constant robotics

Serial Transport Protocol v3.0

specification

clib

www.constantrobotics.com

CONTENTS

DOCUMENT VERSIONS

Table 1 – Document versions.

Version	Release date	What was changed
3.0	12.12.2022	Specification Serial Transport Protocol version 3.0.

PROTOCOL VERSIONS

Table 2 – Protocol versions.

Version	Release date	What's new
3.0	12.12.2022	1. The logic of information exchange has been changed. 2. Unused package types have been removed.

DESCRIPTION

Serial Transport Protocol version **3.0** (hereinafter referred to as the protocol) is designed for the transmission of any types of data over serial ports. According to OSI classification, the protocol is a transport layer protocol. The protocol provides reliable data delivery up to 4 153 343 bytes. The protocol is designed for reliable data transmission over radio channels and allows data exchange under packet loss conditions. The protocol allows organizing up to 256 virtual data channels on one serial port. The Protocol has a minimum number of service bytes (only 10 bytes) added to the transmitted data in each packet.

PROTOCOL PARAMETERS

Table 3 – Protocol parameters.

Parameter	Value
OSI level	OSI transport level
Packet loss handling	The sender performs re-sending of packets, if necessary, ensuring data redundancy.
Maximum data packet size	1024 bytes, including 1014 data bytes and 10 service bytes.
Maximum size of related data items for transmission	4 153 343 bytes.
Adaptability to communication channel bandwidth	The packets should be sent at an interval consistent with the channel capacity.

DATA EXCHANGE PRINCIPLE

The protocol specifies only one type of packets – packets for data transmission (**DATA** packets). The principle of data exchange by means of the Protocol is the following: the data array for transmission is split into sections of fixed length (1014 bytes) with the exception of the last section, which contains the remaining data. The data of each section are packed in a packet (10 service bytes are added to the data). The packets generated with a maximum size of 1024 bytes are sent in series via the serial port. The receiver, once another data package is received, will copy its contents to the clipboard. To ensure reliable data delivery, the sender can resend packets. The number of resending actions is determined by the sender (determined by the user). The protocol allows splitting the transmitted data into separate virtual channels (up to 256 channels – logical ports) for sending several types of data content through

one serial port with their splitting on the receiving side. Figure 1 shows the principle of information exchange using the Serial Transport Protocol version 3.0.

