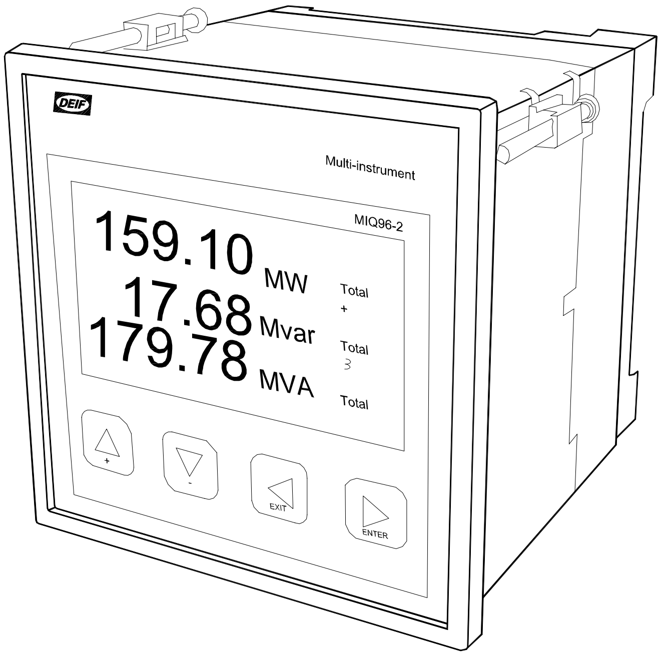
User's manual

Serial Interface, MIQ96-2

4189320012A



- *RS485: Modbus RTU*

DEIF A/S

DEIF A/S
Frisenborgvej 33, DK-7800 Skive
Denmark

Tel.:	*+45 9614 9614*
Fax:	*+45 9614 9615*
E-mail:	*deif@deif.com*

List of contents

1 INTRODUCTION

The MIQ96-2 implements a subset of the AEG Modicon Modbus RTU serial communications standard [reference 1, Modicon Modbus Protocol Reference Guide PI - MBUS - 300 Rev. E]. Modbus is a single master multiple slave protocol suitable for a multi-drop configuration as provided by the RS485 connection. Up to 32 devices can be connected in this way.

2 TRANSACTIONS

Communication operates on a master-slave basis where only one device (the master) can initiate transactions called 'Requests'. The other devices (slaves) respond by supplying the requested data to the master. This is called the 'Request - response cycle'.

Master to slave request:

Device address	Function code	nx8 bit data bytes	Error check

Slave to master response:

Device address	Function code	nx8 bit data bytes	Error check

2.1 Request

This master to slave transaction takes the form:

Device address: Master addressing a slave (address 0 is used for the broadcast address, which all slave devices recognise).

Function code: E.g. 03 asks the slave to read its registers and respond with their contents.

Data bytes: Tells the slave which register to start at and how many registers to read.

2.2 Response

This slave to master transaction takes the form:

Device address: To let the master know which slave is responding.

Function code: This is an echo of the request function code.

Data bytes: Contains the data collected from the slave.

2.3 Request - response cycle example

I_1	160.00 A = 16000*10^{-2} A
Data type 32 bit float	FE 00 3E $80_{(16)}$
Data held in Modbus addresses	$30036_{(10)}$ & $30037_{(10)}$ $30036_{(10)} - 30000_{(10)} = 36_{(10)} \equiv 00\ 24_{(16)}$

2.3.1 Request frame

		Starting register	Register count	CRC
Slave address	Function code	HI LO	HI LO	LO HI
21	03	00 24	00 02	

2.3.2 Response frame

			Register data	CRC
Slave address	Function code	Byte count	HI LO HI LO	LO HI
21	03	04	FE 00 3E 80	

3 FRAMING

There are two types of message framing for the serial communications, ASCII or RTU. The MIQ96-2 supports RTU framing.

3.1 RTU framing

In RTU mode, messages start and end with a silent interval of at least 3.5 character times (t1-t2-t3-t4 as shown below). The advantage of this mode of framing is that it enables a greater character density and a better data throughput. However, each message must be transmitted in a continuous stream. If a silent interval of more than 1.5 character times occurs before completion of the frame, the device flushes the incomplete message and assumes that the next byte will be the address field of a new message.

Start	Address	Function	Data	CRC check	End
t1-t2-t3-t4	8 bits	8 bits	nx8 bits	16 bits	t1-t2-t3-t4

The cyclic redundancy check (CRC) field is two bytes, containing a 16 bit binary value. The CRC value is calculated by the transmitting device, which appends the CRC to the message. The receiving device recalculates a CRC during receipt of the message, and compares the calculated value to the actual value it received in the CRC field. If the two values are not equal this will result in an error. The CRC-16 calculation is an industry standard method used for error detection.

One frame is transmitted as 1 start bit, 8 data bits and 2 stop bits. If parity is selected, the frame is transmitted as 1 start bit, 8 data bits and 1 stop bit.

