multicomp PRO

Programming Manual

MP701125 Series Smart Digital Power Meter

SCPI

SCPI Introduction

SCPI (Standard Commands for Programmable Instruments) is a standardized instrument programming language that builds on existing standards IEEE 488.1 and IEEE 488.2 and follows the floating point rules of IEEE 754 standard, ISO 646 message exchange 7-bit encoding notation (equivalent to ASCII programming) and many other standards.

This section introduces the format, symbols, parameters, and abbreviations of the SCPI command.

Instruction Format

Command is consisting of a keyword, separator, parameter domain and end marker. Take the following command as an example.

:VOLTage:RANGe 150
VOLTage, RANGe is keyword, ":" and blank is separator, "150" is parameter (some commands have multiple parameters and separated by ","), the line separator or carriage return after the command is the end mark.

For the convenience of description, the following conventions are adopted for subsequent symbols.

- Square Brackets "[ ]"
  The contents in square brackets (command keywords) can be omitted.

- Braces "{ }"
  It represents the parameter in command string.

- Angle Braces "< >"
  The parameter enclosed in the angle brackets must be a numerical parameter.

- Vertical Bar "|"
  It is used to separate multiple parameters.

- End Mark: line separator <LF> (0x0A)or carriage return <CR> (0x0D)

Parameter Description

The data type of programming parameters include numeric, character and Boolean type. Regardless of the type, it is expressed as an ASCII. For more details, see the following table.

Symbol	Meaning	Example
<NR1>	Integer	123, 0123
<NR2>	Fixed Floating point number	123., 12.3, 0.123, .123

<NR3>	Floating point number	123, 12.3, 123E+3
<NRF>	It may be <NR1>, <NR2> or <NR3>	
<Boolean>	Boolean data	0|1|ON|OFF

Shorthand Rule

All the commands are case-insensitive. The commands can be all input in uppercase letters or in lowercase letters. For abbreviations, it should enter all the uppercase letters that exist in the command syntax.

Communication Interface and Setting

The detailed description refer to "Communication Setting" and "Communication Interface" in the User's Manual.

SCPI Command

Command List

Instruction	Function
*IDN?	Query the instrument information.
*RST	Restore to the factory setting.
*STB?	Query status byte register.
*SAV	Save the current setting into nonvolatile memory for future use.
:HOLD	Turn on or off hold mode.
:MODE	Set the measurement mode of voltage/current.
:VOLTage:RANGe	Set the voltage range.
:VOLTage:AUTo	Turn on or off auto range of voltage.
:CURRent:RANGe	Set the current range.
:CURRent:AUTo	Turn on or off auto range of current.
:RATe	Set update rate.
:AVERaging	Set the average switch and average count.
:MUTe	Turn on or off mute.
:ALARm:FLAG?	Query the alarm state.
:ALARm:CURRent:HIGH	Set the upper limit of current alarm.
:ALARm:CURRent:LOW	Set the lower limit of current alarm.
:ALARm:POWer:HIGH	Set the upper limit of power alarm.
:ALARm:POWer:LOW	Set the lower limit of power alarm.
:ALARm:TIMe	Set alarm delay.

:UPDAte:COUNt?	Query the current update count.
:MEASure:FREQuency:VOLTage?	Query the measured frequency of voltage.
:MEASure:VOLTage?	Query the currently measured value of voltage.
:MEASure:CURRent?	Query the currently measured value of current.
:MEASure:POWer:ACTive?	Query the currently measured value of power.
:MEASure:PFACtor?	Query the currently measured value of power factor.
:SYSTem:ERRor?	Query error information
:LOCK	Set the lock key on the front panel.

Instruction Parse

*IDN?

Function Query the instrument information.

Syntax *IDN?

Example *IDN?

-> MP701125+,012345678,F1.02

Description The return format of instrument information is <manufacturer>,<model>,<serial number>,< firmware version>

*RST

Function Restore to the factory setting.

Syntax *RST

Example *RST

Description Except communication configuration parameter (instruction type, baud rate, address), other configuration parameter will restore to the factory setting.

*STB?

Function Query status byte register.

Syntax *STB?

