Serial Memory Loader
for
Arduino Nano

—

SOEN 422 - Fall 2021 Project - Document #2

Dr. Michel de Champlain
Department of Computer Science and Software Engineering
Concordia University, Montreal, Canada

November 9, 2021

Table of Contents

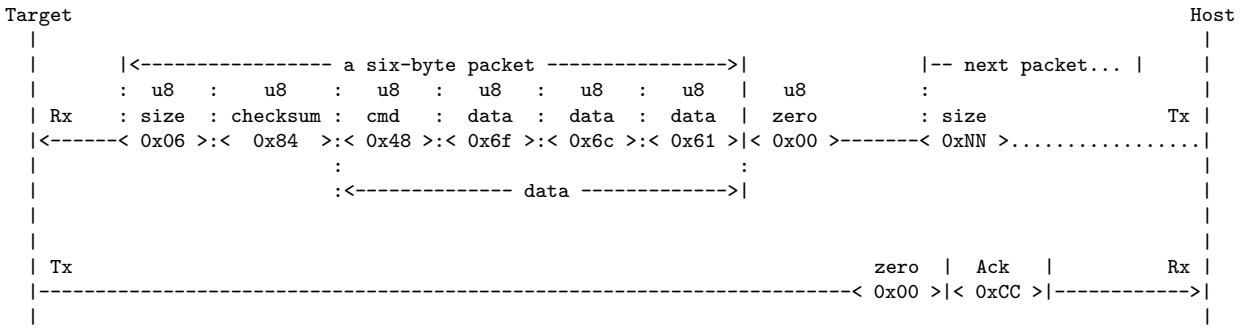
1 Serial Memory Loader

This paper describes the serial memory (in RAM) loader utility which loads the target (VM) memory without the need for proprietary USB connections between hosts and targets. The loader uses packets to provide reliable UART synchronous communications.

The UART device uses two wires (Tx and Rx) for UART communications. The baud rate is 9600 baud, and its serial format is fixed at 8 data bits, no parity, and a one-stop bit (8N1).

1.1 Basic Packet Format

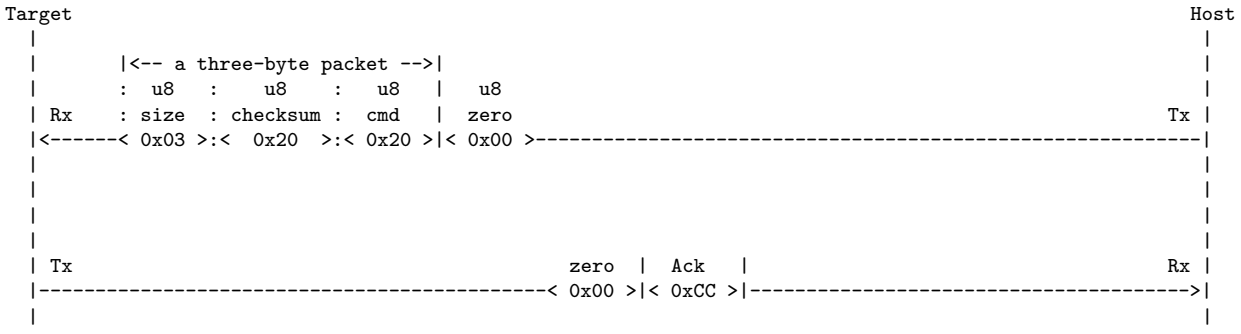
All serial communications are done via defined packets that are acknowledged (Ack) or not acknowledged (Nak) by the devices. The packets use the same format for receiving and sending packets. This includes the method used to acknowledge the successful or unsuccessful reception of a packet. The basic packet format is shown below:



The example above shows the Host transmitting a command to the Target (using a six-byte packet) while the bottom line is the response (Ack) from the Target to the Host. A checksum result is calculated from the data (0x48+0x6f+0x6c+0x61 = 0x184) and is truncated to 8 bits as 0x84. The next bytes to be sent are the 4 data bytes in this packet. The Host sends zeros until a non-zero response is received. The Target sends zeros until it is ready to Ack or Nak the packet that is being sent by the Host. Neither device (host or target) should transfer a non-zero byte until it has received a response after transmitting a packet.

