1 MeCom Protocol Specification

1.1 Introduction

This document describes the Meerstetter Engineering GmbH (meCom) communication standard. The objective of this standard is to ensure link-layer and application-layer compatibility between Devices and Host Systems and Service stations. This document is intended to assist designers of Meerstetter Engineering GmbH equipment by providing a high-level common reference document.

The Protocol uses a client-server architecture. This type of system has a number of client nodes (the user control devices) that issue explicit commands to the server node and process responses. The Server node will not typically transmit data without a request from the client node, and does not communicate with other servers.

1.2 Frame Structure

Link layer transmissions are sent in small blocks of data, called frames. Each frame is made up of several smaller groups, called fields. The table below illustrates the basic construction of frames.

Field	Parameter	Field Type	Length	Description
1	Control (Source)	ASCII char	8 Bits	See Control Field
2	Address	UINT8	16 Bits	See Device Address
3	Sequence Nr.	UINT16	32 Bits	See Sequence Number
4	Payload	N * 8 Bits	N * 8 Bits	See Application Protocol Structure
5	Frame CRC	UINT16	32 Bits	See Frame CRC Algorithm
6	End-of-Frame	ASCII char	8 Bits	ASCII <CR>

1.2.1 Control Field

The control field identifies the type of device sending this frame.
The identifiers have been selected such that the receiving device, in order to detect the start of a frame, can synchronize to a character with a value of 0x2x.

Device Type	ASCII	HEX
Device	!	0x21
Interface 1	#	0x23
Interface 2	$	0x24
Interface 3	%	0x25
Interface 4	&	0x26

1.2.2 Address

This address field represents always the device address. The device response does also contain the own device address.

1.2.3 Sequence Number

The sequence number is always replayed by the device to help the client system to ensure that this is the right answer on a specific question. It is recommended to initialize the sequence number on the client machine to a random value and increase it on every message.

1.2.4 Frame CRC

The Frame Checksum is a CRC-CCITT (CRC-16) calculated by both the sender and the receiver of a frame. It ensures that the frame was not corrupted by the transmission medium. The Frame Checksum is represented by four HEX digits. See at the C Code Example in the Appendix.

1.3 Application Protocol Structure

The Application Protocol is a Device Layer 7 protocol. It is used to control the Device from a Master Software. In this context, the Master Software acts as a client and the Device acts as a server.

1.3.1 Set Commands

These operations are used to set a specific parameter or trigger an action on the Device. The server responds in either of the following ways:

a. Send acknowledge message

b. Send error message

Set Commands have always two capital letters after the sequence number followed by the specified parameters for the current command. Numbers are transmitted as Hex characters and have always the same length. The frame is terminated by a CR. If the Device accepts the command, it will answer with an Acknowledge. The Acknowledge is always returning the CRC of the prior set command.

Set Command Structure:

source	address	sequence nr.	**Command**	**value 1**	**value n**	CRC	<CR>
ASCII	UINT8	UINT16	**2 ASCII**	**…**	**…**	UINT8	ASCII

ACK:

source	address	sequence nr.	**CRC from Set Command**	<CR>
ASCII	UINT8	UINT16	**UINT16**	ASCII

1.3.2 Query Commands

These operations are used to query a specific parameter or status of the Device. The server responds in either of the following ways:

a) Send response message as indicated in chapter Software Interface.

b) Send error message

Query Commands have always a question mark and two capital letters after the sequence number followed by the specified parameters for the current command. Numbers are transmitted as Hex characters and have always the same length. The frame is terminated by a CR. If the Device accepts the query, it will answer with the specified values for the current query.

Query:

source	address	sequence nr.	**Command**	**value 1**	**value n**	CRC	<CR>
ASCII	UINT8	UINT16	**…**	**…**	**…**	UINT8	ASCII

Server Response:

source	address	sequence nr.	**value 1**	**value n**	CRC	<CR>
ASCII	UINT8	UINT16	**…**	**…**	UINT8	ASCII

1.3.3 Parameter Types

For all operations requiring an argument, this directly follows the command mnemonic. Since mnemonics have fixed length, no special delimiter is required.
Numerical values are represented as follows:

Parameter Type	Length on Data string	Representation	Range
UINT4	1 ASCII character	0…F	0 … 15
UINT8	2 ASCII characters	2 HEX digits	0 … 255
UINT16	4 ASCII characters	4 HEX digits	0 … 65535
UINT32	8 ASCII characters	8 HEX digits	0 … 4294967295
INT8	2 ASCII characters	2 HEX digits	-128 … 127
INT16	4 ASCII characters	4 HEX digits	-32768 … 32767
INT32	8 ASCII characters	8 HEX digits	-2147483648 … 2147483647
FLOAT32	8 ASCII characters	8 HEX digits	-1.2E+38 … 3.4E+38 (IEEE754)

Example: A UINT16 value of decimal 23456 is transmitted as 4 ASCII chars: 5BA0

The memory area of the FLOAT32 value is copied to a UINT32 memory area and transferred as it would be a UINT32.

1.3.4 Server Error Codes

Any SET or GET command may result in an error condition on the server side. In order for the client application to recover from as many error conditions as possible, the server indicates the error codes listed in below.

The Error Message has the this format:

Server Response:

source	address	sequence nr.	**Error Identifier**	**Error Code**	CRC	<CR>
ASCII	UINT8	UINT16	**ASCII char +**	**UINT8**	UINT8	ASCII

Errors 0 … 99 → Common errors
Errors 100 … 255 → Device specific errors

Error Code	Symbol	Description
1	EER_CMD_NOT_AVAILABLE	Command not available
2	EER_DEVICE_BUSY	Device is busy
3	ERR_GENERAL_COM	General communication error
4	EER_FORMAT	Format error
5	EER_PAR_NOT_AVAILABLE	Parameter is not available
6	EER_PAR_NOT_WRITABLE	Parameter is read only
7	EER_PAR_OUT_OF_RANGE	Value is out of range
8	EER_PAR_INST_NOT_AVAILABLE	Instance is not available

2 Appendix A: C code examples

2.1 Send Frame function

This Function is used to send a Frame to the Device.

- ucAddress is the Device Address
- usSequenceNr is the random sequence Number which will be returned by the Device
- ucLength is the length of the payload field
- ucData is the pointer to the payload data

```
void UART1_SendFrame(unsigned char ucAddress, unsigned short usSequenceNr, unsigned char ucLength, unsigned char *ucData)
{
    const unsigned char ucHex[16] =
    {'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
    unsigned char uc; unsigned short usCRC = 0;

    uc = 0x21;                        UART1_Send(uc); CRC16Algorithm(&usCRC, uc);
    uc = ucHex[ucAddress / 16];       UART1_Send(uc); CRC16Algorithm(&usCRC, uc);
    uc = ucHex[ucAddress % 16];       UART1_Send(uc); CRC16Algorithm(&usCRC, uc);

    uc = ucHex[usSequenceNr/4096];    UART1_Send(uc); CRC16Algorithm(&usCRC, uc);
    uc = ucHex[(usSequenceNr/256)%16]; UART1_Send(uc); CRC16Algorithm(&usCRC, uc);
    uc = ucHex[(usSequenceNr%256)/16]; UART1_Send(uc); CRC16Algorithm(&usCRC, uc);
    uc = ucHex[(usSequenceNr%256)%16]; UART1_Send(uc); CRC16Algorithm(&usCRC, uc);

    for(uc = 0; uc < ucLength; uc++)
    {
        UART1_Send(*ucData);
        CRC16Algorithm(&usCRC, *ucData);
        ucData++;
    }
    UART1_Send(ucHex[usCRC/4096]);
    UART1_Send(ucHex[(usCRC/256)%16]);
    UART1_Send(ucHex[(usCRC%256)/16]);
    UART1_Send(ucHex[(usCRC%256)%16]);
    UART1_Send(0x0D);
}
```

2.2 Frame CRC Algorithm

The used standard is: CRC-CCITT (CRC-16)

```
void CRC16Algorithm(unsigned short *CRC, unsigned char Ch)
{
    unsigned int genPoly = 0x1021; //CCITT CRC-16 Polynominal
    unsigned int uiCharShifted = ((unsigned int)Ch & 0x00FF) << 8;
    *CRC = *CRC ^ uiCharShifted;
    for (int i = 0; i < 8; i++)
        if ( *CRC & 0x8000 ) *CRC = (*CRC << 1) ^ genPoly;
        else *CRC = *CRC << 1;
    *CRC &= 0xFFFF;
}
```