Example *STB? -> 4

Description If returned value is 4, it represents the status byte register set to 00000100; It means the error queue is not empty, which also means an error has been generated.

*SAV

Function Save the current setting into nonvolatile memory for next time use.

Syntax *SAV

Example *SAV

:HOLD

Function Turn on/off hold mode.

Syntax :HOLD {<Boolean>}

:HOLD?

Example :HOLD OFF

:HOLD? -> 0

:MODE

Function Set the measurement mode of voltage/current.

Syntax :MODE {AC|ACDC|DC}

:MODE?

AC, ACDC = (RMS); DC

Example :MODE ACDC

:MODE? -> ACDC

:VOLTage:RANGe

Function Set the voltage range.

Syntax :VOLTage:RANGe {<Voltage>}

:VOLTage:RANGe?

<Voltage> = 75,150,300,600

Example :VOLTage:RANGe 150

:VOLTage:RANGe? -> 150

:VOLTage:AUTo

Function Turn on or off auto range of voltage.

Syntax :VOLTage:AUTo {<Boolean>}

:VOLTage:AUTo?

Example :VOLTage:AUTo 1

:VOLTage:AUTo? -> 1

:CURRent:RANGe

Function Set the current range.

Syntax :CURRent:RANGe {<Current>}

:CURRent:RANGe?

<Current> = 0.5, 2, 8, 20

Example :CURRent:RANGe 2

:CURRent:RANGe? -> 2

:CURRent:AUTo

Function Turn on or off auto range of current.

Syntax :CURRent:AUTo {<Boolean>}

:CURRent:AUTo?

Example :CURRent:AUTo 1

:CURRent:AUTo? -> 1

:RATe

Function Set update rate.

Syntax :RATe {<Time>}

:RATe?

<Time> = 0.1,0.25,0.5,1,2,5

Example :RATe 0.25

:RATe? -> 0.25

:AVERaging

Function Set the average switch and average count.

Syntax :AVERaging {<Average>}

:AVERaging?

<Average> = OFF,8,16,32,64

·OFF = Average is turned off

·8,16,32,64 = Average is turned on and it represents the average count.

Example :AVERaging 16

:AVERaging? -> 16

:MUTe

Function Turn on or off mute.

Syntax :MUTe {<Boolean>}

:MUTe?

Example :MUTe 1

:MUTe? ->1

:ALARm:FLAG?

Function Query the alarm state.

Syntax :ALARm:FLAG? {<Type>,<State>}

<Type> = CURRENT,POWER

CURRENT = current； POWER = power

<State> = DISABLE,WAITING， RUNNING， OK， LOW， HIGH

·DISABLE = the test is forbidden;

·WAITING = wait for connect to the load;

·RUNNING = testing;

·OK = the test is completed, the test result within the lower and upper limit;

·LOW = the test is completed, the test result is below the lower limit;

·HIGH = the test is completed, the test result is higher than theupper limit.

Example :ALARm:FLAG? CURRENT -> RUNNING

:ALARm:CURRent:HIGH

Function Set the upper limit of current alarm.

Syntax :ALARm:CURRent:HIGH {<NRf>}

:ALARm:CURRent:HIGH?

Example :ALARm:CURRent:HIGH 10.1

:ALARm:CURRent:HIGH? -> 10.1

:ALARm:CURRent:LOW

Function Set the lower limit of current alarm.

Syntax :ALARm:CURRent:LOW {<NRf>}

:ALARm:CURRent:LOW?

Example :ALARm:CURRent:LOW 1.1

:ALARm:CURRent:LOW? -> 1.1

:ALARm:POWer:HIGH

Function Set the upper limit of power alarm.

Syntax :ALARm:POWer:HIGH {<NRf>}

:ALARm:POWer:HIGH?

Example :ALARm:POWer:HIGH 1000.1

:ALARm:POWer:HIGH? -> 1000.1

:ALARm:POWer:LOW

Function Set the lower limit of power alarm.

Syntax :ALARm:POWer:LOW {<NRf>}

:ALARm:POWer:LOW?

Example :ALARm:POWer:LOW 10.1

:ALARm:POWer:LOW -> 10.1

:ALARm:TIMe

Function Set alarm delay.

