BABUC ABC
PROGRAMMER'S MANUAL

For instruments version 4 & 5

SUMMARY

1 Introduction

Babuc ABC supports two types of communication protocols: a proprietary Lastem and a MODBUS protocol. This document describes the proprietary protocols only, to allow non Lastem technical personnel to develop communication protocols when using the Babuc ABC V. 4.02 acquisition systems.

2 References

[1] Operator's Manual for Babuc ABC

3 LSI-Lastem Proprietary Protocol

3.1 General Information

3.1.1 Communication Protocol

The acquisition stations behave as slaves when they receive a query from the masters. Therefore, they can never independently initiate data transfers.

Data are requested and transmitted, between the master and the slave, using the serial communication line to send binary *frames*, which group several data within a single package of characters so that they can be transmitted in sequence.

The frame also contains information that enables to route a data request to one station instead of another, to attach additional information to the request, to include actual transmittable data and to check the validity of the data received.

The master always sends requests within a single frame. The replies of the slaves can contain one or more frames, because the quantity of the transmitted data can be larger than the information contained in a single frame (*multiframe* transmissions). The master sends one request at a time. For each request sent, the slave replies at least with one frame.

The master doesn't necessarily reply every time it receives a frame. If the slave transmits data contained in a single frame, the master may not reply to the slave because the latter is not expecting the acknowledgement of the receipt. However, the slave is also able to send the previously transmitted frame every time it receives a request from the master. If the slave sends data that have to be contained in several frames (*multiframe* transmission), each frame received from the master should correspond to the transmission of a receipt acknowledgement (*Ack*). As a result, the slave also sends the next frame. This condition is repeated until the master receives a frame containing the last frame transmitted (*LastFrame*). At this point, the master can choose whether to send a reply or not.

The safety of the transmission and the accuracy of the received data are guaranteed by the CRC 16-bit check digit, which is calculated according to the content of the frame transmitted. If the CRC's received and calculated do not correspond, the master issues a special command requesting the frame to be re-transmitted. If the frame received by the slave is incorrect, no reception anomaly is signalled in order to avoid affecting the communications between the master and another slave. If the master doesn't receive a reply from the slave within a specific interval of time or the reply received is incorrect, the master may ask the slave to retransmit the frame for an endless number of times.

The three synchronism characters transmitted at the beginning of the frame ensure that the frame is received correctly, even in presence of other spurious characters originating, for example, from the enabling of radio carriers. The communication protocol is able to correctly acknowledge the frame even if only one of the three synchronism characters is received.

The slave replies to a request within a maximum interval of 3 seconds. Therefore, the master must wait at least for said interval before regarding the slave's reply as not received.

3.1.2 Physical Level

The physical transmission protocol uses bit rates ranging from 1200 and 19200 (included), 8 characters per bit, 1 stop bit and no parity. The frames transmitted can have lengths that survey from a few to 2048 bytes. The transmitted data, with a 16 or 32 bit binary format, contain the most significant bit in the smaller address and the less significant bit in the larger address. The floating point value has 32 bits and is compliant with standard IEEE-754.

The protocol correctly manages point-to-point and multipoint connections and several slave stations (up to a maximum of 32 slave stations on an RS-485 line).

Babuc ABC can appropriately pilot the RTS (*request to send*) signal in order to enable half-duplex transmissions via single frequency radio communication lines. This signal has to be coupled to the transceiver system in order to adequately pilot the carrier enabling signal during transmission.
The RTS signal is enabled before the beginning of the frame transmission and is appropriately disabled at the end of the transmission. The interval of time between the enabling of the signal and the beginning of the frame transmission can be changed from a minimum of 100 ms to a maximum of 990 ms with steps of 100 ms.

3.1.3 Storage and transmitted data format

The memory of Babuc ABC may contain one or more *surveys*; a survey is a set of processes/events that are grouped so that they can univocally decoded. Data are decoded by means of some *headers* that are stored at the time the survey is opened in the memory. These headers contain general information on the survey (structure *RelMemHeader*), on the processes and the acquisition/pre-processing channels originating from the processes (structure *ChMemHeader*). Each survey contains a *RelMemHeader* header and one or more *ChMemHeader* channel headers.

