ASCII-Serial-Com Data Format

Justin Hugon

2021-11-29

1 Version 0

Need to be careful of the bit-width of register numbers!

This is the format for a serial communication protocol based on ASCII. The protocol is not meant to be efficient, but to be easily human readable.

This protocol defines a host and a device (like USB). Command types have different meanings depending on if they are sent from a host or device

1.1 Data Frame

Data frames are made up of 8-bit bytes. Only the following symbols are present in frames:

a-zA-Z0-9,.> newline (\n) and space

All of the characters are 8-bit ASCII (with the MSB set to 0).
The maximum frame size is 64 bytes to reduce buffer size in microcontrollers.

1.1.1 Data Frame Format

The data frame consists of the following:

- The start of frame character '>'
- One byte of version information for ASCII Serial Com; must be ASCII 0
- One byte of application version and format information represented in the characters A-Z0-9
- One byte of command and/or message type as lower case letters a-z
- Data is then sent as hex 0-9A-F (the letters must be capital)

  - The protocol may use the comma character ‘,’ to separate fields
  - Applications may use the space character to group data
  - 54 or less bytes
- The end of data character ‘.’
- A CRC code represented as 4 bytes of ASCII hex 0-9A-F (capitals), so from 0000-FFFF. The CRC should be run from the start of frame character to the end of data character (inclusive). Use CRC-16-DNP (`http://reveng.sourceforge.net/crc-catalogue/16.htm#crc.cat-bits.16`).
- The end of frame character ‘\n’

Examples:

```
>00r0F.9AD2\n
>02w0F 003FFF92.EA89\n
```

1.2 Command and/or Message Types

Command/Message Type Summary Table

Command/Type			
Description	Code	Host Sent Data	Dev Sent Data
Read register	r	Reg num (4-byte)	Reg num (4-byte) comma (‘,’) reg content
Write register	w	Reg num (4-byte) comma (‘,’) reg content	Reg num (4-byte) for success
Data stream frame	s	frame number (2-byte) comma (‘,’) data	frame number (2-byte) comma (‘,’) data
Device stream on	n		
Device stream off	f		
Error message	e	2 hex digits of error code, comma (‘,’), command code, comma (‘,’), first 9 bytes of data of message that caused error	2 hex digits of error code, comma (‘,’), command code, comma (‘,’), first 9 bytes of data of message that caused error
No-op	z	Optional: any data	Optional: any data

- r: Read register: The host sends this command with a register number as 4-bytes of data in capital hex (so from 0000-FFFF). The device will respond with an ‘r’ message with data register number in 4-bytes then comma (‘,’) then register value, both in capital hex.

- w: Write register: The host sends this command with data: register number as 4-bytes of data (so from 0000-FFFF) then comma (‘,’) then register value, both in capital hex. On success, the device will respond with a ‘w’ message with the register number as 4-bytes of data (in capital hex).
- s: Data stream frame: The same format is sent from the host to the device or the device to the host. For streaming from the host, the host just starts streaming. For streaming to the host, the host must send the ‘n’ and ‘f’ commands to start and stop streaming. The data consists of a stream frame number, a ‘,’ character, and the data. The stream frame number and data must be 2 bytes of ASCII capital hex (so from 00-FF). Spaces may be put into the data to act as e.g. field separators. The stream frame number must increment by one with each frame sent.
- n: Device stream on: The host sends this with no data to tell the device to start streaming data.
- f: Device stream off: The host sends this with no data to tell the device to stop streaming data.
- e: Usually device to host, replying with error and beginning of message that caused error.
- z: No-op: used for testing and similar. No meaning. May contain any data.

Error Code Table

Code	Description
0x00	Unknown Error
0x01	No Error
0x10	Data too long for frame
0x11	Problem computing checksum
0x12	Invalid frame
0x13	Invalid frame: missing or misplaced ‘.’
0x14	Invalid frame: non-hex char where hex char expected
0x15	Command not implemented
0x16	Unexpected command
0x17	Data too short
0x20	Register block is null (invalid)
0x21	Register number out of bounds
0x22	Register value is wrong number of bytes
0x30	Circular buffer out of bounds
0x31	Circular buffer tried to pop from empty buffer
0x40	File read error
0x41	File write error