Syntax :ALARm:TIMe {<NRf>}

:ALARm:TIMe?

Example :ALARm:TIMe 20.2

:ALARm:TIMe? -> 20.2

:UPDAte:COUNt?

Function Query the current update count.

Syntax :UPDAte:COUNt?

Example :UPDAte:COUNt? -> 763

Description Each time the data is updated, the number of updates will be increased by one. By detecting the difference in the number of updates before and after, it can determine whether the data update event occurs, so as to obtain the latest updated data.

:MEASure:FREQuency:VOLTage

Function Query the measured frequency of voltage.

Syntax :MEASure:FREQuency:VOLTage?

Example :MEASure:FREQuency:VOLTage? -> 50.00

:MEASure:VOLTage?

Function Query the currently measured value of voltage.

Syntax :MEASure:VOLTage?

Example :MEASure:VOLTage? -> 110.36

:MEASure:CURRent?

Function Query the currently measured value of current.

Syntax :MEASure:CURRent?

Example :MEASure:CURRent? -> 10.23

:MEASure:POWer:ACTive?

Function Query the currently measured value of power.

Syntax :MEASure:POWer:ACTive?

Example :MEASure:POWer:ACTive? -> 30.5

:MEASure:PFACtor?

Function Query the currently measured value of power factor.

Syntax :MEASure:PFACtor?

Example :MEASure:PFACtor? -> 0.519

:SYSTem:ERRor?

Function Query the last error code and information.

Syntax :SYSTem:ERRor?

Example :SYSTem:ERRor? -> -113,"Undefined header"

Description If there is no error, then it returns 0,"No error"

:LOCK

Function Set the lock key on the front panel.

Syntax :LOCK {<Boolean>}

:LOCK?

Example :LOCK 1

:LOCK? -> 1

MP701125 + Instruction Parse

*IDN?

Function Query the instrument information.

Syntax *IDN?

Example *IDN?

-> MP701125+,012345678,F1.02

Description The return format of instrument information is <manufacturer>, <model>, <serial number>, < firmware version>.

*RST

Function Restore to the factory setting.

Syntax *RST

Example *RST

Description Except communication configuration parameter (instruction type, baud rate, address), other configuration parameter will restore to the factory setting.

*STB?

Function Query status byte register.

Syntax *STB?

Example *STB? -> 4

Description If returned value is 4, it represents the status byte register set to 00000100; it means the error queue is not empty, which also means an error has been generated.

*SAV

Function Save the current setting into nonvolatile memory for next time use.
Syntax *SAV
Example *SAV

:HOLD

Function Turn on/off hold mode.
Syntax :HOLD {<Boolean>}
:HOLD?
Example :HOLD OFF
:HOLD? -> 0

:VOLTage:RANGe

Function Set the voltage range.
Syntax :VOLTage:RANGe {<Voltage>}
:VOLTage:RANGe?
<Voltage> = 60,600
Example :VOLTage:RANGe 60
:VOLTage:RANGe? -> 60

:VOLTage:AUTo

Function Turn on or off auto range of voltage.
Syntax :VOLTage:AUTo {<Boolean>}
:VOLTage:AUTo?
Example :VOLTage:AUTo 1
:VOLTage:AUTo? -> 1

:CURRent:RANGe

Function Set the current range.
Syntax :CURRent:RANGe {<Current>}
:CURRent:RANGe?
<Current> = 0.05, 0.1, 10

Example :CURRent:RANGe 0.05
:CURRent:RANGe? -> 0.05

:CURRent:AUTo

Function Turn on or off auto range of current.
Syntax :CURRent:AUTo {<Boolean>}
:CURRent:AUTo?
Example :CURRent:AUTo 1
:CURRent:AUTo? -> 1

:AVERaging

Function Set the average switch state and average count.
Syntax :AVERaging {<Average>}
:AVERaging?
<Average> = OFF, 8,16,32,64
·OFF = Average is turned off.
·8,16,32,64 = Average is turned on and it represents the average count.
Example :AVERaging 16
:AVERaging? -> 16

:MUTe

Function Turn on or off mute key.
Syntax :MUTe {<Boolean>}
:MUTe?
Example :MUTe 1
:MUTe? ->1

:ALARm:FLAG?

