Bootloader

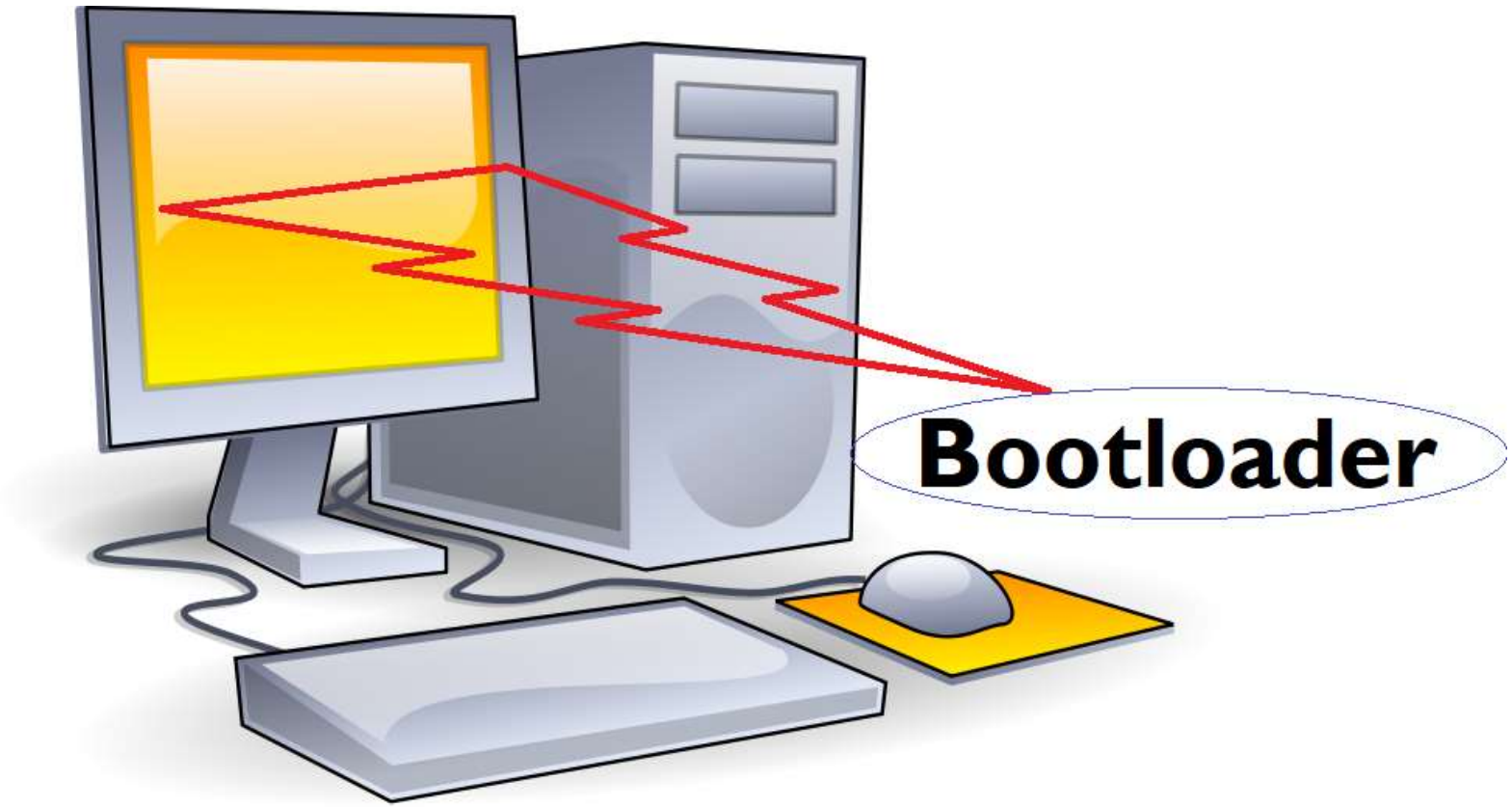
Many binary formats to use any application or firmware file, Majorly converting and using formats is

