Application Note - Serial Protocol

AST-Robotics Serial Protocol

Introduction

The aim of this document is to describe the serial communication protocol used in ST by the AST-Robotics team. This document describes the bit-level message format and some of the higher level commands.

Index

1 Data format

The serial protocol defines two layers: a first lower layer, hereafter named “Layer 1”, and a second upper layer, “Layer 2”.

The Layer 1 packet is defined as follows:

Type	Encoded Payload	EOF
Bytes	N	1

Where:

- **Encoded Payload:** is the message exchanged by the protocol between the module and a host. Its length can change but it is limited by the physical MTU (Maximum Transmission Unit).
  The actual payload is encoded by a byte stuffing function, described later in this document, in order to avoid any EOF character in its content.
- **EOF (0xF0):** byte representing the end of the packet. It has the value: 0xF0.

The Encoded Payload of Layer 1 must be processed by a function which performs the **reverseByteStuffing**, described later, in order to obtain the Payload. The resulting Payload is a Layer 2 packet which has got the following format:

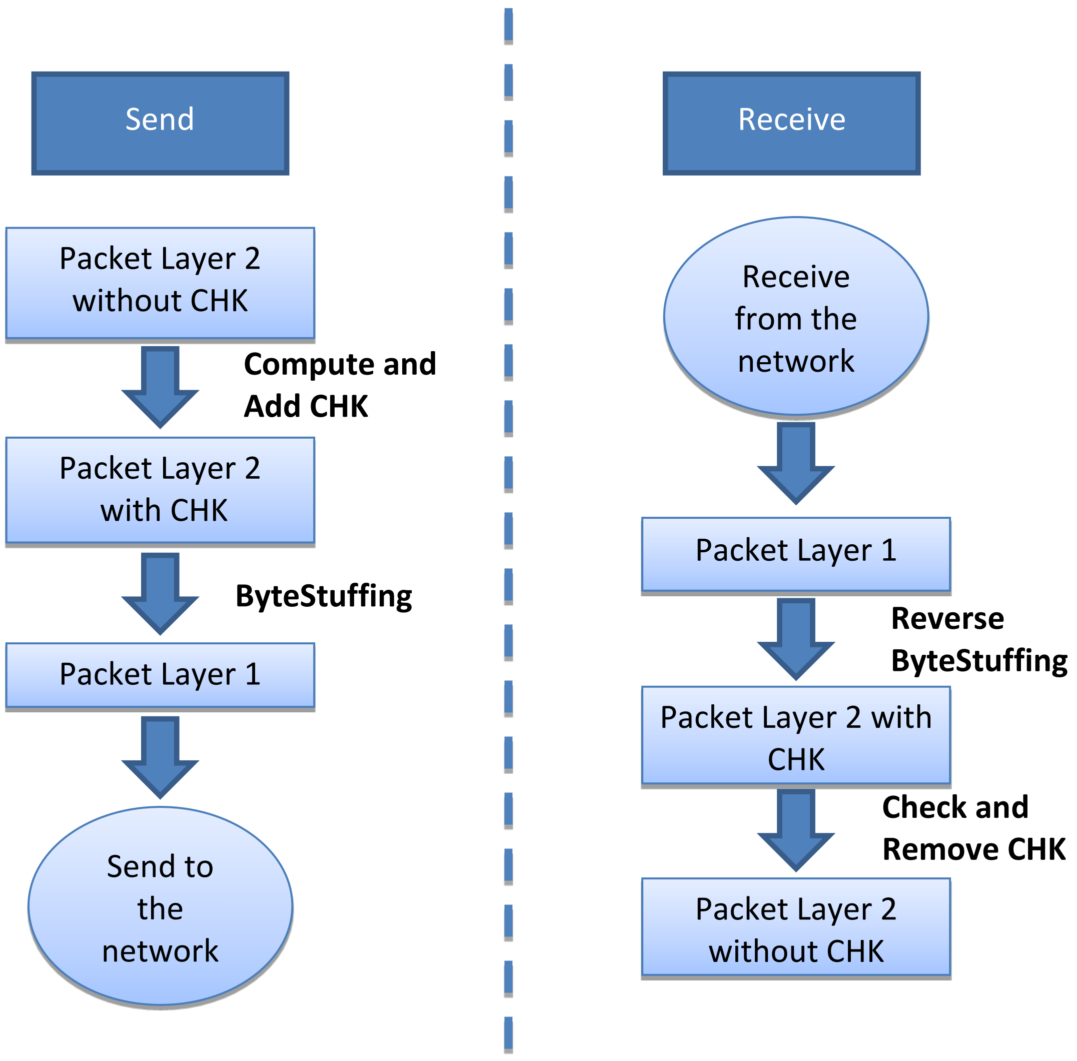
Type	Destination Address	Source Address	Command (CMD)	Payload	CHK
Bytes	1	1	1	N	1

Where:

- **Destination Address:** symbolic address of the destination. It enables unicast messages on a shared bus.
- **Source Address:** symbolic address of the source.
- **Command:** code representing the issued command. It specifies how the payload should be interpreted.
- **Payload:** data which are interpreted with respect to the CMD field.
- **CHK:** the computed sum of all the bytes in the packet must be zero (see section 2.1).

2 Processes involved in sending/receiving packets

The following figure shows the processes needed in order to send and receive a packet. The application of the two functions for checksum and bytestuffing, and their reverses, ensures the correct interpretation of the CMD and its payload.