Where n > 1 data is transmitted most significant byte first. The CRC check is transmitted least significant byte first.

4 SUPPORTED FUNCTIONS AND USAGE

Code	Code	Function	References
DEC	HEX		
3	03	To read from holding registers	(4XXXX memory references)
4	04	To read from input registers	(3XXXX memory references)
6	06	To write to a single holding register	(4XXXX memory references)
16	10	To write to one or more holding registers	(4XXXX memory references)
17	11	Report slave ID	6 characters
77	4D	Read measurement string	1 byte value code (request)
82	52	Re-read output buffer	Use after broadcast request

4.1 Code 03 read from holding registers

Reads the binary content of holding registers (4X references) in the slave. Broadcast is also supported.

4.1.1 Request frame

The query message specifies the starting register and quantity of registers (1 to 16) to be read.

Here is an example of a request to read registers 40009 ... 40010 from slave device 33:

		Starting register	Register count	CRC
Slave address	Function code	HI LO	HI LO	LO HI
21	03	00 09	00 02	

4.1.2 Response frame

The register data in the response message are packed 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.

Data are scanned in the slave at the rate of 16 registers per scan. The response is returned when the data are completely assembled.

Here is an example of a response to the query:

			Register data	CRC
Slave address	Function code	Byte count	HI LO HI LO	LO HI
21	03	04	75 03 42 15	

The contents of register 40009 are shown as the two byte values of 75 03 hex. The contents of registers 40009 ... 40010 are 75 03 and 42 15 hex.

4.2 Code 04 read from input registers

Reads the binary content of input registers (3X references) in the slave. Broadcast is also supported.

4.2.1 Request frame

The query message specifies the starting register and quantity (1 to 16) of registers to be read.

Here is an example of a request to read registers 30036 ... 30037 from slave device 33:

		Starting register	Register count	CRC
Slave address	Function code	HI LO	HI LO	LO HI
21	04	00 24	00 02	

4.2.2 Response frame

The register data in the response message are packed 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.

Data are scanned in the slave at the rate of 16 registers per scan. The response is returned when the data are completely assembled.

Here is an example of a response to the query:

			Register data	CRC
Slave address	Function code	Byte count	HI LO HI LO	LO HI
21	04	04	FE 00 3E 80	

The contents of register 30036 are shown as the two-byte values of FE 00 hex. The contents of registers 30036 ... 30037 are FE 00 and 3E 80 hex.

4.3 Code 06 write to a single holding register

Pre-sets a value into a single holding register (4X references). When broadcast, the function pre-sets the same register reference in all attached slaves.

4.3.1 Request frame

The query message specifies the register reference to be pre-set. Registers are addressed starting at 1; register 1 is addressed as 1.

Here is an example of a request to pre-set register 40010 to 42 15 hex in slave device 33:

		Register address	Register data	CRC
Slave address	Function code	HI LO	HI LO	LO HI
21	06	00 0A	42 15	

4.3.2 Response frame

The normal response is an echo of the query, returned after the register contents have been pre-set.

Here is an example of a response to the query:

		Register address	Register data	CRC
Slave address	Function code	HI LO	HI LO	LO HI
21	06	00 0A	42 15	

4.4 Code 16 (10 HEX) write to one or more registers

Pre-sets values into a sequence of holding registers (4X references). When broadcast the function pre-sets the same register references in all attached slaves.

4.4.1 Request frame

The query message specifies the register references to be pre-set. Registers are addressed starting at 1; register 1 is addressed as 1.

Here is an example of a request to pre-set two registers starting at 40001 to 41 42 and 43 44 hex (Enter Password ABCD), in slave device 33:

Slave	Function	Starting register	Register count	Byte	Register data	CRC
Address	Code	HI LO	HI LO	COUNT	HI LO HI LO	LO HI
21	16	00 01	00 02	04	41 42 43 44	

4.4.2 Response frame

The normal response returns the slave address, function code, starting address and quantity of registers pre-set.

Here is an example of a response to the query shown above:

		Starting register	Register count	CRC
Slave address	Function code	HI LO	HI LO	LO HI
21	16	00 01	00 02	

If the password is not correct (L1, L2 or BP), the response to the query is:

		Starting register	Register count	CRC
Slave address	Function code	HI LO	HI LO	LO HI
21	16	00 01	00 00	

4.5 Code 17 (11HEX) report slave ID

Returns a description of the type of controller present at the slave address.

4.5.1 Request frame

Here is an example of a request to report the ID of slave device 33:

		CRC
Slave address	Function code	LO HI
21	11	

4.5.2 Response frame

The format of a normal response is shown below:

			Register data	CRC
Slave address	Function code	Byte count	HI LO HI LO HI LO	LO HI
21	11	06	4D 49 37 31 33 30	

4.6 Code 77 (4D HEX) read measurement string

Reads the measurement value as an ASCII string. Broadcast is also supported.

