UDS Bootloader

1.1 UDS Bootloader Overview:

Embedded systems are one of the keys to making our world smarter. Small, dedicated processors provide the foundation for much of the information arising from infrastructure, vehicles, and more. One aspect of an embedded system creates a problem. Namely, the firmware of an individual ECU is difficult to update once a unit has been installed. Typical loading through headers and debug tools becomes nearly impossible once the unit becomes physically inaccessible. A bootloader solves this issue by providing a gateway to the hardware through established communication lines, such as RS-232, CAN, and Bluetooth.

The Simma Software CAN Bootloader provides a target-specific and compact solution for reprogramming ECUs efficiently and securely. Our bootloaders have been used worldwide for over fifteen years, beginning with J1939 and CAN based systems. Typically, the bootloader reserves a small portion of the ECU's internal memory for itself, leaving the remaining memory for the main application. The average code size of the bootloader package ranges from 2-4 KB of memory, depending on the needs of the customer and specifics of the ECU. If requested, our bootloaders can be secured via HMAC_SHA256 encryption, as well as support certain types of compression and digital signing of incoming hex and s-record files.

When the ECU comes out of reset, the bootloader code runs to determine if a valid application has been loaded. If an application has been detected, the bootloader will wait for a short period (100ms-2 seconds) and then jump to the application. The wait is implemented to allow for the user to overwrite an unresponsive application, if necessary. If no application is detected, the bootloader will continuously listen for the commands required to start the loading process.

1.2 FlashUDS:

The PC bootloader application, named flashUDS.exe, works with Simma Software VNA 232/USB/Bluetooth to send the new flash image across the UDS network. flashUDS allows the user to specify the port number for the adapter and the s-record or hex file to be used.

Example: flashUDS 2 app.s19

1.3 Programming Message Flow Overview:

Step 1: The first message sent by the PC tool instructs the application that is currently running to reset. The tool expects the application to send back an acknowledgement to indicate that a reset will occur, and then perform the reset within a certain amount of time, which is configurable based upon customer requirements. The purpose of the reset is to relaunch the bootloader for the remaining steps of the operation. If an application has to have been added to the ECU, the bootloader will acknowledge the command, but will not reset.

Step 2: The second message sent to the ECU is the start of a seed-key procedure, to ensure that the PC request is authorized to begin the bootloading process. Upon receiving the request, the target will send the seed information, which the PC tool will use to generate a key.

Step 3: The next command involves an unlock procedure to ensure that information is being transmitted correctly between the PC tool and the ECU. The tool sends a key to unlock the bootloader with the version information received in the previous communication. The bootloader checks to ensure that it is the expected resulting key; if it is, the bootloader will begin the new loading session proper.

Step 4: With the bootloader unlocked, the PC tool will instruct the bootloader to begin the process of erasing the application space. Once complete, the bootloader will respond, indicating the success/failure of the erase. If successful, the bootloader will prepare to receive the application data.

Step 5: (UDS) Next, the PC tool will send a data transfer request with the starting address and size of the data to be sent. If successfully received, the ECU will provide a positive response to begin the downloading process. The PC tool will then send the data as a "data transfer" service message type, as the size specified in the request. Once complete, the PC tool repeats this step until the app to be written has been fully transferred.

Step 6: The PC tool should now transmit a 'check download' message to ensure that the transmission was fully received. The bootloader will perform a checksum to confirm the data, and send an acknowledgement of the result.

1.4 UDS Packet Structure:

Service	Data

Service Requests:

Service ID equals 0x10: Diagnostic Session Control Request
Parm[1] equals 0x2: Programming Session
DLC: 2 bytes

Service ID equals 0x11: ECU Reset Request
Param[1] equals 0x1: Hard Reset
DLC: 2 bytes

Service ID equals 0x27: Security Access Request
Param[1] equals 0x1: Request Seed
DLC: 2 bytes

Param[1] equals 0x2: Send Key
DLC: 2 bytes

Service ID equals 0x31: Routine Control Request
Param[1] equals 0x1, Param [2] equals 0x0, Param[3] equals 0x5: Erase Flash
DLC: 4 bytes

Param[1] equals 0x1, Param [2] equals 0x0, Param[3] equals 0xA: Check Download
DLC: 4 bytes

Service ID equals 0x34: Download Request
Param[1]: Data Format Identifier (0 for no compression/encryption)
Param[2]: Number of bytes for data size/address (bits 7-4 size, bits 3-0 address)
Param[3-n]: Data transfer address (byte number n dependent on Param[2])
Param[n-m]: Data transfer size (byte number m dependent on Param[2])
DLC: m bytes

Service ID equals 0x36: Transfer Data Request
Param[1]: Sequence Number
Param[2-r]: Data (Number of bytes n determined by Download Request value)
DLC: r bytes

Service ID equals 0x37: Transfer Data Exit Request
DLC: 1 byte

Service Responses:

Service ID equals 0x50: Diagnostic Session Control Response
DLC: 1 byte

Service ID equals 0x51: ECU Reset Response
DLC: 1 byte

Service ID equals 0x67: Security Access Response
DLC: 1 byte

Service ID equals 0x71: Routine Control Response
DLC: 1 byte

Service ID equals 0x74: Request Download Response
DLC: 1 byte

Service ID equals 0x76: Transfer Data Response
DLC: 1 byte

Service ID equals 0x77: Transfer Data Exit Response
DLC: 1 byte

Service ID equals 0x7E: Tester Present Response
DLC: 1 byte