The system enables to store the data in linear or circular format. The linear format enables to store the data until the physical limit of the memory is completely taken up. Then, storage is stopped issuing a full memory error. The circular format enables the system to overwrite the oldest data with the newest. This occurs when the physical space available in the memory ends. As a result, the new processes or events are stored starting from the beginning of the survey, after the last channel header.

With the circular format, the space available for storage *circularity* surveys from the end of the last channel header of the current survey to the physical end of the memory. For this reason, if the memory contains other previously stored surveys, which take up only a part of the available memory, the last survey can store the date in circular format only in the remaining memory space, without altering the content of previous surveys. An extreme situation is represented by the unavailability of other storage operations, which occurs when the current or the previous surveys have taken up all the physical space of the memory.

All the processes or events stored in the memory belong to several types of structures. Each of these has an *FECommon* structure, which enables to recognise the data that follow the *FECommon* structure itself. The conversion function that must convert the binary data received from the acquisition system decodes the binary data, starting from the *FECommon* structure and continuing with the next process, moving to the next *FECommon* structure and so on until all the data are received.

Between the processes, it is also possible to store a *FinderMemHeader* structure, which is used internally by the system to reduce some of the data search operations. For a correct conversion, these structures should be ignored.

The section here below provides a schematic layout of the content of the memory with two stored surveys (*FEElab* identifies any type of process or event):

RelMemHeader	*Survey header*	
ChMemHeader	*Channel header*	*block headers channels*
ChMemHeader		
...		
ChMemHeader		
FEElab	*Processed*	*block processed*
FEElab		
...		
FEElab		
FinderMemHeader	*Block used for the rapid search of data*	
FEElab		
FEElab		
RelMemHeader	*Survey header*	
ChMemHeader	*Channel header*	*block headers channels*
ChMemHeader		
...		
ChMemHeader		
FEElab	*Processed*	*block processed*
FEElab		
...		
FEElab		
FEElab		

The available request codes enable to transmit all the survey headers (one or more *RelMemHeader* structures), the current survey header, which contains the index, along with the channel headers (one or more *ChMemHeader* structures) and the processed data, starting from a set date/time.

3.1.4 Transmission time

The time required to transmit data from Babuc ABC to the external master changes according to the following factors:

- Transmission speed (bit rate)
- Length of the information to transmit
- Overhead set by the frame structure
- Response time of the system

3.1.4.1 Transmission speed

The transmission speed, expressed in bps (bits per second), indicates the amount of bits that can be transmitted along the communication line within the unit of time of a second. In ideal conditions, with a continuous flow of data, the amount of bytes transmitted on the communication line can be calculated using the following expression:

$$\frac{BitRate}{BitsPerByte + NrStartBit + NrStopBit}$$

Where:

BitRate: Communication speed expressed in bps
BitsPerByte: Number of bits that constitute a byte (8)
NrStartBit: Number of start bits (for Babuc ABC: 1)
NrStopBit: Number of stop bits (for Babuc ABC: 1)

3.1.4.2 Length of the information to transmit

This information changes according to the type of information sent on the communication line. For example, if the transmission is related to instantaneous data acquired by the system at a given time, the dimension of the data transmitted corresponds to the number of acquisition channels multiplied by 4, which is equivalent to the number of bytes that form a single number in floating point format.

The examples here below list the transmitted bytes for some typical configurations of Babuc ABC

Type of transmission and configuration of the system	Byte n.
Transmission of instantaneous values from a system configured with 10 channels	40
Transmission of hourly and daily processed data referred to a specific day from a system configured with 10 channels and *EEMinMedMaxDvst word* type processes	≈4000
Transmission of hourly and daily processed data referred to a specific month from a system configured with 10 channels and *EEMinMedMaxDvst word* type processes	≈120000
Transmission of hourly and daily processed data referred to a specific day from a system configured with 5 channels and *EEMinMedMaxDvst word* type processes for 4 channels and *EEEolo3* processes for a wind direction channel	≈2600
Transmission of hourly and daily processed data referred to a specific month from a system configured with 5 channels and *EEMinMedMaxDvst word* processes for 4 channels and *EEEolo3* processes for a wind direction channel	≈78000
Transmission of the operating configuration of the system (system parameters, survey parameters, library of operating codes, calibration of amplifiers, etc.):	≈16000
Transmission of instantaneous values grouped by *EE10Ist word* processes for 8 channels (for a total of 80 values) with selectable acquisition time	216
Transmission of 80 *EEMinMedMaxDvst word* processes divided by several channels	1280

