	Read command: NOT SUPPORTED	**1:CMD ERR** if command is issued
	Set command: **DLERE=1**	**0:OK** if parameters accepted **1:CMD ERR** if logging functions not enabled **2:PARAM ERR** if *n* not equal to 1
	Help command: **DLERE=?**	**1:EXECUTE** if accepted **1:CMD ERR** if logging functions not enabled **.**
DLESE (**D**ata **L**ogger **E**vents **S**end **E**nable) Enables or disables the sending of the events data logger's data by email using the GPRS terminal.	Read command: **DLESE?**	Status of events data send if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **DLESE=*n***	**0:OK** if parameters accepted **1:CMD ERR** if RS232 port not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **DLESE=?**	List of ***num:description*** if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level

DLMRD (**D**ata **L**ogger **M**in/max **R**ea**D**) Reads the min/max stored values.	Read command: **DLMRD?**	Min/max values in CSV format if accepted **1:CMD ERR** if logging functions not enabled
	Set command: NOT SUPPORTED	**1:CMD ERR** if command is issue
	Help command: NOT SUPPORTED	**1:CMD ERR** if command is issued
DLMRE (**D**ata **L**ogger **M**in/max **RE**set) Resets the min/max stored values.	Read command: NOT SUPPORTED	**1:CMD ERR** if command is issued
	Set command: **DLMRE=1**	**0:OK** if parameters accepted **1:CMD ERR** if logging functions not enabled **5:ACCESS ERR** if insufficient access level
	Help command: **DLMRE=?**	**1:EXECUTE** if accepted **1:CMD ERR** if logging functions not enabled **.** **5:ACCESS ERR** if insufficient access level
DLUSE (**D**ata **L**ogger measure **U**nits **S**end **E**nable) Enables or disables the sending of the measure units of the loggers data by email using the GPRS terminal. Not enabling this option makes the data more compact to send by wireless communication.	Read command: **DLUSE?**	Status of measure units send if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **DLUSE=*n***	**0:OK** if parameters accepted **1:CMD ERR** if RS232 port not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **DLUSE=?**	List of ***num:description*** if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level
DLTST (**D**ata **L**ogger **T**est **S**tart **T**ime) Reads or sets the date and time start value used for the "step check" test. **WARNING: when setting the date and time, always check the value returned because malformed input string may lead to incorrect time result.**	Read command: **DLTST?**	Value of date and time if accepted **1:CMD ERR** if logging functions not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **DLTST=*tmstring*** Required access level: **2**	Value of date and time set **1:CMD ERR** if logging functions not enabled **5:ACCESS ERR** if insufficient access level
	Help command: **DLTST=?**	Format of the string to be input **1:CMD ERR** if logging functions not enabled **5:ACCESS ERR** if insufficient access level
DLTPT (**D**ata **L**ogger **T**est sto**P** **T**ime) Reads or sets the date and time stop value used for the "step check" test. **WARNING: when setting the date and time,**	Read command: **DLTPT?**	Value of date and time if accepted **1:CMD ERR** if logging functions not enable **5:ACCESS ERR** if insufficient access level

always check the value returned because malformed input string may lead to incorrect time result.	Set command: **DLTPT=*tmstring***	Value of date and time set **1:CMD ERR** if logging functions not enabled **5:ACCESS ERR** if insufficient access level
	Help command: **DLTPT=?**	Format of the string to be input **1:CMD ERR** if logging functions not enabled **5:ACCESS ERR** if insufficient access level
DLTTI (**D**ata **L**ogger **T**est **T**ime **I**nterval) Reads or sets the time interval used for the “step check” test. The time is in minutes.	Read command: **DLTTI?**	Value of time interval if accepted **1:CMD ERR** if logging functions not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **DLTTI=*n***	**0:OK** if parameters accepted **1:CMD ERR** if logging functions not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **DLTTI=?**	***min <> max (units)*** if accepted **1:CMD ERR** if logging functions not enabled **5:ACCESS ERR** if insufficient access level
DTSON (**D**ata logger **T**est **S**tart **ON**) Starts the “step check” test mode and reads the status of the operation (1 = in progress , 0 = terminated). If the stop date is less than the current time and date, an error code is returned	Read command: **DTSON?**	Status of operation if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **DTSON=1**	**0:OK** if parameters accepted **2:PARAM ERR** if *n* not equal to 1 **3:EXEC ERR** if operation not possible **5:ACCESS ERR** if insufficient access level
	Help command: **DTSON=?**	**1:EXECUTE** if accepted **5:ACCESS ERR** if insufficient access level
DTSOF (**D**ata logger **T**est **S**tart **OF**f) Stops or suspend the “step check” test mode.	Read command: NOT SUPPORTED	**1:CMD ERR** if command is issued
	Set command: **DTSOF=1**	**0:OK** if parameters accepted **2:PARAM ERR** if *n* not equal to 1 **5:ACCESS ERR** if insufficient access level
	Help command: **DTSOF=?**	**1:EXECUTE** if accepted **5:ACCESS ERR** if insufficient access level

COMMANDS AND FUNCTIONS EQUIVALENT TO THE “10-DIAGNOSTIC” MENU		
Name and description	Modes	Returned values or codes
CALIC (**CAL**ibration **I**mmediate **C**ommand) performs an immediate calibration cycle and reads the status of the operation (1 = in progress , 0 = terminated).	Read command: **CALIC?**	Status of calibration operation if accepted
	Set command: **CALIC=1**	**0:OK** if parameters accepted **2:PARAM ERR** if *n* not equal to 1
	Help command: **CALIC=?**	**1:EXECUTE** if accepted

ATSIC (**A**uto-**T**e**S**t **I**mmediate **C**ommand) Performs an auto-test cycle.	Read command: NOT SUPPORTED	**1:CMD ERR** if command is issued
	Set command: **ATSIC=1**	**0:OK** if parameters accepted **2:PARAM ERR** if *n* not equal to 1 **5:ACCESS ERR** if insufficient access level
	Help command: **ATSIC=?**	**1:EXECUTE** if accepted **5:ACCESS ERR** if insufficient access level
MSIEN (**M**easure **SI**mulation **EN**able) Enables or disables the measure simulation function.	Read command: **MSIEN?**	Status of measure sim. function if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **MSIEN=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **MSIEN=?**	List of ***num:description*** if accepted **5:ACCESS ERR** if insufficient access level
STBYC (STandBY Command) set converter in stand-by mode **Once this command is issued , the only way to wakeup the converter is by the keyboard**	Read command: **STBYC =?**	List of ***num:description*** if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **STBYC =*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level

COMMANDS AND FUNCTIONS EQUIVALENT TO THE "11-INTERNAL DATA" MENU		
Name and description	Modes	Returned values or codes
L2ACD (**L**evel **2** **A**ccess **C**o**D**e) Reads or sets the level 2 programmable access code. It can be set to zero to disable all L2 access requests. **WARNING: do not forget the code entered!**	Read command: **L2ACD?**	Programmed code if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **L2ACD=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **L2ACD=?**	***min <> max*** if accepted **5:ACCESS ERR** if insufficient access level
LFDIC (**L**oad **F**actory **D**efaults **I**mmediate **C**ommand) Loads the factory default parameters. **WARNING: all the current working parameters eventually modified by the user will be lost!**	Read command: NOT SUPPORTED	**1:CMD ERR** if command is issued
	Set command: **LFDIC=1**	**0:OK** if parameters accepted **2:PARAM ERR** if *n* not equal to 1 **3:EXEC ERR** if default data corrupted **5:ACCESS ERR** if insufficient access level
	Help command: **LFDIC=?**	**1:EXECUTE** if accepted **5:ACCESS ERR** if insufficient access level
LUDIC (**L**oad **U**ser **D**ata **I**mmediate **C**ommand) Loads the user preset parameters.	Read command: NOT SUPPORTED	**1:CMD ERR** if command is issued

WARNING: all the current working parameters eventually modified by the user will be lost!	Set command: **LUDIC=1**	**0:OK** if parameters accepted **2:PARAM ERR** if *n* not equal to 1 **3:EXEC ERR** if user data corrupted **5:ACCESS ERR** if insufficient access level
	Help command: **LUDIC=?**	**1:EXECUTE** if accepted **5:ACCESS ERR** if insufficient access level
SUDIC (**S**ave **U**ser **D**ata **I**mmediate **C**ommand) Saves the current working parameters as user preset data.	Read command: NOT SUPPORTED	**1:CMD ERR** if command is issued
	Set command: **SUDIC=1**	**0:OK** if parameters accepted **2:PARAM ERR** if *n* not equal to 1 **3:EXEC ERR** if memory failure **5:ACCESS ERR** if insufficient access level
	Help command: **SUDIC=?**	**1:EXECUTE** if accepted **5:ACCESS ERR** if insufficient access level
SFDIC (**S**ave **F**actory **D**efaults **I**mmediate **C**ommand) Saves the current working parameters as factory defaults preset data.	Read command: NOT SUPPORTED	**1:CMD ERR** if command is issued
	Help command: **SFDIC=?**	**1:EXECUTE** if accepted **5:ACCESS ERR** if insufficient access level
CMRIC (**C**omplete **M**emory **R**eset **I**mmediate **C**ommand) Resets all the working parameters to the default values. **WARNING: all the current working parameters eventually modified by the user and the calibration coefficients will be lost!**	Read command: NOT SUPPORTED	**1:CMD ERR** if command is issued
	Help command: **CMRIC=?**	**1:EXECUTE** if accepted **5:ACCESS ERR** if insufficient access level
SRNUM (**S**e**R**ial **NUM**ber) Reads the device serial number.	Read command: **SRNUM?**	Value of serial number if accepted **5:ACCESS ERR** if insufficient access level
	Set command: NOT SUPPORTED	**1:CMD ERR** if command is issued
	Help command: NOT SUPPORTED	**1:CMD ERR** if command is issued
TONTM (**T**otal **ON** **T**i**M**e) Reads the total functioning time of the device.	Read command: **TONTM?**	Value of total on time if accepted
	Set command: NOT SUPPORTED	**1:CMD ERR** if command is issued
	Help command: NOT SUPPORTED	**1:CMD ERR** if command is issued
CFFKT (**C**oe**FF**icient **KT**) Reads or sets the value of the gain coefficient KT	Read command: **CFFKT?**	Value of the coefficient if accepted **5:ACCESS ERR** if insufficient access level
	Help command: **CFFKT=?**	***min <> max*** if accepted **5:ACCESS ERR** if insufficient access level
CFFKR (**C**oe**FF**icient **KR**) Reads or sets the value of the gain coefficient KR	Read command: **CFFKR?**	Value of the coefficient if accepted **5:ACCESS ERR** if insufficient access level

	Set command: **CFFKR=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **CFFKR=?**	***min <> max*** if accepted **5:ACCESS ERR** if insufficient access level
CFFKS (**C**oe**FF**icient **KS**) Reads or sets the value of the gain coefficient KS	Read command: **CFFKS?**	Value of the coefficient if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **CFFKS=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **CFFKS=?**	***min <> max*** if accepted **5:ACCESS ERR** if insufficient access level
CFFKZ (**C**oe**FF**icient **KZ**) Reads or sets the value of the zero coefficient KZ	Read command: **CFFKZ?**	Value of the coefficient if accepted **5:ACCESS ERR** if insufficient access level
	Help command: **CFFKZ=?**	***min <> max*** if accepted **5:ACCESS ERR** if insufficient access level
ICALE (**I**gnore **CAL**ibration **E**rrors) Enables or disables the calibration errors recognition.	Read command: **ICALE?**	Status of cal. error recognition if accepted **5:ACCESS ERR** if insufficient access level
	Set command: **ICALE=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **ICALE=?**	List of ***num:description*** if accepted **5:ACCESS ERR** if insufficient access level
C1CP1 (**C**urrent output **1** **C**alibration **P**oint **1**) Reads or sets the value of the current output calibration point 1 (4 mA).	Read command: **C1CP1?**	Value of the cal. point if accepted **1:CMD ERR** if current output not enabled **5:ACCESS ERR** if insufficient access level
	Help command: **C1CP1=?**	***min <> max*** if accepted **1:CMD ERR** if current output not enabled **5:ACCESS ERR** if insufficient access level
C1CP2 (**C**urrent output **1** **C**alibration **P**oint **2**) Reads or sets the value of the current output calibration point 2 (20 mA).	Read command: **C1CP2?**	Value of the cal. point if accepted **1:CMD ERR** if current output not enabled **5:ACCESS ERR** if insufficient access level
	Help command: **C1CP2=?**	***min <> max*** if accepted **1:CMD ERR** if current output not enabled **5:ACCESS ERR** if insufficient access level

A1CP1 (**A**nalog input **1** **C**alibration **P**oint **1**) Reads or sets the value of the current output calibration point 1 (4 mA or min. voltage value).	Read command: **A1CP1?**	Value of the cal. point if accepted **1:CMD ERR** if analog input not enabled **5:ACCESS ERR** if insufficient access level
	Help command: **A1CP1=?**	***min <> max*** if accepted **1:CMD ERR** if analog input not enabled **5:ACCESS ERR** if insufficient access level
A1CP2 (**A**nalog input **1** **C**alibration **P**oint **2**) Reads or sets the value of the current output calibration point 2 (20 mA or max. voltage value).	Read command: **A1CP2?**	Value of the cal. point if accepted **1:CMD ERR** if analog input not enabled **5:ACCESS ERR** if insufficient access level
	Help command: **A1CP2=?** Required access level: **2**	***min <> max*** if accepted **1:CMD ERR** if analog input not enabled **5:ACCESS ERR** if insufficient access level

