MODBUS
PROTOCOL MANUAL
PLC1, PLC2 AND POS2 BOARD

Language: English 0899.5512 E/1

11/2006

Table of Contens

RTU-MODBUS PROTOCOL MANUAL IN THE PLC1, PLC2 AND POS2 BOARD

Below is an explanation about the operation of the PLC1, PLC2 and POS2 boards in the RTU- Modbus network.

The baud rate is defined by a specific parameter and the following values are possible:

1 – 1200bps
2 – 2400bps
3 – 4800bps
4 – 9600bps (Factory Setting)
5 – 19200bps
6 – 38400bps

The communication is made through RS-232C, without parity, 8 bits and 2 stop bits.

The use of MIW-02 converters is required for the network implementation, which convert the RS-232C (point-to-point) into RS-485 (multipoint).

The PLC address in the network is defined by a specific parameter and can be set between 1 and 247 (0 is the broadcast address). The factory setting is 1.

Obs.: The system must be restarted after modifying the address parameters or baudrate.

What can be programmed on the board using the RTU-Modbus?

1 - Write/read parameters and markers (commands 3, 6 and 16):
By means of the RTU-Modbus protocol in the board one can read and write the board parameters (P750...P899), drive parameters (P000…P490) and macro parameters (PM0…PM31), in addition to the word and float markers. This operation can be carried out in only one parameter or in a parameter group.

2 - Analog Input Reading (command 4):
Analog drive inputs 1 and 2 (addresses 101 and 102) and analog board inputs (address 1 when available) can be read. The value is given in 15 bits, that is, for input values from 0 to 100% the read value can be 0 to 32,767.

3 - Control/Status of digital inputs and outputs and bit registers (commands 1, 2, 5 and 15):
It is posible the read (check the status) and write (control) the board or drive digital outputs, as well as, check in the digital input states (read). Besides that, the bit registers (retentive or non-retentive) can be read (status) and write (control). This operation can be performed by using one bit or a group of bits.

Note: If the user program uses any PLC digital output, this program will have priority over the write operation through the Modbus, i. e., the user program overwrites the status imposed by the Modbus protocol.

4 - Board identification Reading (command 43):
Through the command 43 you can read the board identification data, such as, manufacturer (WEG), model (PLC1.01, for example) and the firmware version (V1.40, for example).

Detailed Protocol Description:

1.1 MODBUS-RTU

O The Modbus protocol was developed in 1979. Currently this s a disclosed protocol and used widely by several equipment manufacturers. The RTU-Modbus communication for the PLC1 board has been developed according to the following documents:

1) MODBUS Protocol Reference Guide Rev. J, MODICON, June 1996.

2) MODBUS Application Protocol Specification, MODBUS.ORG, may 8th 2002.

The message format used by the elements that are part of the Modbus network, as well as the services (or functions) that are available via network and how these elements exchange the data via network are defined in these documents.

1.1.1 Transmission Modes

The ASCII and the RTU transmission modes are defined in the protocol specification. The modes define how the message bytes are transmitted. It's not possible to use the two transmission modes in the message network.

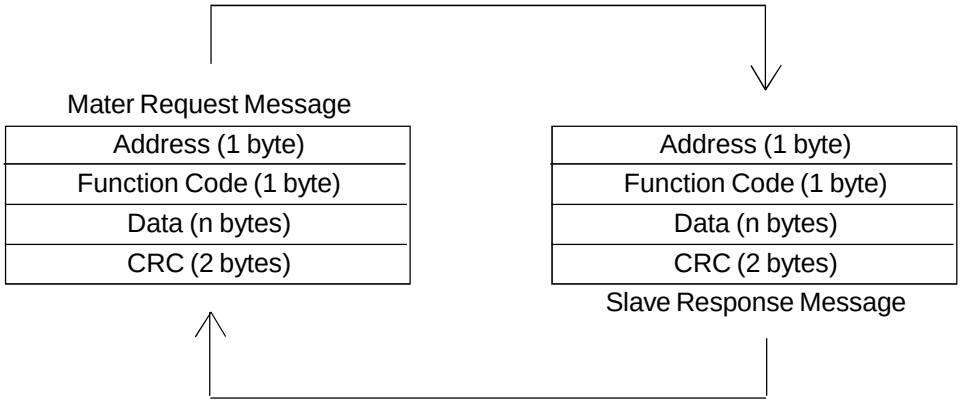
In the RTU mode, each word transmitted has a 1 start bit, eight data bits, 1 parity bit (optional) and 1 stop bit (2 stop bits if the parity bit is not used). The sequence of bits for transmission of a byte is as follows:

Start	B0	B1	B2	B3	B4	B5	B6	B7	Stop	Stop

In the RTU transmission mode, each byte is transmitted as being a single word with it value as a hexadecimal one. The board uses only this transmission mode for communication and does not, therefore, allow communication in the ASCII mode.

