ELECTRICAL MULTIFUNCTION METER COMMUNICATION PROTOCOL

MULTIMETERS MODBUS-RTU COMMUNICATION PROTOCOL

1) MODBUS PROTOCOL

Modbus is a master-slave communication protocol able to support up to 255 slaves organized as a bus or as a star network;
The phisical link layer can be RS232 for a point to point connection or RS485 for a network.
The communication is half-duplex.
The network messages can be Query-Response or Broadcast type.
The Query-Response command is transmitted from the Master to an estabilished Slave and generally it is followed by an answering message.
The Broadcast command is transmitted from the Master to all Slaves and is never followed by an answer.

MODBUS use two modes for transmission.
A) ASCII Mode: uses a limited character set as a whole for the comunication.

B) RTU Mode: binary,with time frame synchronization,fasterthan the ASCII Mode,uses half so long data block than the ASCII Mode; max 127 unit .

analyzers employ RTU mode.

GENERIC MESSAGE STRUCTURE:

START OF FRAME	ADDRESS FIELD	FUNCTION CODE	DATA FIELD	ERROR CHECK	END OF FRAME

START OF FRAME = Starting message marker
ADDRESS FIELD = Includes device address in wich you need to comunicate in Query-Response mode.
In case the message is a Broadcast type it includes 00.
FUNCTION CODE = Includes the operation code that you need to perform.
DATA FIELD = Includes the data field.
ERROR CHECK = Field for the error correction code.
END OF FRAME = End message marker.

Communication frame structure:

Mode RTU

Bit per byte = mode a) 1 Start, 8 Bit, 1 Parity, 1 Stop
mode b) 1 Start, 8 Bit, 2 Stop
START OF FRAME = silence on line for time >=4 characters
ADDRES FIELD = 1 character
FUNCTION CODE = 1 character
DATA FIELD = N characters
ERROR CHECK = 16 bit CRC
END OF FRAME = silence on line for time >=4 characters

CRC GENERATION

Example of the CRC-16 generation with "C" language:

```
static unsigned char auchCRCHi [ ] = {
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, 0x01, 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
} ;

static unsigned char auchCRCLo [ ] = {
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 CRC16 (ptMsg, usDataLen)
unsigned char *ptMsg;                    / * message to calculate CRC upon * /
unsigned short usDataLen;                  / * number of bytes in message * /
{
      unsigned char uchCRCHi = 0xFF;          / * CRC high byte * /
      unsigned char uchCRCLo =0xFF;           / * CRC low byte * /
      unsigned uIndex;

      while (usDataLen--)                / * pass through message buffer * /
      {
         uIndex = uchCRCHi ^ *ptMsg++;        /.* calculate the CRC * /
         uchCRCHi = uchCRCLo ^ auchCRCHi [ uIndex ] ;
         uchCRCLo = auchCRCLo [ uIndex ]
      }
      return (uchCRCHi « 8 | uchCRCLo ) ;
}
```

Note: The"Error Check (CRC)" field must be computed reffering to the characters from the first of ADDR to the last of DATA inclusive.

2) READING OF THE REGISTERS (Function Code $ 03)

Reads the binary contents of holding registers (2X references) in the slave.
Broadcast is not supported.
The Query message specified the starting register and quantity of register to be read.

QUERY:

START OF FRAME	ADDRESS FIELD	FUNCTION CODE	START ADDRESS	No. OF REGISTERS	ERROR CHECK	END OF FRAME

START OF FRAME = Starting message marker.
ADDRESS FIELD = device address (00...FF HEX) (1 byte).
FUNCTION CODE = Operation code (03 HEX) (1 byte).
START ADDRESS = First register address to be read (2 byte).
No.OF REGISTERS= Number of registers (max 32) to be read (4 byte for 1 meas value).
ERROR CHECK = Check sum.
END OF FRAME = End message marker.

WARNING:
It is possible to read more than one variable at the same time (max 16) only if their addresses are consecutive and the variables on the same line cannot be divided.

The register data in the response message are packet as two bytes per register,with the binary contents right justified within each byte.

For each register,the first byte contains the high order bits and the second contains the low order bits.

The wait time for response is preferred 300 to 500msecond.