COMMANDS AND FUNCTIONS USED FOR READING THE PROCESS PARAMETERS

Name and description	Modes	Returned values or codes
FRSRN (**F**low **R**ate **S**cale **R**ange **N**umber) Reads the scale range currently in use on the instrument. Valid only if the dual-range function is active.	Read command: **FRSRN?**	Scale number: (1 or 2) if dual range active **1:CMD ERR** if dual range not enabled
	Set command: NOT SUPPORTED	**1:CMD ERR** if command is issued
	Help command: NOT SUPPORTED	**1:CMD ERR** if command is issued
FRVPC (**F**low **R**ate **V**alue **P**er**C**ent) Reads or sets the flow rate value in the percent form. Returns two comma-separated fields, the percent symbol and the numeric value. Percent value is referred to the full scale currently active. The flow rate value can be set only when the simulation mode is active.	Read command: **FRVPC?**	Returns *%,value* string
	Set command: **FRVPC=*n***	**0:OK** if parameters accepted **2:PARAM ERR** if *n* out of range or simulation mode not enabled
	Help command: **FRVPC=?**	***min <> max (%)***
FRVTU (**F**low **R**ate **V**alue **T**echnical **U**nits) Reads the flow rate value and its technical measure unit. Returns two comma-separated fields, the technical unit symbol and the numeric value.	Read command: **FRVTU?**	Returns *measure-unit,value* string
	Set command: NOT SUPPORTED	**1:CMD ERR** if command is issued
	Help command: NOT SUPPORTED	**1:CMD ERR** if command is issued
VTTPV (**V**olume **T**otalizer **T**otal **P**ositive **V**alue) Reads the positive total totalizer value and its technical measure unit. Returns two comma-separated fields, the technical unit symbol and the numeric value.	Read command: **VTTPV?**	Returns *measure-unit,value* string
	Set command: NOT SUPPORTED	**1:CMD ERR** if command is issued
	Help command: NOT SUPPORTED	**1:CMD ERR** if command is issued
VTPPV (**V**olume **T**otalizer **P**artial	Read command: **VTPPV?**	Returns *measure-unit,value* string

Positive **V**alue) Reads the positive partial totalizer value and its technical measure unit. Returns two comma-separated fields, the technical unit symbol and the numeric value.	Set command: NOT SUPPORTED	**1:CMD ERR** if command is issued
	Help command: NOT SUPPORTED	**1:CMD ERR** if command is issued
VTTNV (**V**olume **T**otalizer **T**otal **N**egative **V**alue) Reads the negative total totalizer value and its technical measure unit. Returns two comma-separated fields, the technical unit symbol and the numeric value.	Read command: **VTTNV?**	Returns *measure-unit,value* string
	Set command: NOT SUPPORTED	**1:CMD ERR** if command is issued
	Help command: NOT SUPPORTED	**1:CMD ERR** if command is issued
VTPNV (**V**olume **T**otalizer **P**artial **N**egative **V**alue) Reads the negative partial totalizer value and its technical measure unit. Returns two comma-separated fields, the technical unit symbol and the numeric value.	Read command: **VTPNV?**	Returns *measure-unit,value* string
	Set command: NOT SUPPORTED	**1:CMD ERR** if command is issued
	Help command: NOT SUPPORTED	**1:CMD ERR** if command is issued
CUTPV (**CU**rrency **T**otalizer **P**ositive **V**alue) Reads the converted currency value relative to the partial positive totalizer and its currency unit. Returns two comma-separated fields, the unit symbol and the numeric value.	Read command: **CUTPV?**	Returns *currency-symbol,value* string **1:CMD ERR** if currency mode not enabled
	Set command: NOT SUPPORTED	**1:CMD ERR** if command is issued
	Help command: NOT SUPPORTED	**1:CMD ERR** if command is issued
CUTNV (**CU**rrency **T**otalizer **N**egative **V**alue) Reads the converted currency value relative to the partial negative totalizer and its currency unit. Returns two comma-separated fields, the unit symbol and the numeric value.	Read command: **CUTNV?**	Returns *currency-symbol,value* string **1:CMD ERR** if currency mode not enabled
	Set command: NOT SUPPORTED	**1:CMD ERR** if command is issued
	Help command: NOT SUPPORTED	**1:CMD ERR** if command is issued
AIN1V (**A**nalog **IN**put **1** **V**alue) Reads the value of the analog input 1 and its associated technical unit. Returns two comma-separated fields, the unit symbol and the numeric value.	Read command: **AIN1V?**	Returns *measure-unit,value* string **1:CMD ERR** if analog input 1 not enabled
	Set command: NOT SUPPORTED	**1:CMD ERR** if command is issued
	Help command: NOT SUPPORTED	**1:CMD ERR** if command is issued
BATTS (**BATT**ery **S**tatus) Reads the estimated residual capacity of the battery. Returns the percent symbol and value separated by the comma.	Read command: **BATTS?**	Returns *%,value* string
	Set command: NOT SUPPORTED	**1:CMD ERR** if command is issued
	Help command: NOT SUPPORTED	**1:CMD ERR** if command is issued
ALARM (**ALARM** status) Reads the alarm status of the instrument. Returns a CSV list of all active alarm.	Read command: **ALARM?**	Returns all active alarms in CSV format
	Set command: NOT SUPPORTED	**1:CMD ERR** if command is issued
	Help command: NOT SUPPORTED	**1:CMD ERR** if command is issued

AUXILIARY SET COMMAND

Name and description	Modes	Returned values or codes
ACODE (Access CODE)	Read command: **ACODE?** Required access level: **2**	String value if command accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **ACODE =*string***	**0:OK** if parameters accepted **1:CMD ERR** if GPRS not enabled **2:PARAM ERR** if *string* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **ACODE =?**	**NUMERIC RANGE** if accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
MODSV (MODel and Software Version)	Read command: **MODSV?**	String value if command accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
CLIST(Command LIST)	Read command: **CLIST?**	String value if command accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
CFLST (Configuration LiST)	Read command: **CFLST?**	The list command if command accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level

COMMANDS AND FUNCTIONS SPECIFIC TO GPRS COMMUNICATION

Name and description	Modes	Returned values or codes
GPAPN (**GP**rs **A**ccess **P**oint **N**ame) Reads or sets the GPRS network access point name. This parameter must be set accordingly to what is specified by the wireless operator and it is a string of max. 31 characters length.	Read command: **GPAPN?**	String value if command accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **GPAPN=*string***	**0:OK** if parameters accepted **1:CMD ERR** if GPRS not enabled **2:PARAM ERR** if *string* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **GPAPN=?**	**MAX 31 CHR STRING** if accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
GPUSR (**GP**rs **US**e**R**) Reads or sets the GPRS network access user name. This parameter must be set accordingly to what is specified by the wireless operator and it is a string of max. 15 characters length.	Read command: **GPUSR?**	String value if command accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **GPUSR=*string***	**0:OK** if parameters accepted **1:CMD ERR** if GPRS not enabled **2:PARAM ERR** if *string* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **GPUSR=?**	**MAX 15 CHR STRING** if accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
GPPSW (**GP**rs **P**a**S**s**W**ord) Sets the GPRS network access password. This parameter must be set accordingly to what is specified by the wireless operator and it is a string of max. 7 characters length.	Read command: NOT SUPPORTED	**1:CMD ERR** if command issued **5:ACCESS ERR** if insufficient access level
	Set command: **GPPSW=*string***	**0:OK** if parameters accepted **1:CMD ERR** if GPRS not enabled **2:PARAM ERR** if *string* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **GPPSW=?**	**MAX 7 CHR STRING** if accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level

GPEMF (**GP**rs **EM**ail **F**rom) Reads or sets the GPRS network email sender address. This parameter must be set to an existing mail address. The maximum length is 31 characters. **Address of sender : this address will be used in case of the mail server cannot deliver the message to the final destination. In this case the message will be bounced to the sender, so it is possible to detect the error.**	Read command: **GPEMF?**	String value if command accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **GPEMF=*string***	**0:OK** if parameters accepted **1:CMD ERR** if GPRS not enabled **2:PARAM ERR** if *string* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **GPEMF=?**	**MAX 31 CHR STRING** if accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
GPEMT (**GP**rs **EM**ail **T**o) Reads or sets the GPRS network email receiver address. This parameter must be set to an existing mail address. The maximum length is 31 characters. **Address where the converter send the e-mail (data)**	Read command: **GPEMT?**	String value if command accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **GPEMT=*string***	**0:OK** if parameters accepted **1:CMD ERR** if GPRS not enabled **2:PARAM ERR** if *string* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **GPEMT=?**	**MAX 31 CHR STRING** if accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
GPURL (**GP**rs **URL**) Reads or sets the GPRS network URL used to fetch the dynamic document in the http server that performs the time synchronization. The trailing "/" preceding the URL and the eventual "http://" must be omitted. The maximum length is 31 characters.	Read command: **GPURL?**	String value if command accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **GPURL=*string***	**0:OK** if parameters accepted **1:CMD ERR** if GPRS not enabled **2:PARAM ERR** if *string* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **GPURL=?**	**MAX 31 CHR STRING** if accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
GPASN (**GP**rs **A**uthorized **S**ender **N**umber) Reads or sets the GPRS network SMS authorized number from which the device can accept SMS commands. The value can match exactly the sender number or can be the first part only, leaving the possibility to use adjacent valid sender numbers. The maximum length is 18 characters.	Read command: **GPASN?**	String value if command accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **GPASN=*string***	**0:OK** if parameters accepted **1:CMD ERR** if GPRS not enabled **2:PARAM ERR** if *string* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **GPASN=?**	**MAX 18 CHR STRING** if accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
GPAUT (**GP**rs **A**uthentication **T**ype) Reads or sets the GPRS authentication type used to access the network. This parameter must be set accordingly to what specifies the wireless operator and it can be: 0=Normal (PAP), 1=Secure (CHAP), 2=None.	Read command: **GPAUT?**	value of auth. type if command accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **GPAUT=*n***	**0:OK** if parameters accepted **1:CMD ERR** if GPRS not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **GPAUT=?**	***min <> max*** if accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
GPESE (**GP**rs **E**mail **S**end **E**nable) Enables or disables the sending of the emails using the GPRS terminal.	Read command: **GPESE?**	Status of email send functions if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level

Valid only if the RS232 port is installed and enabled. Valid only if the GPRS terminal is installed and enabled.	Set command: **GPESE=*n***	**0:OK** if parameters accepted **1:CMD ERR** if RS232 port not enabled **2:PARAM ERR** if *n* out of rang **5:ACCESS ERR** if insufficient access level
	Help command: **GPESE=?**	List of ***num:description*** if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level
GPDTS (**GP**rs **D**a**T**a **S**end) Sends an email containing data coming from the enabled loggers using the GPRS terminal.	Read command: NOT SUPPORTED	**1:CMD ERR** if read command is issued
	Set command: **GPDTS=1**	**0:OK** if parameters accepted **1:CMD ERR** if RS232 port not enabled **2:PARAM ERR** if *n* not equal to 1 **5:ACCESS ERR** if insufficient access level
	Help command: **GPDTS=?**	**1:EXECUTE** if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level
GPCFS (**GP**rs **C**on**F**iguration **S**end) Sends an email containing all the configuration data of the instrument using the GPRS terminal.	Read command: NOT SUPPORTED	**1:CMD ERR** if read command is issued **5:ACCESS ERR** if insufficient access level
	Set command: **GPCFS=1**	**0:OK** if parameters accepted **1:CMD ERR** if RS232 port not enabled **2:PARAM ERR** if *n* not equal to 1 **5:ACCESS ERR** if insufficient access level
	Help command: **GPCFS=?**	**1:EXECUTE** if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level
GPESI (**GP**rs **E**mail **S**end **I**nterval) Reads or sets the value of the time interval used to send the emails.	Read command: **GPESI?**	Email send time interval if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **GPESI=*n***	**0:OK** if parameters accepted **1:CMD ERR** if RS232 port not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **GPESI=?**	List of ***num:description*** if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level
GPSCI (**GP**rs **S**ms **C**heck **I**nterval) Reads or sets the value of the time interval used to check the sms.	Read command: **GPSCI?**	SMS check interval if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **GPSCI=*n***	**0:OK** if parameters accepted **1:CMD ERR** if RS232 port not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **GPSCI=?**	List of ***num:description*** if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level
GPCSE (**GP**rs **C**lock **S**ync **E**nable) Enables or disables the synchronization of the clock using the GPRS terminal.	Read command: **GPCSE?**	Status of clock sync functions if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **GPCSE=*n***	**0:OK** if parameters accepted **1:CMD ERR** if RS232 port not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level

	Help command: **GPCSE=?**	List of ***num:description*** if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level
GPSMP (**GP**rs **SM**tp **P**ort) Reads or sets the GPRS network SMTP port used to exchange the email data. Usually is the TCP port 25.	Read command: **GPSMP?**	value of SMTP port if command accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **GPSMP=*n***	**0:OK** if parameters accepted **1:CMD ERR** if GPRS not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **GPSMP=?**	***min <> max*** if accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
GPHTP (**GP**rs **HT**tp **P**ort) Reads or sets the GPRS network HTTP port used to exchange the time synchronization data. Usually is the TCP port 80, but BEWARE: some wireless operator block this port and reroute it to their proxy servers.	Read command: **GPHTP?**	value of HTTP port if command accepted **1:CMD ERR** if GPRS not enable **5:ACCESS ERR** if insufficient access level
	Set command: **GPHTP=*n***	**0:OK** if parameters accepted **1:CMD ERR** if GPRS not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **GPHTP=?**	***min <> max*** if accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
GPSMA (**GP**rs **SM**tp **A**ddress) Reads or sets the GPRS network SMTP server address used to exchange the email data. It must be specified in the IP-range notation.	Read command: **GPSMA?**	IP address if command accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **GPSMA=*ip-range***	**0:OK** if parameters accepted **1:CMD ERR** if GPRS not enabled **2:PARAM ERR** if *ip-range* malformed **5:ACCESS ERR** if insufficient access level
	Help command: **GPSMA=?**	***xxx.xxx.xxx.xxx*** if accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
GPHTA (**GP**rs **HT**tp **A**ddress) Reads or sets the GPRS network HTTP server address used to exchange the clock synchronization data. It must be specified in the IP-range notation.	Read command: **GPHTA?**	IP address if command accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **GPHTA=*ip-range***	**0:OK** if parameters accepted **1:CMD ERR** if GPRS not enabled **2:PARAM ERR** if *ip-range* malformed **5:ACCESS ERR** if insufficient access level
	Help command: **GPHTA=?**	***xxx.xxx.xxx.xxx*** if accepted **1:CMD ERR** if GPRS not enabled **5:ACCESS ERR** if insufficient access level
GPSME (**GP**rs **SM**s **S**end **E**nable) Enables or disables the functions related to the SMS using the GPRS terminal.	Read command: **GPSME?**	Status of SMS functions if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level
	Set command: **GPSME=*n***	**0:OK** if parameters accepted **1:CMD ERR** if RS232 port not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **GPSME=?**	List of ***num:description*** if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level
GPSSN (Gprs Sms Send Number) telephone number to sent the SMS	Read command: **GPSSN?**	Number set if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level

	Set command: **GPSSN =*n***	**0:OK** if parameters accepted **1:CMD ERR** if RS232 port not enabled **2:PARAM ERR** if *n* out of range **5:ACCESS ERR** if insufficient access level
	Help command: **GPSSN =?**	**MAX 19 CHR STRING** if accepted **1:CMD ERR** if RS232 port not enabled **5:ACCESS ERR** if insufficient access level

4. HTP PROTOCOL (HYPER TEXT PROTOCOL)

4.1. INTRODUCTION

HTP protocol is a set of command used only with the RS 232 serial port of the converter.