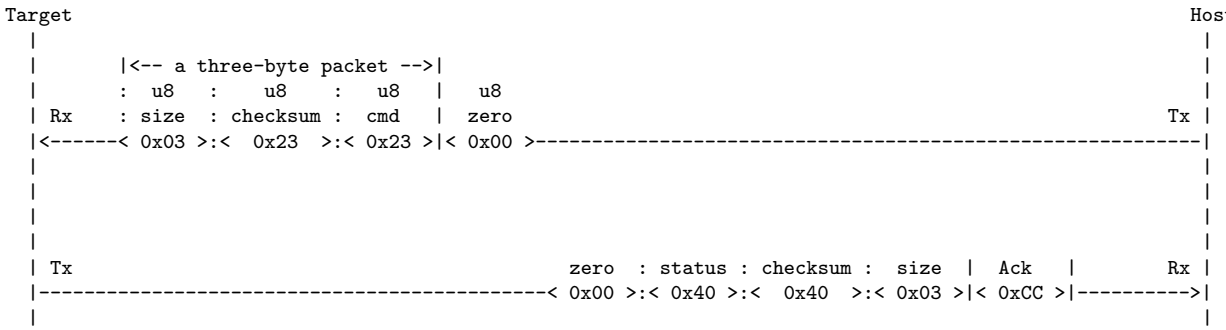
1.2 Example 1 - Host Sending a Ping Command to Target

This first example illustrates the Host sending a `Ping` command (0x20), in a three-byte packet, to the loader on Target. The first byte is the size, the checksum, and the ping command. Thereafter, the Host device should wait for the first non-zero data before looking for the Ack (0xCC) byte.



1.3 Example 2 - Host Sending a GetStatus Command to Target

The second example illustrates the Host sending a **GetStatus** command (0x23) to the loader on Target, which returns its status back to the Host. The first non-zero data received from the flash loader after the Ack (0xCC) is the first byte of the returned status packet. This status packet consists of the size, checksum, and status. If the status is 0x40, this indicates that the last command issued completed successfully. Any value other than 0x40 indicates that the previous command failed for the reason indicated by the data value returned. For a complete list of return codes, see the command GetStatus.



2 Command Definitions

This section defines the list of commands that can be sent to the serial loader (on Target). The first byte of the data in a packet should always be one of the defined commands, followed by data or parameters as determined by the command that is sent.

The following two definitions are the values used to indicate successful or unsuccessful transmission of a packet: Ack (0xCC) and Nak (0x33).

2.1 Ping Command

The `Ping` command (0x20) is a simple ping request to the serial loader on Target in a three-byte packet. The first byte is the size, the checksum, and the ping command. Since there is only one byte in the command, the checksum is simply equal to command code. The ping command has no real return status, the receipt of an Ack can be interpreted as a successful ping to the loader.

2.2 GetStatus Command

The `GetStatus` command (0x23) is the transmission of status request to the loader on Target, which returns its status back to the Host. The first non-zero data received from the flash loader after the Ack (0xCC) is the first byte of the returned status packet. This status packet consists of the size, checksum, and status. If the status is 0x40, this indicates that the last command issued completed successfully. Any value other than 0x40 indicates that the previous command failed for the reason indicated by the data value returned.
The following list is the possible status values that are returned from the serial memory loader when a GetStatus command is sent to the Host:

```
#define CommandReturn_Success     0x40
#define CommandReturn_UnknownCmd  0x41
#define CommandReturn_InvalidCmd  0x42
#define CommandReturn_InvalidAddr 0x43
#define CommandReturn_MemoryFail  0x44
```