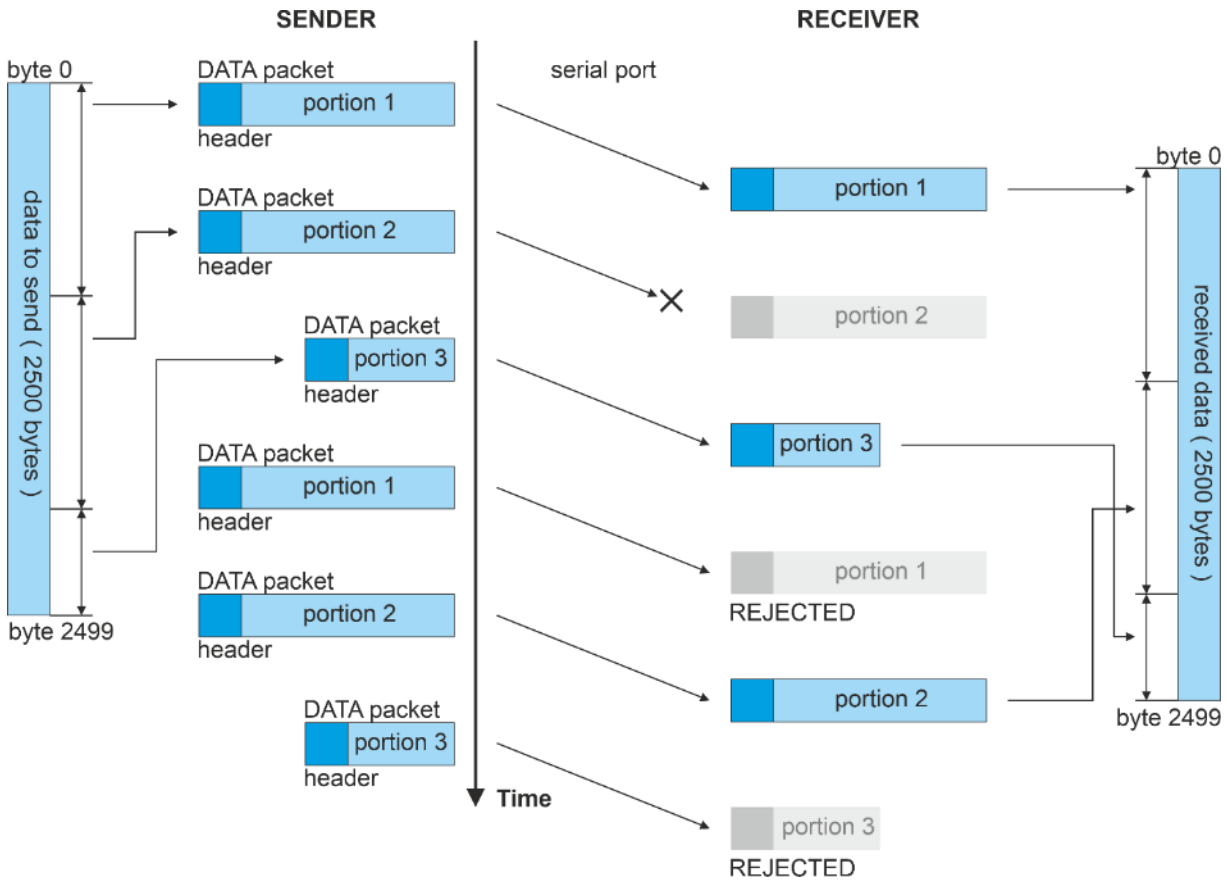

Figure 1 – Data exchange principle.

The figure 1 shows a simple data sending example. 2500 byte size was taken as an example. According to the protocol specification, the maximum size of the transmitted packet is 1024 bytes including 1014 bytes of data and 10 service bytes. In this example, the data volume will be divided into 3 fragments of 1014 bytes, and 472 bytes for the last fragment. **Data is divided into sections in sequence starting with the first byte (that is a continuous single-dimension byte array).** For each data section, **DATA** packets containing 10 bytes of service data (header) are generated. Packets are sent to the receiver in sequence. The packets should be sent at an interval consistent with the channel capacity. The required interval between packet sending (the interval between the beginning of sending the current packet and the beginning of another packet sending) must be calculated for each transmitted packet according to the following formula:

$$T_{sec} = \frac{N * 8}{B_{bps}}, \quad (1)$$

where T_{sec} is the interval between packets (seconds); B_{bps} is the Serial port capacity (bits per second); N is the size of transmitted packet in bytes.

The figure shows that a **DATA** packet is initially sent with the first portion of data. Then the second **DATA** packet is sent. Let us suppose the second packet was lost. The next packet received by the receiver contains the third portion of data. After receiving the **DATA** packet with the third portion of data, the data has not yet been collected. To provide reliable data delivery, the sender resends the packets. Let us suppose the number of resend actions was set as two. Once the third packet is sent, the sender resends the packets, starting with the first one. The **DATA** packet with the first portion of data will be rejected by the receiver, since it has already been received. The re-sent **DATA** packet with the second portion of data will be successfully processed by the receiver. After that, the data will be considered as received and all subsequent packets with these data will be rejected.

DATA PACKET FORMAT

DATA packets have maximum length of 1024 bytes and are intended for data sending. DATA packets have the following format.

Byte No.	0	1	2	3	4	5	6	7	8	9	10	…	N
filed, hex	0xAA	0xA0 | buffer_ID	logic_port	packet_ID		max_packet_ID		packet_size		CRC	data[0]	…	data[n]

Table 4 - DATA Packet Fields.

Field	Value	Description
Start byte	0xAA	The start byte has a fixed value and is used to identify DATA packets in the serial port buffer.
Header	0xA0	The packet header occupies the first 4 bits of the second byte of the packet. It has a fixed value.
buffer_ID	0 (0x0) to 15 (0xF)	Data buffer identifier. It occupies the last 4 bits of the second byte of the packet. The value of the data buffer identifier is generated as follows: Each related data item (e.g., data structure) is marked sequentially with its respective data buffer number from 0 (0x0) to 15 (0xF). After 15 (0xF) identifier, the numbering starts again (0x0) (0, 1, … 15, 0).
logic_port	0 (0x00) to 255 (0x0FF)	Logical port number. The protocol allows the transmitted data to be divided into 256 logical streams for transmitting different meaningful data over a single serial port.
packet_ID **(big endian)**	0 (0x0000) to 4095 (0x0FFF)	Packet ID. The data sent are divided into packets, each of which is assigned its own ID in sequence from 0 (0x0000) to 4095 (0x0FFF). The ID value is sent in **big endian** format.
max_packet_ID **(big endian)**	0 (0x0000) to 4095 (0x0FFF)	Maximum packet ID for the data being sent (last packet identifier). If the size of data being sent is less than 1014 bytes, all data will be sent in one packet. This means that packet_ID and max_packet_ID fields will be identical and equal to 0 (0x0FFF). The ID value is sent in **big endian** format.
packet_size **(big endian)**	11 to 1024	Packet size. It can take values from 11 (0x000B - minimum packet size) to 1024 (0x0400 - maximum packet size). The packet size is transmitted in **big endian** format.
CRC	-	Header checksum. This is the low byte of the first 9 bytes of the DATA package: $\mathrm{CRC} = (uint8\_t)(\mathrm{byte0} + \mathrm{byte1} + \mathrm{byte2} + \mathrm{byte3} + \mathrm{byte4} + \mathrm{byte5} + \mathrm{byte6} + \mathrm{byte7} + \mathrm{byte8} + \mathrm{byte9})$
data[n]	any	Data being sent. Maximum size of data included in the packet is 1014 bytes.