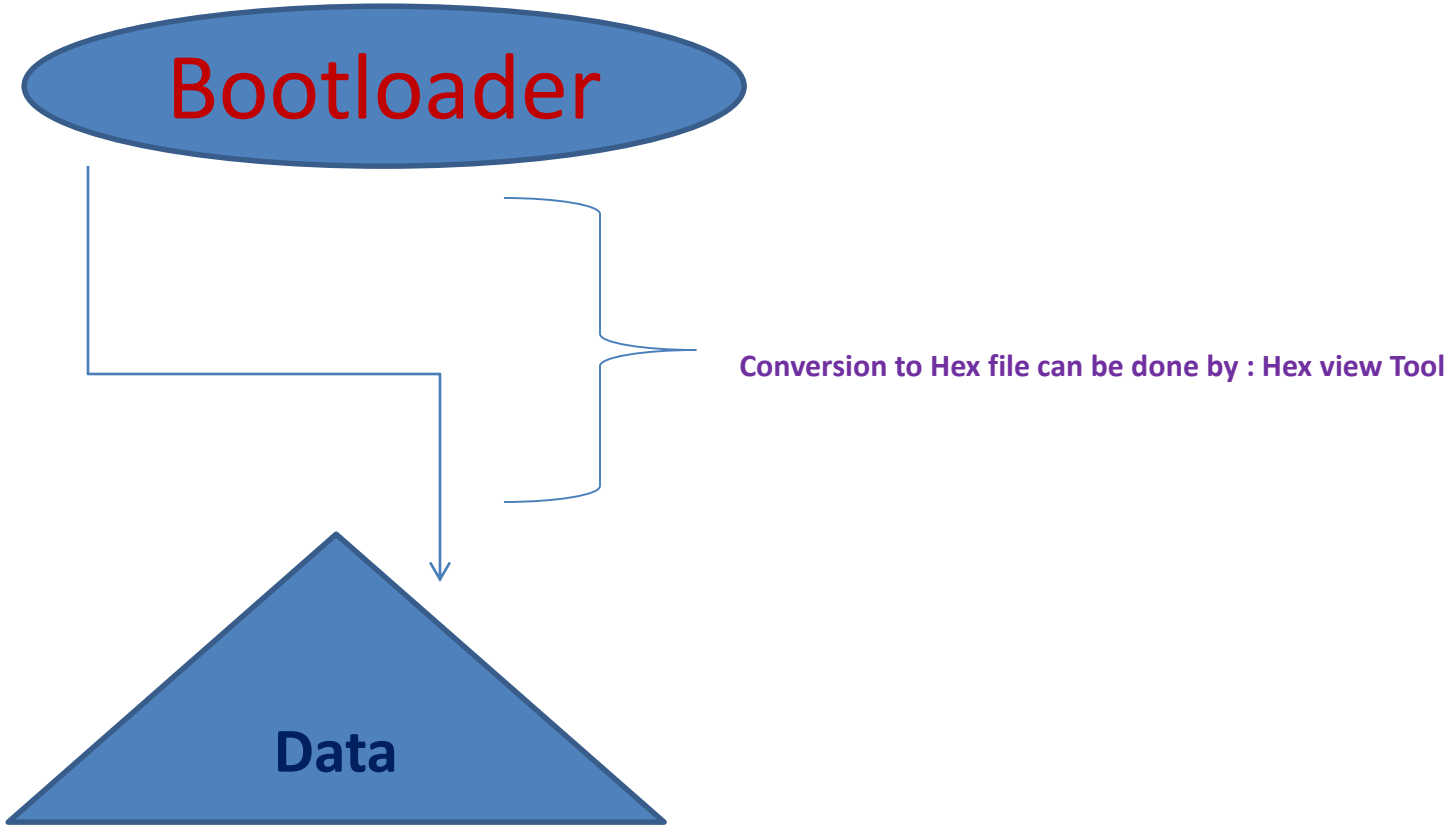
1. Hex format &
2. SRE format for the standards such as Intel and Motorola.