1.1.2 Message Structure in the RTU-Mode

A The RTU-Modbus network operates in the Master-Slave system. In this system there can be up to 247 slaves, but only one master.
Every communication starts with the master making a request to a slave and the slave answering to what has been requested. The structure is the same in both telegrams (request and answer): Address, Function Code, Data and CRC. Only the data field may vary in size depending on the request.



Address:

The master starts the communication by sending a byte with the slave address to which the message should be sent. On sending the response, the salve also starts the telegram with its own address. The master can also send a message to the 0 (zero) address which means that this message is destined to all the slaves of the network (broadcast). In this case no slave will answer to the master.

Function Code:

This field also contains a single byte and the master specifies the type of service or function requested to the slave (read. write, etc.). According to the protocol, each function is used to access a specific type of data.

Data Field:

This field has a variable size. The format and content depend on the function used and on the transmitted values. This field is described in the function description.

CRC:

A The last part of the telegram is the field for checking transmission errors. The method used is the CRC-16 (Cycling Redundancy Check).
This field is formed by two bytes. The least significant byte (CRC-) is transmitted first and only then the most significant byte (CRC+) is transmitted.
The CRC calculation is started by loading a 16 bit variable (referenced from now on as CRC variable) with the value FFFFh. After that, proceed according to the following routine:

1. The first message byte is submitted (only the data bits - start bit, parity bit and stop bit are not used) to an XOR logic (OR exclusive) with the 8 least significant bits of the CRC variable returning the result back to the CRC variable..

2. Then the CRC variable is shifted one position to the right, in the direction of the least significant bit and the position of the most significant bit is filled out with 0 (zero).

3. After this shift, the flag bit (the bit that has been shifted outside the CRC variable) is analyzed as follows:
   - ☑ If the bit value is 0 (zero), nothing is done.
   - ☑ SIf the bit value is 1, the CRC variable content is submitted to an XOR logic with a A001h constant value and the result is returned to the CRC variable.

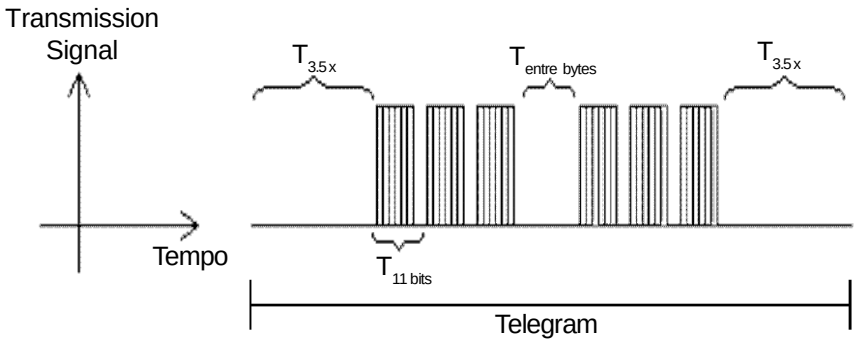
4. Steps 2 and 3 are repeated until the eight displacements have been carried out.

5. Steps 1 to 4 are repeated using the next message byte until the whole message has been processed.

The end content of the CRC variable is the value of the CRC field that is transmitted at the end of the telegram. The least significant part is transmitted first (CRC-) followed by the most significant part (CRC+).

Time Between Messages:

There is no specific character in the RTU-mode that indicates the beginning or the end of a telegram.
So what indicates when a new message begins or when it ends is the interruption of the data transmission in the network for a period longer than 3,5 times the transmission time of a data word (11 bits). Thus, if a telegram starts after this minimum time has elapsed without transmission the network elements assume that the received character represents the beginning of a new telegram. In the same way, the network elements will assume that the telegram transmission has ended after this time has elapsed again.
If during the transmission of a telegram, the time between the byte transmission is longer that this minimum established time, the telegram will be considered invalid because this the board will disregard the bytes already received and will assemble a new telegram with the bytes that are being transmitted.
The table below shows the times for two different communication rates.



Transfer Rate	$T_{11\ bits}$	$T_{3.5x}$
9600 bits/seg	1.146 ms	4.010 ms
19200 bits/seg	573 µs	2.005 ms

$T_{11\ bits}$ = Time for transmitting one telegram word.
$T_{between\ bytes}$ = Time between bytes (it can not be longer than $T_{3.5x}$).
$T_{3.5x}$ = Minimum interval to indicate the beginning and the end of the telegram (3.5 x T_{11bits}).