4.6.1 Request frame

The query message specifies the value code of the measurement to be read.

Here is an example of a response to read total real power from slave device 33:

			CRC
Slave address	Function code	Value code	LO HI
21	4D	04	

4.6.2 Response frame

The ASCII string in the response message is packed as data bytes. The quantity of data bytes depends on the value code.

Here is an example of the query:

			String data	CRC
Slave address	Function code	Byte count	1. 2. 3. 4. 5. 6. 7. 8.	LO HI
21	4D	08	2B 32 31 2E 31 33 35 6B	

4.6.3 Value codes

The value codes are described in the following table:

Value code DEC	Value code HEX	Measurement value	Byte count	Example string data
00	00	Energy counter 1	15	"0000004.46kWh "
01	01	Energy counter 2	15	"0000001.24kvarh"
02	02	Energy counter 3	15	"0000005.71kWh "
03	03	Energy counter 4	15	"0000002.86kvarh"
04	04	Total active power	8	"+21.135k"
05	05	Phase active power L_1	8	"+7046.3 "
06	06	Phase active power L_2	8	"+7037.3 "
07	07	Phase active power L_3	8	"+7051.1 "
08	08	Total reactive power	12	"1208.7 var L"
09	09	Phase reactive power L_1	12	"0400.2 var L"
10	0A	Phase reactive power L_2	12	"0406.4 var L"
11	0B	Phase reactive power L_3	12	"0400.9 var L"
12	0C	Total I	7	"93.671 "
13	0D	I_1	7	"31.227 "
14	0E	I_2	7	"31.222 "
15	0F	I_3	7	"31.222 "
16	10	Average U	7	"226.06 "
17	11	U_1	7	"226.08 "
18	12	U_2	7	"225.83 "
19	13	U_3	7	"226.27 "
20	14	Total apparent power	7	"21.170k"
21	15	Phase apparent power L_1	7	"7057.3 "
22	16	Phase apparent power L_2	7	"7049.0 "
23	17	Phase apparent power L_3	7	"7062.8 "
24	18	Total power factor	8	"+0.998 L"
25	19	Phase power factor L_1	8	"+0.998 L"
26	1A	Phase power factor L_2	8	"+0.998 L"
27	1B	Phase power factor L_3	8	"+0.998 L"
28	1C	Frequency	7	"46.008 "
29	1D	Frequency	7	"46.008 "
30	1E	Frequency	7	"46.008 "
31	1F	Frequency	7	"46.008 "
32	20	Total power angle	7	"+003.26"
33	21	Phase power angle L_1	7	"+003.25"
34	22	Phase power angle L_2	7	"+003.30"
35	23	Phase power angle L_3	7	"+003.25"
36	24	I_N	6	"93.67 "
37	25	Angle 12	7	"+000.00"

38	26	Angle 23	7	"+000.01"
39	27	Angle 31	7	"-000.01"
40	28	Average U_{XY}	6	"000.3 "
41	29	U_{12}	6	"000.2 "
42	2A	U_{23}	6	"000.2 "
43	2B	U_{31}	6	"000.2 "
44	2C	Dynamic demand value 1	13	"Pt=+9.818kW "
45	2D	Dynamic demand value 2	12	"Qt=6.504kvar"
46	2E	Dynamic demand value 3	12	"St=12.89kVA "
47	2F	Dynamic demand value 4	12	"It=56.91 A "
48	30	Max. demand since reset 1	13	"Pt=+11.26kW "
49	31	Max. demand since reset 2	12	"Qt=14.64kvar"
50	32	Max. demand since reset 3	12	"St=18.46kVA "
51	33	Max. demand since reset 4	12	"It=81.01 A "
52	34	Time stamp MD 1	12	"03.SEP 14:11"
53	35	Time stamp MD 2	12	"03.SEP 14:10"
54	36	Time stamp MD 3	12	"03.SEP 14:10"
55	37	Time stamp MD 4	12	"03.SEP 14:12"

4.7 Code 82 (52 HEX) re-read output buffer

This function should be used after the broadcast request. The addressed slave transmits the response frame of the previous request.

4.7.1 Request frame

Here is an example of a request to re-read the output buffer of slave device 33:

		CRC
Slave address	Function code	LO HI
21	52	

4.7.2 Response frame

The response to the query depends on the previous function code.

5 ERROR RESPONSES

When a slave detects an error other than a CRC error, a response will be sent to the master. The most significant bit of the function code byte will be set to 1 (i.e. the function code sent from the slave will be equal to the function code sent from the master plus 128). The following byte will be an exception code indicating the type of error that occurred. The slave will ignore transmissions received from the master with CRC errors.

