Communication Protocol Manual

JOFRA CTC, ITC, MTC, ETC and Compact

Contents

1 Introduction

This document describes the serial communication between the PC and the Jofra CTC, ITC, MTC, ETC and Compact calibrators. The communication is based on telegrams complying with Ametek's ADK-protocol. The communication uses a serial RS232 connection.

2 Protocol

The ADK-protocol is based on the Master/Slave principle. The PC is always to be regarded as the master, and the calibrator as the slave. This means that the calibrator never starts the communication for itself, but solely replies to demands from the PC. When talking about writing and reading, the communication is seen from the PC's point of view, which means that writing is information from the PC to the calibrator, whereas reading is information from the calibrator to the PC.

2.1 Variables

All variables are stated in the format Big-Indian, i.e. starting with the most significant variables followed by the less significant ones etc. (contrary to the Intel notation).
The following designations are used in this manual:

char	1 byte unsigned integer, value 0 – 255.
byte	1 byte unsigned integer, value 0 – 255.
int	2 byte signed integer, value –32,768 – 32,767.
unsigned int	2 byte unsigned integer, value 0 – 65,535.
float	4 byte floating point number, complying with the IEEC standard.
bool	1 byte flag, value 0=False, 1=True.
string[X]	Array of X characters, occupying X+1 bytes. The last byte is always zero.

Numbers followed by an h (e.g. 07h) indicate a hexadecimal number.

2.2 Telegram structure

A telegram is constructed as a serial flow of bytes:

Section:	Length:	Description:
Telegram number.	2 bytes. (unsigned int)	States the telegram type.
Data bytes.	Depends on the telegram type.	If a telegram contains data, they are placed here.
CRC-checksum.	2 bytes. (unsigned int)	CRC-sum for the sections Telegram type & Data bytes.

2.3 Packing and Unpacking of telegrams

Before transmitting the telegram, it is packed to ensure that the end of the telegram can be detected. The telegram must always be terminated by the character EOT = 04h, and therefore no other bytes in the telegram must have this value. To prevent this, a byte with the value 04h is translated into two characters 1Bh + FCh, and if the telegram contains a byte with the value 1Bh, this is translated into the two characters 1Bh + E5h. The telegram will thus consist of a number of bytes different from the value 04h and terminated with the value 04h.

2.4 Error detection

The error detection ensures that defective telegrams can be refused. A simple kind of CRC-checksum is used. The CRC sum does not include the last 3 bytes of a telegram. At the start, the CRC-check- sum of the type unsigned int has the value 0. The following routine is carried out for each telegram byte:

1. CRC-checksum = CRC-checksum Xor (telegram byte × 256)
2. To be repeated 8 times:
   If the CRC-checksum > 7FFFh:
   CRC-checksum = (2 × CRC-checksum) Xor 8005h
   if not:
   CRC-checksum = 2 × CRC-checksum

The following code example is a function written in C, generating the CRC-checksum:

```
unsigned int calcCrc(unsigned int crcSum, unsigned char
datachar)
{
  unsigned char count;

  crcSum = crcSum ^ (datachar * 256);

  for (count = 0; count <= 7; count++)
  {
    if (crcSum > 0x7FFF)
      crcSum = (crcSum * 2) ^ 0x8005;
    else
      crcSum = crcSum * 2;
  }
  return crcSum;
```

}

2.5 Establishment of connection

A telegram from the PC is answered by a telegram with the same telegram type, if necessary “empty” (i.e. without data bytes), to acknowledge the receipt or containing specific data. If a telegram is received with CRC errors, it is entirely ignored by the recipient. The PC waits for at least 1 second after sending the telegram, until the time-out occurs and another transmission from the PC can be tried. After 3 attempts, the transmission is off and the connection is regarded as interrupted. Afterwards you may try to establish a new connection (by transmitting a Log-on telegram). Some telegrams are checked for range errors. If an error is recognized the acknowledge telegram returns a ‘1’. Acknowledge telegrams with no errors return a ‘0’.

2.6 Examples of the content of telegrams

In general a telegram consists of

<Telegram number (2 bytes)>, <Data bytes (x bytes)>, <CRC check sum (2 bytes)>, <04 (1 byte)>

As an illustration the log-on telegram is composed as

<01 (2 bytes)>, <CRC (2 bytes)>, <04 (1 byte)>

The log-on telegram has no data bytes. The CRC check sum is calculated as the CRC sum of the telegram number and the data bytes. Note if the composed telegram holds the byte value 04 hex, this byte is substituted by the escape sequences explained in section ’2.3 Packing and Unpacking of Telegrams’. I.e. the finished telegam does not hold the value 04 hex, except the final byte which is always 04 hex to indicate the end of the telegram.