1.2 Operation in the Modbus-RTU

The boards operate as slaves of the Modbus-RTU network. The communication starts with the master in the network requesting some service to a network address. If the board has been configured for the corresponding address, then it processes the request and responds to the master as requested.
The boards use a serial interface to communicate with the Modbus-RTU network. There are two possibilities for the physical connection between the master in the network and a board: interfaces RS232 and RS485.

1.2.1 RS-232

- ☑ Used for the point-to-point connection (between a single slave and master).
- ☑ Maximum distance: 10 meters.
- ☑ Signal levels meet the EIA STANDARD RS-232C.
- ☑ Three wires: transmission (TX), reception (RX) and reference (0V).

1.2.2 RS-485

- ☑ Available through the MIW-02 converter connected to the RS-232 board.
- ☑ Used for the multipoint connection (several salves and one master).
- ☑ Maximum distance: 1000 meters (uses shielded cable).
- ☑ Signal levels meet the EIA STANDARD RS-485.

1.3 Board Setting in the Modbus-RTU Network

So the board can communicate correctly in the network, in addition to the physical connection, it is necessary to configure its address in the network as well as the respective transfer rate.

1.3.1 Board Address in the Network

- ☑ This address is defined by parameter 764 or 788.
- ☑ Each slave in the network must have its own address.
- ☑ The network master has no address.
- ☑ Even if the connection is made point-to-point, the slave address in the network must be known.

1.3.2 Transmission Rate

- ☑ Defined by parameter 765 or 789.
- ☑ Transmission Rate: 1200, 2400, 4800, 9600 or 19200 kbytes/sec.
- ☑ Parity: None.
- ☑ All slaves, and also the network master, must use the same communication rate and in the same parity.

1.4 Access to the Board and Drive Data

All board and drive parameters, as well as the board digital inputs and outputs can be accessed through the network.

1.4.1 Available Functions and Response Times

The board, parameters and markers have been defined as holding and input registers. In addition to these registers, it is also possible to directly access digital board or drive inputs and outputs, and bit markers by using the bit type functions of the Modbus. The following services (or functions) are available for accessing these bits and registers:

☑ Read Coils
Function: Reads the bit markers and the board and drive digital outputs
Function code: 01.
Broadcast: not supported.
Response time: 5 to 10 ms.

☑ Read Input Status
Function: Reads board and drive digital inputs.
Function code: 02
Broadcast: not supported.
Response time 5 to 10 ms.

☑ Read Holding Registers
Function: Reads word and float markers, board, drive and macro parameters.
Function code: 03.
Broadcast: not supported.
Response time 5 to 10 ms.

☑ Read Input Registers
Function: Reads board and drive analog inputs.
Function code: 04.
Broadcast: not supported.
Response time: 5 to 10 ms.

☑ Write Single Coil
Function: Writes in bit markers and digital board and drive digital outputs.
Function code: 05.
Broadcast: supported.
Response time 5 to 10 ms.

☑ Write Single Register
Function: Writes in a word marker, board, drive and macro parameter and board and drive digital outputs.
Function code: 06.
Broadcast: supported.
Response time 5 to 10 ms.

☑ Write Multiple Coils
Function: Writes in a multiple bit markers or board and drive digital outputs.
Function code: 15.
Broadcast: supported.
Response time 5 to 10 ms.

☑ Write Multiple Registers
Function: Writes on multiple word or float markers and board, drive and macro parameters, board and drive digital outputs
Function code: 16.
Broadcast: supported.
Response time: 10 to 20 ms for each written register.

☑ Read Device Identification
Function: Reads the board manufacturer, model and the firmware version
Function code: 43.
Broadcast: not supported.
Response time: 5 to 1o ms.

Note: The slaves of the Modbus-RTU network are addressed from 1 to 247. The master uses the 0 (zero) address to send a message common to all slaves (broadcast).

1.4.2 Data Addressing

The addressing of the board data is made with an offset equal to zero, which means that the address number is equal to the given number. The parameters, markers and the digital inputs/outputs are made available at the 0 (zero) address.