Transfer Data

- The Request to **upload** data by the tester to initiate **the data** transfer from the **ECU to Tester** or to **download** data by the tester to initiate data transfer from **Tester to ECU**
- The above mode of transferring data has to be preceding with either **upload or download** the data using the **respective services**

Assumption Scenario




Bootloader
Conversion to Hex file can be done by : Hex view Tool
Data

Hex File

01 4D 13 DE 01 21 01 10 10 01 01 .. … ……

02 4D 12 DE 01 41 01 10 10 01 01 .. … ……

03 4D 15 DE 01 21 01 05 10 01 01 .. … ……

04 4D 17 DE 01 27 01 10 10 01 01 .. … ……

05 4D 18 DE 01 61 01 10 10 01 01 .. … ……

06 4D 12 DE 01 21 01 10 10 01 01 .. … ……

07 4D 1B DE 01 81 01 10 10 01 01 .. … ……

Terminologies – Request Transfer Data

Block Sequence Counter

Block Sequence Counter initialize from **0x00** and starts from **0x01** to the **subsequent range** till 0xFF is **maximum**

Transfer Request Parameter Record

The data to be **upload or download** to be included in the request message

Terminologies – **Upload Download Functional unit**

Address and Length Format Identifier

- The number of bytes used for the Memory Address and Memory Size parameters is defined by Address and Length Format Identifier (low and high nibble).
- It is also possible to use a fixed Address and Length Format Identifier & **unused bytes** within the Memory Address or Memory Size parameter are padded with the value 00 hex in the higher range address locations.
  - **Bit 7-4 : length of memory size parameter**
  - **Bit 3-0 : length of memory address parameter**

Terminologies – **Upload Download Functional unit**

Memory Size (Un Compressed)

- During the Transfer Data service the size of memory used by the server is compared with amount of data transferred & uncompressed memory size.
- This increases the programming security.
- The number of bytes used for this size is defined by the high nibble (bit 7 - 4) of the Address Format Identifier.

Terminologies – **Upload Download Functional unit**

Memory Address

- The parameter Memory address is the **starting address** of server memory to which data is to be written.
- After recognizing the starting address server will keep on transfer a data to consecutive address till the end byte of data
- The number of bytes used for this address is defined by the low nibble (bit 3 - 0) of the **Address Format Identifier.**
- Byte #m in the Memory Address parameter is always the least significant byte of the address being referenced in the server.
- The most significant byte of the address can be used as a memory Identifier.

Terminologies – **Upload Download Functional unit**

Length Format Identifier

- This parameter is a one-byte value with each nibble encoded separately:
  - bit 7 - 4: length (number of bytes) of the Max Number Of Block Length parameter.
  - bit 3 - 0: reserved by standard to be set to 0 hex.
- The format of this parameter is compatible to the format of the address and Length format Identifier parameter contained in the request message, except that the lower nibble has to be set to 0 hex.

Max Number of Block Length

- This parameter is used by the Request Download positive response message to inform the client how many data bytes shall be included in each Transfer Data request message from the client.
- This length reflects the complete message length.