RESPONSE:

START OF FRAME	ADDRESS FIELD	FUNCTION CODE	No. OF BYTES	D0 D1 .. Dn	ERROR CHECK	END OF FRAME

START OF FRAME = Starting message marker.
ADDRESS FIELD = device address (00...FF HEX) (1byte).
FUNCTION CODE = Operation code (03 HEX) (1 Byte).
No.OF SEND BYTES = Number of data bytes (00...?? HEX) (1 byte). 1 register requires 2 data bytes.
D0, D1, .., Dn = data bytes (00...?? HEX) (Nr. of register x 2 = n. byte).
ERROR CHECK = Check sum.
END OF FRAME = End message marker .

See the TABLE OF REGISTERS to the sect. 5 and see the EXAMPLE to the sect. 6:

3) SETUP OF THE PARAMETERS (Function Code $ 10)

Write values into a sequence of holding registers (2X references).

WARNING: It is possible to write more than one variable at the same time only if their addresses are consecutive and the variables on the same line cannot be divided. (max of 4 consecutive register on the same message).

QUERY:

START OF FRAME	ADDRESS FIELD	FUNCTION CODE	START ADDRESS	NO. OF REGISTERS	NO. OF BYTES	D0 D1 .. Dn	ERROR CHECK	END OF FRAME

START OF FRAME = Starting message marker.
ADDRESS FIELD = device address (00...FF HEX) (1 byte).
FUNCTION CODE = Operation code (10 HEX) (1 byte).
START ADDRESS = First register address to be written (2 byte).
No. OF REGISTER = Number of registers to be written (1 to 4,...) (2 byte).
No. OF BYTES = Number of data bytes (HEX) (1 byte): 1register requires 2 data bytes.
D0,D1,..,Dn = Data bytes (00...? HEX) (1 byte) (Nr.of register x 2 = n. byte).
ERROR CHECK = Check sum.
END OF FRAME = End message marker.

The normal response retums the slave address, function code,starting address and quantity of register preset.

RESPONSE:

START OF FRAME	ADDRESS FIELD	FUNCTION CODE	START ADDRESS	NO.OF REGISTERS	ERROR CHECK	END OF FRAME

START OF FRAME = Starting message marker.
ADDRESS FIELD = device address (00...FF HEX) (1 byte).
FUNCTION CODE = Operation code (10 HEX) (1 byte).
START ADDRESS = First register address to be written (2 byte).
No. OF REGISTER = Number of registers to be written (2 byte).
ERROR CHECK = Check sum.
END OF FRAME = End message marker.

See the TABLE OF NPM REGISTERS to the sect. 5 and see the EXAMPLE to the sect. 6 :

4) ERROR MESSAGE FROM SLAVE TO MASTER

When a slave device receives a not valid query, it does transmit an error message.

RESPONSE:

```
==========================================
START      | ADDRESS | FUNCTION | ERROR | ERROR | END
OF FRAME | FIELD     | CODE     | CODE  | CHECK | OF FRAME
==========================================
```

START OF FRAME = Starting message marker.
ADDRESS FIELD = device address (00...FF HEX) (1 byte).
FUNCTION CODE = Operation code with bit 7 high (1 byte).
ERROR CODE = Message containing communication failure (1 byte).
ERROR CHECK = Check sum.
END OF FRAME = End message marker.

ERROR EXAMPLE

QUERY

Field Name	Example (Hex)
Slave Address	01
Function Code	03
Starting Address Hi	00
Starting Address Lo	00
Number Of Word Hi	00
Number Of Word Lo	05
Error Check (CRC)	??
	??

RESPONSE

Field Name	Example (Hex)
Slave Address	01
Function Code	83 (1)
Error Code	02 (2)
Error Check (CRC)	??
	??

(1): Function Code transmitted by master with bit 7 high.
(2): Error type:
01= Illegal Function
02= Illegal data address
03= Illegal data value

5) TABLE OF NPM REGISTERS

The following table shown all the NPM registers.