Function Query the alarm state.
Syntax :ALARm:FLAG? {<State>}
<State> = 0- not detecting, 1- PASS, 2- NG
Example :ALARm:FLAG?-> 0 # acquire the alarm state.

:ALARm:VOLTageFLAG?

Function Query the voltage alarm state.
Syntax :ALARm:VOLTageFLAG? {<State>}
<State> = 0- not detecting, 1- PASS, 2- NG
Example :ALARm:VOLTageFLAG?-> 0 # acquire the voltage alarm state.

:ALARm:SWItch:Total

Function Set the main switch of alarm.
Syntax :ALARm:SWItch:Total {<bool>}
:ALARm:SWItch:Total?
Example :ALARm:SWItch:Total ON
:ALARm:SWItch:Total? -> ON
Description If it need to read the main switch or other switch, change the third parameter Total.

:ALARm:ALARmpar:VOLTage:HIGH

Function Set the upper limit of voltage alarm.
Syntax :ALARm:ALARmpar:VOLTage:HIGH {<NR3>}
:ALARm:ALARmpar:VOLTage:HIGH?
Example :ALARm:ALARmpar:VOLTage:HIGH 250.5
:ALARm:ALARmpar:VOLTage:HIGH? -> 250.5
Description If it need to read the upper limit of voltage alarm or the upper limit of other parameter, change the third parameter **VOLTage**.

Example Set the upper limit of current alarm.
:ALARm:ALARmpar:CURRent:HIGH 1.065
:ALARm:ALARmpar:VOLTage:HIGH? -> 1.065

Set the upper limit of power.
:ALARm:ALARmpar**:**POWer:ACTive:HIGH 500
:ALARm:ALARmpar**:**POWer:ACTive:HIGH? -> 500

:ALARm:ALARmpar:VOLTage:LOW

Function Set the lower limit of voltage alarm.
Syntax :ALARm:ALARmpar:VOLTage:LOW {<NR3>}
:ALARm:ALARmpar:VOLTage:LOW?
Example :ALARm:ALARmpar:VOLTage:LOW 250.5
:ALARm:ALARmpar:VOLTage:LOW? -> 250.5
Description If it need to read the lower limit of voltage alarm or the lower limit of other parameter, change the third parameter **VOLTage**.

Example Set the lower limit of current alarm.
:ALARm:ALARmpar:CURRent:LOW 1.065
:ALARm:ALARmpar:VOLTage:LOW? -> 1.065

Set the lower limit of power.
:ALARm:ALARmpar:POWer:ACTive:LOW 500
:ALARm:ALARmpar:POWer:ACTive:LOW? -> 500

:ALARm:ALARmpar:DELy

Function Set the alarm delay.
Syntax :ALARm:ALARmpar:DELy {<NR1>}
:ALARm:ALARmpar:DELy?
Example :ALARm:ALARmpar:DELy 5
:ALARm:ALARmpar:DELy? -> 5

:ALARm:ALARmpar:BEEp

Function Set the alarm beeper times.
Syntax :ALARm:ALARmpar:BEEp{<NR1>}
:ALARm:ALARmpar:BEEp?
Example :ALARm:ALARmpar:BEEp 10
:ALARm:ALARmpar:BEEp? -> 10

:MEASure:FREQuency:VOLTage

Function Query the measuring frequency of voltage.
Syntax :MEASure:FREQuency:VOLTage?
Example :MEASure:FREQuency:VOLTage? -> 50.00

:MEASure:VOLTage?

Function Query the currently measured value of AC voltage.
Syntax :MEASure:VOLTage?
Example :MEASure:VOLTage? -> 110.36

:MEASure:VOLTage:PEAK+?

Function Query the currently measured value of voltage positive peak.
Syntax :MEASure:VOLTage:PEAK+?
Example :MEASure:VOLTage:PEAK+? -> 110.36

:MEASure:VOLTage:PEAK-?