2.1 Checksum Algorithm

The checksum algorithm ensures that the packet you handle contains the correct information.

The algorithms needed to compute and to verify the checksum integrity are explained by the following pseudo-code snippets.

ComputeandAddCHK

```
begin
    uint_8 CHK = 0
    for i=0:layer2length
        CHK = CHK - layer2data(i)
    end
    layer2data(layer2length+1) = CHK
end
```

CheckandRemoveCHK

```
begin
    uint_8 CHK = 0
    for i=0:layer2length
        CHK = CHK + layer2length
    end
    layer2length = layer2length - 1
    return CHK == 0
end
```

2.2 Byte Stuffing

Byte stuffing is a process that transforms a sequence of data bytes that may contain 'illegal' or 'reserved' values into a potentially longer sequence that contains no occurrences of those values.

In this case the EOF character identifies the end of the packet, therefore the packet must not contain any other occurrence of EOF character.

For this reason the following special characters are defined:

- **TMsg_EOF (0xF0):** is the EOF of layer 1 packet
- **TMsg_BS (0xF1):** is the byte stuffing escape character
- **TMsg_BS_EOF (0xF2):** is the substitution for TMsg_EOF

The byte stuffing algorithm, used in sending actions, is defined as follows:

- Given Layer 2 message, for each character:
    - Substitutes TMsg_BS with TMsg_BS followed by TMsg_BS (double the character).
      0xF1 become 0xF1 0xF1.
    - Substitutes each TMsg_EOF with TMsg_BS followed by TMsg_BS_EOF.
      0xF0 become 0xF1 0xF2.

As it can be seen, the length of the packet of Layer 2 may change in the process.

The reverse byte stuffing algorithm can be defined in the same way:

- Given a Layer 1 payload, for each character

- Substitute the expression (TMsg_BS followed by TMsg_BS) with a single TMsg_BS. 0xF1 0xF1 become 0xF1.
- Substitute the expression (TMsg_BS followed by TMsg_BS_EOF) with TMsg_EOF. 0xF1 0xF2 become 0xF0.

3 Standard Commands

A set of fundamental and standard commands are defined in this document. The protocol can be extended in order to accommodate many more commands which are specifically related to the target application.

The standard commands are summarized in the following table:

Command	CMD Value	Meaning
CMD_PING	0x01	This is the standard ping command, the device will reply accordingly.
CMD_Read_PresString	0x02	Requests the presentation string which contains basic device information (name and version)
CMD_Reset	0x0F	Requests a reboot of the device
CMD_Reply_Add	0x80	This value is added to the value of the field CMD to form the command for the reply. E.g., 0x81 = reply to the PING command

This list specifies the value of the field CMD which can be inserted in the request packet (from the host to the module). The module will reply adding **CMD_Reply_Add** to the value of CMD.

Note that the length of the packets is determined by low level function using terminator and escape special characters. For this reason, the real length of the packet could not be equivalent to the message length. Moreover, the data contained in the Layer 2 payload are serialized before being copied in a packet so that, using a function to deserialize them, the data are independent from the architecture (big/little-endian).

The commands, in Layer 2 format, are explained here in more details:

3.1 CMD_PING

The packet is formed as follows:

Type	Destination Address	Source Address	Command
Length (Bytes)	1	1	1
Value	XX	YY	CMD_PING

The module will answer with the same packet format but source and destination addresses will be obviously swapped and the command field will contain CMD_PING + CMD_Reply_Add.

3.2 CMD_Read_PresString

The request packet is equal to CMD_PING where CMD is substituted by CMD_Read_PresString.
As reply you will receive:

Type	Destination Address	Source Address	Command	Payload
Length (Bytes)	1	1	1	K
Value	YY	XX	CMD_Read_PresString+ CMD_Reply_Add	String of K characters

3.3 CMD_Reset

The request packet is equal to CMD_PING where CMD is substituted by CMD_Reset. The module will reboot and no ACK will be sent. The host can check the new state of the module using other application dependent commands.

4 PING Example

In this example, we consider the PING command, how to form the request and its response at “Layer 1” and “Layer 2” (see the general Serial Protocol document for more information about layers).

4.1 Request

With the hypothesis of:

- Sender Address = TMsg_EOF = 0xF0
- Destination Address = 0x42

The Layer 2 request packet is:

Type	Destination Address	Source Address	Command	CHK
Length (Bytes)	1	1	1	1
Value	0x42	0xF0	CMD_PING = 0x1	205 = 0xCD

After the bytestuffing algorithm the “Layer 1” packet will be:

0x42	TMsg_BS = 0xF1	TMsg_BS_EOF = 0xF2	0x1	0xCD	TMsg_EOF =0xF0

4.2 Reply

The packet is received at “Layer 1”. After the reverse bytestuffing and checksum the original packet is parsed (at “Layer 2”).

Upon a request of PING the “Layer 2 ” reply will be the following:

Type	Destination Address	Source Address	Command	CHK
Length (Bytes)	1	1	1	1
Value	0xF0	0x42	CMD_PING + CMD_Reply_Add = 0x81	77 = 0x4D

After the bytestuffing algorithm the “Layer 1” packet will be:

TMsg_BS = 0xF1	TMsg_BS_EOF = 0xF2	0x42	0x81	0x4D	TMsg_EOF =0xF0

5 Revision history

Document revision history

Date	Revision	Changes
24-Set-2012	1.0	Initial release