DRIVE Parameters		
Parameter Number	Modbus Address	
	Decimal	Hexadecimal
P000	0	0000h
P100	100	0064h
⋮	⋮	⋮
P490	490	01EAh

BOARD Parameters		
Parameter Number	Modbus Address	
	Decimal	Hexadecimal
P750	750	02Eeh
P800	800	0320h
⋮	⋮	⋮
P899	899	0383h

Macro Parameters		
Parameter Number	Modbus Address	
	Decimal	Hexadecimal
PM0	5000	1380h
⋮	⋮	⋮
PM31	5031	13A7h

Retentive WORD Markers		
Marker Number	Modbus Address	
	Decimal	Hexadecimal
MW6000	6000	1770h
⋮	⋮	⋮
MW6149	6149	1805h

Volatile WORD Markers		
Marker Number	Modbus Address	
	Decimal	Hexadecimal
MW7000	7000	1B58h
⋮	⋮	⋮
MW7649	7649	1D1E

Retentive FLOAT Markers		
Marker Number	Modbus Address	
	Decimal	Hexadecimal
MF9500	9500	251Ch
⋮	⋮	⋮
MF9524	9524	2534h

Volatile FLOAT Markers		
Marker Number	Modbus Address	
	Decimal	Hexadecimal
MF9000	9000	2328h
⋮	⋮	⋮
MF9174	9174	2306h

Retentive BIT Markers		
Marker Number	Modbus Address	
	Decimal	Hexadecimal
MX1000	1000	03E8h
⋮	⋮	⋮
MX1671	1671	0687h

Volatile BIT Markers		
Marker Number	Modbus Address	
	Decimal	Hexadecimal
MX2000	2000	07D0h
⋮	⋮	⋮
MX3407	3407	0D4Fh

System BIT Markers		
Marker Number	Modbus Address	
	Decimal	Hexadecimal
SX0	200	00C8h
SX1	201	0069h

System WORD Markers		
Marker Number	Modbus Address	
	Decimal	Hexadecimal
SW0	8000	1F40h
⋮	⋮	⋮
SW5	8005	1F45h

Analog BOARD Inputs		
Analog Input Number	Modbus Address	
	Decimal	Hexadecimal
IW1	1	1h

Analog DRIVE Inputs		
Analog Input Number	Modbus Address	
	Decimal	Hexadecimal
IW101	101	0065h
IW102	102	0066h

Analog BOARD Outputs		
Analog Output Number	Modbus Address	
	Decimal	Hexadecimal
QW1	8201	2009h
QW2	8202	200Ah

Analog DRIVE Outputs		
Analog Output Number	Modbus Address	
	Decimal	Hexadecimal
QW101	8301	2060h
QW102	8302	206Eh

Digital BOARD Inputs		
Digital Input Number	Modbus Address	
	Decimal	Hexadecimal
IX1	1	1h
⋮	⋮	⋮
IX9	9	9h

Digital DRIVE Inputs		
Digital Input Number	Modbus Address	
	Decimal	Hexadecimal
IX101	101	0065h
⋮	⋮	⋮
IX106	106	006Ah

Digital BOARD Outputs		
Digital Output Number	Modbus Address	
	Decimal	Hexadecimal
QX1	1	1h
⋮	⋮	⋮
QX6	6	6h

Digital DRIVE Outputs		
Digital Output Number	Modbus Address	
	Decimal	Hexadecimal
QX101	101	0065h
QX102	102	0066h
QX103	103	0067h

1.4.3 Address Ranges

The following table shows the elements that can be used in communication, their address ranges and the quantities available in each board model.