MEASURED VALUES (Function code $ 03)

Register HEX	Word	Description	U.M.	
$1000	2	3-PHASE SYSTEM VOLTAGE	[V]	(Unsigned
$1002	2	PHASE VOLTAGE L_{1-N}	[V]	(Unsigned)
$1004	2	PHASE VOLTAGE L_{2-N}	[V]	(Unsigned)
$1006	2	PHASE VOLTAGE L_{3-N}	[V]	(Unsigned)
$1008	2	LINE VOLTAGE L_{1-2}	[V]	(Unsigned)
$100A	2	LINE VOLTAGE L_{2-3}	[V]	(Unsigned)
$100C	2	LINE VOLTAGE L_{3-1}	[V]	(Unsigned)
$100E	2	3-PHASE SYSTEM CURRENT	[mA]	(UnSigned)
$1010	2	LINE CURRENT L_1	[mA]	(UnSigned)
$1012	2	LINE CURRENT L_2	[mA]	(UnSigned)
$1014	2	LINE CURRENT L_3	[mA]	(UnSigned)
$1016	2	3-PHASE SYS. POWER FACTOR	[-]	(Signed)
$1018	2	POWER FACTOR L_1	[-]	(Signed)
$101A	2	POWER FACTOR L_2	[-]	(Signed)
$101C	2	POWER FACTOR L_3	[-]	(Signed)
$101E	2	3-PHASE SYSTEM COS∅	[-]	(Signed)
$1020	2	PHASE COS$\varnothing_1$	[-]	(Signed)
$1022	2	PHASE COS$\varnothing_2$	[-]	(Signed)
$1024	2	PHASE COS$\varnothing_3$	[-]	(Signed)
$1026	2	3-PHASE S. APPARENT POWER	[VA]	(UnSigned)
$1028	2	APPARENT POWER L_1	[VA]	(UnSigned)
$102A	2	APPARENT POWER L_2	[VA]	(UnSigned)
$102C	2	APPARENT POWER L_3	[VA]	(UnSigned)
$102E	2	3-PHASE SYS. ACTIVE POWER	[W]	(UnSigned)
$1030	2	ACTIVE POWER L_1	[W]	(UnSigned)
$1032	2	ACTIVE POWER L_2	[W]	(UnSigned)
$1034	2	ACTIVE POWER L_3	[W]	(UnSigned)
$1036	2	3-PHASE S. REACTIVE POWER	[VAR]	(UnSigned)
$1038	2	REACTIVE POWER L_1	[VAR]	(UnSigned)
$103A	2	REACTIVE POWER L_2	[VAR]	(UnSigned)
$103C	2	REACTIVE POWER L_3	[VAR]	(UnSigned)
$103E	2	3-PHASE SYS. ACTIVE ENERGY	[100*Wh]	(Unsigned)
$1040	2	3-PHASE S. REACTIVE ENERGY	[100*VARh]	(Unsigned)
$1046	2	FREQUENCY	[mHz]	(Unsigned)
$1048	2	NEUTRAL CURRENT	[mA]	(Unsigned)

SPECIAL ENERGY COUNTER MEASURED VALUES ONLY FOR NPM WITH TARIFF FUNCTION (TIME BANDS)
MODBUS DATA REGISTERS FOR ENERGY COUNTERS

Only for NPM with tariff function (time bands) the energy counter registers are different as follow:

Register HEX	Word	Description	U.M.	
$103E	2	3-PHASE SYS. ACTIVE ENERGY TIMEBAND 1	[100*Wh]	(Unsigned)
$1040	2	3-PHASE S. REACTIVE ENERGY TIMEBAND 1	[100*VARh]	(Unsigned)
$1042	2	3-PHASE SYS. ACTIVE ENERGY TIMEBAND 2	[100*Wh]	(Unsigned)
$1044	2	3-PHASE S. REACTIVE ENERGY TIMEBAND 2	[100*VARh]	(Unsigned)

The RESET ENERGY COUNTER FUNCTION in this case reset all the counters.