Terminologies – **Upload Download Functional unit**

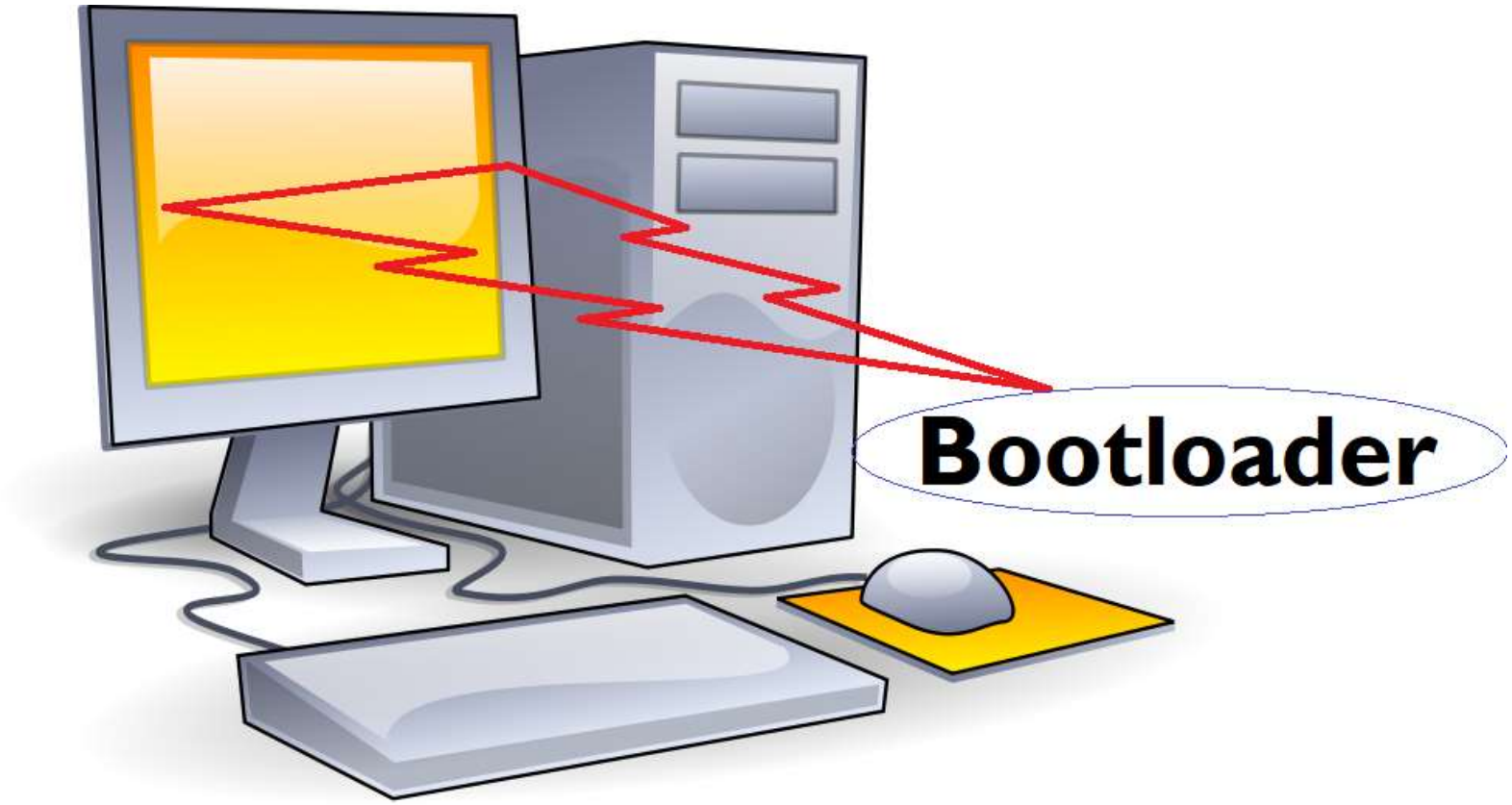
Block Sequence Counter

- The Block Sequence Counter parameter value starts **at 01 hex** with the first Transfer Data request that follows the
- Request Download (34 hex) or Request Upload (35 hex) service. Its value is **incremented by one** for each subsequent
- Transfer Data request. At the value one-byte value with **each nibble encoded separately**