Element	Tipo	Cards					
		PLC1 V1.7X CFW-09		PLC2 V1.1X CFW-09		POS2 V1.2X CFW-09	
		[Beginning] [End]	Quantity	[Beginning] [End]	Quantity	[Beginning] [End]	Quantity
Retentive Bit Marker	Marcador de Bit Retentivos	%MX1000 %MX1671	672	%MX1000 %MX1671	672	%MX1000 %MX1671	672
Volatile Bit Marker	Marcador de Bit Voláteis	%MX2000 %MX3407	1308	%MX2000 %MX3407	1308	%MX2000 %MX3407	1308
Retentive Word Marker	Marcador de Word Retentivos	%MW6000 %MW6099	100	%MW6000 %MW6099	100	%MW6000 %MW6099	100
Volatile Word Marker	Marcador de Word Voláteis	%MW7000 %MW7649	650	%MW7000 %MW7649	650	%MW7000 %MW7649	650
System Bit Marker	Marcador de Bit de Sistema	%SX0	1	%SX0 %SX1	2	%SX0	1
System Word Marker	Marcador de Word de Sistema	%SW0 %SW5	6	%SW0 %SW5	6	%SW0 %SW5	6
Retentive Float Marker	Marcador de Float Retentivos	%MF9500 %MF9524	25	%MF9500 %MF9524	25	%MF9500 %MF9524	25
Volatile Float Marker	Marcador de Float Voláteis	%MF9000 %MF9174	175	%MF9000 %MF9174	175	%MF9000 %MF9174	175
User Parameters	Parâmetros do Usuário	%UW800 %UW899	100	%UW800 %UW899	100	%UW800 %UW899	100
Próprio Digital Inputs	Entradas Digitais Próprias	%IX1 %IX9	9	%IX1 %IX9	9	%IX1 %IX9	9
Digital Drive Inputs	Entradas Digitais do Drive	%IX101 %IX106	6	%IX101 %IX106	6	%IX101 %IX106	6
Próprio Digital Outputs	Saídas Digitais Próprias	%QX1 %QX6	6	%QX1 %QX6	6	%QX1 %QX6	6
Digital Drive Outputs	Saídas Digitais do Drive	%QX101 %QX103	3	%QX101 %QX103	3	%QX101 %QX103	3
Próprio Analog Inputs	Entradas Analógicas Próprias	-	-	%IW1	1	%IW1	1
Analog Drive Inputs	Entradas Analógicas do Drive	%IW101 %IW102	2	%IW101 %IW102	2	%IW101 %IW102	2
Próprio Analog Outputs	Saídas Analógicas Próprias	-	-	%QW1 %QW2	2	-	-
Macro Parameters	Parâmetros de Macro	%PM0 %PM31	32	%PM0 %PM31	32	%PM0 %PM31	32

Table 1.1 *- WLP address ranges*

1.4.4 System Markers Functions

The following markers have functions pre-defined by the system and must be used for this.

Marker	Reading Function	Writing Function
%SX0	Reading Function	Writing Function
%SX1	Drive Enabling Return	Enables Drive
%SW0	Motor PTC Sensor Input	-
%SW1	Drive Speed Return (rpm)	-
%SW2	Drive Speed Return (13/15 bits)	-
%SW3	-	Generates User Fault
%SW4	Card Fault Return	-
%SW5	-	Logical Command of the Drive

Table 1.2 *- System marker functions*

1.5 Detailed Functions Descriptions

This item contains a detailed description of the functions available in the Modbus-RTU communication boards. To be able to make the telegrams, it is important to observe the following:

- ☑ The values are always transmitted in hexadecimal format.
- ☑ The data address, the quantity of data and the register values are always represented in 16 bits. Because of this, it is necessary to transmit these fields using two bytes (high and low). To access bits, the representation of a bit depends on the function used.
- ☑ The telegrams, for both requests and responses, cannot pass 256 bytes.

1.5.1 Function 01 - Read Coils

It reads the content of a group of bits (bit markers, system markers or digital outputs of a board or drive) that must necessarily be in a numeric sequence. This function has the following structure for the read and response telegrams (the values are always hexadecimal and each field represents a byte):

Resquest (Master)	Response (Slave)
Slave address	Slave address
Function	Function
Address oh the initial bit (byte high)	Field Byte Count (number of the data bytes)
Address oh the initial (byte low)	Byte 1
Number of bits (byte high)	Byte 2
Number of bits (byte low)	Byte 3
CRC-	etc...
CRC+	CRC-
	CRC+

Each bit of the response is placed in a position of the data bytes sent by the slave. The first byte, in bits 0 to 7, receives the first 8 bits through the initial address indicated by the master. The other bytes (if the number of read bits is greater than 8), continue the sequence. If the number of bits read is not a multiple of 8, the remaining bits of the last byte must be filled out with 0 (zero).

Example: digital output reading, DO1 to DO6 at address 1:

Request (Master)		Response (Slave)	
Field	Value	Field	Value
Slave address	01h	Slave address	01h
Function	01h	Function	01h
Initial bit (high)	00h	Initial bit (high)	00h
Initial bit (low)	01h	Initial bit (low)	01h
Number of bits (high)	00h	Number of bits (high)	00h
Number of bits (low)	06h	Number of bits (low)	06h
CRC-	Edh	CRC-	Edh
CRC+	C8h	CRC+	C8h

Observation: do not forget that the digital outputs of the drive (RL1, RL2 and RL3) are represented in the board as DO101, DO102 and DO103, respectively.