Function Query the currently measured value of voltage negative peak.
Syntax :MEASure:VOLTage:PEAK-?
Example :MEASure:VOLTage:PEAK-? -> -110.36

:MEASure:CURRent?

Function Query the currently measured value of current.
Syntax :MEASure:CURRent?
Example :MEASure:CURRent? -> 10.23

:MEASure:CURRent:PEAK+?

Function Query the currently measured value of current positive peak.
Syntax :MEASure:CURRent:PEAK+?
Example :MEASure:CURRent:PEAK+? -> 14.53

:MEASure:CURRent:PEAK-?

Function Query the currently measured value of current negative peak.
Syntax :MEASure:CURRent:PEAK-?
Example :MEASure:CURRent:PEAK-? -> -14.53

:MEASure:POWer:ACTive?

Function Query the currently measured value of active power.
Syntax :MEASure:POWer:ACTive?
Example :MEASure:POWer:ACTive? -> 30.5

:MEASure:POWer:APParent?

Function Query the currently measured value of apparent power.
Syntax :MEASure:POWer:APParent?
Example :MEASure:POWer:APParent? -> 30.5

:MEASure:POWer:PFACtor?

Function Query the currently measured value of power factor.
Syntax :MEASure:PFACtor?
Example :MEASure:PFACtor? -> 0.519

:MEASure:POWer:PHAse?

Function Query the currently measured value of phase.
Syntax :MEASure:PHAse?
Example :MEASure:PHAse? -> 60.5

:SYSTem:ERRor?

Function Query the last error code and information.
Syntax :SYSTem:ERRor?
Example :SYSTem:ERRor? -> -113,"Undefined header"
Description If there is no error, then it returns 0,"No error"

:LOCK

Function Set the lock key state on the front panel.
Syntax :LOCK {<Boolean>}
:LOCK?
Example :LOCK 1
:LOCK? -> 1

Acquire Newest Measurement Data

The measurement data will acquire in breaks when in auto range, the acquired data is “nan” via“:MEASure:###:###?”. If user wants to acquire the newest measurement data, you need to exit break state and then re-acquire the data. By detecting the difference in the number of updates before and after, it can determine whether the data update event occurs. To obtain the latest updated data, the specific method is as follows.

:UPDAte:COUNt? -> 101
:UPDAte:COUNt? -> 101
…
:UPDAte:COUNt? -> 102 # data update event occurs
:MEASure:VOLTage? -> 110.36
:MEASure:CURRent? -> 10.23
…

Modbus Programming Manual

Modbus Introduction

Modbus is a widely used field bus protocol. Multiple slave machines can easily network with the host through Modbus, the host computer can be PC or PLC. Modbus has two varieties, which is Modbus-RTU and Modbus-ASC. MP701125 only supports Modbus-RTU.

Communication Interface and Setting

Communication Data Format

During communication, data is return by word (word- two bytes). In each returned word, MSB first, then LSB. If two bytes are continuously return (such as floating point number or long integer), MSB first, then LSB.

Data Format	Number of Register	Number of Byte	Description
Byte Data		1	
Integer Data	1	2	A return, MSB first, then LSB.
Long Integer Data	2	4	Return in two words, MSB first, then LSB.
Floating Point Data			

Interconversion of Word and Float

A register in Modbus protocol is 16 bits, that is a word. The previous section mentioned that floating-point take up two registers, i.e., two words. After receiving byte data, user needs to convert a word to a floating-point or a floating-point number to a word.

The following code is a good example for interconversion of word and float point.

```
/* C program for converting a floating point number to two words */
void FloatToWord(float Data,u16 *Word)
{
 union
 {
        float Data;
        unsigned char Byte[4];
 }FloatData;
 FloatData.Data=Data;
 Word[0]=(u16)FloatData.Byte[3]<<8|FloatData.Byte[2];
 Word[1]=(u16)FloatData.Byte[1]<<8|FloatData.Byte[0];
}
/* C program for converting two words to a floating point number */
float WordToFloat(const u16 *Word)
```

```
{
union
{
        float Data;
        unsigned char Byte[4];
}FloatData;
FloatData.Byte[3]=(Word[0]>>8)&0xFF;
FloatData.Byte[2]=(Word[0])&0xFF;
FloatData.Byte[1]=(Word[1]>>8)&0xFF;
FloatData.Byte[0]=(Word[1])&0xFF;
return FloatData.Data;
}
```

Modbus-RTU

Function code 03H， read multiple words

This command can read at least one word. The following example issues a read command from the master station to slave station 1, reading two consecutive words that start from address 0096H (150).