Checksum

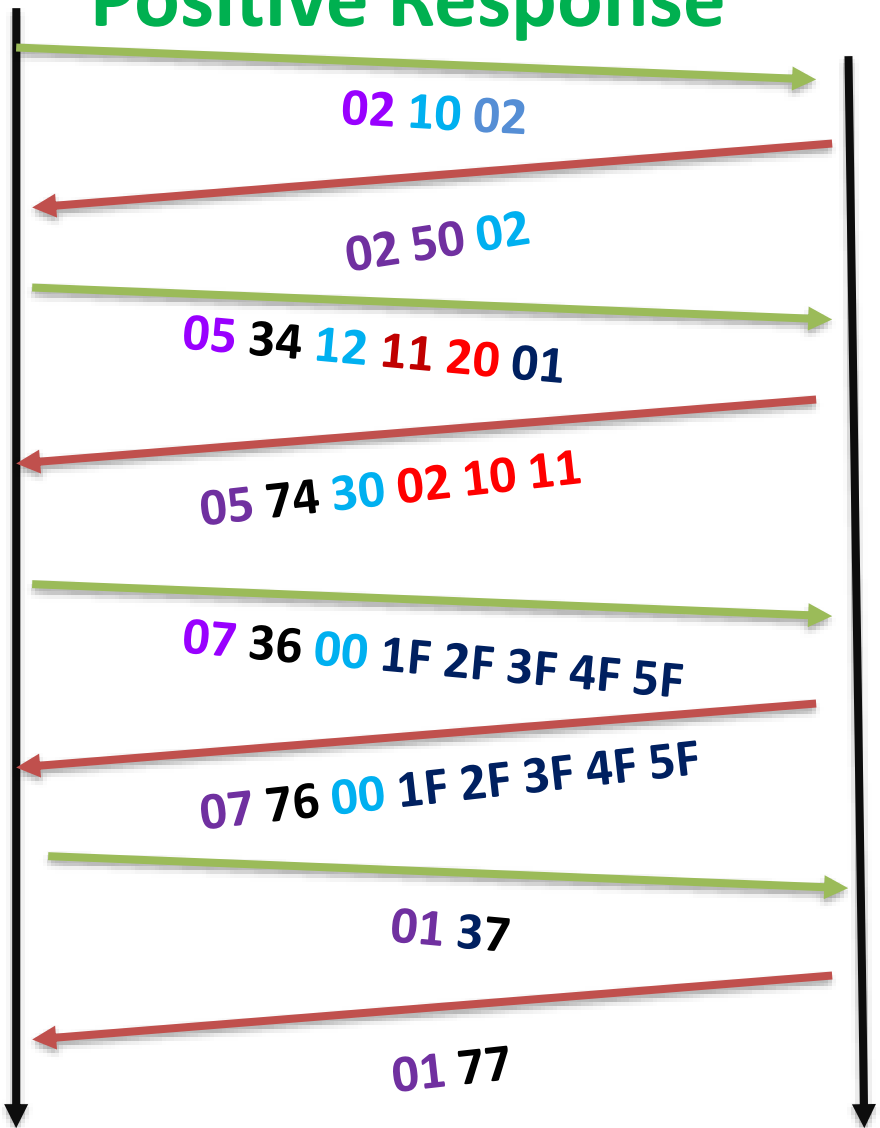
- To compare the uncompressed data and compressed data check sum is used at the end.
- Once the process gets started check sum will note down the end value address and **which compare before the entire** data transfer done.
- If the check sum matches then it is considered as **data transferred successfully**