1.5.2 Function 02 - Read Input Status

Reads the content of a group of digital inputs of the board and the drive, which must necessarily be in numeric sequence. This function has the following structure for the read and response telegrams (the values are always hexadecimal and each field represents a byte):

Resquest (Master)	Response (Slave)
Slave address	Slave address
Function	Function
Address oh the initial bit (byte high)	Field Byte Count (number of the data bytes)
Address oh the initial (byte low)	Byte 1
Number of bits (byte high)	Byte 2
Number of bits (byte low)	Byte 3
CRC-	etc...
CRC+	CRC-
	CRC+

Each bit of the response is placed in a position of the data bytes sent by the slave. The first byte, in bits 0 to 7, receives the first 8 bits from the initial address indicated by the master. The other bytes (if the number of read bits is greater than 8), continue the sequence. If the number of bits read is not a multiple of 8, the remaining bits of the last byte must be filled in with 0 (zero).

Example: reading of digital inputs, D12 to D17 in address 1:

Request (Master)		Response (Slave)	
Field	Value	Field	Value
Slave address	01h	Slave address	01h
Function	02h	Function	02h
Initial bit (high)	00h	Initial bit (high)	00h
Initial bit (low)	02h	Initial bit (low)	02h
Number of bits (high)	00h	Number of bits (high)	00h
Number of bits (low)	06h	Number of bits (low)	06h
CRC-	59h	CRC-	59h
CRC+	C8h	CRC+	C8h

In the example, since the number of bits read in less than 8, the slave needed only 1 byte for the response. The value of the byte was 21h, which in binaries takes the form of 0010 0001. Since the number of bits read is equal to 6, only the six less significant bits are of interest, which possess digital input values of 2 to 7. The other bits, as they were not requested, are filled in with 0 (zero).

Observation: do not forget that the digital inputs of the drive (DI1...DI6) are represented in the board as DI101...DI106, respectively.

1.5.3 Function 03 - Read Holding Register

Reads the content of a group of word and float markers or board/drive parameters that must necessarily be in numeric sequence. This function has the following structure for the read and response telegrams (the values are always hexadecimal, and each field represents one byte):

Resquest (Master)	Response (Slave)
Slave address	Slave address
Function	Function
Address oh the initial bit (byte high)	Field Byte Count
Address oh the initial (byte low)	Data 1 (high)
Number of bits (byte high)	Data 1 (low)
Number of bits (byte low)	Data 2 (high)
CRC-	Data 2 (low)
CRC+	Etc...
	CRC-
	CRC+

Exemplo: leitura dos valores de velocidade (P002) e corrente do motor (P003) do drive no endereço 1:

Request (Master)		Response (Slave)	
Field	Value	Field	Value
Slave address	01h	Slave address	01h
Function	02h	Function	03h
Initial bit (high)	00h	Byte Count	04h
Initial bit (low)	02h	P002 (high)	03h
Number of bits (high)	00h	P002 (low)	84h
Number of bits (low)	06h	P003 (high)	00h
CRC-	59h	P003 (low)	35h
CRC+	C8h	CRC-	7Ah
		CRC+	49h

Important observation about FLOAT markers and macro parameters:

As a macro parameter float marker has 4 bytes, the master must request two registers to read a float, for example:

To read MF9000, the address is 9000 and the quantity must be 2, in other words, 4 bytes (2 words) will be returned, which represent the float marker in the IEEE 754 format (IEEE Standard Floating Point Format).

If an odd number of registers is requested, error 2 will be returned.

1.5.4 Function 04 - Read Input Register

Reads the contents of the board and drive analog inputs. This function has the following structure for the read and response telegrams (the values are always hexadecimal, and each field represents a byte):

Resquest (Master)	Response (Slave)
Slave address	Slave address
Function	Function
Address oh the initial bit (byte high)	Field Byte Count
Address oh the initial (byte low)	Data 1 (high)
Number of bits (byte high)	Data 1 (low)
Number of bits (byte low)	Data 2 (high)
CRC-	Data 2 (low)
CRC+	Etc...
	CRC-
	CRC+

Example: reading of analog inputs 101 and 102 of the drive in address 1:

Request (Master)		Response (Slave)	
Field	Value	Field	Value
Slave address	01h	Slave address	01h
Function	04h	Function	04h
Initial bit (high)	00h	Byte Count Field	04h
Initial bit (low)	65h	AI101 (high)	1Fh
Number of bits (high)	00h	AI101 (low)	A0h
Number of bits (low)	02h	AI102 (high)	0Dh
CRC-	67h	AI102 (low)	20h
CRC+	D4h	CRC-	F9h
		CRC+	3^{A}h

Each register is always formed by two bytes (high and low). For the example, AI101 = 1FA0h, which in decimals is equal to 8096, and AI102 = 0D20h = 3360. Since the analog inputs vary between 0 and 32767, this reading represents, respectably, 24.7% and 10.25% of the input end of scale value.

Observation: do not forget that the analog inputs of the drive (AI1 and AI2) are represented in the board as AI101 and AI102, respectively.

