PiiGAB
PROCESSINFORMATION

Description of M-Bus ASCII protocol

www.piigab.com

PiiGAB Processinformation i Göteborg AB • Anders Carlssons gata 7 • S-417 55 Göteborg • Sweden
Tel: +46(0)31 559977 • Fax: +46(0)31 227071 • email: info@piigab.se

Contents

1. Document Information

The M-Bus ASCII protocol is developed by PiiGAB with the purpose of helping users to an easy way to read M-Bus meters without the need to have specific M-Bus drivers in the computer.

1.1 Versions

Version	Detail
0.00.00.001	The very first version.
2.00.00.001	The first v2 version.
2.00.01.001	Changed timeout error from T to T and M.
2.00.02.001	Added error O for maximum number of read items.
2.00.03	Several small adjustments.
2.01.00	Adjustments for PiiGAB M-Bus 900S and several other.

1.2 Targeted devices

This document is only for *PiiGAB M-Bus 900S* and *PiiGAB M-Bus 900 V2*. For PiiGAB M-Bus 900V1, see section *3.1 Message Structure Obsolete*.

2. Protocol description

The core of the protocol is the OPC-item which is built up from Channel, Device and Tag. If you have experience from OPC-servers this is probably well known. The reason for using this technique is that we have implemented the same dynamic M-Bus client in the PiiGAB M-Bus 900S as we have in the PiiGAB M-Bus OPC Server. The positive consequence of this is that you can use the same configuration tool as for the M-Bus OPC Server. You simply take the configuration file and upload it to the PiiGAB M-Bus 900S instead of using it together with the OPC Server. This also means that you can test your configuration file with the monitor function using the demo version of the PiiGAB M-Bus Explorer before you upload the file to the PiiGAB M-Bus 900S. The only thing needed from the M-Bus ASCII client is the name of your OPC-items.. See example below. For the M-Bus ASCII protocol all OPC-items must have the data type set to string.

2.1 Message Structure

The communication structure is a typical *request/response* message structure. You can choose if you want to have a checksum control for each message. The checksum should normally be used if you are using RS232 or RS485. If you are using TCP/IP or UDP/IP a checksum is already a part of a lower communication layer and is therefore not needed although it is allowed.
The messages buildup is <start byte><text area><stop byte> where the start byte is one of *STX*, *ACK* or *NAK* and the stop byte is always *ETX*.
The text area consists of only readable tokens. All numbers like *TID*, *PID*, *ADR* and *CHECKSUM* are formatted in a special way. All are eight bit binary numbers formatted as two characters. Examples: Decimal 127_{10} = $7F_{16}$ => “7F”. The M-Bus ASCII is obviously optimized to be readable, but it does not make good use of bandwidth.
The user must be careful when creating the checksum. In a normal binary based protocol the checksum of two bytes 0x7F and 0x81 is (0x7F+0x81)&0xFF = 0. But in M-Bus ASCII we create the checksum by summing up the integer values of individual characters: (‘7’+’F’+’8’+’1’)&0xFF = (55+70+56+49)&0xFF = 0xE6. In this way “7f” generates a different checksum from “7F”. This does not create any problem unless the case of a telegram is changed. We recommend using uppercase letters for numbers as well as OPC items.

2.1.1 Single item read message format without checksum:

Request:

STX	**TID**	**PID**	**ADR**	**ITEM**	**ETX**

Positive answer:

ACK	**TID**	**PID**	**ADR**	**DATA**	**ETX**

Negative answer:

NAK	**TID**	**PID**	**ADR**	**ERROR**	**ETX**

2.1.2 Single item read message format with checksum:

Request:

STX	**TID**	**PID**	**ADR**	**ITEM**	**CRC**	**ETX**

Positive answer:

ACK	**TID**	**PID**	**ADR**	**DATA**	**CRC**	**ETX**

Negative answer:

NAK	**TID**	**PID**	**ADR**	**ERROR**	**ETX**

2.1.3 Single item write message format without checksum: Implementing in V2.x

Request:

STX	**TID**	**PID**	**ADR**	**ITEM|DATA**	**ETX**

Positive answer:

ACK	**TID**	**PID**	**ADR**	**ETX**

Negative answer:

NAK	**TID**	**PID**	**ADR**	**ERROR**	**ETX**

2.1.4 Single item write message format with checksum: Implementing in V2.x

Request:

STX	**TID**	**PID**	**ADR**	**ITEM|DATA**	**CRC**	**ETX**

Positive answer:

ACK	**TID**	**PID**	**ADR**	**CRC**	**ETX**

Negative answer:

NAK	**TID**	**PID**	**ADR**	**ERROR**	**ETX**

2.1.5 Multiple item read message format without checksum:

The maximum number of items in one read request is ten items.

Request:

STX	**TID**	**PID**	**ADR**	**ITEM1;ITEM2;....ITEM10**	**ETX**

Positive answer:

ACK	**TID**	**PID**	**ADR**	**DATA1;DATA2;....DATA10**	**ETX**

Negative answer:

NAK	**TID**	**PID**	**ADR**	**ERROR**	**ETX**

2.1.6 Multiple item read message format with checksum:

Request:
The maximum number of items in one read request is ten items.