An example of an illegal request and the corresponding exception response is shown below. The request in this example is to read registers 0201H to 0209H. If these addresses are not supported in the slave then the following occurs:

Request message:

		Starting register	Register count	CRC
Address	Function code	HI LO	HI LO	LO HI
21	04	02 01	00 08	6D B4

Exception response message:

Address	Function code	Exception code	CRC
21	84	02	C1 91

5.1 Exception codes

Code	Name	Meaning
01	ILLEGAL FUNCTION	The function code transmitted is not one of the functions supported by the slave.
02	ILLEGAL DATA ADDRESSES	The data address received in the request is not an allowable value for the slave. Write to password protected registers.
03	ILLEGAL DATA VALUE	The value referenced in the data field transmitted by the master is not within range for the selected data address. The register count is greater than 16 (functions 03 and 04).
06	SLAVE DEVICE BUSY	The slave is engaged in processing a long duration program command. The master should re-transmit the message later when the slave is free.

6 MODBUS REGISTER MAP

The Modbus register map consists of the following columns:

Code, Address, Contents, Data type, Indicator, Values, Register type, Conditional, Min., Max., Step and Password.

Code:
Function codes as described in section 4.0.

Address:
16-bit register address starting from zero. Most Modbus master devices add 40000 decimal to the actual address of the register.

Contents:
Description of parameters assigned to registers.

Data type:

UNSIGNED INTEGER	Range 0 ... 65535 One 16-bit register
SIGNED INTEGER	Range -32768 ... 32767 One 16-bit register
ASCII TEXT	Range 32 ... 159 16-bit registers (two ASCII codes per register)
BINARY FLAGS	Each bit of a 16-bit register can be used as a binary flag

Indicator:
Each bit of a 16-bit register can be either assigned as flags or filled with binary data.

Values:
Definitions of settings and data values.

Register type:
Declares whether a register is to be read/write register (setting) or a read register (data).

Conditional:
Lists any dependencies that exist between settings.

Min., Max., Step:
The minimum and maximum numerical range and the incremental step size.

Password:
There is a numerical password that allows save/abort settings and a factory accessible password constructed from the serial number that allows entry/exit to and from the calibration and configuration settings.

Code	Address		Contents	Data	Ind.	Reg. type
			SYSTEM DATA			
04	30001	30003	Model number	T12		Data
04	30004		Serial number	T1		Data
04	30005		Software ref. 1	T1		Data
04	30006		Energy counter 1 exponent	T2		Data
04	30007		Energy counter 2 exponent	T2		Data
04	30008		Counter 3 exponent	T2		Data
04	30009		Counter 4 exponent	T2		Data
			MEASUREMENTS			
04	30010	30011	Energy counter 1	T3		Data
04	30012	30013	Energy counter 2	T3		Data
04	30014	30015	Energy counter 3	T3		Data
04	30016	30017	Energy counter 4	T3		Data
04	30018	30019	Total active power	T6		Data
04	30020	30021	Phase active power L_1	T6		Data
04	30022	30023	Phase active power L_2	T6		Data
04	30024	30025	Phase active power L_3	T6		Data
04	30026	30027	Total reactive power	T6		Data
04	30028	30029	Phase reactive power L_1	T6		Data
04	30030	30031	Phase reactive power L_2	T6		Data
04	30032	30033	Phase reactive power L_3	T6		Data
04	30034	30035	Total I	T5		Data
04	30036	30037	I_1	T5		Data
04	30038	30039	I_2	T5		Data
04	30040	30041	I_3	T5		Data
04	30042	30043	Average U	T5		Data
04	30044	30045	U_1	T5		Data
04	30046	30047	U_2	T5		Data
04	30048	30049	U_3	T5		Data
04	30050	30051	Total apparent power	T5		Data
04	30052	30053	Phase apparent power L_1	T5		Data
04	30054	30055	Phase apparent power L_2	T5		Data
04	30056	30057	Phase apparent power L_3	T5		Data
04	30058	30059	Total power factor	T7		Data
04	30060	30061	Phase power factor L_1	T7		Data
04	30062	30063	Phase power factor L_2	T7		Data
04	30064	30065	Phase power factor L_3	T7		Data
04	30066		Frequency	T1		Data
04	30067		Frequency	T1		Data

Values/Dependencies	Min.	Max.	Step	Pass.
MIQ96-2				0
				0
Software version	208			0
(=6 if incorrect divider @40026)(1)	-6	9	1	0
(=6 if incorrect divider @40027)(1)	-6	9	1	0
(=6 if incorrect divider @40028)(1)	-6	9	1	0
(=6 if incorrect divider @40029)(1)	-6	9	1	0
Total export active energy (default)	-99999999	899999999	1	0
Total import reactive energy (default)	-99999999	899999999	1	0
Pulse output 1	-99999999	899999999	1	0
Pulse output 2	-99999999	899999999	1	0
W				0
W				0
W				0
W				0
var L(if > 0); var C (if < 0)				0
var L(if > 0); var C (if < 0)				0
var L(if > 0); var C (if < 0)				0
var L(if > 0); var C (if < 0)				0
A				0
A				0
A				0
A				0
V				0
V				0
V				0
V				0
VA				0
VA				0
VA				0
VA				0
				0
				0
				0
				0
mHz	00.000	65.535	0.001Hz	0
mHz	00.000	65.535	0.001Hz	0