A standard application suitable for this purpose is the Windows program "**HyperTerminal**".

4.2. DESCRIPTION AND USE

Connect the flow meter to the PC with the IF21 cable.

On the flow meter, set the function "**IF2 prot.= HTP**" on the menu "**7-Communication**".

Start the HyperTerminal application and set its property as follows:

- COM port: select DIRECT CONNECTION with the port where the IF2 cable is plugged and open the dialog for configuring the port, set the following property:
  - Bits per second: 38400
  - Data bits: 8
  - Parity: none (no parity)
  - Stop bits: 1
  - Flow control: none (no control lines nor xon/xoff characters used)
- Open the Settings tab and set terminal type as "ANSI", leave all parameters set as default, open "ASCII setup" dialog and check the following:
  - ASCII Sending: check the "Send line ends…" box
  - Line delay: set to 1000 milliseconds
  - Character delay: set to 10 milliseconds
  - Verify that the box "Echo typed characters locally" is UNCHECKED, if it is checked, uncheck it
  - ASCII Receiving: check the last box "Wrap lines…", all others must be unchecked

When the settings are finished and if it is not yet on-line, click on the icon "connect" and if you want, save the settings with a new name. Try to type the command **MODSV?** and confirm it pressing the "enter" key. The flow meter should respond with its model name and software version. If the flow meter doesn't respond, verify that it is alive and all the connections and the settings are ok. Try also to send the command more times, since the flow meter may shut down the communication automatically when it detects no line activity for more than 30 seconds. When the communication is successful, the system is ready to enter the configuration data, as explained in the next section.

It is also possible to retrieve and to save the complete set of configuration parameters of the instrument. To do so, select "Transfer" on the menu bar of the Hyper terminal and choose "Capture text". Give a file name in the dialog box that will appear. Write on the screen the command "**ACODE=xxxxx,CFLST?**" , where xxxxx stays for the Level 2 code set in the instrument, and press "enter". You will see a series of lines scrolling on the display. When the listing is finished, go to the "Transfer" menu, select "Capture Text" and finally "Stop". The file saved contains the complete configuration list.

If now you want to set the complete parameters list into the instrument, proceed as illustrated.

- Open the saved file with Notepad or similar application and delete the first and the last lines, where you see the "**ACODE=xxxxx,CFLST?**" command and the "**0:OK**" answer.
- Save the modified file and close the Notepad application.
- Send the following command to the instrument: "**ACODE=xxxxx,L2ACD=0**", where the xxxxx stays for the actual L2 code set in the instrument. This disables the Level 2 code, otherwise it should be necessary to repeat the command "ACODE=xxxxx" at each line.
- Select "Transfer" on the menu bar of the Hyper terminal, and choose "Send Text": a dialog box window will appear
- Set the filename that you have modified before as the file to be send in the dialog box and press "enter"
- The display will show a series of lines scrolling, with all the configuration parameters contained in the file.
- Wait until the last command is sent.
- Eventually set again the Level 2 code with the command "L2ACD=xxxxx". Do not forget the code entered now!

NOTE: some commands sent with this method probably will fail with a "1:CMD ERR", "2:PARAM ERR" or "5:ACCESS ERR" message. This is normal because some parameters can be read at certain access level but they can't be written at the same level, or some others can be read but they can't be written in the actual configuration context.

4.3. HTP PROTOCOL

The HTP protocol (with only one "T", to be not confused with the HTTP protocol, which is a completely different thing) is a very simple way to exchange data between the flow meter and an application like the Windows-based Hyper-Terminal. With the HTP protocol it is possible to read and to write the parameters of the flow meter and to perform some internal functions. The data are exchanged using ASCII strings terminated by the carriage-return character. The maximum size of the input or the output strings is about 1000 characters when a terminal is attached directly to the serial port, or 160 characters if the communications is performed using SMS. When the maximum size is exceeded in the input, the entire string entered is discarded and an error message is issued. When the maximum size is exceeded in the output, the execution of the commands is still performed but the output information are lost and an error message is issued. The output string length depends exclusively on the type of commands given in the input, some commands produces a very short answer and some other gives a very long string. Care must be taken to not saturate the output capacity, otherwise the expected information will be lost.

IMPORTANT: Before accessing or modifying some parameters, a privilege level must be acquired. This can be done sending an access code that matches the L2 code of flow meter as first command in the input string (ACODE=*n*). The code lifetime is limited to the execution of the string that contain it, when the input string is evaluated completely, it expires. Thus it is necessary to provide the access code each time a parameter must be changed or inspected. If the L2 code of the instrument is set to zero, the access code is no more necessary.

Special characters. The following characters have special meaning in the protocol and thus they can't be used for other purposes:

- **<CR>** carriage-return character, value 13 decimal, 0D hexadecimal, terminates the input string and starts the elaboration
- **<LF>** line-feed character, value 10 decimal, 0A hexadecimal, may follows the <CR> but it is never considered
- **?** question mark character, value 63 decimal, 3F hexadecimal, it is an *operator* character
- **=** equal sign character, value 61 decimal, 3D hexadecimal, it is another *operator* character
- **:** colon character, value 58 decimal, 3A hexadecimal, it is the *comment-separator* character
- **,** comma character, value 44 decimal, 2C hexadecimal, it is the *command-separator* character

General input syntax. The information are entered in the flow meter as text line strings, with one or more *command-sequences* terminated by the <CR> character. The *command-sequences* are executed in the same order as they are found in the string. The execution of the commands contained in the input string does not start until the <CR> character is received. The optional <LF> character that may follows the <CR> is not considered but it is accepted because usually the Hyper-Terminal or similar application sends also this extra character when the <CR> is sent.

As a rule, an input string is composed of one or more *command-sequences*, terminated by the <CR> character and an optional <LF> character.

The *command-sequence* is composed by the following elements, exactly in this order:

- A five-letter mnemonic *command*, always present
- An *operator*, always present
- An optional *value*, present only when requested by the *operator* type
- An optional *comment-separator*, may be present if it is also present the *value*
- A *comment*, present only if it is also present the *comment-separator*
- An optional *command-separator*, present only if another *command-sequence* follows it

With the exception of the *comment* element, no other extra characters or spaces are allowed in the *command-sequence*.

Command. The commands are always represented by a five-letters mnemonic code and are case insensitive, so for example the command MODSV can be written "MODSV", "Modsv", "modsv", "mOdSv" and in any combination of upper / lower case letters.

Operator. The operators permit to choose one of the three possible functions associated to the command at which they are attached and they are:

- **READ**, indicated by the **?** symbol. It is used to read values.
- **SET**, indicated by the **=** symbol. It is used to set values.
- **HELP**, indicated by the **=?** sequence of symbols. It is used to display a set of options or a range of permissible values related to the *command*.

Value. The values can be numbers, strings or special formatted fields like the date / time or the IP addresses, depending on what it is expected by the *command*. Numeric values are always checked for validity range and strings are checked for length. IP addresses and date / time fields are checked only for the correct syntax but not for the values, so please be careful, because in case of misspelled characters or wrong numeric values the result may be different from what it is expected. In case of floating-point numbers, the decimal point symbol to be used is the dot (**.**), not the comma (**,**).

Comment-separator. This is an optional element and it is indicated by the **:** symbol.

Comment. This element may be present only when the *value* to input belongs to a list of options, composed by numbers and descriptions. Normally the user doesn't have to supply both, but in case of copying and pasting some values coming from a previously listed configuration, this ensures the full compatibility between the output and the input formats.

Command-separator. This element is required when more than one *command-sequence* is submitted in an input string and it is indicated with the **,** symbol.

For each *command-sequence* recognized and executed, the flow meter returns one of the following output types, depending on it:

- a *result code*, when a function execution was requested
- an *expression*, when a parameter or a process data value was requested
- a *list of options* or a *range of values*, when an help on a parameter was requested

Each answer is separated from the other by the comma symbol (the same used as *command-separator*). The complete output string is terminated by a **<CR><LF>** sequence. Unrecognized *command-sequences* are silently discarded without response and without halting the execution of the next sequence, if it is present. Illegal parameter's values and operations performed in wrong contexts are reported and identified by error codes.

Result-code. The format of the results is the following: *code-number:description*, without any blank spaces separating the number from the description. There are six possible *result-codes:*

- **0:OK**, the execution was correct
- **1:CMD ERR**, wrong context, execution was not possible due to a configuration limit or wrong working conditions
- **2:PARAM ERR**, the expected parameter was out of the allowed range
- **3:EXEC ERR**, the execution of the command was not successful due to an internal error condition
- **4:RANGE ADJ**, the entered parameter caused an internal automatic adjustment on other ranges
- **5:ACCESS ERR**, the execution of the command was not possible due to an insufficient privilege level
- **6:BUFFER FULL**, the input or the output strings exceed the maximum allowable space.

Expression. An *expression* may be a number, a string or a combination of both. Usually this type of output is given when a read operation on some parameter is requested.

List of Option. This type of output is given when a help on a parameter that requires multiple options is requested. The structure is similar of that used for the *result-codes: option-number:description*.

Range of values. This type of output is given when a help on a parameter that requires a value is requested. In this case the output takes this form: *minimum-value <> maximum-value (units).*

Mnemonic command list. For the list of the command see the ETP protocol section The list of mnemonic are the same of the ETP protocol.

5. MODBUS FIELD BUS

5.1. INTRODUCTION

In this section of the manual there is the description of the Modbus field bus implemented in the converter.
The Modbus field bus is available in the RS232 and RS485 serial port of the converter.

5.2. RS485 HARDWARE CONNECTION

For the hardware connection see the relative section in this manual.

5.3. DATA WORD FORMAT

The data bytes travelling in serial form on the communication line are enclosed in ***words*** which have a fixed length of 10 bits:

1 START BIT
8 DATA BITS = 1 BYTE DI DATI
1 STOP BIT

Each word contains one byte of data plus additional bits which serve to synchronise and make the communication safer. These extra bits are added automatically in the transmission phase by the transmitter integrated circuit. In the reception phase, the reverse operation is executed by the receiver integrated circuit: the eight data bits are extracted and the others are eliminated. These operations are executed entirely on a hardware level.
The 8 data bits must be serialised staring from bit 0 (the least significant one).

5.4. COMMUNICATION SPEED

The millennium series instruments have 4 communication speeds::

- 4800 bps
- 9600 bps
- 19200 bps
- 38400 bps

5.5. SERIAL PORT SETTINGS

Serial port settings:

- Data bits: 8
- Parity: Menu «**7-Communication»**, function - «**Parity**»
- Stop bits: 1
- Flow control: none (no control lines no xon/xoff characters used)

5.6. PARAMETER SETTINGS

MODBUS PARAMETERS SETTINGS FOR THE CONVERTER:

- Menu «**7-Communication»**, function - «**Parity**»: set the control of the parity for the byte frame in MODBUS communication. This function affect all the serial port that communicate with MODBUS protocol. The possible value are: EVEN (default), NONE and ODD.
- Menu «**7-Communication»**, function - «**RS485 pr**.»: the function select protocol type for the RS 485 serial port. Select the value MODBUS.
- Menu «**7-Communication»**, function - «**RS232 pr**.»: the function select protocol type for the RS 232 serial port. Select the value MODBUS.

Refer to the relative section of the serial port selected for the other type of parameter like port speed etc.

5.7. STANDARD FUNCTIONS

For the MODBUS protocol, the converter ha the function of SLAVE device. The MASTER of the MODBUS network see the Converter as a set of register of 1 bit or 16 bit.
The data are supplied with list of fixed address grouped in tables with different length.

- **FUNCTION 01**, send of Status Batch. This command return variables of bit type.

Addresses(Hex format)	Bit Description	Converter Model
0000	1 = batch process is running, 0 = batch process correctly terminated	ML 210
0001	1 = batch process is suspended, 0 = batch process is running or terminated	ML 210

- **FUNCTION 03**, send process data, data logger data, data logger events and batch memory. This command return 16 bit variables that are linked to form floating-point (float) o 32 bit long-integer (long) variables type. The high side of the word (MSW) is on even address, the low side of the word (LSW) is in the immediately following address. The addresses are in HEX format.