STX	TID	PID	ADR	ITEM1;ITEM2;....ITEM10	CRC	ETX

Positive answer:

ACK	TID	PID	ADR	DATA1;DATA2;....DATA10	CRC	ETX

Negative answer:

NAK	TID	PID	ADR	ERROR	ETX

2.2 Message descriptions

2.2.1 Field description

Field		Dec	Hex	Symbol	Length	Description
STX	Start Character	2	02	Const	1 Byte	Start of request frame.
TID[1]	Transaction Identifier	0-255	0-FF	”00”-”FF”H	2 Char	The client should increase TID with 1 for each request Recopied by the server from the received request
PID[1]	Protocol Identifier	0-255	0-FF	”00”-”FF”H	2 Char	“00”=Without checksum (TCP, UDP) “01”=With checksum (serial) Recopied by the server from the received request
ADR[1]	Address of 900 for RS485 (point to point)	0-255	0-FF	”00”-”FF”H	2 Char	Indicates which of the 256 PiiGAB M-Bus 900S on a serial line is being addressed. Recopied by the server from the received request
ITEM	OPC Item	-	-	8-bit ASCII	N Char	Format Channel.Device.Tag Should be printable characters excluding ’|’ and ’;’
CRC[1]	Checksum	-	-	“00” - “FF”H	2 Char	Checksum byte obtained executing the sum, bitwise-and 255 (0xFF), of all transmitted ASCII from TID to ITEM or TID to DATA included. Checksum is never used on NAK responses.
ETX	Termination Character	3	03	Const	1 Byte	End character of the request and response frames.
ACK	Positive answer	6	06	Const	1 Byte	Response with value.
DATA	Values when reading and writing	-	-	8-bit ascii	N Char	N characters of data according to item or write type.
NAK	Negative answer	21	15	Const	1 Byte	Response with error code.
ERROR	Error code	-	-	-	1 Char	Error code details see section 2.3.3.

[1]TID, PID, ADR and CRC are obtained by writing the single byte hex value as a two character string.

2.2.2 Control characters

Character	Name	Dec	Hex	Description
|	Vertical bar	124	7C	Delimiter between OPC Item and writevalue
;	Semicolon	59	3B	Delimiter between OPC Items when multirequest

Total length of a request message is 1500 characters including STX and ETX.

2.2.3 Error description

If there are problems to answer the requested OPC-item the PiiGAB M-Bus 900S will send an error character instead of the normal response message.

Code	Dec	Hex	Description
C	67	43	Invalid checksum.
D	68	44	Wrong data type. The data type must be VT_BSTR.
I	73	49	The request contains an OPC-item which doesn't exist.
M	77	4D	Timeout MBusHub's master port. No response from the meter.
O	79	4F	Too many OPC-items in the request.
T	84	54	Timeout slave port. No response from the Master port.
V	86	56	Unable to write outside value range. Ex: Range 10-20 cannot handle value 22.

2.3 Examples

2.3.1 Read request

If you want to ask for the value of a namespace Channel.Device.Tag which might be total energy in a meter connected over a network. The only thing you need to know is the name of the OPC-item. Let's say the name is "A.B.C" then the total string you must send to the PiiGAB M-Bubs 900S if you want to use checksum calculation is:

<STX><TID1><TID2><PID1><PID2><ADR1><ADR2>A.B.C<CRC1><CRC2><ETX>

2.3.1.1 Read request without checksum

Protocol = 0

	TID		PID		ADR		OPC Item	
STX	'0'	'0'	'0'	'0'	'0'	'0'	A.B.C	ETX
02	30	30	30	30	30	30	41 2E 42 2E 43	03

2.3.1.2 Read request with checksum

Protocol = 1

	TID		PID		ADR		OPC Item	CRC		
STX	'0'	'0'	'0'	'1'	'0'	'0'	A.B.C	'4'	'3'	ETX
02	30	30	30	31	30	30	41 2E 42 2E 43	34	33	03

2.3.2 Checksum calculation

Checksum byte is obtained executing the sum of all transmitted ASCII from TID to OPC Item or Data included.

To calculate the checksum manually you can use the calculator in Windows. Choose Hex and Byte on the calculator.

Try to use big letters A B C D E F in the ASCII frame. In hexadecimal f = F, during the crc calculation f and F represents different ASCII numbers.

The checksum in the example above is:

- (30+30+30+31+30+30+41+2E+42+2E+43)mod 256 = 43
  or
- (30+30+30+31+30+30+41+2E+42+2E+43)&FF = 43

then change the binary number 43 to the two characters '4' and '3'.

3. Appendix

3.1 Message Structure Obsolete

This was the first telegram structure and it's no longer used.
Note: PiiGAB M-Bus 900 V1 use this structure.

Single item read message format:

Request:

STX	**TID**	**ADR**	**ITEM**	**CRC**	**ETX**

Positive answer:

ACK	**TID**	**ADR**	**DATA**	**CRC**	**ETX**

Negative answer:

NAK	**TID**	**ADR**	**ERROR**	**CRC**	**ETX**