Command Message of Master Station

Slave address	01H
Function code	03H
Position of initial data	00H (MSB)
	96H (LSB)
Data number (calculating in word)	00H
	02H
CRC (Check Low)	24H (LSB)
CRC (Check High)	27H (MSB)

Respond Message of Slave Station (Normal)

Slave address	01H
Function code	03H
Data number (calculated in byte)	04H
Start data address 0096H	40H (MSB)
	DDH (LSB)
The second data address 0097H	1EH (MSB)
	B8H (LSB)
CRC (Check Low)	76H (LSB)
CRC (Check High)	1BH (MSB)

Respond Message of Slave Station (Abnormal)

Slave address	01H
Function code	83H
Error Code	02H
CRC (Check Low)	C0H (LSB)
CRC (Check High)	F1H (MSB)

Function code 10H, written multiple words

This command can write at least one word. The following example issues a write command from the master station to slave station 1, written data of two words 0003H and 0002H from the start address 0065H(101) . That is write 0003H into address 0065H, write 0002H into address 00066H. The slave replies to the master station when the write is completed.

Command Message of Master Station

Slave address	01H
Function code	10H
Position of initial data	00H
	65H
Data number (calculating in word)	00H (MSB)
	02H (LSB)
Data number (calculated in byte)	04H
The first data address	00H (MSB)
	03H (LSB)
The second data address	00H (MSB)
	02H (LSB)
CRC (Check Low)	44 (LSB)
CRC (Check High)	79 (MSB)

Respond Message of Slave Station (Normal)

Slave address	01H
Function code	10H
Position of initial data	00H (MSB)
	65H (LSB)
Data number (calculating in word)	00H (MSB)
	02H (LSB)
CRC (Check Low)	51H (LSB)
CRC (Check High)	D7H (MSB)

Respond Message of Slave Station (Abnormal)

Slave address	01H
Function code	90H
Error Code	02H
CRC (Check Low)	CDH (LSB)
CRC (Check High)	C1H (MSB)

Description of Error Code

Error code parsing for respond message of slave station (abnormal) as shown in the following table.

Error Code	Name	Description
01	Illegal function code	The slave machine does not support this function code.
02	Illegal data address	The starting data position or a combination of the starting data position and the number of transmitted data received from the machine is not allowed.
03	Illegal data value	Data received from the machine is not allowed.

Register List

Register List

*Notes: R represents it can be read, that is support command 03H. W represents it can be wirtten, that is support command 10H.

Data Name	Data Format	Unit	Initial Address	Number of Register	Read/Write	Remarks
Product Information						
Product information	ASCII		0	50	R	"MP701125,012345678,F1.02"
Retain			50	50	R	
Parameter Setting						
Measurement mode	U16		100	1	R/W	0 (AC+DC), 1 (AC), 2 (DC)
Voltage range	U16		101	1	R/W	0 (Auto), 1 (75V), 2 (150V), 3 (300V), 4 (600V)
Current range	U16		102	1	R/W	0 (Auto), 1 (0.5A), U162 (2A), 3 (8A), 4 (20A)
Update cycle	U16		103	1	R/W	0 (0.1s), 1 (0.25s), 2 (0.5s), 3 (1s), 4 (2s), 5 (5s)
Average	U16		104	1	R/W	0 (the average is turned off), 1 (8 times), 2 (16 times), 3 (32 times), 4 (64 times)
Data hold	U16		105	1	R/W	0 (forbidden), 1 (enabled)
Display	U16		106	1	R/W	0 (display PF value), 1 (display frequency value)
Mute	U16		107	1	R/W	0 (forbidden), 1 (enabled)
Upper limit of current alarm	Float	A	108	2	R/W	0.000~40.000，When the upper limit and the lower limit is set to 0 at the same time, it represents the alarm is forbidden.
Lower limit of current alarm	Float	A	110	2	R/W	
Upper limit of power limit	Float	W	112	2	R/W	0.000~48000.0，When the upper limit and the lower limit is set to 0 at the same time, it represents the alarm is forbidden.
Lower limit of power limit	Float	W	114	2	R/W	
Alarm delay	Float	S	116	2	R/W	0.0~99.9
Measurement data type	U16		120	1	R/W	0: real-time measurement data 1: recently update TRMS data
Parament Configuration of Instrument						
Default setting	U16		140	1	W	0 (forbidden), 1 (set the parameter to

