Communication Protocol between Arduino and Application

# Purpose

This document describes the binary communication protocol between the Arduino Uno, which is measuring the voltages of the circuit and the application running at the other side of the virtual RS-232 interface.

# Definitions

Server – The Arduino board delivers most of the data to the application and is activated by the application. The board is called ‘server’.

Client – The Windows/Desktop application, which is reading the information is called ‘client’

# Requirements

The client is responsible to start and stop the measurement. The board shall not send out any data before measurement has been started by server

The client shall set the number of data to be read. The client shall acknowledge the command.

The transport of measured data shall be tuned to high performance.

# State machine

Server\_Stop – Waiting for activation

Server\_Stopping – Waiting for confirmation by client

Server\_Running – Server is sending data

Client\_Stop – Waiting for activation

Client\_Start – Client is retrieving data

# Assumptions

The server has 6 ADCs which return values between 0 and 1023 (10 bits of payload).

# Commands

Every ADC value of 1023 (11 1111 11112) shall be reduced to 1022 (11 1111 11002). The reason is that two 0xFF values will be seen as start of a command or additional information during data stream of client.

## Client-commands to server (Application -> Arduino)

The following commands may be sent from client to server:

### Set number of analog channels

‘a{number of channels}’

Allowed states for client: ‘Client\_Stop’

Allowed states for server: ‘Server\_Stop’

Defines the number of channels that shall be read from the server board. The parameter {number of channels} is the ASCII representation of the number. The following numbers are allowed: 1, 2, 3, 4, 5 and 6.

The data will be retrieved from the corresponding ADC channel.

### Go Command

‘g’

Allowed states for client: ‘Client\_Stop’

Allowed states for server: ‘Server\_Stop’

Starts the measurement of data and streaming to the client. The number of channels have to be set before.

### Stop Command

‘s’

Allowed states for client: ‘Client\_Start’

Allowed states for server: ‘Server\_Start’

Stops the measurement of data and streaming to the client. The number of channels have to be set before.

The stop will be acknowledged by multiple 0xFF 0xFF 0x01– messages being sent from server to client. The server has to confirm the reception by the stop confirmation command.

The server transitions to the ‘Server\_Stopping’ status.

### Stop Confirmation Command

‘t’

Allowed states for server: ‘Server\_Stopping’

Confirms the reception of the Stop Sequence request.

If the server has received the comment, it will switch to ‘Server\_Stop’.

## Server-Information to client (Arduino -> Application)

### Sample Sequence

The server sends the analog values of the channels (up to 6) in a value chunk of 10 bits for each channel. It is task of the server to maintain the number of requested channels. The number of channels may only be modified during ‘Client\_Stop’.

For bandwidth-saving, the data of the channels are put sequentially in the bit streams. The following table should show the payload for the given number of channels:

|  |  |  |
| --- | --- | --- |
| # of Channels | Bits | Bytes |
| 1 | 10 = 1 Byte + 2 Bits | 2 Bytes |
| 2 | 20 = 2 Bytes + 4 Bits | 3 Bytes |
| 3 | 30 = 3 Bytes + 6 Bits | 4 Bytes |
| 4 | 40 = 5 Bytes | 5 Bytes |
| 5 | 50 = 6 Bytes + 2 Bits | 7 Bytes |
| 6 | 60 = 7 Bytes + 4 Bits | 8 Bytes |

The example following table describes a message for 3 analog channels, which had been initialized by ‘a3’. The table does not represent byte order on network traffic, it represents byte order of communication API



Not used bytes will be filled by ‘0’.

### Server information

An information message is started by two 0xFF messages, which can be distinguished from the data stream. The data stream will contain have a 10 bits of ‘1’ in a row, since the value 1023 is replaced by 1022.

The following bytes describe the information. The length of the information is dependent on the type of message.

### Resynchronisation sequence

A triple 0xFF message is sent to resynchronize the data stream.

The first byte after the resynchronization sequence contains the first 8 bits of the first data analog channel.

### Stop stream sequence

The message 0xFF 0xFF 0x01 indicates that no further stream will be sent to client.

Client and server should switch to ‘Server\_Stop’ and ‘Client\_Stop’. This message will be sent in reply to a Stop command (‘s’) by client.

This message will be sent until the reception has been confirmed by client.

### Error sequence

The message 0xFF 0xFF 0xFE 0xXX indicates that an error has occurred.

The fourth byte might be used for error reason.