2.3 Download Command

The `Download` command (0x21) is sent to the memory loader to indicate where to store data and how many bytes to send with the `SendData` commands that follow. The command consists of two 32-bit values that are both transferred the most significant bit (MSB) first. The first 32-bit value is the address to start programming data into, while the second is the 32-bit size of the data to send. This command also triggers an override of the full memory area to be stored, so this command takes longer than other commands. This results in a longer time to receive the Ack/Nak back from the Target. This command should be followed by a `GetStatus` command to ensure that the Program Address and Program Size parameters are valid for the device running the memory loader. The format of the packet to send this command is a follows:

```
Byte[0]  = 11
Byte[1]  = checksum(Bytes[2..10])
Byte[2]  = 0x21 (cmd)
Byte[3]  = Program Address [31:24]
Byte[4]  = Program Address [23:16]
Byte[5]  = Program Address [15:8]
Byte[6]  = Program Address [7:0]
Byte[7]  = Program Size [31:24]
Byte[8]  = Program Size [23:16]
Byte[9]  = Program Size [15:8]
Byte[10] = Program Size [7:0]
```

2.4 SendData Command

The `SendData` (0x24) command should only follow a `Download` command or another `SendData` command if more data is needed. Consecutive `SendData` commands automatically increment address and continue programming from the previous location. The caller should limit transfers of data to a maximum of 8 bytes of packet data to allow the memory to be loaded successfully and not overflow input buffers of the serial interfaces. The command terminates programming once the number of bytes indicated by the `Download` command has been received. Each time this function is called it should be followed by a `GetStatus` command to ensure that the data was successfully programmed into the memory. If the memory loader sends a Nak to this command, the memory loader does not increment the current address to retransmit the previous data.

```
Byte[0]  = 11
Byte[1]  = checksum(Bytes[2..10])
Byte[2]  = 0x24 (cmd)
Byte[3]  = Data[0]
Byte[4]  = Data[1]
Byte[5]  = Data[2]
Byte[6]  = Data[3]
Byte[7]  = Data[4]
Byte[8]  = Data[5]
Byte[9]  = Data[6]
Byte[10] = Data[7]
```

2.5 Run Command

The `Run` command (0x22) tells the memory loader to execute from the address passed as the parameter in this command. This command consists of a single 32-bit value that is interpreted as the address to execute. The 32-bit value is transmitted MSB first, and the memory loader sends an Ack signal back to the host device before actually executing the code at the given address. This allows the host to know that the command was received successfully, and the code is now running.

```
Byte[0] = 7
Byte[1] = checksum(Bytes[2..6])
Byte[2] = 0x22 (cmd)
Byte[3] = Execute Address[31:24]
Byte[4] = Execute Address[23:16]
Byte[5] = Execute Address[15:8]
Byte[6] = Execute Address[7:0]
```

2.6 Reset Command

The `Reset` command (0x25) tells the memory loader device to reset. This is useful when downloading a new executable image that overwrote the memory loader. This is useful when a new image has overwritten the memory loader and wants to start from a full reset. This allows the initial stack pointer to be read by the hardware and set up for the new code, unlike the run command. It can also reset the memory loader if a critical error occurs and the host device wants to restart communication with the memory loader.

```
Byte[0] = 3
Byte[1] = checksum(Byte[2])
Byte[2] = 0x25 (cmd)
```