Code	Address		Contents	Data	Ind.	Reg. type
04	30068		Frequency	T1		Data
04	30069		Frequency	T1		Data
04	30070		Total power angle	T2		Data
04	30071		Phase power angle L_1	T2		Data
04	30072		Phase power angle L_2	T2		Data
04	30073		Phase power angle L_3	T2		Data
04	30074	30075	I_N	T5		Data
04	30076		$Angle_{12}$	T2		Data
04	30077		$Angle_{23}$	T2		Data
04	30078		$Angle_{31}$	T2		Data
04	30079	30080	Average U_{XY}	T5		Data
04	30081	30082	U_{12}	T5		Data
04	30083	30084	U_{23}	T5		Data
04	30085	30086	U_{31}	T5		Data
04	30087	30088	Dynamic demand value 1	T6		Data
04	30089	30090	Dynamic demand value 2	T6		Data
04	30091	30092	Dynamic demand value 3	T6		Data
04	30093	30094	Dynamic demand value 4	T6		Data
04	30095	30096	Max. demand since reset 1	T6		Data
04	30097	30098	Max. demand since reset 1	T6		Data
04	30099	30100	Max. demand since reset 1	T6		Data
04	30101	30102	Max. demand since reset 1	T6		Data
04	30103	30104	Time stamp MD 1	T8		Data
04	30105	30106	Time stamp MD 2	T8		Data
04	30107	30108	Time stamp MD 3	T8		Data
04	30109	30110	Time stamp MD 4	T8		Data
04	30111		Time into period (minutes)	T1		Data
04	30112		U1 THD%	T1		Data
04	30113		U2 THD%	T1		Data
04	30114		U3 THD%	T1		Data
04	30115		U12 THD%	T1		Data
04	30116		U23 THD%	T1		Data
04	30117		U31 THD%	T1		Data
04	30118		I1 THD%	T1		Data
04	30119		I2 THD%	T1		Data
04	30120		I3 THD%	T1		Data

Values/Dependencies	Min.	Max.	Step	Pass.
mHz	00.000	65.535	0.001Hz	0
mHz	00.000	65.535	0.001Hz	0
0.01 deg	-180.00	+179.99	0.01 deg	0
0.01 deg	-180.00	+179.99	0.01 deg	0
0.01 deg	-180.00	+179.99	0.01 deg	0
0.01 deg	-180.00	+179.99	0.01 deg	0
A				0
0.01 deg	-180.00	+179.99	0.01 deg	0
0.01 deg	-180.00	+179.99	0.01 deg	0
0.01 deg	-180.00	+179.99	0.01 deg	0
V				0
V				0
V				0
V				0
Total active power				0
Total absolute reactive power				0
Total apparent power				0
Total I				0
Total active power				0
Total absolute reactive power				0
Total apparent power				0
Total I				0
				0
				0
				0
				0
				0
0.01%	0.00	400.00	0.01%	0
0.01%	0.00	400.00	0.01%	0
0.01%	0.00	400.00	0.01%	0
0.01%	0.00	400.00	0.01%	0
0.01%	0.00	400.00	0.01%	0
0.01%	0.00	400.00	0.01%	0
0.01%	0.00	400.00	0.01%	0
0.01%	0.00	400.00	0.01%	0
0.01%	0.00	400.00	0.01%	0

Code	Address		Contents	Data	Ind.	Reg. type
16	40000	40001	Enter password L1 & L2 & BP	T11	A…Z	Write only
16	40002	40004	Enter configuration password	T12	A…Z	Write only
16	40005	40006	Set password level 1	T11	A…Z	Write only
16	40007	40008	Set password level 2	T11	A…Z	Write only
3, 6, 16	40009	40010	Time	T9		Setting
3, 6, 16	40011	40012	Date	T10		Setting
6	40013		Reset counter & MD	T1	Bit-0	Write only
					Bit-1	
					Bit-2	
					Bit-3	
					Bit-8	
					Bit-9	
					Bit-10	
3	40014		Calibration voltage in V	T1		Read only
3	40015		Calibration current in A/10	T1		Read only
3, 6	40016		Voltage tr. primaries in V/10(4)	T1		Setting
			bit # 0…13		1…15999	
			bit # 14…15		0…3	
3, 6	40017		Voltage tr. secondaries in V(5)	T1		Setting
3, 6	40018		Current tr. ratio(6)	T1		Setting
3, 6	40019		Connection mode (7)	T1	1	Setting
					9	
					25	
					5	
					7	
3, 6	40020		Communication settings	T1	0	Setting
					1	
					2	
					3	
					4	
					5 (10)	
					6 (10)	
					7 (10)	
					Bit-3	
					Bit-4	
					Bit-5	
					Bit-6	
					Bit-7	
3, 6	40021		Communication address	T1	1…247	Setting