Another example, telegram #6 - write coefficients

<06 (2 bytes)>, <12 bytes data>, <2 bytes CRC>, <04 (1 byte)>

3 RS232 connection

3.1 Communication parameters

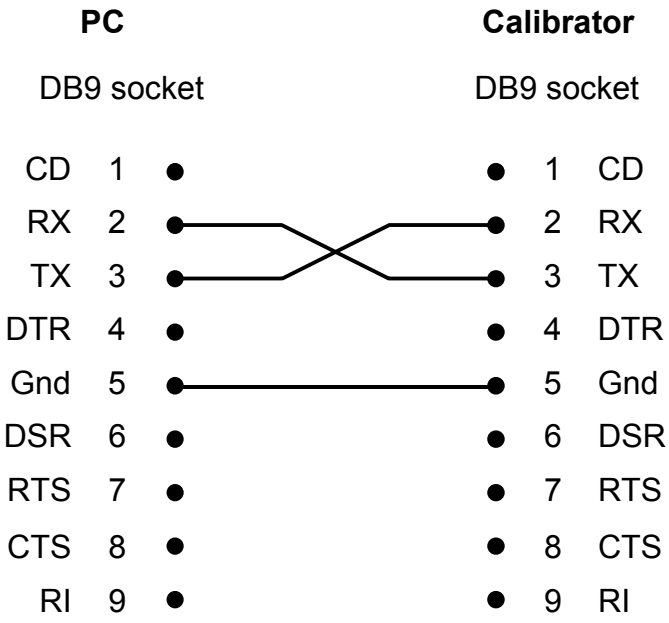
The serial communication is based on a standard RS232 communication. The communication parameters are specified as follows:

- 9600 Baud
- 8 data bits
- No parity
- 1 stop bit

No hardware handshake is used.

3.2 Cable

The cable between the PC and the calibrator is designed in the following way:



4 Telegrams

The communication always starts with the PC setting the calibrator in communication mode (Log-on). At the same time any existing functions are interrupted. As long as the calibrator is in communication mode, the user interface is disconnected. This is shown by the text "REMOT." in the display of the calibrator. It is only possible to read or write when the calibrator is in remote mode.
When the PC logs off, the user interface of the calibrator is reactivated.

Telegram #1 - Log-on

Number:	1
Subject:	The connection between the PC and the calibrator is established.
Reading/writing:	Reading
Comments:	The returned data identify the instrument type, the protocol consulted and the version number of the software of the calibrator. At the same time the calibrator is set in remote mode.
Data, PC → calibrator:	-
Data, PC ← calibrator:	6 bytes: Instrument type (unsigned int): 2091: C-140 2092: C-320 2093: C-320-2 2094: C-650 2095: C-650-2 2096: ITC-155 A 2097: ITC-320 A 2098: ITC-650 A 2099: CTC-140 A 2100: CTC-320 A 2101: CTC-320 B 2102: CTC-650 A 2103: CTC-650 B 2104: MTC-140 A 2105: MTC-320 A 2106: MTC-320 B 2107: MTC-650 A 2108: MTC-650 B 2109: CTC-1200 A 2200: ETC-125 A 2201: ETC-400 A 2202: ETC-400 R Protocol version (unsigned int): 101 (V1.01) Software version (unsigned int): 100 (V1.00)

Telegram #2 - Log off

Number:	2
Subject:	The communication is interrupted.
Reading/writing:	-
Comments:	The remote mode is closed and the user interface of the calibrator is activated.
Data, PC → calibrator:	-
Data, PC ← calibrator:	-

Telegram #3 - Not Applicable

Telegram #4 - Write SET temperature

Number:	4
Subject:	SET temperature.
Reading/writing:	Writing
Data, PC → calibrator:	4 bytes: SET temperature in °C (float)
Data, PC ← calibrator:	-

Telegram #5 - Not Applicable

Telegram #6 - Not Applicable

Telegram #7 - Not Applicable

Telegram #8 - Not Applicable

Telegram #9 - Read serial number

Number:	9
Subject:	Serial number.
Reading/writing:	Reading
Data, PC → calibrator:	-
Data, PC ← calibrator:	13 bytes: Serial number (string[12])

Telegram #10 - Not Applicable

Telegram #11 - Read calibration date

Number:	11
Subject:	Calibration date
Reading/writing:	Reading
Data, PC → calibrator:	-
Data, PC ← calibrator:	4 bytes: Day, 1..31 (char) Month, 1..12 (char) Year, 1998..2025 (unsigned int)

Telegram #12 - Write calibration date

Number:	12
Subject:	Calibration date
Reading/writing:	Writing
Data, PC → calibrator:	4 bytes: Day, 1..31 (char) Month, 1..12 (char) Year, 1998..2025 (unsigned int)
Data, PC ← calibrator:	-