3.1.4.3 Overhead set by the frame structure

The transmission frame enables to regulate the traffic of information on the communication line. This adds some additional information to the actual data transmitted.

The influence of the length of the frame structure on the transmission varies according to the nature and therefore the dimension of the transmitted data and by the division of the frames of the set of information transmitted by the system to the master.

The length of the frame, which can be selected by the user, changes mainly according to the type of transmission media used for communication purposes and consequently according to the quality and reliability of the transmission line. It is therefore advisable to use a frame with a length of 512 bytes up to a maximum of 1024 bytes, for dedicated or low speed communication lines, and frame lengths of 32 or 64 bytes for communication lines that are particularly noisy.

For radio transmissions it is necessary to progressively reduce the length of the frame in order to reduce to the minimum the noise on the radio carrier during the interval during which the frame is transmitted. The length should not however be too low in order not to affect the transmission time of the set of data to be sent. Each frame start has a waiting time that is necessary to guarantee the perfect synchronism of the two radio systems every time the communication between the transmitting and the receiving stations is inverted.

3.1.4.4 Response time of the system

Babuc ABC replies to a set of data request or command codes within different time limits, which vary according to the time needed by the system to recover the information stored in its memory and to process the commands received.

Generally, the response time is almost immediate for the vast majority of the requests accepted. The only exception is represented by the transmission of data processed starting from a date/time set by the master, because to perform this type of command, the system must run a query function within the data memory to be able to transmit the desired information only. This search is completed within a variable time limit, which is influenced by 3 factors:

- Number of processes stored in the data memory
- Dimension of the data memory
- Physical distance of the first process with date/time corresponding to the expected one at the end of the survey

The search starts from the last process and continues towards the beginning of the , which contains the processes with older dates/times. This facilitates the search of data whose date/time is close to the real time to the detriment of those with older date/time.

In the most disadvantageous extreme situation, with system connected to the largest memory card full of processes and the master requesting data with date/time corresponding to the oldest process, Babuc ABC transmits the first frame within 3 seconds. The subsequent frames are sent, after the reception of the *Ack* frame, instantaneously. Only a few tenths of seconds, necessary to form the subsequent frames, are necessary if their dimensions are large, that is above 512 bytes.

Given the above, the sections that follow list some real data transmission examples along with the related occupancy time of the communication line. The examples imply the use of a good quality communication line.

Type of transmission	**Bit Rate (bps)**	**Dimension of the frame (bytes)**	**Time required (mm:ss)**
Transmission of instantaneous values from a system with 10 channels	9600	1024	< 00:01
As above	1200	64	< 00:01
As above, via radio with a carrier enabling time of 500 ms	1200	128	< 00:02
Transmission of the hourly and daily data processed referred to a specific day, from a system configured with 10 channels and *EEMinMedMaxDvst word* type processes	9600	1024	< 00:07
As above	1200	64	< 00:50
As above, via radio with a carrier enabling time of 500 ms	1200	128	< 01:15
Transmission of the hourly and daily data processed referred to a specific month, from a system configured with 10 channels and *EEMinMedMaxDvst word* type processes	9600	1024	< 03:30
As above	1200	64	< 25:00
As above, via radio with a carrier enabling time of 500 ms	1200	128	< 40:00
Transmission of the hourly and daily data processed referred to a specific day, from a system configured with 5 channels and *EEMinMedMaxDvst word* processes for 4 channels and *EEEolo3* processes for a wind direction channel	9600	1024	< 00:04
As above	1200	64	< 00:30
Transmission of the hourly and daily data processed referred to a specific month, from a system configured with 5 channels and *EEMinMedMaxDvst word* processes for 4 channels and *EEEolo3* processes for a wind direction channel	9600	1024	< 02:00
As above	1200	64	< 15:00
Transmission of the operating configuration of the system (system parameters, survey parameters, library, operating codes, calibration of the amplifiers, etc.):	9600	1024	< 00:30
As above	1200	64	< 03:20
Transmission of instantaneous values grouped by *EE10Ist word* processes for 8 channels (for a total value of 80 values) with selectable acquisition times	9600	1024	< 00:01
Transmission of 80 *EEMinMedMaxDvst word* processes divided by several channels	9600	1024	< 00:02