Values/Dependencies	Min.	Max.	Step	Pass.
				0
				0
				1
				2
				1
				1
Reset counter 1				1
Reset counter 1				
Reset pulse output counter 1				
Reset pulse output counter 2				
Synchronise MD				
Reset last period MD				
Reset MD values				
			1 V	0
	10 A/10 = 1 A	50 A/10 = 5 A	0.1 V	0
2300 for 230 V			0.1 V	2
Unsigned integer value	1	15999	1	
Unsigned exponent	0	3	1	
	10	775	1 V, 5 V	2
	1	4000	1	2
Single phase				2
3 phase 3 wire balanced				
3 phase 4 wire balanced				
3 phase 3 wire unbalanced				
3 phase 4 wire unbalanced				
1200 baud				2
2400 baud				
4800 baud				
9600 baud				
19200 baud				
38400 baud				
57600 baud				
115200 baud				
2 stop bits; 0 – 1 stop bit				
Odd parity; 0 – Even parity				
Parity; 0 – No parity				
7 bits; 0 – 8 bits (read only)				
> 10 ms response time				
	1	247	1	2

Code	Address		Contents	Data	Ind.	Reg. type
3, 6	40022		MD setting bits # 0…7	T1	0	Setting
					1…255	
			Bits # 8…15		0	
					1	
					2…15	
3, 6	40023		Counter mode 2, bits # 0…7(3)	T1	Bit-0	Setting
					Bit-1	
					Bit-2	
					Bit-3	
					Bit-5	
					Bit-6	
					Bit-7	
			Counter mode 1, bits # 8…15(3)		Bit-8	
					Bit-9	
					Bit-10	
					Bit-11	
					Bit-13	
					Bit-14	
					Bit-15	
3, 6	40024		Pulse output mode	T1		Setting
			Output mode 2, bits # 0…7(3)			
			Output mode 1, bits # 8…15(3)			
3, 6	40025		Counter 1 divider	T1		Setting
3, 6	40026		Counter 2 divider	T1		Setting
3, 6	40027		Counter 3 divider	T1		Setting
3, 6	40028		Counter 4 divider	T1		Setting
	40029	40079	RESERVED			
3, 6	40080		Starting current	T1		Setting
3, 6	40081		Quartz frequency correction	T2		Setting
3, 6	40082		Calibration status	T1	Bit-0	Setting
					Bit-1	
					Bit-2	
					Bit-3	
					Bit-4	
					Bit-5	
					Bit-6	
					Bit-7	
					Bit-8	

Values/Dependencies	Min.	Max.	Step	Pass.
Disable				2
Time constant (window period; interval of sub-period)				
Thermal function				
Fixed window				
Sliding window; # of periods				
Enable quadrant 1				2
Enable quadrant 2				
Enable quadrant 3				
Enable quadrant 4				
Absolute value				
Inverted value				
Reactive energy; 0 – Active energy				
Enable quadrant 1				
Enable quadrant 2				
Enable quadrant 3				
Enable quadrant 4				
Absolute value				
Inverted value				
Reactive energy; 0 – Active energy				
Same as counter mode				2
Same as counter 2 mode				
Same as counter 1 mode				
1, 10, 100, 1000, 10000(1)				2
1, 10, 100, 1000, 10000(1)				2
1, 2, 5, 10, 20, 50, …, 50000(1)				2
1, 2, 5, 10, 20, 50, …, 50000(1)				2
320 for 0.2%				3
	-128	127	1	3
I_1, range HI				3
I_2, range HI				
I_3, range HI				
I_1, range LO				
I_2, range LO				
I_3, range LO				
U_1				
U_2				
U_3				

Code	Address		Contents	Data	Ind.	Reg. type
					Bit-9	
					Bit-10	
					Bit-11	
					Bit-12	
					Bit-13	
					Bit-14	
6	40083		Calibration request	T1	Bit-0	Write only
					Bit-1	
					Bit-2	
3, 6	40101		Language	T1	0	Setting
					1	
					2	
					3	
					5	
					6	
3, 6	40102		Active access level	T1		Setting
16	40110	40111	Set energy counter 1(2)	T3		Write only
16	40112	40113	Set energy counter 2(2)	T3		Write only

Values/Dependencies	Min.	Max.	Step	Pass.
Power angle phase L_1, range HI				
Power angle phase L_2, range HI				
Power angle phase L_3, range HI				
Power angle phase L_1, range LO				
Power angle phase L_2, range LO				
Power angle phase L_3, range LO				
Calibrate voltage inputs				3
Calibrate current inputs				
Calibrate phase angles				
English				2
Français				
Deutsch				
Español				
Russian				
Danish				
Only 0 can be written	0	3	1	0
Counter 1 must be halted	-99999999	899999999	1	2
Counter 2 must be halted	-99999999	899999999	1	2

Note 1: If counter 1 or counter 2 dividers are not set to 1, 10, 100, 1000 or 10000, the counter does not show correct decade units (k, M, …).
If counter c or counter d dividers are not set to 1, 2, 5, 10, 20, …, the pulse counter value will be incorrect.