NOTE: when the request is for the data of the data logger or the event logger that is not valid (data not still collected), the returned value are = FFFFFFFF (hex).

Addresses table:

Addresses(Hex format)	Description	Converter Model
0000-0025	process data	ML 210 – ML 211 – ML 212 – ML 110
0064-02E3	data logger value, grouped in 32 blocks contiguous from 20 registries of memory to 16 bit for	ML 210 – ML 211 – ML 212 – ML 110

	every block	
03E8-04E7	data logger events, grouped in 64 blocks contiguous from 4 register of memory to 16 bit for every block	ML 210 – ML 211 – ML 212 – ML 110
07D0-084F	values of batch memories, grouped in 64 blocks contiguous from 8 registers of memory to 16 bit for every block	ML 210
0BB8	index of the batch memory current in use	ML 210

Process Parameters

Addresses(Hex format)	**Description**	**Converter Model**
0000-0001	Flow rate in % (float)	ML 210 – ML 211 – ML 212 – ML 110
0002-0003	Flow rate in technical unit (float)	ML 210 – ML 211 – ML 212 – ML 110
0004-0005	Totalizer for total volume positive T+(V+ for ML211) (long)	ML 210 – ML 211 – ML 212 – ML 110
0006-0007	Totalizer for partial volume positive P+ (V- for ML211) (long)	ML 210 – ML 211 – ML 212 – ML 110
0008-0009	Totalizer for total volume negative T- (E+ for ML211) (long)	ML 210 – ML 211 – ML 212 – ML 110
000A-000B	Totalizer for partial volume negative P- (E- for ML211) (long)	ML 210 – ML 211 – ML 212 – ML 110
000C-000D	Clock value in seconds (long)	ML 210 – ML 211 – ML 212 – ML 110
000E-000F	Input value AIN1 in technical unit (opt.) (float)	ML 210
0010-0011	Input value AIN2 in technical unit (opt.) (float)	ML 210
0012-0013	Thermal power value in % (float)	ML 211
0014-0015	Thermal power value in technical unit (float)	ML 211
0016-0017	Delta T value in technical unit (float)	ML 211
0018-0019	Temperature value T1 in technical unit (float)	ML 211
001A-001B	Temperature T2 value in technical unit (float)	ML 211
001C-001D	Set-point value in % (float)	ML 212
001E-001F	Output value in % (float)	ML 212
0020-0021	Deviation value in % (float)	ML 212
0022	Process flags: Bit 0 (LSB) = excitation is too fast for the sensor connected / temperature error ML211 Bit 1 = max alarm is active Bit 2 = min alarm is active Bit 3 = flow rate exceeds the scale range, overflow Bit 4 = too many impulse to emit	ML 210 – ML 211 – ML 212 – ML 110

	Bit 5 = measurement signal is disturbed or sensor disconnect Bit 6 = empty tube Bit 7 = coils not working or sensor disconnect Bit 8 = second measurement scale is active Bit 9 = flow rate is lower than the cut-off threshold Bit10 = flow rate negative Bit11 = new measurement value is available for the display Bit12 = counter block signal is active Bit13 = batch in progress Bit14 = calibration cicle in progress Bit15 (MSB) = flow rate simulation in progress	
0023	Input Flags AIN1/2: Bit 2 = input error AIN1 Bit 3 = input error AIN2 (bit 0-1 e 4-15 not used, value = 0)	ML 210
0024	Flags ML211: Bit 0 (LSB) = max alarm thermal power Bit 1 = min alarm thermal power Bit 2 = max alarm delta T Bit 3 = min alarm delta T Bit 4 = max alarm temp. T1 Bit 5 = min alarm temp. T1 Bit 6 = max alarm temp. T2 Bit 7 = min alarm temp. T2 (bit 8-15 not used, value = 0)	ML 211
0025	Flags ML212: Bit 0 (LSB) = command error actuator Bit 1 = deviation error Bit 2 = Input error AIN1 Bit 3 = Input error AIN2 Bit 4 = manual regulation active Bit 5 = safety mode active (bit 6-15 not used, value = 0)	ML 212

Data Logger values

Addresses(Hex format)	Description	Converter Model
0064-0077	Data block record n.1 of the Data logger	ML 210 – ML 211 – ML 212 – ML 110
0078-008B	Data block record n.2 of the Data logger	ML 210 – ML 211 – ML 212 – ML 110
…	…	ML 210 – ML 211 – ML 212 – ML 110
…	…	ML 210 – ML 211 – ML 212 – ML 110
…	…	ML 210 – ML 211 – ML 212 – ML 110
02D0-02E3	Data block record n.32 of the Data logger	ML 210 – ML 211 – ML 212 – ML 110

In the following table there is the description of the structure of data of the first generation of the Data Logger, the following records are the same but in different addresses respect the new version.

Addresses(Hex format)	Description	Converter Model
0064-0065	Date and time of the record in second (long)	ML 210 – ML 211 – ML 212 – ML 110
0066-0067	Totalizer value positive (long)	ML 210 – ML 211 – ML 212 – ML 110
0068-0069	Totalizer value negative (long)	ML 210 – ML 211 – ML 212 – ML 110
006A-006B	Flow rate in technical unit (float)	ML 210 – ML 211 – ML 212 – ML 110
006C-006D	Energy value pos. (ML211) (long) / Input AIN1 (float)	ML 210 – ML 211
006E-006F	Energy value neg. (ML211) (long) / Input AIN2 (float)	ML 210 – ML 211
0070-0071	Thermal power in technical unit (float)	ML 211
0072-0073	Delta T value in technical unit (float)	ML 211
0074-0075	Temperature value T1 in technical unit (float)	ML 211
0076-0077	Temperature value T2 in technical unit (float)	ML 211

The returned values are at FFFFFFFF (hex) when the request data is not valid in the data logger (data not still collected).

Data Logger events

Addresses(Hex format)	Description	Converter Model
03E8-03EB	Data block record n.1 of the Event logger	ML 210 – ML 211 – ML 212 – ML 110
03EC-03EF	Data block record n.2 of the Event logger	ML 210 – ML 211 – ML 212 – ML 110
…	…	ML 210 – ML 211 – ML 212 – ML 110
…	…	ML 210 – ML 211 – ML 212 – ML 110
…	…	ML 210 – ML 211 – ML 212 – ML 110
04E4-04E7	Data block record n.64 of the Event logger	ML 210 – ML 211 – ML 212 – ML 110

In the following table there is the description of the data structure of the first generation of the event logger, the following records are the same but in different addresses respect the new version.

Addresses(Hex format)	Description	Converter Model
03E8-03E9	Date/Time of the record in seconds (long)	ML 210 – ML 211 – ML 212 – ML 110

03EA-03EB	Events (long):
	
	Bit 0 = err. Temp. probe (ML211) / Batch Alarm
	Bit 1 = max alarm flow rate
	Bit 2 = min alarm flow rate
	Bit 3 = scale range value overflow
	Bit 4 = output saturated impulses
	Bit 5 = input error measurement
	Bit 6 = measuring tube empty
	Bit 7 = coils excitation interrupted
	Bit 8 = max alarm thermal power (ML211)
	Bit 9 = min alarm thermal power (ML211)
	Bit10 = max alarm delta T (ML211)
	Bit11 = min alarm delta T (ML211)
	Bit12 = max alarm temp. T1 (ML211)
	Bit13 = min alarm temp. T1 (ML211)
	Bit14 = max alarm temp. T2 (ML211)
	Bit15 = min alarm temp. T2 (ML211)
	Bit16 = current loop 4-20 mA interrupted
	Bit17 = power supply error
	Bit18-23 = not used, value = 0
		ML 210 – ML 211 – ML 212 – ML 110

	Bit24 = command error actuator (ML212)
	Bit25 = deviation error (ML212)
	Bit26 = input error AIN1 (ML212 / ML210 opz.)
	Bit27 = input error AIN2 (ML212 / ML210 opz.)
	Bit28 = manual setting (ML212)
	Bit29 = safety output regulator (ML212)
	Bit30-31 = not used, value = 0	

The returned values are at FFFFFFFF (hex) when the request data is not valid in the data logger (data not still collected).

NOTA: when the value of the single long-integer variable, which contains the event flags, is 000300FF (hex) the meaning is a 'Restart of the converter' and the single bit has not significant.

Batch memory values

Addresses(Hex format)	Description	Converter Model
07D0-07D7	Data block memory batch n. 0	ML 210
07D8-07DF	Data block memory batch n. 1	ML 210
…	…	ML 210
…	…	ML 210
…	…	ML 210
0848-084F	Data block memory batch n. 15	ML 210

In the following table there is the structure of the data refer to the first batch memory, the other memories are the same but in different addresses.

Addresses(Hex format)	Description	Converter Model
07D0-07D3	Name of the product refer to the batch memory (8 length string characters)	ML 210
07D4	Counter of the number of batch with this product	ML 210
07D5	Safety timer value in tenths of second (10 = 1.0 seconds)	ML 210
07D6-07D7	Batch value in technical (long)	ML 210

NOTE: for reading the batch memories is necessary to set ON the Batch function, batch function.
If the batch function is disabled the system return the exception code n. 4.

Index value of the batch memory current in use

Addresses(Hex format)	Description	Converter Model
0BB8	Index value of the batch memory current in use	ML 210

NOTE: for reading the index value of the batch memory is necessary to activate the batch function.
If the batch function is disabled the system return the exception code n. 4.

- **FUNCTION 05**, execution of command of type on/off. For the standard MODBUS, this function is used only for set the state of variable type "coil" or bit, in this context the function of the variable is for change the activation state of a specific function. The two 16 bit words that follow the command are used for indicate the function type (address) and action (on or off). The "off" action don't have practical effect. To every address a various function corresponds. The code number for "on" is FF00 (hex), for "off" is 0000. Others values product the exception n.3.

Addresses(Hex format)	Function Description	Converter Model
0000	start / stop batch	ML 210
0001	reset batch	ML 210
0002	reset totalizers	ML 210
0003	reset data logger	ML 210
0004	reset events logger	ML 210

NOTE: for reading the index value of the batch memory is necessary to activate the batch function.
If the batch function is disabled the system return the exception code n. 4.

- **FUNCTION 08**, standard diagnose function of the MODBUS protocol. The following table lists the sub-commands:

Addresses (Hex format)	Function Description	Converter Model
0000	return query data	ML 210 – ML 211 – ML 212 – ML 110
0001	restart communications	ML 210 – ML 211 – ML 212 – ML 110
0004	listen mode only	ML 210 – ML 211 – ML 212 – ML 110
000A	clear counters	ML 210 – ML 211 – ML 212 – ML 110
000B-0012	return counters value	ML 210 – ML 211 – ML 212 – ML 110

- **FUNCTION 16**, function for to set parameters in the converter. As the same way as the reading function, it is possible to write value in the memory of the meter in the format of 16 bit variables that can be append together to form floating point variables (float) or long-integer (long) with 32 bit. The high side (MSW) start at even address, the low side (LSW) start immediately the following. The addresses are expressed in hex format.

Addresses (Hex format)	Function Description	Converter Model
07D0-084F	Values batch memory, grouped in 64 contiguous blocks of 8 register of memory at 16 bit for every block.	ML 210

0BB8	Batch Memory index current in use	ML 210

For the meaning of the memory location and their address go to the relative reading section.

5.8. CUSTOM FUNCTION

FUNCTION 110, execution of the ETP command. With this function it is possible to send a text string to the meter in the ETP protocol format (see the relative section), the ETP command is embedded in the MODBUS standard protcol format.
The max length of the string to send is 251 characters. The returned string form the meter is embedded with the same structure. Also in this case the max length of the returned string is 251 characters.
With the ETP command is possible to set and to return the value of every parameter of the converter.
For the complete list of the ETP commands see the relative section.

Example of the sequence of bytes for return the model and the software version of the converter with the function 110 and the ETP command: MODSV.
In this example the address of the converter is 0x01.
ETP command is not case sensitive, "MODSV?" = "modsv?"

The MODBUS master send this sequence of bytes to the converter for reading the model and the software version: 01 6E 6D 6F 64 73 76 3F 0D 6F FE

Byte (hex format)	**Description**
0x01	Address of the converter
0x6E	Function 110(dec) = 0x6E(hex)
0x6D	‘m’
0x6F	‘o’
0x64	‘d’
0x73	‘s’
0x76	‘v’
0x3F	‘?’
0x0D	Character 13
0x6F	Modbus checksum
0xFE	Modbus checksum

Example of the returned string form the converter:

Byte (hex format)	**Description**
0x01	Address of the converter
0x6E	Function 110(dec) = 0x6E(hex)
0x4D	‘M’
0x4C	‘L’
0x20	‘ ’
0x31	‘1’
0x31	‘1’
0x30	‘0’
0x20	‘ ’
0x56	‘V’
0x45	‘E’

0x52	‘R’
0x2E	‘.’
0x33	‘3’
0x2E	‘.’
0x36	‘6’
0x30	‘0’
0x20	‘ ’
0x41	‘A’
0x70	‘p’
0x72	‘r’
0x20	‘ ’
0x31	‘1’
0x34	‘4’
0x20	‘ ’
0x32	‘2’
0x30	‘0’
0x30	‘0’
0x38	‘8’
0x0D	Carriage return
0x0A	Line feed
0x73	Modbus checksum
0xFE	Modbus checksum

The model and the software version of the converter returned is: “ML 110 VER.3.60 Apr 14 2008”

Example of the sequence of bytes for set the Nominal Diameter of the converter with the function 110 and the ETP command: PDIMV.
In this example the address of the converter is 0x01.