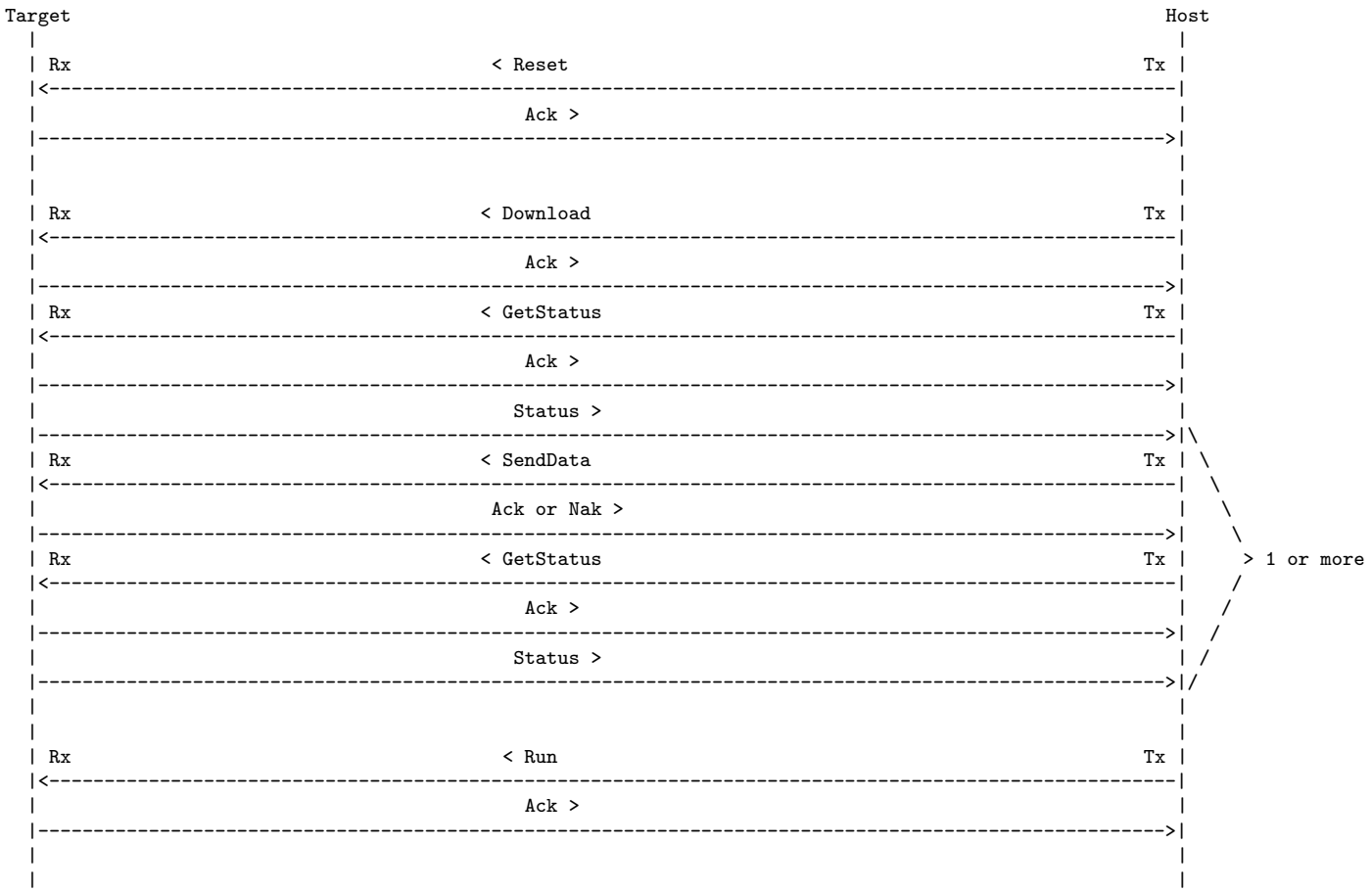
3 Loader Program Using a Typical Sequence of Commands

The section illustrates a typical serial memory loading from Host to Target.

A loader program must be developed on the host platform to send the commands from Host to Target using the Arduino Nano JTAG (USB/UART) connection. This typical loading sequence of serial communication commands (Reset/Download//Run) can be represented using the EBNF notation as follows:

```
SerialMemoryLoading = ResetCommand DownloadCommand RunCommand .
DownloadCommand     = GetStatusCommand { SendDataCommand GetStatusCommand }+ .

where { }+  means 1 or more
```



Reference

This serial memory loader has been inspired by the following application note from TI which was using packets to provide reliable UART synchronous communications:

Using the Stellaris Serial Flash Loader, Application Note AN01242, Texas Instruments, 2009.
https://www.ti.com/lit/an/spma029/spma029.pdf?ts=1606270750048