ASCII-Serial-Com Software Architecture

Justin Hugon

2020-06-13

This document describes the high-level architecture for a serial communication protocol based on ASCII.

1 Overview

Assuming UART receives data into a buffer and data in another buffer is transmitted. ASCII-Serial-Com is part of a data processing loop (likely low priority) on a microcontroller.

For just the basic register control interface (not streaming), the device could just have a function in it's event loop like:

```
void pollAsciiSerialCom(uint8_t* inBuffer, uint8_t inBufSize, uint8_t*
    outBuffer, uint8_t outBufSize, void* registers, uint16_t regSize);
```

That would then call functions below to read from inbuffer, frame, check, unpack, and dispatch functions to the appropriate function for action. The outBuffer is there for responses, and registers are there for reads and writes.

A macro would define the bit-width of each register.

May want to have an argument that is the register to store the number of checksum errors in.

2 Receiver

- Framer: identifies and returns frames out of the incoming serial byte stream
  - Inputs: byte stream
  - Outputs: data frame
- Checker: computes and verifies the checksum of the frame
  - Inputs: data frame

  - Outputs: valid flag

- Unpacker: unpacks the message type (command) and data from the frame
  - Inputs: data frame
  - Outputs: message type, data

- Dispatcher (optional): sends different message types to different functions or threads
  - Inputs: message type, data

3 Transmitter

- Packer: packs the message type (command) and data into a frame
  - Inputs: command, data
  - Outputs: data frame without checksum or end of frame

- Checker: computes and verifies the checksum of the frame (can be part of packer)
  - Inputs: data frame without checksum or end of frame
  - Outputs: data frame

- Streamer: puts the frame into the message stream

4 Shared Components

- Checksum calculator
- Message type validation
- Data validation for message type (for python implementation)