The MODBUS master send this sequence of bytes to the converter for setting the nominal diameter to the value 10 mm : 01 6E 50 44 49 4D 56 3D 31 30 0D 0D A0 61

Byte (hex format)	**Description**
01	Address of the converter
6E	Function 110(dec) = 0x6E(hex)
50	‘P’
44	‘D’
49	‘I’
4D	‘M’
56	‘V’
3D	‘=’
31	‘1’
30	‘0’
0D	Character 13
A0	Modbus checksum
61	Modbus checksum

Example of the returned string form the converter: 01 6E 30 3A 4F 4B 0D 0A 31 A1

Byte (hex format)	Description
01	Address of the converter
6E	Function 110(dec) = 0x6E(hex)
30	‘0’
3A	‘:’
4F	‘O’
4B	‘K’
0D	Carriage return
0A	Line feed
31	Modbus checksum
A1	Modbus checksum

5.9. EXAMPLE OF SOURCE CODE FOR VB .NET 2005 PROJECT

The code below is intended as an example of source code for return the process data from the flow meter via Modbus protocol.
The example uses the serial port control present in the tool box of the Visual Basic Net 2005
For the hardware communication it is used a converter RS232 – RS485 connected to the RS232 serial port of the pc and the RS485 port of the converter.
The control line used in the example for the control of the transmission data in the RS232-RS485 converter is the RTS line.
The example program is an executable application that run in a Windows XP operative system. For this reason the commutation of the RTS line is not immediately reached because the Windows XP is multithreading system and the application haven’t the direct control of the hardware.
In other words there isn’t a direct control of the UART of the mother board by the application.
For this reason there is possible to insert a delay time by the function **A.delay** in the Menu «**7-Communication»** of the flow meter for compensate the retarder.

```
Public Class Form1

    Public Delegate Sub myDelegate()


    Dim afIntChkHigh(0 To 255) As Integer
    Dim afIntChkLow(0 To 255) As Integer


    Dim vfStrErrorString As String

    Dim vfStrCommand As String

    Dim vfStrCrc As String


    Private Sub Form1_FormClosing(ByVal sender As Object, ByVal e As
System.Windows.Forms.FormClosingEventArgs) Handles Me.FormClosing
        End
    End Sub


    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
```