Note 2: The counter is halted when all quadrants are disabled (register address 40024).

Note 3: For description of quadrants – please see User's Manual page 19.

Note 4: All values except 0 are acceptable. The exponent (bits 14 and 15) effect to the energy counter decimal places.

Note 5: List of values for voltage tr. secondary – register 40018:
10 … 137 step 1, 140 … 775 step 5.

Note 6: List of values for current tr. ratio – register 40019:
1 … 63 step 1, 65 … 315 step 5, 320 … 630 step 10, 650 … 3150 step 50, 4000.
Any other value between 1 and 4000 is rounded to the nearest upper value in the list.

Note 7: Connection mode value:
Bit 0: Set: I1 is connected; Reset: I1 is not connected (I1, P1, Q1, S1, are 0).
Bit 1: Set: I2 is connected; Reset: I2 is not connected (I2, P2, Q2, S2, are 0).
Bit 2: Set: I3 is connected; Reset: I3 is not connected (I3, P3, Q3, S3, are 0).
Bit 3: Set: 3 phase balanced (Pt = P1 x 3); Reset unbalanced or single phase.
Bit 4: Set: 4 wire; Reset: 3 wire (only for 3 phase balanced mode).
At least one bit (0, 1, 2) must be set. If not, all of them are set to 1 (value 7).
Bit 3 can be set only when bit 0 or bit 1 or bit 2 is set.

Value 1	1b (1W)	Single phase connection.
Value 5	3u (2W3)	Three phase three-wire connection with unbalanced load.
Value 7	4u (3W4)	Three-phase four-wire connection with unbalanced load.
Value 9	3b (1W3)	Three-phase three-wire connection with balanced load.
Value 25	4b (1W4)	Three-phase four-wire connection with balanced load.

7 MODBUS DATA TYPES

Registers defined in the Modbus database will define data as one of the data types described in the following table:

Type	Value/bit mask	Description
T1		Unsigned value (16 bit) Example: 12345 stored as 12345 = 3039$_{(16)}$
T2		Signed value (16 bit) Example: 12345 stored as -12345 = CFC7$_{(16)}$
T3		Signed long value (32 bit) Example: 123456789 stored as 123456789 = 075B CD 15$_{(16)}$
T4		Text string Two characters per 16 bit register
T5	 Bits # 31..24 Bits # 23..00	Unsigned measurement (32 bit) Decade exponent (signed 8 bit) Binary unsigned value (24 bit) Example: 123456*10^{-3} stored as FD01 E240$_{(16)}$
T6	 Bits # 31..24 Bits # 23..00	Signed measurement (32 bit) Decade exponent (signed 8 bit) Binary signed value (24 bit) Example: - 123456*10^{-4} stored as FCFE 1DC0$_{(16)}$
T7	 Bits # 31..24 Bits # 23..16 Bits # 15..00	Power factor (32 bit) Sign: Import/export (00/FF) Sign: Inductive/capacitive (00/FF) Unsigned value (16 bit), 4 decimal places Example: 0.9876 CAP stored as 00FF 2694$_{(16)}$
T8	 Bits # 31..24 Bits # 23..16 Bits # 15..08 Bits # 07..00	Time stamp (32 bit) Minutes 00 - 59 (BCD) Hours 00 - 23 (BCD) Day of month 01 - 31 (BCD) Month of year 01 - 12 (BCD) Example: 15:42, 1. SEP stored as 4215 0109$_{(16)}$
T9	 Bits # 31..24 Bits # 23..16 Bits # 15..08 Bits # 07..00	Time (32 bit) 1/100s 00 - 99 (BCD) Seconds 00 - 59 (BCD) Minutes 00 - 59 (BCD) Hours 00 - 24 (BCD) Example: 15:42:03.75 stored as 7503 4215$_{(16)}$
T10	 Bits # 31..24 Bits # 23..16 Bits # 15..00	Date (32 bit) Bit# 31..24 Day of month 01 - 31 (BCD) Bit# 23..16 Month of year 01 - 12 (BCD) Bit# 15..00 Year (unsigned integer) 1998..4095 Example: 10, SEP 1998 stored as 1009 07CE$_{(16)}$
T11		Text string 4 characters Two characters per 16 bit register
T12		Text string 6 characters Two characters per 16 bit register

8 CRC CHECKING AND GENERATING

In RTU mode messages include an error-checking field which is based on a CRC method. The CRC field checks the contents of the entire message. It is applied regardless of any parity check method used for the individual characters of the message.