1.5.5 Function 05 - Write Single Coil

This function is used to write a value on a bit marker, system marker or digital output. The bit value is represented using two bytes, where FF00h represents the bit equal to 1, and 0000h represents the bit equal to 0 (zero). It has the following structure (the values are always hexadecimal, and each field represents one byte):

Resquest (Master)	Response (Slave)
Slave address	Slave address
Function	Function
Address oh the initial bit (byte high)	Bit address (byte high)
Address oh the initial (byte low)	Bit address (byte low)
Number of bits (byte high)	Bit value (byte high)
Number of bits (byte low)	Bit value (byte low)
CRC-	CRC-
CRC+	CRC+

Example: switch on digital output 2, of the board, in address 1:

Request (Master)		Response (Slave)	
Field	Value	Field	Value
Slave address	01h	Slave address	01h
Function	05h	Function	05h
Bit number (byte high)	00h	Bit number (high)	00h
Bit number (byte low)	02h	Bit number (low)	02h
Bit value (byte high)	FFh	Bit value (high)	FFh
Bit value (byte low)	00h	Bit value (low)	00h
CRC-	2Dh	CRC-	2Dh
CRC+	Fah	CRC+	Fah

For this function, the response of the slave is an identical copy of the master's request.

1.5.6 Function 06 – Write Single Register

This function is used to write a value in a word marker, board parameter or drive, analog outputs of the board or the drive. It cannot be used to write in a float marker. It has the following structure (the values are always hexadecimal, and each field represents a byte):

Resquest (Master)	Response (Slave)
Slave address	Slave address
Function	Function
Address oh the initial bit (byte high)	Parameter address (byte high)
Address oh the initial (byte low)	Parameter address (byte low)
Number of bits (byte high)	Parameter value (byte high)
Number of bits (byte low)	Parameter value (byte low)
CRC-	CRC-
CRC+	CRC+

Example: writing of the speed reference equal to 900 rpm, in a user's parameter (P800), in address 1.

Request (Master)		Response (Slave)	
Field	Value	Field	Value
Slave address	01h	Slave address	01h
Function	06h	Function	06h
Parameter (high)	03h	Parameter (high)	03h
Parameter (byte low)	20h	Parameter (byte low)	20h
Value (high)	03h	Value (high)	03h
Value (low)	84h	Value (low)	84h
CRC-	88h	CRC-	88h
CRC+	D7h	CRC+	D7h

For this function the response of the slave is an identical copy of the solicitation made by the master. The word markers or parameters are directly addressed by their numbers, in the example above P800=0320h.

1.5.7 Function 15 – Write Multiple Coils

This function allows values to be written for a group of bit markers or digital outputs of the board or drive, which must be in numeric sequence. It can also be used to write in only one bit (the values are always hexadecimal, and each field represents a byte):

Resquest (Master)	Response (Slave)
Slave address	Slave address
Function	Function
Address of the initial bit (byte high)	Address oh the initial bit (byte high)
Address of the initial (byte low)	Address oh the initial (byte low)
Number of bits (byte high)	Number of bits (byte high)
Number of bits (byte low)	Number of bits (byte low)
Field Byte Count (umber of data bytes)	CRC-
Byte 1	CRC+
Byte 2	
Byte 3	
etc...	
CRC-	
CRC+	
CRC+	

The value of each bit that is being written is placed in a position of the data bytes sent by the master. The first byte, in bits 0 to 7, receives the first eight bits starting from the initial address indicated by the master. The other bytes (if the number of written bits is greater than 8), continue in sequence.

If the number of written bits is not a multiple of 8, the rest of the bits of the last byte must be filled in with 0 (zero).
Example: switch on digital outputs 4 and 5 of the board, in address 1:

Request (Master)		Response (Slave)	
Field	Value	Field	Value
Slave Address	01h	Slave address	01h
Function	0Fh	Function	0Fh
Initial bit (byte high)	00h	Initial bit (byte high)	00h
Initial bit (byte low)	04h	Initial bit (byte low)	04h
Number of bits (byte high)	00h	Number of bits (byte high)	00h
Number of bits (byte low)	02h	Number of bits (byte low)	02h
Byte Count	01h	CRC-	95h
Bit value	03h	CRC+	CBh
CRC-	6Fh		
CRC+	56h		

As only two bits are being written, the master needed only 1 byte to transmit the data. The transmitted values are in the two least significant bits of the byte that contains the bit value. The other bits of this byte were left with value 0 (zero).