MAN 1000102637 EN Version: B Status: RL (released | freigegeben) printed: 10.01.2018

```
Try

    'load checksum

    afIntChkHigh(0) = &H0
    afIntChkHigh(1) = &HC1
    afIntChkHigh(2) = &H81
    afIntChkHigh(3) = &H40
    afIntChkHigh(4) = &H1
    afIntChkHigh(5) = &HC0
    afIntChkHigh(6) = &H80
    afIntChkHigh(7) = &H41
    afIntChkHigh(8) = &H1
    afIntChkHigh(9) = &HC0
    afIntChkHigh(10) = &H80
    afIntChkHigh(11) = &H41
    afIntChkHigh(12) = &H0
    afIntChkHigh(13) = &HC1
    afIntChkHigh(14) = &H81
    afIntChkHigh(15) = &H40

    afIntChkHigh(16) = &H1
    afIntChkHigh(17) = &HC0
    afIntChkHigh(18) = &H80
    afIntChkHigh(19) = &H41
    afIntChkHigh(20) = &H0
    afIntChkHigh(21) = &HC1
    afIntChkHigh(22) = &H81
    afIntChkHigh(23) = &H40
    afIntChkHigh(24) = &H0
    afIntChkHigh(25) = &HC1
    afIntChkHigh(26) = &H81
    afIntChkHigh(27) = &H40
    afIntChkHigh(28) = &H1
    afIntChkHigh(29) = &HC0
    afIntChkHigh(30) = &H80
    afIntChkHigh(31) = &H41

    afIntChkHigh(32) = &H1
    afIntChkHigh(33) = &HC0
    afIntChkHigh(34) = &H80
    afIntChkHigh(35) = &H41
    afIntChkHigh(36) = &H0
    afIntChkHigh(37) = &HC1
    afIntChkHigh(38) = &H81
    afIntChkHigh(39) = &H40
    afIntChkHigh(40) = &H0
    afIntChkHigh(41) = &HC1
    afIntChkHigh(42) = &H81
    afIntChkHigh(43) = &H40
    afIntChkHigh(44) = &H1
    afIntChkHigh(45) = &HC0
    afIntChkHigh(46) = &H80
    afIntChkHigh(47) = &H41

    afIntChkHigh(48) = &H0
afIntChkHigh(49) = &HC1
afIntChkHigh(50) = &H81
afIntChkHigh(51) = &H40
afIntChkHigh(52) = &H1
afIntChkHigh(53) = &HC0
afIntChkHigh(54) = &H80
afIntChkHigh(55) = &H41
afIntChkHigh(56) = &H1
afIntChkHigh(57) = &HC0
afIntChkHigh(58) = &H80
afIntChkHigh(59) = &H41
afIntChkHigh(60) = &H0
afIntChkHigh(61) = &HC1
afIntChkHigh(62) = &H81
afIntChkHigh(63) = &H40

afIntChkHigh(64) = &H1
afIntChkHigh(65) = &HC0
afIntChkHigh(66) = &H80
afIntChkHigh(67) = &H41
afIntChkHigh(68) = &H0
afIntChkHigh(69) = &HC1
afIntChkHigh(70) = &H81
afIntChkHigh(71) = &H40
afIntChkHigh(72) = &H0
afIntChkHigh(73) = &HC1
afIntChkHigh(74) = &H81
afIntChkHigh(75) = &H40
afIntChkHigh(76) = &H1
afIntChkHigh(77) = &HC0
afIntChkHigh(78) = &H80
afIntChkHigh(79) = &H41

afIntChkHigh(80) = &H0
afIntChkHigh(81) = &HC1
afIntChkHigh(82) = &H81
afIntChkHigh(83) = &H40
afIntChkHigh(84) = &H1
afIntChkHigh(85) = &HC0
afIntChkHigh(86) = &H80
afIntChkHigh(87) = &H41
afIntChkHigh(88) = &H1
afIntChkHigh(89) = &HC0
afIntChkHigh(90) = &H80
afIntChkHigh(91) = &H41
afIntChkHigh(92) = &H0
afIntChkHigh(93) = &HC1
afIntChkHigh(94) = &H81
afIntChkHigh(95) = &H40

afIntChkHigh(96) = &H0
afIntChkHigh(97) = &HC1
afIntChkHigh(98) = &H81
afIntChkHigh(99) = &H40
afIntChkHigh(100) = &H1
afIntChkHigh(101) = &HC0
afIntChkHigh(102) = &H80
afIntChkHigh(103) = &H41
afIntChkHigh(104) = &H1
afIntChkHigh(105) = &HC0
afIntChkHigh(106) = &H80
afIntChkHigh(107) = &H41
afIntChkHigh(108) = &H0
afIntChkHigh(109) = &HC1
afIntChkHigh(110) = &H81
afIntChkHigh(111) = &H40

afIntChkHigh(112) = &H1
afIntChkHigh(113) = &HC0
afIntChkHigh(114) = &H80
afIntChkHigh(115) = &H41
afIntChkHigh(116) = &H0
afIntChkHigh(117) = &HC1
afIntChkHigh(118) = &H81
afIntChkHigh(119) = &H40
afIntChkHigh(120) = &H0
afIntChkHigh(121) = &HC1
afIntChkHigh(122) = &H81
afIntChkHigh(123) = &H40
afIntChkHigh(124) = &H1
afIntChkHigh(125) = &HC0
afIntChkHigh(126) = &H80
afIntChkHigh(127) = &H41

afIntChkHigh(128) = &H1
afIntChkHigh(129) = &HC0
afIntChkHigh(130) = &H80
afIntChkHigh(131) = &H41
afIntChkHigh(132) = &H0
afIntChkHigh(133) = &HC1
afIntChkHigh(134) = &H81
afIntChkHigh(135) = &H40
afIntChkHigh(136) = &H0
afIntChkHigh(137) = &HC1
afIntChkHigh(138) = &H81
afIntChkHigh(139) = &H40
afIntChkHigh(140) = &H1
afIntChkHigh(141) = &HC0
afIntChkHigh(142) = &H80
afIntChkHigh(143) = &H41

afIntChkHigh(144) = &H0
afIntChkHigh(145) = &HC1
afIntChkHigh(146) = &H81
afIntChkHigh(147) = &H40
afIntChkHigh(148) = &H1
afIntChkHigh(149) = &HC0
afIntChkHigh(150) = &H80
afIntChkHigh(151) = &H41
afIntChkHigh(152) = &H1
afIntChkHigh(153) = &HC0
afIntChkHigh(154) = &H80
afIntChkHigh(155) = &H41
afIntChkHigh(156) = &H0
afIntChkHigh(157) = &HC1
afIntChkHigh(158) = &H81
afIntChkHigh(159) = &H40

afIntChkHigh(160) = &H0
afIntChkHigh(161) = &HC1
afIntChkHigh(162) = &H81
afIntChkHigh(163) = &H40
afIntChkHigh(164) = &H1
afIntChkHigh(165) = &HC0
afIntChkHigh(166) = &H80
afIntChkHigh(167) = &H41
afIntChkHigh(168) = &H1
afIntChkHigh(169) = &HC0
afIntChkHigh(170) = &H80
afIntChkHigh(171) = &H41
afIntChkHigh(172) = &H0
afIntChkHigh(173) = &HC1
afIntChkHigh(174) = &H81
afIntChkHigh(175) = &H40

afIntChkHigh(176) = &H1
afIntChkHigh(177) = &HC0
afIntChkHigh(178) = &H80
afIntChkHigh(179) = &H41
afIntChkHigh(180) = &H0
afIntChkHigh(181) = &HC1
afIntChkHigh(182) = &H81
afIntChkHigh(183) = &H40
afIntChkHigh(184) = &H0
afIntChkHigh(185) = &HC1
afIntChkHigh(186) = &H81
afIntChkHigh(187) = &H40
afIntChkHigh(188) = &H1
afIntChkHigh(189) = &HC0
afIntChkHigh(190) = &H80
afIntChkHigh(191) = &H41

afIntChkHigh(192) = &H0
afIntChkHigh(193) = &HC1
afIntChkHigh(194) = &H81
afIntChkHigh(195) = &H40
afIntChkHigh(196) = &H1
afIntChkHigh(197) = &HC0
afIntChkHigh(198) = &H80
afIntChkHigh(199) = &H41
afIntChkHigh(200) = &H1
afIntChkHigh(201) = &HC0
afIntChkHigh(202) = &H80
afIntChkHigh(203) = &H41
afIntChkHigh(204) = &H0
afIntChkHigh(205) = &HC1
afIntChkHigh(206) = &H81
afIntChkHigh(207) = &H40
afIntChkHigh(208) = &H1
afIntChkHigh(209) = &HC0
afIntChkHigh(210) = &H80
afIntChkHigh(211) = &H41
afIntChkHigh(212) = &H0
afIntChkHigh(213) = &HC1
afIntChkHigh(214) = &H81
afIntChkHigh(215) = &H40
afIntChkHigh(216) = &H0
afIntChkHigh(217) = &HC1
afIntChkHigh(218) = &H81
afIntChkHigh(219) = &H40
afIntChkHigh(220) = &H1
afIntChkHigh(221) = &HC0
afIntChkHigh(222) = &H80
afIntChkHigh(223) = &H41

afIntChkHigh(224) = &H1
afIntChkHigh(225) = &HC0
afIntChkHigh(226) = &H80
afIntChkHigh(227) = &H41
afIntChkHigh(228) = &H0
afIntChkHigh(229) = &HC1
afIntChkHigh(230) = &H81
afIntChkHigh(231) = &H40
afIntChkHigh(232) = &H0
afIntChkHigh(233) = &HC1
afIntChkHigh(234) = &H81
afIntChkHigh(235) = &H40
afIntChkHigh(236) = &H1
afIntChkHigh(237) = &HC0
afIntChkHigh(238) = &H80
afIntChkHigh(239) = &H41

afIntChkHigh(240) = &H0
afIntChkHigh(241) = &HC1
afIntChkHigh(242) = &H81
afIntChkHigh(243) = &H40
afIntChkHigh(244) = &H1
afIntChkHigh(245) = &HC0
afIntChkHigh(246) = &H80
afIntChkHigh(247) = &H41
afIntChkHigh(248) = &H1
afIntChkHigh(249) = &HC0
afIntChkHigh(250) = &H80
afIntChkHigh(251) = &H41
afIntChkHigh(252) = &H0
afIntChkHigh(253) = &HC1
afIntChkHigh(254) = &H81
afIntChkHigh(255) = &H40


'-------------------
'-------------------
'-------------------
```

```
'-------------------
'-------------------


afIntChkLow(0) = &H0
afIntChkLow(1) = &HC0
afIntChkLow(2) = &HC1
afIntChkLow(3) = &H1
afIntChkLow(4) = &HC3
afIntChkLow(5) = &H3
afIntChkLow(6) = &H2
afIntChkLow(7) = &HC2
afIntChkLow(8) = &HC6
afIntChkLow(9) = &H6
afIntChkLow(10) = &H7
afIntChkLow(11) = &HC7
afIntChkLow(12) = &H5
afIntChkLow(13) = &HC5
afIntChkLow(14) = &HC4
afIntChkLow(15) = &H4

afIntChkLow(16) = &HCC
afIntChkLow(17) = &HC
afIntChkLow(18) = &HD
afIntChkLow(19) = &HCD
afIntChkLow(20) = &HF
afIntChkLow(21) = &HCF
afIntChkLow(22) = &HCE
afIntChkLow(23) = &HE
afIntChkLow(24) = &HA
afIntChkLow(25) = &HCA
afIntChkLow(26) = &HCB
afIntChkLow(27) = &HB
afIntChkLow(28) = &HC9
afIntChkLow(29) = &H9
afIntChkLow(30) = &H8
afIntChkLow(31) = &HC8

afIntChkLow(32) = &HD8
afIntChkLow(33) = &H18
afIntChkLow(34) = &H19
afIntChkLow(35) = &HD9
afIntChkLow(36) = &H1B
afIntChkLow(37) = &HDB
afIntChkLow(38) = &HDA
afIntChkLow(39) = &H1A
afIntChkLow(40) = &H1E
afIntChkLow(41) = &HDE
afIntChkLow(42) = &HDF
afIntChkLow(43) = &H1F
afIntChkLow(44) = &HDD
afIntChkLow(45) = &H1D
afIntChkLow(46) = &H1C
afIntChkLow(47) = &HDC

afIntChkLow(48) = &H14
afIntChkLow(49) = &HD4
afIntChkLow(50) = &HD5
afIntChkLow(51) = &H15
afIntChkLow(52) = &HD7
afIntChkLow(53) = &H17
afIntChkLow(54) = &H16
afIntChkLow(55) = &HD6
afIntChkLow(56) = &HD2
afIntChkLow(57) = &H12
afIntChkLow(58) = &H13
afIntChkLow(59) = &HD3
afIntChkLow(60) = &H11
afIntChkLow(61) = &HD1
afIntChkLow(62) = &HD0
afIntChkLow(63) = &H10

afIntChkLow(64) = &HF0
afIntChkLow(65) = &H30
afIntChkLow(66) = &H31
afIntChkLow(67) = &HF1
afIntChkLow(68) = &H33
afIntChkLow(69) = &HF3
afIntChkLow(70) = &HF2
afIntChkLow(71) = &H32
afIntChkLow(72) = &H36
afIntChkLow(73) = &HF6
afIntChkLow(74) = &HF7
afIntChkLow(75) = &H37
afIntChkLow(76) = &HF5
afIntChkLow(77) = &H35
afIntChkLow(78) = &H34
afIntChkLow(79) = &HF4

afIntChkLow(80) = &H3C
afIntChkLow(81) = &HFC
afIntChkLow(82) = &HFD
afIntChkLow(83) = &H3D
afIntChkLow(84) = &HFF
afIntChkLow(85) = &H3F
afIntChkLow(86) = &H3E
afIntChkLow(87) = &HFE
afIntChkLow(88) = &HFA
afIntChkLow(89) = &H3A
afIntChkLow(90) = &H3B
afIntChkLow(91) = &HFB
afIntChkLow(92) = &H39
afIntChkLow(93) = &HF9
afIntChkLow(94) = &HF8
afIntChkLow(95) = &H38

afIntChkLow(96) = &H28
afIntChkLow(97) = &HE8
afIntChkLow(98) = &HE9
afIntChkLow(99) = &H29
afIntChkLow(100) = &HEB
afIntChkLow(101) = &H2B
afIntChkLow(102) = &H2A
afIntChkLow(103) = &HEA
afIntChkLow(104) = &HEE
afIntChkLow(105) = &H2E
afIntChkLow(106) = &H2F
afIntChkLow(107) = &HEF
afIntChkLow(108) = &H2D
afIntChkLow(109) = &HED
afIntChkLow(110) = &HEC
afIntChkLow(111) = &H2C

afIntChkLow(112) = &HE4
afIntChkLow(113) = &H24
afIntChkLow(114) = &H25
afIntChkLow(115) = &HE5
afIntChkLow(116) = &H27
afIntChkLow(117) = &HE7
afIntChkLow(118) = &HE6
afIntChkLow(119) = &H26
afIntChkLow(120) = &H22
afIntChkLow(121) = &HE2
afIntChkLow(122) = &HE3
afIntChkLow(123) = &H23
afIntChkLow(124) = &HE1
afIntChkLow(125) = &H21
afIntChkLow(126) = &H20
afIntChkLow(127) = &HE0

afIntChkLow(128) = &HA0
afIntChkLow(129) = &H60
afIntChkLow(130) = &H61
afIntChkLow(131) = &HA1
afIntChkLow(132) = &H63
afIntChkLow(133) = &HA3
afIntChkLow(134) = &HA2
afIntChkLow(135) = &H62
afIntChkLow(136) = &H66
afIntChkLow(137) = &HA6
afIntChkLow(138) = &HA7
afIntChkLow(139) = &H67
afIntChkLow(140) = &HA5
afIntChkLow(141) = &H65
afIntChkLow(142) = &H64
afIntChkLow(143) = &HA4

afIntChkLow(144) = &H6C
afIntChkLow(145) = &HAC
afIntChkLow(146) = &HAD
afIntChkLow(147) = &H6D
afIntChkLow(148) = &HAF
afIntChkLow(149) = &H6F
afIntChkLow(150) = &H6E
afIntChkLow(151) = &HAE
afIntChkLow(152) = &HAA
afIntChkLow(153) = &H6A
afIntChkLow(154) = &H6B
afIntChkLow(155) = &HAB
afIntChkLow(156) = &H69
afIntChkLow(157) = &HA9
afIntChkLow(158) = &HA8
afIntChkLow(159) = &H68

afIntChkLow(160) = &H78
afIntChkLow(161) = &HB8
afIntChkLow(162) = &HB9
afIntChkLow(163) = &H79
afIntChkLow(164) = &HBB
afIntChkLow(165) = &H7B
afIntChkLow(166) = &H7A
afIntChkLow(167) = &HBA
afIntChkLow(168) = &HBE
afIntChkLow(169) = &H7E
afIntChkLow(170) = &H7F
afIntChkLow(171) = &HBF
afIntChkLow(172) = &H7D
afIntChkLow(173) = &HBD
afIntChkLow(174) = &HBC
afIntChkLow(175) = &H7C

afIntChkLow(176) = &HB4
afIntChkLow(177) = &H74
afIntChkLow(178) = &H75
afIntChkLow(179) = &HB5
afIntChkLow(180) = &H77
afIntChkLow(181) = &HB7
afIntChkLow(182) = &HB6
afIntChkLow(183) = &H76
afIntChkLow(184) = &H72
afIntChkLow(185) = &HB2
afIntChkLow(186) = &HB3
afIntChkLow(187) = &H73
afIntChkLow(188) = &HB1
afIntChkLow(189) = &H71
afIntChkLow(190) = &H70
afIntChkLow(191) = &HB0

afIntChkLow(192) = &H50
afIntChkLow(193) = &H90
afIntChkLow(194) = &H91
afIntChkLow(195) = &H51
afIntChkLow(196) = &H93
afIntChkLow(197) = &H53
afIntChkLow(198) = &H52
afIntChkLow(199) = &H92
afIntChkLow(200) = &H96
afIntChkLow(201) = &H56
afIntChkLow(202) = &H57
afIntChkLow(203) = &H97
afIntChkLow(204) = &H55
afIntChkLow(205) = &H95
afIntChkLow(206) = &H94
afIntChkLow(207) = &H54
afIntChkLow(208) = &H9C
afIntChkLow(209) = &H5C
afIntChkLow(210) = &H5D
afIntChkLow(211) = &H9D
afIntChkLow(212) = &H5F
afIntChkLow(213) = &H9F
afIntChkLow(214) = &H9E
afIntChkLow(215) = &H5E
afIntChkLow(216) = &H5A
afIntChkLow(217) = &H9A
afIntChkLow(218) = &H9B
afIntChkLow(219) = &H5B
afIntChkLow(220) = &H99
afIntChkLow(221) = &H59
afIntChkLow(222) = &H58
afIntChkLow(223) = &H98

afIntChkLow(224) = &H88
afIntChkLow(225) = &H48
afIntChkLow(226) = &H49
afIntChkLow(227) = &H89
afIntChkLow(228) = &H4B
afIntChkLow(229) = &H8B
afIntChkLow(230) = &H8A
afIntChkLow(231) = &H4A
afIntChkLow(232) = &H4E
afIntChkLow(233) = &H8E
afIntChkLow(234) = &H8F
afIntChkLow(235) = &H4F
afIntChkLow(236) = &H8D
afIntChkLow(237) = &H4D
afIntChkLow(238) = &H4C
afIntChkLow(239) = &H8C

afIntChkLow(240) = &H44
afIntChkLow(241) = &H84
afIntChkLow(242) = &H85
afIntChkLow(243) = &H45
afIntChkLow(244) = &H87
afIntChkLow(245) = &H47
afIntChkLow(246) = &H46
afIntChkLow(247) = &H86
afIntChkLow(248) = &H82
afIntChkLow(249) = &H42
afIntChkLow(250) = &H43
afIntChkLow(251) = &H83
afIntChkLow(252) = &H41
afIntChkLow(253) = &H81
afIntChkLow(254) = &H80
afIntChkLow(255) = &H40

'-------------------
'-------------------
'-------------------
'-------------------
```

```
        '-------------------


        Me.Text = System.Windows.Forms.Application.ProductName
        lblTitle.Text = System.Windows.Forms.Application.ProductName
        lblDescription.Text = "The program read the process data with Modbus command 03" & vbCrLf & _
                        "The address of the Flow meter is 01" & vbCrLf & _
                        "For the connection of the PC to the RS 485 port of the Flow meter is used an" & vbCrLf &
_
                        "RS 232-485 converter and the PC command the RTS signal of the serial port for" & vbCrLf
& _
                        "the trasmission of the data"
        lblProcessData.Text = ""

        SerialPort1.PortName = "COM1"

        SerialPort1.BaudRate = 9600

        SerialPort1.DataBits = 8
        SerialPort1.Parity = IO.Ports.Parity.None
        SerialPort1.StopBits = IO.Ports.StopBits.One

        SerialPort1.ReadBufferSize = 512
        SerialPort1.WriteBufferSize = 512

        SerialPort1.Encoding = System.Text.Encoding.Default


        SerialPort1.Open()

        SerialPort1.RtsEnable = False
        Application.DoEvents()

        Threading.Thread.Sleep(500)

        SerialPort1.DiscardInBuffer()
        SerialPort1.DiscardOutBuffer()


        'Command 3: Read process data
        vfStrCommand = Chr(1) & Chr(3) & Chr(0) & Chr(0) & Chr(0) & Chr(38)

        If False = fBolCalcCRC(vfStrCommand, vfStrCrc) Then
            Return
        End If

        vfStrCommand = vfStrCommand & vfStrCrc


        Me.Show()
        Application.DoEvents()


        Do

            TextBox2.Text = ""
            SerialPort1.RtsEnable = False

            SerialPort1.Write(vfStrCommand)

            Threading.Thread.Sleep(10)

            SerialPort1.RtsEnable = True
            Application.DoEvents()

            Threading.Thread.Sleep(200)

        Loop

    Catch vObjExcept As Exception

        vfStrErrorString = "Form name: " & Me.Name & vbCrLf & _
                           "" & vbCrLf & _
                           "Error source: " & vObjExcept.Source & vbCrLf & _
                           "" & vbCrLf & _
                           "Error StackTrace: " & vObjExcept.StackTrace & vbCrLf & _
                           "" & vbCrLf & _
                           "Error Message: " & vObjExcept.Message

        MsgBox(vfStrErrorString, MsgBoxStyle.OkOnly + MsgBoxStyle.Critical, Me.Text)
        Return
    End Try

End Sub

Function fBolCalcCRC(ByVal vStrString As String, ByRef vStrCRC As String) As Boolean

    Dim vIntCrcHigh As Integer
    Dim vIntCrcLow As Integer

    Dim vIntN As Integer
    Dim vByteA As Byte

    Try


        vIntCrcLow = &HFF
        vIntCrcHigh = &HFF

        For vIntN = 1 To Len(vStrString)
            vByteA = Asc(Mid(vStrString, vIntN, 1))
            vByteA = vByteA Xor vIntCrcLow
            vIntCrcLow = afIntChkHigh(vByteA) Xor vIntCrcHigh
            vIntCrcHigh = afIntChkLow(vByteA)
        Next

        vStrCRC = Chr(vIntCrcLow) & Chr(vIntCrcHigh)

        Return True

    Catch vObjExcept As Exception
```

```
            vfStrErrorString = "Form name: " & Me.Name & vbCrLf & _
                             "" & vbCrLf & _
                             "Error source: " & vObjExcept.Source & vbCrLf & _
                             "" & vbCrLf & _
                             "Error StackTrace: " & vObjExcept.StackTrace & vbCrLf & _
                             "" & vbCrLf & _