						the default value)
Save parameter	U16		141	1	W	0 (forbidden), 1 (save the parameter into system storage for next use)
Measurement Data						
Voltage value	Float	V	150	2	R	The numerical value is related to the measurement model.
Current value	Float	A	152	2	R	The numerical value is related to the measurement model.
Active power	Float	W	154	2	R	
Power factor	Float		156	2	R	
Frequency of voltage	Float	Hz	158	2	R	
Alarm state of current	U16		160	1	R	0(alarm forbidden), 1 (wait for connect to the load), 2 (testing), 3 (the result is normal), 4 (the result is low), 5 (the result is high)
Alarm state of power	U16		161	1	R	
Data update count	U16		162	1	R	The latest measurements are available when changes in this data are detected.

Expression of Special Data

- Floating point 9.91E+37 in measurement data, which represents invalid data, window displays " --- ";
- Floating point 9.9E+37 in measurement data, which represents the data is overrange or overflow, window displays " --oL-" or "--oF-".

Appendix 1 CRC Calculation

```
const unsigned char aucCRCHi[] = {
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40
};

const unsigned char aucCRCLo[] = {
    0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7,
    0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E,
    0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09, 0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9,
    0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC,
    0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
    0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32,
    0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D,
    0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A, 0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38,
    0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF,
    0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
    0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1,
    0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4,
    0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F, 0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB,
    0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA,
    0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
    0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0,
```

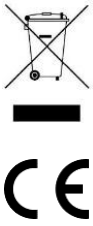
```
    0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97,
    0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C, 0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E,
    0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89,
    0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
    0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83,
    0x41, 0x81, 0x80, 0x40
};

unsigned short usMBCRC16( unsigned char    * pucFrame, unsigned short usLen )
{
    unsigned  char  ucCRCHi  =  0xFF;
    unsigned  char  ucCRCLo  =  0xFF;
    int                     iIndex;

    while( usLen-- )
    {
        iIndex = ucCRCLo ^ *( pucFrame++ );
        ucCRCLo = ( unsigned char)( ucCRCHi ^ aucCRCHi[iIndex] );
        ucCRCHi = aucCRCLo[iIndex];
    }
    return ( unsigned short )( ucCRCHi << 8 | ucCRCLo );
}

unsigned char SendBuf[30];
void main(void)
{
    unsigned short CRC;
    unsigned short SendLen;
    SendLen = 0;

    SendBuf[SendLen++] = 0x01;
    SendBuf[SendLen++] = 0x03;
    SendBuf[SendLen++] = 0x00;
    SendBuf[SendLen++] = 0x96;
    SendBuf[SendLen++] = 0x00;
    SendBuf[SendLen++] = 0x02;
    CRC = usMBCRC16(SendBuf,SendLen); /*Start to caclulating CRC */
    SendBuf[SendLen++] = CRC&0xFF;        /* CRC LSB */
    SendBuf[SendLen++] = (CRC>>8)&0xFF;    /* CRC MSB */
}
```



INFORMATION ON WASTE DISPOSAL FOR CONSUMERS OF ELECTRICAL & ELECTRONIC EQUIPMENT.

When this product has reached the end of its life it must be treated as Waste Electrical & Electronics Equipment (WEEE). Any WEEE marked products must not be mixed with general household waste, but kept separate for the treatment, recovery and recycling of the materials used. Contact your local authority for details of recycling schemes in your area.

Made in China
150 Armley Road, Leeds, LS12 2QQ (UK)
Riverside One, Sir John Rogerson Quay, Dublin 2 (EU)