3.2 Frame

A frame includes the following information (it has been coded in C because this is the most common language used for this kind of applications):

Element	Meaning
unsigned char *TheSyn[3]*	Synchronism characters at the beginning of the frame. The first character contains value hex FD, while the subsequent ones contain value hex FF
unsigned char *ID*	*ID* of the slave station; value ranging from 2 and 254; towards the master the *ID* takes value 1
unsigned char *NrFrame*	Number of the current frame transmitted; it is generally always equal to 0 for transmissions carried out using one frame only; for multiframe transmissions, value *NrFrame* is increased for each frame transmitted by the slave.
unsigned short *FrameLength*	Identifies the quantity of information contained in a frame. Information includes all the bytes ranging from *TheSyn[0]* and *EOT* (included).
unsigned char *OpCode*	Identifies the operating code of the request; see §Meaning of OpCode
Data	Variable number of bytes containing the data to transmit; this field may be blank
unsigned short *CRC*	16 bit *CRC* of the data transmitted between the *ID* and the last *Data* bytes (included)
unsigned char *EOT*	Identifies the end of the frame; it always contains value ASCII *EOT*

3.2.1 CRC Calculation

CRC is calculated according to the polynomial

$$X^{16}+X^{15}+X^{2}+1$$

The following function is the one used to calculate the 16 bit CRC:

```
unsigned short CRC16(unsigned char *PtrBuf, unsigned short NrChar)
{
unsigned short UnsValA,
                        UnsValB,
                         j;
unsigned char CharValA,
                   i,
                   PEBit;

for (j=0, UnsValB=0; j<NrChar; j++) {
    CharValA = (*PtrBuf++) ^ (unsigned char)UnsValB;
    UnsValB = UnsValB >> 8;
    for (i=0, PEBit=0; i<8; i++)
        PEBit ^= (CharValA >> i) & 0x01;
    if (PEBit)
        UnsValB ^= 0xC001;
    UnsValA = (unsigned)CharValA << 6;
    UnsValB ^= UnsValA;
    UnsValA = UnsValA << 1;
    UnsValB ^= UnsValA;
    }
return UnsValB;
}
```

Where: *PtrBuf* is the pointer to the data buffer
NrChar number of characters present in the data buffer

As explained in paragraph *Communication protocol, the* 16 or 32 bit data with binary format transmitted by the stations contain the most significant byte in the smaller address and the less significant byte in the larger address. It is therefore necessary to invert the sequence of bytes that form these data, after they have been received. The receiving and transmitting master should calculate CRC before and after the conversion of the 16 and 32 bit values.

3.2.2 Meaning of OpCode

The value of *OpCode* identifies the type of request sent or the meaning of the frame transmitted.
A set of codes is dedicated to the management and regulation of the communication. For example, if the master receives an incorrect frame, after the CRC check, it will send a frame with the request code set to *SendLastFrame*. Other codes are grouped as if they were part of the group of command transmission or data request codes.