                             "Error Message: " & vObjExcept.Message

            MsgBox(vfStrErrorString, MsgBoxStyle.OkOnly + MsgBoxStyle.Critical, Me.Text)
            Return False
        End Try

    End Function


    Private Sub SerialPort1_DataReceived(ByVal sender As Object, ByVal e As
System.IO.Ports.SerialDataReceivedEventArgs) Handles SerialPort1.DataReceived


        Try
            TextBox2.Invoke(New myDelegate(AddressOf decode), New Object() {})


        Catch vObjExcept As Exception

            vfStrErrorString = "Form name: " & Me.Name & vbCrLf & _
                             "" & vbCrLf & _
                             "Error source: " & vObjExcept.Source & vbCrLf & _
                             "" & vbCrLf & _
                             "Error StackTrace: " & vObjExcept.StackTrace & vbCrLf & _
                             "" & vbCrLf & _
                             "Error Message: " & vObjExcept.Message

            MsgBox(vfStrErrorString, MsgBoxStyle.OkOnly + MsgBoxStyle.Critical, Me.Text)
            Return
        End Try
    End Sub

    Private Sub decode()


        Dim vIntBytesToRead As Integer

        Dim vStrRecevebuffer As String = ""

        Dim vIntN As Integer

        Dim vDblTemp As Single
        Dim vLngTemp As Long
        Dim afByteTemp(0 To 3) As Byte

        Try


            vIntBytesToRead = SerialPort1.BytesToRead
            If vIntBytesToRead >= 81 Then

                lblProcessData.Text = ""
                ReDim afByteReceveBuffer(0 To vIntBytesToRead - 1)

                SerialPort1.Read(afByteReceveBuffer, 0, vIntBytesToRead)

                For vIntN = 0 To vIntBytesToRead - 1
                    vStrRecevebuffer = vStrRecevebuffer & Chr(afByteReceveBuffer(vIntN))
                Next

                If False = fBolCalcCRC(Mid(vStrRecevebuffer, 1, vIntBytesToRead - 2), vfStrCrc) Then
                    Return
                End If

                If vfStrCrc <> (Chr(afByteReceveBuffer(79)) & Chr(afByteReceveBuffer(80))) Then
                    TextBox2.Text = "Bad data ---> Check sum error!"
                    Return
                End If



                TextBox2.Text = TextBox2.Text & hex_view(bytes_to_string(afByteReceveBuffer, vIntBytesToRead)) &
vbCrLf

                'Flow rate in %
                afByteTemp(0) = afByteReceveBuffer(6)
                afByteTemp(1) = afByteReceveBuffer(5)
                afByteTemp(2) = afByteReceveBuffer(4)
                afByteTemp(3) = afByteReceveBuffer(3)
                vDblTemp = BitConverter.ToSingle(afByteTemp, 0)
                lblProcessData.Text = "Flow rate in % = " & Format(vDblTemp, "###0.00") & vLngTemp & vbCrLf

                'Flow rate in technical unit
                afByteTemp(0) = afByteReceveBuffer(10)
                afByteTemp(1) = afByteReceveBuffer(9)
                afByteTemp(2) = afByteReceveBuffer(8)
                afByteTemp(3) = afByteReceveBuffer(7)
                vDblTemp = BitConverter.ToSingle(afByteTemp, 0)
                lblProcessData.Text = lblProcessData.Text & "Flow rate in technical unit = " & Format(vDblTemp,
"###0.000") & vLngTemp & vbCrLf

                'Totalizer for total volume positive T+    (V+ for ML211)
                afByteTemp(0) = afByteReceveBuffer(14)
                afByteTemp(1) = afByteReceveBuffer(13)
                afByteTemp(2) = afByteReceveBuffer(12)
                afByteTemp(3) = afByteReceveBuffer(11)
                vLngTemp = BitConverter.ToUInt32(afByteTemp, 0)
                lblProcessData.Text = lblProcessData.Text & "Totalizer for total volume positive T+   (V+ for ML211) =
" & vLngTemp & vbCrLf

                'Totalizer for partial volume positive P+ (V- for ML211)
                afByteTemp(0) = afByteReceveBuffer(18)
                afByteTemp(1) = afByteReceveBuffer(17)
```

```
        afByteTemp(2) = afByteReceveBuffer(16)
        afByteTemp(3) = afByteReceveBuffer(15)
        vLngTemp = BitConverter.ToUInt32(afByteTemp, 0)
        lblProcessData.Text = lblProcessData.Text & "Totalizer for partial volume positive P+ (V- for ML211) = " & vLngTemp & vbCrLf

        'Totalizer for total volume negative T- (E+ for ML211)
        afByteTemp(0) = afByteReceveBuffer(22)
        afByteTemp(1) = afByteReceveBuffer(21)
        afByteTemp(2) = afByteReceveBuffer(20)
        afByteTemp(3) = afByteReceveBuffer(19)
        vLngTemp = BitConverter.ToUInt32(afByteTemp, 0)
        lblProcessData.Text = lblProcessData.Text & "Totalizer for total volume negative T- (E+ for ML211) = " & vLngTemp & vbCrLf

        'Totalizer for partial volume negative P- (E- for ML211)
        afByteTemp(0) = afByteReceveBuffer(26)
        afByteTemp(1) = afByteReceveBuffer(25)
        afByteTemp(2) = afByteReceveBuffer(24)
        afByteTemp(3) = afByteReceveBuffer(23)
        vLngTemp = BitConverter.ToUInt32(afByteTemp, 0)
        lblProcessData.Text = lblProcessData.Text & "Totalizer for partial volume negative P- (E- for ML211) = " & vLngTemp & vbCrLf

        'Clock value in seconds
        afByteTemp(0) = afByteReceveBuffer(30)
        afByteTemp(1) = afByteReceveBuffer(29)
        afByteTemp(2) = afByteReceveBuffer(28)
        afByteTemp(3) = afByteReceveBuffer(27)
        vLngTemp = BitConverter.ToUInt32(afByteTemp, 0)
        lblProcessData.Text = lblProcessData.Text & "Clock value in seconds = " & vLngTemp & vbCrLf

        'Input value AIN1 in technical unit (opt.)
        afByteTemp(0) = afByteReceveBuffer(34)
        afByteTemp(1) = afByteReceveBuffer(33)
        afByteTemp(2) = afByteReceveBuffer(32)
        afByteTemp(3) = afByteReceveBuffer(31)
        vLngTemp = BitConverter.ToSingle(afByteTemp, 0)
        lblProcessData.Text = lblProcessData.Text & "Input value AIN1 in technical unit (opt.) = " & vLngTemp & vbCrLf

        'Input value AIN2 in technical unit (opt.)
        afByteTemp(0) = afByteReceveBuffer(38)
        afByteTemp(1) = afByteReceveBuffer(37)
        afByteTemp(2) = afByteReceveBuffer(36)
        afByteTemp(3) = afByteReceveBuffer(35)
        vLngTemp = BitConverter.ToSingle(afByteTemp, 0)
        lblProcessData.Text = lblProcessData.Text & "Input value AIN2 in technical unit (opt.) = " & vLngTemp & vbCrLf

        'Thermal power value in % (ML211)
        afByteTemp(0) = afByteReceveBuffer(42)
        afByteTemp(1) = afByteReceveBuffer(41)
        afByteTemp(2) = afByteReceveBuffer(40)
        afByteTemp(3) = afByteReceveBuffer(39)
        vLngTemp = BitConverter.ToSingle(afByteTemp, 0)
        lblProcessData.Text = lblProcessData.Text & "Thermal power value in % (ML211) = " & vLngTemp &
vbCrLf

        'Thermal power value in technical unit (ML211)
        afByteTemp(0) = afByteReceveBuffer(46)
        afByteTemp(1) = afByteReceveBuffer(45)
        afByteTemp(2) = afByteReceveBuffer(44)
        afByteTemp(3) = afByteReceveBuffer(43)
        vLngTemp = BitConverter.ToSingle(afByteTemp, 0)
        lblProcessData.Text = lblProcessData.Text & "Thermal power value in technical unit (ML211) = " &
vLngTemp & vbCrLf

        'Delta T value in technical unit (ML211)
        afByteTemp(0) = afByteReceveBuffer(50)
        afByteTemp(1) = afByteReceveBuffer(49)
        afByteTemp(2) = afByteReceveBuffer(48)
        afByteTemp(3) = afByteReceveBuffer(47)
        vLngTemp = BitConverter.ToSingle(afByteTemp, 0)
        lblProcessData.Text = lblProcessData.Text & "Delta T value in technical unit (ML211) = " & vLngTemp
& vbCrLf

        'Temperature value T1 in technical unit (ML211)
        afByteTemp(0) = afByteReceveBuffer(54)
        afByteTemp(1) = afByteReceveBuffer(53)
        afByteTemp(2) = afByteReceveBuffer(52)
        afByteTemp(3) = afByteReceveBuffer(51)
        vLngTemp = BitConverter.ToSingle(afByteTemp, 0)
        lblProcessData.Text = lblProcessData.Text & "Temperature value T1 in technical unit (ML211) = " &
vLngTemp & vbCrLf

        'Temperature value T2 in technical unit (ML211)
        afByteTemp(0) = afByteReceveBuffer(58)
        afByteTemp(1) = afByteReceveBuffer(57)
        afByteTemp(2) = afByteReceveBuffer(56)
        afByteTemp(3) = afByteReceveBuffer(55)
        vLngTemp = BitConverter.ToSingle(afByteTemp, 0)
        lblProcessData.Text = lblProcessData.Text & "Temperature value T2 in technical unit (ML211) = " &
vLngTemp & vbCrLf

        'Set-point value in % (ML212)
        afByteTemp(0) = afByteReceveBuffer(62)
        afByteTemp(1) = afByteReceveBuffer(61)
        afByteTemp(2) = afByteReceveBuffer(60)
        afByteTemp(3) = afByteReceveBuffer(59)
        vLngTemp = BitConverter.ToSingle(afByteTemp, 0)
        lblProcessData.Text = lblProcessData.Text & "Set-point value in % (ML212) = " & vLngTemp & vbCrLf

        'Output value in % (ML212)
        afByteTemp(0) = afByteReceveBuffer(66)
        afByteTemp(1) = afByteReceveBuffer(65)
        afByteTemp(2) = afByteReceveBuffer(64)
        afByteTemp(3) = afByteReceveBuffer(63)
        vLngTemp = BitConverter.ToSingle(afByteTemp, 0)
        lblProcessData.Text = lblProcessData.Text & "Output value in % (ML212) = " & vLngTemp & vbCrLf

        'Deviation value in % (ML212)
```

```
            afByteTemp(0) = afByteReceveBuffer(70)
            afByteTemp(1) = afByteReceveBuffer(69)
            afByteTemp(2) = afByteReceveBuffer(68)
            afByteTemp(3) = afByteReceveBuffer(67)
            vLngTemp = BitConverter.ToSingle(afByteTemp, 0)
            lblProcessData.Text = lblProcessData.Text & "Deviation value in % (ML212) = " & vLngTemp & vbCrLf

            'Process flags
            If (afByteReceveBuffer(72) And &H1) = &H1 Then
               lblProcessData.Text = lblProcessData.Text & "Process flags = excitation is too fast for the sensor
connected / temperature error ML211" & vbCrLf
            End If

            If (afByteReceveBuffer(72) And &H2) = &H2 Then
               lblProcessData.Text = lblProcessData.Text & "Process flags = max alarm is active" & vbCrLf
            End If

            If (afByteReceveBuffer(72) And &H4) = &H4 Then
               lblProcessData.Text = lblProcessData.Text & "Process flags = min alarm is active" & vbCrLf
            End If

            If (afByteReceveBuffer(72) And &H8) = &H8 Then
               lblProcessData.Text = lblProcessData.Text & "Process flags = flow rate exceeds the scale range,
overflow" & vbCrLf
            End If

            If (afByteReceveBuffer(72) And &H10) = &H10 Then
               lblProcessData.Text = lblProcessData.Text & "Process flags = too many impulse to emit" & vbCrLf
            End If

            If (afByteReceveBuffer(72) And &H20) = &H20 Then
               lblProcessData.Text = lblProcessData.Text & "Process flags = measurement signal is disturbed or
sensor disconnect" & vbCrLf
            End If

            If (afByteReceveBuffer(72) And &H40) = &H40 Then
               lblProcessData.Text = lblProcessData.Text & "Process flags = empty tube" & vbCrLf
            End If

            If (afByteReceveBuffer(72) And &H80) = &H80 Then
               lblProcessData.Text = lblProcessData.Text & "Process flags = coils not working or sensor disconnect"
& vbCrLf
            End If

            If (afByteReceveBuffer(71) And &H1) = &H1 Then
               lblProcessData.Text = lblProcessData.Text & "Process flags = second measurement scale is active" &
vbCrLf
            End If

            If (afByteReceveBuffer(71) And &H2) = &H2 Then
               lblProcessData.Text = lblProcessData.Text & "Process flags = flow rate is lower than the cut-off
threshold" & vbCrLf
            End If

            If (afByteReceveBuffer(71) And &H4) = &H4 Then
               lblProcessData.Text = lblProcessData.Text & "Process flags = flow rate negative" & vbCrLf
```

```
    End If

    If (afByteReceveBuffer(71) And &H8) = &H8 Then
      lblProcessData.Text = lblProcessData.Text & "Process flags = new measurement value is available
for the display" & vbCrLf
    End If

    If (afByteReceveBuffer(71) And &H10) = &H10 Then
      lblProcessData.Text = lblProcessData.Text & "Process flags = counter block signal is active" & vbCrLf
    End If

    If (afByteReceveBuffer(71) And &H20) = &H20 Then
      lblProcessData.Text = lblProcessData.Text & "Process flags = batch in progress" & vbCrLf
    End If

    If (afByteReceveBuffer(71) And &H40) = &H40 Then
      lblProcessData.Text = lblProcessData.Text & "Process flags = calibration cicle in progress" & vbCrLf
    End If

    If (afByteReceveBuffer(71) And &H80) = &H80 Then
      lblProcessData.Text = lblProcessData.Text & "Process flags = flow rate simulation in progress" &
vbCrLf
    End If

    'Flags ML 211
    If (afByteReceveBuffer(74) And &H2) = &H2 Then
      lblProcessData.Text = lblProcessData.Text & "Process flags = input error AIN1" & vbCrLf
    End If

    If (afByteReceveBuffer(74) And &H4) = &H4 Then
      lblProcessData.Text = lblProcessData.Text & "Process flags = input error AIN2" & vbCrLf
    End If

    'Flags ML 211
    If (afByteReceveBuffer(76) And &H1) = &H1 Then
      lblProcessData.Text = lblProcessData.Text & "Process flags = max alarm thermal power" & vbCrLf
    End If

    If (afByteReceveBuffer(76) And &H2) = &H2 Then
      lblProcessData.Text = lblProcessData.Text & "Process flags = min alarm thermal power" & vbCrLf
    End If

    If (afByteReceveBuffer(76) And &H4) = &H4 Then
      lblProcessData.Text = lblProcessData.Text & "Process flags = max alarm delta T" & vbCrLf
    End If

    If (afByteReceveBuffer(76) And &H8) = &H8 Then
      lblProcessData.Text = lblProcessData.Text & "Process flags = min alarm delta T" & vbCrLf
    End If

    If (afByteReceveBuffer(76) And &H10) = &H10 Then
      lblProcessData.Text = lblProcessData.Text & "Process flags = max alarm temp. T1" & vbCrLf
    End If

    If (afByteReceveBuffer(76) And &H20) = &H20 Then
      lblProcessData.Text = lblProcessData.Text & "Process flags = min alarm temp. T1" & vbCrLf
```

```
                End If

                If (afByteReceveBuffer(76) And &H40) = &H40 Then
                    lblProcessData.Text = lblProcessData.Text & "Process flags = max alarm temp. T2" & vbCrLf
                End If

                If (afByteReceveBuffer(76) And &H80) = &H80 Then
                    lblProcessData.Text = lblProcessData.Text & "Process flags = min alarm temp. T2" & vbCrLf
                End If

                'Flags ML212
                If (afByteReceveBuffer(78) And &H1) = &H1 Then
                    lblProcessData.Text = lblProcessData.Text & "Process flags = command error actuator" & vbCrLf
                End If

                If (afByteReceveBuffer(78) And &H2) = &H2 Then
                    lblProcessData.Text = lblProcessData.Text & "Process flags = deviation error" & vbCrLf
                End If

                If (afByteReceveBuffer(78) And &H4) = &H4 Then
                    lblProcessData.Text = lblProcessData.Text & "Process flags = Input error AIN1" & vbCrLf
                End If

                If (afByteReceveBuffer(78) And &H8) = &H8 Then
                    lblProcessData.Text = lblProcessData.Text & "Process flags = Input error AIN2" & vbCrLf
                End If

                If (afByteReceveBuffer(78) And &H10) = &H10 Then
                    lblProcessData.Text = lblProcessData.Text & "Process flags = manual regulation active" & vbCrLf
                End If

                If (afByteReceveBuffer(78) And &H20) = &H20 Then
                    lblProcessData.Text = lblProcessData.Text & "Process flags = safety mode active" & vbCrLf
                End If

            End If

        Catch vObjExcept As Exception

            vfStrErrorString = "Form name: " & Me.Name & vbCrLf & _
                               "" & vbCrLf & _
                               "Error source: " & vObjExcept.Source & vbCrLf & _
                               "" & vbCrLf & _
                               "Error StackTrace: " & vObjExcept.StackTrace & vbCrLf & _
                               "" & vbCrLf & _
                               "Error Message: " & vObjExcept.Message

            MsgBox(vfStrErrorString, MsgBoxStyle.OkOnly + MsgBoxStyle.Critical, Me.Text)
            Return
        End Try

    End Sub


End Class
```