VALUES STORED IN EEPROM (Function.code $03)

Register HEX	Word	Word Description	U.M.	
$1060	2	MAX ISTANT. CURRENT L1	[mA]	(Unsigned)
$1062	2	MAX ISTANT. CURRENT L2	[mA]	(Unsigned)
$1064	2	MAX ISTANT. CURRENT L3	[mA]	(Unsigned)
$1066	2	MAX ISTANT. 3-PHASE . ACTIVE POWER	[W]	(Unsigned)
$1068	2	MAX ISTANT. 3-PHASE . APPAR. POWER	[VA]	(Unsigned)
$106A	2	MAX 15' MEAN .CURRENT L1	[mA]	(Unsigned)
$106C	2	MAX 15' MEAN .CURRENT L2	[mA]	(Unsigned)
$106E	2	MAX 15' MEAN .CURRENT L3	[mA]	(Unsigned)
$1070	2	MAX 15' MEAN .3-PHASE ACTIVE POWER	[W]	(Unsigned)

READ & WRITE PARAMETERS (Function code $03 & $10)

Register	Word	Description	Range
$11A0	2	TRANSFORM RATIO IL1-IL2-IL3 KCT	1÷2000
$11A2	2	TRANSFORM RATIO KVT	1÷400
$11A4	2	kWhr/kVARhr PULSE WEIGHT	1÷4 1 = 10 WhR-VARhr / PULSE 2 = 100 WhR-VARhr / PULSE 3 = 1 kWhR-VARhr / PULSE 4 = 10 kWhR-VARhr / PULSE
$11A6	2	TRANSFORM RATIO I NEUTRAL KCTN	1÷2000

WRITE PARAMETERS (Function code $10)

Register	Word	Description	WRITE Value	
			MSB word	LSB word
$11B0	2	RESET ENERGY COUNTERS	$11B0	$55AA
$11B2	2	RESET MAX. ISTANT. STORED I	$11B2	$55AA
$11B4	2	RESET 15' MEAN STORED	$11B4	$55AA
$11B6	2	RESET ALL STORED & COUNTERS	$11B6	$55AA

NOTE: WHEN THE INSTRUMENT CAN'T MEASURE IT SEND 0000 AS VALUE.

6) READING EXAMPLE

This is an example of transmitted data to NPM at address 01, requesting 16 variables, as follows:

$101E	2	3-PHASE SYSTEM COS∅	[-]	(Signed)
$1020	2	PHASE COS$\varnothing_1$	[-]	(Signed)
$1022	2	PHASE COS$\varnothing_2$	[-]	(Signed)
$1024	2	PHASE COS$\varnothing_3$	[-]	(Signed)
$1026	2	3-PHASE S. APPARENT POWER	[VA]	(UnSigned)
$1028	2	APPARENT POWER L_1	[VA]	(UnSigned)
$102A	2	APPARENT POWER L_2	[VA]	(UnSigned)
$102C	2	APPARENT POWER L_3	[VA]	(UnSigned)
$102E	2	3-PHASE SYS. ACTIVE POWER	[W]	(UnSigned)
$1030	2	ACTIVE POWER L_1	[W]	(UnSigned)
$1032	2	ACTIVE POWER L_2	[W]	(UnSigned)
$1034	2	ACTIVE POWER L_3	[W]	(UnSigned)
$1036	2	3-PHASE S. REACTIVE POWER	[VAR]	(UnSigned)
$1038	2	REACTIVE POWER L_1	[VAR]	(UnSigned)
$103A	2	REACTIVE POWER L_2	[VAR]	(UnSigned)
$103C	2	REACTIVE POWER L_3	[VAR]	(UnSigned)

EXAMPLE

Stream data send to NPM (H suffix mean hex data format):

01H NPM address
03H Read function
10H Address of 1st register requested (101EH)
1EH
00H Nr of Register requested (2 regs for each variable =32 registers = 0020H)
20H
20H CRC
D4H CRC

Response from NPM:

01H NPM address
03H Read function
40H Nr. of send bytes
… Follow 64 bytes of data
…
…
05H CRC
11H CRC

If response from NPM doesn't happen:
- check connection from NPM and RS232/RS485 converter ;
- check if data outgoing from the RS232 serial port of the PC come in the RS232/485 converter
- try to increase the wait time for response (300 to 500mS is good);
- check if the transmitted data stream is **EXACTLY** as in example, monitoring the data on the RS485 serial line with a terminal (eg. Hyperterminal or other emulator);

End of document.