3.2.2.1 Codes influencing the transmission

Value	Code	Field *Data* transmitted by the Master	Field *Data* received by the Master	Meaning
0x00	*Alert*	---	Byte 0: meaning of the error	Identifies an error condition occurred during the transmission of data. See next sections for a list of the possible error codes transmitted. This frame is always transmitted to the master by the slave.
0x01	*NotAck*	---	---	The sent request has been acknowledged as valid target (master or slave), but field OpCode of the frame has not been acknowledged as valid.
0x02	*Ack*	---	---	The request has been correctly received and managed. This frame is transmitted to the master by the slave if the master has sent a command that should not be followed by data transmitted by the slave; or transmitted from the master to the slave every time the master receives a data package transmitted within a multiframe sequence. In this case, the Ack received by the slave enables it to transmit the next frame.
0x03	*SendLastFrame*	---	---	The receiver requests the transmitter to retransmit the last frame. This frame is usually sent when the received frame is invalid (incorrect *CRC*, incorrect length or not compliant *ID's*) and must therefore be transmitted.
0x04	*EndTrasm*	---	---	The master requests the multiframe transmission from the slave to be completed. This frame is generally used instead of frame *Ack*, if the master finds errors that prevent the next frames from being received (memory allocation problems, full disk, etc.).
0x05	*LastFrame*	---	---	This frame is retransmitted by the slave at the end of a multiframe transmission. It indicates that the current frame corresponds to the last of the transmitted series.

3.2.2.2 Codes influencing the transmission

Value	Code	Field *Data* transmitted by the Master	Field *Data* received by the Master	Meaning
0x00	*Alert*	---	Byte 0: meaning of the error	Identifies an error condition occurred during the transmission of data. See next sections for a list of the possible error codes transmitted. This frame is always transmitted to the master by the slave.
0x01	*NotAck*	---	---	The sent request has been acknowledged as valid target (master or slave), but field OpCode of the frame has not been acknowledged as valid.
0x02	*Ack*	---	---	The request has been correctly received and managed. This frame is transmitted to the master by the slave if the master has sent a command that should not be followed by data transmitted by the slave; or transmitted from the master to the slave every time the master receives a data package transmitted within a multiframe sequence. In this case, the Ack received by the slave enables it to transmit the next frame.
0x03	*SendLastFrame*	---	---	The receiver requests the transmitter to retransmit the last frame. This frame is usually sent when the received frame is invalid (incorrect *CRC*, incorrect length or not compliant *ID's*) and must therefore be transmitted.
0x04	*EndTrasm*	---	---	The master requests the multiframe transmission from the slave to be completed. This frame is generally used instead of frame *Ack*, if the master finds errors that prevent the next frames from being received (memory allocation problems, full disk, etc.).
0x05	*LastFrame*	---	---	This frame is transmitted by the slave at the end of a multiframe transmission. It indicates that the current frame corresponds to the last of the transmitted series.

3.2.2.3 Command transmission or data request codes

Value	Code	Field *Data* transmitted by the Master	Field *Data* received by the Master	Meaning
0x06	*TrCnfSysStat*		Structures type *BBCTime* and *BC_SysInf*	Master receiving system information: current date + system configuration.
0x08	*TrDataMemInf*		Structure type *DataMemInf*	Information on the configuration and on the use of the data memory.
0x0C	*TrAllChElab*		Float value for each active channel, corresponding to the last datum acquired, already engineered	Master receiving the last data acquired/pre-processed of all active channels (multiframe reply).
0x0D	*TrAllMemHeaders*		One or more structures, type *RelMemHeader*	Master receiving all the survey headers present in the data memory (multiframe reply).
0x0E	*TrOneMemHeader*	Bytes 0-1: survey index (0xff= last survey); bytes 2-3: 0= transmits only the survey header; 1= transmits all the survey header and the headers of all the channels.	Structure type *RelMemHeader* with one or more *ChMemHeader* structures, if the parameter in bytes 2-3 is equal to 1.	Reception of the selected survey header with all the headers of the survey channels, that can be selected (multiframe reply).
0x0F	*TrMemRel*	Bytes 0-1: survey index (0xff= last survey); Bytes 2-3: 0= one process per frame; 1= several processes per frame; bytes 4-7: BBCTime structure to select processes from a specific date onwards; if this value is 0, all the processes present in the survey are transmitted	One ore more processing structures	Master receiving a survey (only processing structures). If the parameter of field *Data* contained in bytes 2-3 is equivalent to 1, the processes are stored in frames, in blocks that don't necessarily end with a physical end of a process. For this reason, the received data can be converted only after all the frames have been transmitted (multiframe reply)