6. APPENDIX

6.1. RS232 SERIAL PORT

The serial port RS232 present in the aux modules with the RS 232 port option, is used for the connection to printers, modem, personal computer or other devices. The RS232 port is insulated from all other circuits apart from the RS485 port, with which it shares the power supply.

This type of interface is used to exchange data over short distances (max. 10 metres) with one single user, usually a PC, a printer or an external modem. The connection takes place through 2 signal lines plus a common line. It has poor immunity to electrical disturbances and must therefore be made with considerable care in fixed industrial installation environments.

6.1.1. CONNECTING THE CONVERTER VIA RS232 PORT TO A PRINTER

This connection is made using the aux modules with the RS 232 port option. When the print functions are enabled, the instrument prints the process data at established intervals or in the event of an alarm. A print command can also be sent directly from the keyboard. When the RS232 port is used for connecting a printer, the connection cable can be constructed respecting the following requirements:

- use a shielded cable with at least three wires;
- *the shield is connected only to the ML210 end;*
- the length must not exceed 10 meters. Remember that the RS232 interface is more susceptible to disturbance than the RS485 and it is good working practice to minimize the length of the connections;
- on the end for connection to the ML210, fit tips numbered with the terminal number; for the end for connection to the printer, solder the wires to the pins indicated in the following table on a DB25 MALE type connector; the printer has a DB25 FEMALE connector (standard for serial RS232 printers);

the connections are the following:

ML210 TERMINAL	DIRECTION	DB25 PIN
32 (SHIELD)	<======>	do not connect
25 (TD)	=======>	3 (RXD)
23 (CTS)	<=======	20 (DTR)
26 (SG)	<======>	7 (GND)

ATTENTION: the **CTS** line must be connected to the **DTR** line (or equivalent function) of the printer and be used by this latter to indicate when it can accept data from the converter. The printer must be arranged for **HARDWARE** type flow control (**not** XON/XOFF). If the connections or arrangements indicated are not followed, the display and the converter keyboard may be blocked indefinitely because they are waiting for printer availability. If the printer does not use this type of flow control, or if you wish to send the output that would be printed to another device (PC, digital recorders etc.) you must connect the **CTS** signal of the converter to a terminal on the port that supplied a constant voltage of +5 to +15V (**DTR** or **RTS** of the PC or of the terminal).

NECESSARY PARAMETERS:

- Menu «**7-Communication**», function - «**Speed 2**»: set this value to the same speed as the printer connected. The speed may be chosen from 2400, 9600, 19200 and 38400 bps

- Menu «**7-Communication»**, function - «**Print**»: set to «**ON**» if you wish to use the print functions.
- Menu "**7-Comunicazione"**, function "**<u>Print</u> dos.**": set to "**ON**" if you wish to use the print functions of batch.
- Menu "**7-Comunicazione"**, function " **Print data**": set to "**ON**" if you wish to use the print functions of process
- Menu "**7-Comunicazione"**, function "**Print events**": set to "**ON**" if you wish to use the print functions of alarm and converter status
- Menu «**9-Data logger»**, function - «🕒»: set the date and current time.
- Menu «**9-Data logger»**, function - « **Interv.(h)**»: set the print interval in hours

The format of the printed process data is the following:

- standard:

DATE
INSTANTANEOUS FLOW RATE OF THE TECHNICAL UNIT ARRANGED FOR DISPLAY
FLAGS OF ALARM, SAMPLING SPEED, FLOW RATE IN %
TOTAL +
TOTAL -
PARTIAL +
PARTIAL -
ALARM

Example:

```
 10/07/03 18:14
dm3/s    +4.4423
80      +88.23%
T+dm3 123452.222
P+dm3   3452.222
T-dm3    222.112
P-dm3     44.121
NO ALARM
```

- WITH CURRENCY CONVERSION :

DATA
INSTANTANEOUS FLOW RATE OF THE TECHNICAL UNIT ARRANGED FOR DISPLAY
FLAGS OF ALARM, SAMPLING SPEED, FLOW RATE IN % TOTALIZZATORE T+
TOTAL +
PARTIAL +
PARTIAL + CONVERTED
TOTAL –
PARTIAL –
PARTIAL – CONVERTED
ALARM

EXAMPLE:

```
 10/07/03 18:14
dm3/s    +4.4423
 80      +88.23%
T+dm3 123452.222
P+dm3   3452.222
EUR     12343.55
T-dm3    222.112
P-dm3     44.121
EUR       345.67
NO ALARM
```

- PRINT OF STANTARD BATCH:

  DATE
  INSTANTANEOUS FLOW RATE OF THE TECHNICAL UNIT ARRANGED FOR DISPLAY
  PRODUCT NAME (RECEIPT) AND N° OF BATCH
  T+
  P+
  RESIDUAL QUANTITY
  DISBURSED QUANTITY
  ALARM

  EXAMPLE:

```
 10/07/03 18:14
dm3/s    +4.4423
WATER         32567
T+dm3 123452.222
P+dm3   3452.222
STdm3    200.000
CTdm3    201.121
NO ALARM
```

- PRINT OF BATCH WITH CURRENCY CONVERSION

  DATE
  PRODUCT NAME (RECEIPT) AND N° OF BATCH
  RESIDUAL QUANTITY
  DISBURSED QUANTITY
  DISBURSED QUANTITY CONVERTERD
  ALARM

  EXAMPLE:

```
 10/07/03 18:14
WATER         32567
STdm3    200.000
CTdm3    201.121
EUR      402.24
NO ALARM
```

If the dosing mode is NOT active, Printing can be started at any moment by pressing ⇧, otherwise push ⇨.

6.1.2. CONNECTING THE CONVERTER VIA RS232 PORT TO A PC/MODEM

This connection is made using RS232 port in the aux module. Make up the connection cable respecting the following requirements:

- use a shielded cable with at least three wires;
- *the shield is connected only to the converter end;*
- the length must not exceed 10 meters. Remember that the RS232 interface is more susceptible to disturbance than the RS485 and it is good working practice to minimize the length of the connections;
- on the end for connection to the converter, fit tips numbered with the terminal number; for the end for connection to the PC, solder the wires to the pins indicated in the following table on a DB25 or DB9 FEMALE type connector; if you intend to connect to a modem, the cable connectors must be MALE (the modem has FEMALE):

DB25 PC	DB9 PC	DB25 modem	DB9 modem

3 (RXD)	2 (RXD)	2 (RXD)	3 (RXD)
2 (TXD)	3 (TXD)	3 (TXD)	2 (TXD)
7 (GND)	5 (GND)	7 (GND)	5 (GND)

ATTENTION: when connecting to a modem, this latter must be configured because it only works autonomously with data lines (without control lines RTS/DTR/CTS/DSR etc.). For this type of configuration, refer to the modem manual.

NECESSARY PARAMETERS:

- Menu «**7-Communication»**, function - «**Speed 2**»: set this value to the same speed as the PC or modem. The speed may be chosen from 2400, 9600, 19200 and 38400 bps.
- Menu «**7-Communication»**, function - «**Print**»: set to «**OFF**».

- The meter will communicate with ETP protocol as descript ahead.
- The aux modules supply the RS232 port to the meter.
- The command, formatted with ETP protocol, and send to RS232 of the aux module, for example through a PC, are received by the meter that re-transmit the data, toward the door RS485, to the RS485 devices connected. In the same way, the ETP commands received on the RS485 are re-transmitted to the RS232 port connected to the PC.
- This is a typical case of a typical connection between a GSM modem and a local RS485 network of meters.
- Every command send to the meter and every returned response have two address: the "[**address_to**]" (address of destination) and the "[**address_from**]" (address of origin).
- For examples for to send packets from a PC connected to the RS232 of the meter to the standard door RS485 of the meter, it is necessary that the address "[**address_to**]" (address of destination) must be the address of the device connected on port RS4485 that is the device destination of the command.
- This address must be different from the meter address (Address on Menu «**7-Communication»**).
- This address must be also different from the value "232". The value 232 is not valid for this configuration.
- In this way each packet received on the door RS232 will be re-launched on the door RS485 toward the connected meter.
- Besides, for make so that the deriving answer from this could reach the questioning device on the door RS232, it is necessary that the address "[**address_from**]" (address of origin) will be set to the value" 232."
- In this way the data transit between a door and the other between the two connected devices. The meter that stays in the half has only the function of receive and sort the packets of data between a door and the other.

6.2. RS485 SERIAL PORT

Networks are used when there is a need to connect several instruments together. The interface usually used for this type of connection is the RS485. There are various ways to make several instruments communicate in a network; the way adopted in the Millennium series instrumentation is the MASTER-SLAVE type.
Up to 32 devices can be connected with this interface in a single network covering a length of up to 1200 metres with only two wires. It has excellent immunity to electrical disturbance and is therefore used extensively in industry.

6.2.1. CONNECTING THE CONVERTER VIA RS485 SERIAL PORT

The RS485 port is suitable for even long distance and network connections because it is galvanically insulated from all the rest of the appliance except for the RS232 port (if installed), with which it shares the power supply.

PARAMETERS:

- Menu «**7-Communication»**, function -«**Address**»: set this value so that each instrument in the network has a different address. The addresses possible go from 0 to 255., exclude **232**, reserved to re-launched between RS232 e la RS485.
- Menu «**7-Communication»**, function - «**Speed 1**»: set this value to the same speed as the RS485 device connected. The speed may be chosen from 2400, 9600, 19200 and 38400 bps.
- See point 1.3 if you need to transmit data between RS 232 and RS 485

6.2.2. CONNECTING THE CONVERTER TO AN RS485 NETWORK

This network is comprised of a certain number of devices each characterised by a univocal address number (for convenience we shall call this type of appliance “SLAVE”), and by a single referee ("MASTER") that interrogates all of the instruments connected in turn: a maximum number of 32 devices can be connected.

Example of connection in an RS485 network:

```
o~~~~~o~~~~~~~o~~~~~~~o~~~~~~~o~~~~/....../~~~~o~~~~~~~o  Terminal (+)
|     |       |       |       |            |       |
D0    D1      D2      D3      D4           D30     D31
|     |       |       |       |            |       |
o~~~~~o~~~~~~~o~~~~~~~o~~~~~~~o~~~~/....../~~~~o~~~~~~~o  Terminal (-)
```

D0÷D31 = devices connected (any of them may be the network master).

6.2.3. CRITERIA FOR DATA EXCHANGE

Information requests sent from the MASTER are received contemporaneously by all the SLAVES, but only the one addressed replies. For this reason, it is absolutely necessary for all the SLAVES to have different addresses.

The MASTER establishes the time scheduling with which the information must transit and must therefore be the single type of this element in the network; two MASTER would create conflicts in the communication, making the entire system unusable.

6.2.4. TYPES OF DEVICE FOR THE NETWORK CONNECTION

MASTER type devices can be PCs, PLCs or other terminals that collect measurement data from the instruments.
SLAVE type devices on the other hand are the Millennium series converters.

6.2.5. EXTENDING THE NETWORK

When more than 32 instruments have to be connected, these must be divided into separate groups of a maximum of 32 devices. Each group connects to the next through a repeater which has the task of regenerating the necessary electric signals. In any case, given that the maximum number of addresses that can be assigned for the instruments of the Millennium series is 256, networks with more than 256 elements cannot be created.