Serial Debug Protocol

[**Introduction**](#_5sj56bun2upf) **2**

[**Physical layer**](#_5q6y265opf46) **2**

[**Data Link Layer**](#_hvwu1go717or) **2**

[**Transport Layer**](#_pyfhs4l15l1i) **3**

# 

# Introduction

This document describes the different layers of the Serial Debug Protocol to be used for point-to-point communication between two devices: a master device (i.e., a computer) and a single slave (i.e., a microcontroller). This is a simple protocol that provides the ability to read and write up to 8192 values, and provides a simple method for error detection (but no error correction).

# Physical layer

The bit stream will be transmitted over UART with the following characteristics:

Baudrate: 9600 bauds

Data bits: 8

Parity: None

Stop bits: 1

# Data Link Layer

The format of the data on the channel between the two devices will be as follows:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Byte** | **0** | 1 |  |  |  |  |  |  |  |
|  | **1** | 0 |  |  |  |  |  |  |  |
|  | **2** | 0 |  |  |  |  |  |  |  |
|  | **3** | 0 |  |  |  |  |  |  |  |
|  | **4** | 0 |  |  |  |  |  |  |  |
|  | **5** | 0 |  |  |  |  |  |  |  |
|  | **6** | 0 |  |  |  |  |  |  |  |
|  | **7** | 0 |  |  |  |  |  |  |  |
|  | **8** | 0 |  |  |  |  |  |  |  |

* Frames going in both directions (i.e., master to slave, and vice versa) have the same format.
* There are a total of 9 bytes in each frame.
* The start of frame (SOF) byte has a 1 on its most significant bit. Therefore, the SOF byte value ranges from 0x80 to 0xFF.
* The rest of the bytes (1 to 8) have a 0 on their most significant bit. Therefore, these bytes value ranges from 0x00 to 0x7F.
* Each frame contains a total of 63 Transport layer bits (grayed out on the figure). These will be explained in the next section.

When a device receives a SOF byte, it discards any ongoing transfer and starts a new one.

# Transport Layer

The 63 Transport layer bits are interpreted as follows:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **bit** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Word** | **0** | 0 | E | R/W | ID (13 bits) | | | | | | | | | | | | |
|  | **1** | Value (32 bits) | | | | | | | | | | | | | | | |
|  | **2** |
|  | **3** | CRC (16 bits) | | | | | | | | | | | | | | | |

* Error bit (E): It indicates whether there was an error in the transmission.
  + The master will always set this bit to 0 when transmitting.
  + When responding to a read command, the slave will set this bit to 0 if there was no error condition, and will send the corresponding value on the Value field.
  + When responding to a write command, the slave will set this bit to 0 if there was no error condition, and will send a 0 on the Value field as acknowledgement.
  + When responding to a read or write command, the slave will set this bit to 1 if there was an error condition, and will send an error code on the Value field. The master will keep statistics of the errors.
* Read/Write bit (R/W):
  + The master sets this bit to 1 to read a value from the slave.
  + The master sets this bit to 0 to write a value to the slave.
  + The slave sets this bit to 1 to reply to a read command.
  + The slave sets this bit to 0 to reply to a write command.
* ID: This field indicates the identification of the value to read or write. It ranges from 0 to 8191 (13 bits).
* Value:
  + When the master sends a read command, it sets this field to 0.
  + When the master sends a write command, this field contains the value to be written.
  + When a slave responds to a read command, this field contains the read value.
  + When a slave responds to a write command, it sets this field to 0.
  + When there is an error condition, this field contains an error code (see table below).
* CRC: This field contains a cyclic redundancy check (CRC) to allow some level of data integrity check. It is a standard CRC-16-CCITT error detection (CRC polynomial is 0x1021). Words 0 to 2 are used in the calculation. Bit 15 of Word 0 is set to 0 for the calculation (see figure above).
  + The master/slave will calculate the CRC of the message to be transmitted and store it in the CRC field.
  + The master/slave will calculate the CRC of the received message and compare it to the value on the CRC field. If there is a match, the data is processed. Otherwise, the data is discarded. In the case of the slave, an error message is returned when there is a CRC error on the received message.

|  |  |  |
| --- | --- | --- |
| **Error codes** | **Description** | |
| 0 | CRC error | |
| 1 | Inexistent ID | |
| 2 | Value out of range | |