1.5.8 Function 16 – Write Multiple Registers

This function allows values to be written for a group of word markers, float markers, board or drive parameters, analog outputs of the board or drive, which must be in numeric sequence. It can also be used to write in only one parameter (the values are always hexadecimal, and each field represents a byte):

Resquest (Master)	Response (Slave)
Slave address	Slave address
Function	Function
Address of the initial bit (byte high)	Address of the initial bit (byte high)
Address of the initial (byte low)	Address of the initial (byte low)
Number of bits (byte high)	Number of bits (byte high)
Number of bits (byte low)	Number of bits (byte low)
Field Byte Count (umber of data bytes)	CRC-
Data 1	CRC+
Data 2	
Data 3	
etc...	
CRC-	
CRC+	
CRC+	

Example: writing of the acceleration time (P100) = 1.0 s and deceleration time (P101) = 2.0 s, of the drive, in address 20:

Request (Master)		Response (Slave)	
Field	Value	Field	Value
Slave Address	14h	Slave address	14h
Function	10h	Function	10h
Initial register (high)	00h	Initial register (high)	00h
Initial register (low)	64h	Initial register (low)	64h
Number of register (high)	00h	Number of register (high)	00h
Number of register (low)	02h	Number of register (low)	02h
Byte Count	04h	CRC-	04h
P100 (high)	00h	CRC+	D2h
P100 (low)	0Ah		
P101 (high)	00h		
P101 (low)	14h		
CRC-	91h		
CRC+	75h		

Since both parameters have resolutions of one decimal space, for writing of 1.0 and 2.0 seconds, the following values must be transmitted, respectfully: 10 (000Ah) and 20 (0014h).

Important observation about FLOAT markers and macro parameters:

Since a float marker and macro parameters have 4 bytes, the master must send two registers to write in a float, for example:

To write in the MF9000, the address is 9000 and the quantity must be 2, in other words, 4 bytes (2 words) will be sent, that represent the float marker in the IEEE 754 format (IEEE Standard Floating Point Format).

If an odd number of registers is sent, fault 2 will be returned.

1.5.9 Function 43 – Read Device Identification

Auxiliary function, that permits manufacturer reading, model and version of the product firmware. It has the following structure:

Resquest (Master)	Response (Slave)
Slave address	Slave address
Function	Function
MEI Type	MEI Type
Read code	Conformity level
Object number	More Follows
CRC-	Next object
CRC+	Number of objects
	Object Code*
	Object Size*
	Object Value*
	CRC-
	CRC+

* Fields are repeated according to the number of objects. This function allows reading three information categories: Basic, Regular and Extended, and each category is formed by a group of objects. Each object is formed by a sequence of ASCII characters. For the board, only basic information is available, formed by three objects:

Object 00 – VendorName: Always 'WEG'.'
Object 01 – ProductCode: Formed by the product code (PLC1.01, PLC2.00 or POS2.00) where .XX indicates the hardware version.
Object 02 – MajorMinorRevision: indicates the firmware version of the board, in 'VX.XX' format. The read code indicates which information categories are being read, and if the objects are being accessed in sequence or individually. In this case, the board supports codes 01 (basic information in sequence) and 04 (individual access to the objects).

Example: reading of the basic information in sequence, starting at object 00, of the board in address 1:

1.6 Communication Fault

Faults can occur in the transmission of network telegrams, or in the content of the received telegrams. According to the type of fault, the board may or may not send a response to the master.

When the master sends a message to the board, configured in a specific network address, it will not respond to the master if the following occurs:

- ☑ CRC Fault.
- ☑ Time out among the transmitted bytes (3.5 times the transmission time of an 11 bit word).

In case of a successful reception, during telegram treatment, the board can detect problems and send a fault message, indicating the type of problem found:

- ☑ Invalid function (fault code = 1): the solicited function is not implemented for the board.
- ☑ Invalid data address (fault code = 2): data address (parameter or digital E/S) does not exist.
- ☑ Invalid data value (fault code = 3): occurs in the following situations:
  - ☑ The value is out of the permitted range.
  - ☑ Data writing that cannot be altered (read only register, register that does not allow alteration with enabled drive or bits of the logical state).
  - ☑ Writing in a logical command function that is notenabled via serial.

1.6.1 Fault Message

When a fault in the content of a message occurs (not in the transmission of data), the slave must return a message that indicates the type of fault occurred. The faults that can occur in the message treatment to the board are the invalid function faults (code 01), invalid data address (code 02) and invalid data value (code 03).

The fault messages sent by the slave have the following structure:

Response (Slave)
Slave Address
Function Code (with the most significant bit set to 1)
Error Code
CRC-
CRC+