The CRC field is two bytes, containing a 16-bit binary value. The CRC value is calculated by the transmitting device which appends the CRC to the message. The receiving device recalculates a CRC during receipt of the message and compares the calculated value to the actual value it received in the CRC field. If the two values are not equal this will result in an error. The CRC is started by first pre-loading a 16-bit register to all 1's. Then a process begins of applying successive eight-bit bytes of the message to the current contents of the register. Only the eight bits of data in each character are used for generating the CRC. Start and stop bits and the parity bit do not apply to the CRC.

During generation of the CRC, each eight-bit character is exclusive ORed with the register contents. Then the result is shifted in the direction of the least significant bit (LSB) with a zero filled into the most significant bit (MSB) position. The LSB is extracted and examined. If the LSB was a 1, the register is then exclusive ORed with a pre-set, fixed value. If the LSB was a 0, no exclusive OR takes place.

This process is repeated until eight shifts have been performed. After the last (eighth) shift, the next eight-bit byte is exclusive ORed with the register's current value, and the process repeats for eight more shifts as described above. The final content of the register, after all the bytes of the message have been applied, is the CRC value.

8.1 Generating a CRC

Step 1: Load a 16-bit register with FFFF hex (all 1's). Call this the CRC register.

Step 2: Exclusive OR the first eight-bit byte of the message with the low order byte of the 16-bit CRC register, putting the result in the CRC register.

Step 3: Shift the CRC register one bit to the right (toward the LSB), zero-filling the MSB. Extract and examine the LSB.

Step 4: If the LSB is 0, repeat step 3 (another shift). If the LSB is 1, exclusive OR the CRC register with the polynomial value A001 hex (1010 0000 0000 0001).

Step 5: Repeat steps 3 and 4 until eight shifts have been performed. When this is done, a complete eight-bit byte will have been processed.

Step 6: Repeat steps 2...5 for the next eight-bit byte of the message. Continue doing this until all bytes have been processed.

Result: The final contents of the CRC register is the CRC value.

Step 7: When the CRC is placed into the message, its upper and lower bytes must be swapped as described below.

8.2 Placing the CRC into the message

When the 16-bit CRC (two bytes) is transmitted in the message, the low order byte will be transmitted first, followed by the high order byte. When the CRC is appended to the message, the low order byte is appended first, followed by the high order byte. In ladder logic, the CKSM function calculates a CRC from the message contents. For applications using host computers a detailed example of CRC generation is given below.

Example: An example of a C language function performing CRC generation is shown on the following pages. All of the possible CRC values are preloaded into two arrays which are simply indexed as the function increments through the message buffer. One array contains all of the 256 possible CRC values for the high byte of the 16-bit field, and the other array contains all of the values for the low byte. Indexing the CRC in this way provides faster execution than would be achieved by calculating a new CRC value with each new character from the message buffer.

Note: This function performs the swapping of the high/low CRC bytes internally. The bytes are already swapped in the CRC value which is returned from the function. Therefore, the CRC value returned from the function can be directly placed into the message for transmission.

The function takes two arguments:

```
unsigned char *puchMsg;     A pointer to the message buffer containing
                            binary data to be used for generating the
                            CRC
unsigned short usDataLen;   The quantity of bytes in the message buffer
```

The function returns the CRC as a type unsigned short.

8.3 CRC Generation function

```
unsigned short CRC16 (puchMsg, usDataLen)
unsigned char *puchMsg;        /* message to calculate CRC upon */
unsigned short usDataLen;      /* quantity of bytes in message */
{
  unsigned char uchCRCHi - 0xFF; /* high CRC byte initialized */
  unsigned char uchCRCLo = oxFF; /* low CRC byte initialized */
  unsigned uIndex;               /* will index into CRC lookup */
                                 /* table */
  while (usDataLen - -)          /* pass through message buffer*/
    {
      uIndex = uchCRCHi ^ *puchMsgg++;  /* calculate the CRC */
      uchCRCHi = uchCRCLo ^ auchCRCHi (uIndex);
      uchCRCLo = auchCRCLo (uIndex);
    }
  return (uchCRCHi <<8 I uchCRCLo);}
```

8.4 High order byte table

```
/* Table of CRC values for high-order byte */
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, 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, 0x0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40
} ;
```

8.5 Low order byte table

```
/* Table of CRC values for low-order byte */
static 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, 0xID, 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
} ;
```

9 RELATED DOCUMENTS

Ref.	Document	Title
1	PI-MBUS-300 Rev. E	AEG Modicon Modbus Protocol Reference Guide