Request Upload/Dowload



Transfer Data - Download

Positive Response



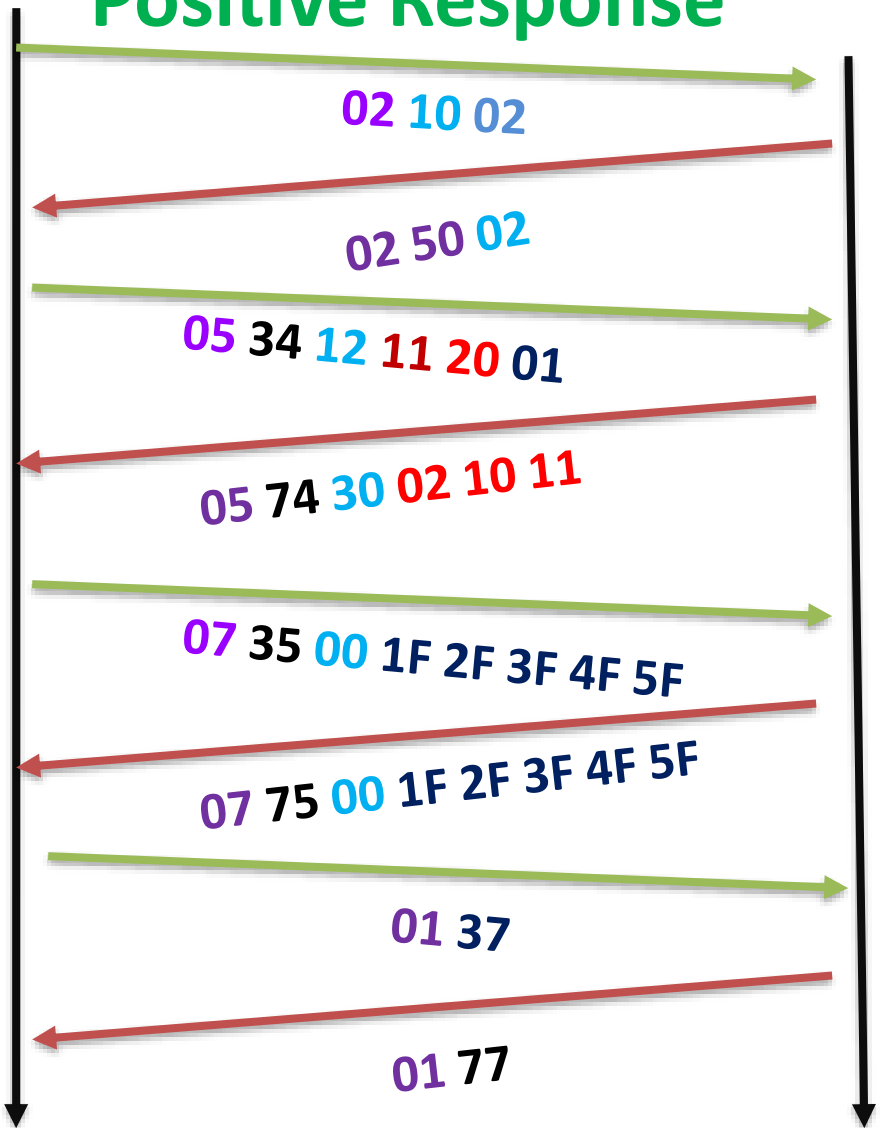
34 – Service Id
12 – Data Format Id
11 - Address & Length format Id
20 - Memory Address
01 - Memory Size

36 – Service Id
00 – Block Seq. Counter
1F 2F 3F 4F 5F - Data

05,07 - PCI Length
74 ,76- Response Sid
30 - Length Format Id
02 10 11 - Max Number of block length

Transfer Data - Upload

Positive Response



34 – Service Id
12 – Data Format Id
11 - Address & Length format Id
20 - Memory Address
01 - Memory Size

36 – Service Id
00 – Block Seq. Counter
1F 2F 3F 4F 5F - Data

05,07 - PCI Length
74 ,76- Response Sid
30 - Length Format Id
02 10 11 - Max Number of block length

List of NRCs Supported

- 0x13 Incorrect Message Length
- 0x24 Request Sequence Error
- 0x31 Request Out of Range
- 0x71 Transfer Data Suspended
- 0x72 General Programming Failure
- 0x73 Wrong Block Sequence Counter
- 0x92 Voltage Too High / Voltage Too Low