Telegram #13 - Read temperature unit and resolution

Number:	13
Subject:	Temperature unit and resolution.
Reading/writing:	Reading
Data, PC → calibrator:	-
Data, PC ← calibrator:	1 byte: Temperature unit (bit 0 of char): 0: °C 1: °F Temperature resolution (bit 1 of char): 0: 1° 1: 0,1°

Telegram #14 - Write temperature unit

Number:	14
Subject:	Temperature unit
Reading/writing:	Writing
Data, PC → calibrator:	1 byte: Temperature unit (char): 0: °C 1: °F
Data, PC ← calibrator:	-

Telegram #15 - Write temperature resolution

Number:	15
Subject:	Temperature resolution.
Reading/writing:	Writing
Data, PC → calibrator:	1 byte: Temperature resolution (char): 0: 0,1° 1: 1°
Data, PC ← calibrator:	-

Telegram #16 - Not Applicable

Telegram #17 - Read maximum SET temperature

Number:	17
Subject:	Maximum SET temperature permitted.
Reading/writing:	Reading
Data, PC → calibrator:	-
Data, PC ← calibrator:	4 bytes: Max. temperature in °C (float)

Telegram #18 - Write maximum SET temperature

Number:	18
Subject:	Maximum SET temperature permitted.
Reading/writing:	Writing
Data, PC → calibrator:	4 bytes: Max. temperature in °C (float)
Data, PC ← calibrator:	-

Telegram #19 - Read slope rate

Number:	19 (Not applicable for ETC)
Subject:	Slope rate.
Reading/writing:	Reading
Data, PC → calibrator:	-
Data, PC ← calibrator:	4 bytes: Slope rate in °C/min. (float)

Telegram #20 - Write slope rate

Number:	20 (Not applicable for ETC)
Subject:	Slope rate. The slope rate is used by both on- and off-line switch test. The slope rate is only used by the on-line switch test when the flag activated in telegram #88 is set. In the on-line switch test the slope rate is used when the SET temperature command is activated. If the activated SET temperature is lower than the actual temperature the slope is negative. A new SET temperatuer may be activated before the current SET temperature is accomplished. The slope rate is saved in EEPROM
Reading/writing:	Writing
Data, PC → calibrator:	4 bytes: Slope rate in °C/min. (float). Valid range is 0.1 – 9.9.
Data, PC ← calibrator:	-

Telegram #21 - Read stability time

Number:	21
Subject:	Stability time
Reading/writing:	Reading
Data, PC → calibrator:	-
Data, PC ← calibrator:	1 byte: Stability time in minutes (char)

Telegram #22 - Write stability time

Number:	22
Subject:	Stability time
Reading/writing:	Writing
Data, PC → calibrator:	1 byte: Stability time in minutes (char)
Data, PC ← calibrator:	-

Telegram #23 - Not Applicable

Telegram #24 - Not Applicable

Telegram #25 - Not Applicable

Telegram #26 - Not Applicable

Telegram #27 - Read maximum temperature

Number:	27
Subject:	Maximum temperature
Reading/writing:	Reading
Data, PC → calibrator:	-
Data, PC ← calibrator:	4 bytes: Max. temperature in °C (float)

Telegram #28 - Read internal ref. sensor resistance

Number:	28
Subject:	Internal reference sensor resistance
Reading/writing:	Reading
Data, PC → calibrator:	-
Data, PC ← calibrator:	4 bytes: Resistance in Ω (float)

Telegram #29 - Read display temperature

Number:	29
Subject:	Display temperature.
Reading/writing:	Reading
Data, PC → calibrator:	-
Data, PC ← calibrator:	4 bytes: Display temperature in °C (float)

Telegram #84 - Read calibrator mode

Number:	84
Subject:	Read calibrator mode
Reading/writing:	Reading
Data, PC → calibrator:	-
Data, PC ← calibrator:	2 bytes: Test mode (byte): 0: Normal 1: Simulation 2: Service Internal status (byte) 1: Temperature setup 2: Switch test 3: Auto step

Telegram #87 - Read slope rate status

Number:	87 (Not applicable for ETC)
Subject:	Slope rate status
Reading/writing:	Reading
Data, PC → calibrator:	-
Data, PC ← calibrator:	1 byte: 0: Slope rate not active (boolean) 1: Slope rate active (boolean)

Telegram #88 - Write slope rate status

Number:	88 (Not applicable for ETC)
Subject:	Slope rate status. The slope rate status is not saved in EEPROM. The slope rate status is reset to 0 when the calibrator goes off-line. The slope rate status controls the use of slope in the on-line switch test.
Reading/writing:	Writing
Data, PC → calibrator:	1 byte. Slope rate status (boolean): 0: Reset slope rate active flag. 1: Set slope rate flag active.
Data, PC ← calibrator:	-

Telegram #93 – Not Applicable

Telegram #94 – Not Applicable