Firmware Loading Interface for Lassen SQ/iQ

Trimble

Revision A
October 2005

Corporate Office

Trimble Navigation Limited
749 North Mary Avenue
Post Office Box 3642
Sunnyvale, CA 94088-3642
U.S.A.
Phone: +1.408.481.7920
Toll-free: +1.800.865.4851
www.trimble.com

Support Offices

For support in Europe, call:
+44.1256.746.221
or send a fax to:
+44.1256.760.148
For support outside Europe, call:
+1.408.481.8786
or send a fax to:
+1.408.481.8011

Release Notice

The following limited warranties give you specific legal rights. You may have others, which vary from state/jurisdiction to state/jurisdiction.

Software and Firmware License, Limited Warranty

This Trimble software and/or firmware product (the "Software") is licensed and not sold. Its use is governed by the provisions of the applicable End User License Agreement ("EULA"), if any, included with the Software. In the absence of a separate EULA included with the Software providing different limited warranty terms, exclusions and limitations, the following terms and conditions shall apply. The Software (which includes all updates thereto) contains valuable trade secrets and is proprietary to Trimble and its suppliers. To the greatest extent permitted by law, such Software may not be modified, copied, disassembled, de-compiled or reverse engineered. Trimble reserves all other rights.

Trimble warrants that this Trimble Software product will substantially conform to Trimble's applicable published specifications for the Software for a period of ninety (90) days, starting from the date of delivery.

Warranty Remedies

Trimble's sole liability and your exclusive remedy under the warranties set forth above shall be, at Trimble's option, to repair or replace any Product or Software that fails to conform to such warranty ("Nonconforming Product") or refund the purchase price paid by you for any such Nonconforming Product, upon your return of any Nonconforming Product to Trimble in accordance with Trimble's standard return material authorization procedures.

Warranty Exclusions and Disclaimer

The foregoing Limited Warranty shall be applied only in the event and to the extent that: (i) the Products and Software are properly and correctly installed, configured, interfaced, maintained, stored, and operated in accordance with Trimble's relevant operator's manual and published specifications, and; (ii) the Products and Software are not modified or misused. The foregoing Limited Warranty shall not apply to and Trimble shall not be responsible for defects or performance problems resulting from (i) the combination or utilization of the Product or Software with products, information, data, systems, interfaces, services, or devices not made, supplied or specified by Trimble; (ii) the operation of the Product or Software under any specification other than, or in addition to, Trimble's standard published specifications for its products; (iii) the unauthorized installation, modification or use of the Product or Software; (iv) damage caused by accident, lightning or other electrical discharge, fresh or salt water immersion or spray; or (v) normal wear and tear on consumable parts (e.g., batteries).

TRIMBLE DOES NOT WARRANT OR GUARANTEE THE RESULTS OBTAINED THROUGH THE USE OF THE SOFTWARE.

THE FOREGOING TERMS OF THE LIMITED WARRANTY STATE TRIMBLE'S ENTIRE LIABILITY, AND YOUR EXCLUSIVE REMEDIES, RELATING TO USE AND PERFORMANCE OF THE PRODUCTS AND SOFTWARE. EXCEPT AS OTHERWISE EXPRESSLY PROVIDED HEREIN, THE PRODUCTS, SOFTWARE, AND ACCOMPANYING DOCUMENTATION AND MATERIALS, AND/OR ANY SOFTWARE OR FIRMWARE AND UPDATES THERETO ARE PROVIDED "AS-IS" AND WITHOUT EXPRESS OR IMPLIED WARRANTY OF ANY KIND BY EITHER TRIMBLE OR ANYONE WHO HAS BEEN INVOLVED IN ITS CREATION, PRODUCTION, INSTALLATION, OR DISTRIBUTION INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NONINFRINGEMENT. THE STATED EXPRESS WARRANTIES ARE IN LIEU OF ALL OBLIGATIONS OR LIABILITIES ON THE PART OF TRIMBLE ARISING OUT OF, OR IN CONNECTION WITH, ANY PRODUCTS OR SOFTWARE.

WITHOUT LIMITING THE GENERALITY OF THE FOREGOING:

> TRIMBLE IS NOT RESPONSIBLE FOR THE OPERATION OR FAILURE OF OPERATION OF GPS SATELLITES OR THE AVAILABILITY OF GPS SATELLITE SIGNALS.

THE SOFTWARE MAY CONTAIN TECHNOLOGY THAT IS NOT FAULT TOLERANT AND IS NOT DESIGNED, MANUFACTURED OR INTENDED FOR USE IN ENVIRONMENTS OR APPLICATIONS IN WHICH THE FAILURE OF THE SOFTWARE WOULD LEAD TO DEATH, PERSONAL INJURY OR SEVERE PHYSICAL OR ENVIRONMENTAL DAMAGE OR SEVERE FINANCIAL LOSS. ANY USE OR DISTRIBUTION BY YOU OR YOUR CUSTOMERS IN CONNECTION WITH ANY SUCH ENVIRONMENT OR APPLICATION SHALL BE AT YOUR AND YOUR CUSTOMERS' SOLE RISK, AND TRIMBLE SHALL HAVE NO LIABILITY WHATSOEVER IN CONNECTION THEREWITH. YOU SHALL INDEMNIFY AND HOLD TRIMBLE AND ITS SUPPLIERS HARMLESS FROM ANY CLAIM BROUGH AGAINST TRIMBLE WHICH ARISES FROM

> YOUR USE OR DISTRIBUTION OF THE SOFTWARE IN CONNECTION WITH SUCH ENVIRONMENTS OR APPLICATIONS.

SOME STATES AND JURISDICTIONS DO NOT ALLOW LIMITATIONS ON DURATION OR THE EXCLUSION OF AN IMPLIED WARRANTY, SO THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

Limitation of Liability

TRIMBLE'S ENTIRE LIABILITY UNDER ANY PROVISION HEREIN SHALL BE LIMITED TO THE GREATER OF THE AMOUNT PAID BY YOU FOR THE PRODUCT OR SOFTWARE LICENSE OR U.S.$25.00. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL TRIMBLE OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES WHATSOEVER UNDER ANY CIRCUMSTANCE OR LEGAL THEORY RELATING IN ANY WAY TO THE PRODUCTS, SOFTWARE AND ACCOMPANYING DOCUMENTATION AND MATERIALS, (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS), REGARDLESS WHETHER TRIMBLE HAS BEEN ADVISED OF THE POSSIBILITY OF ANY SUCH LOSS AND REGARDLESS OF THE COURSE OF DEALING WHICH DEVELOPS OR HAS DEVELOPED BETWEEN YOU AND TRIMBLE. BECAUSE SOME STATES AND JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

Table of Contents

Release Information

The following changes have been made to this document.

Date	Revision	Change
October 2005	A	First release.

1 Introduction

This document describes an interface for programming (loading) firmware into Component Technologies' Lassen SQ/iQ receivers. The interface can be used to develop a tool to upgrade firmware in the field. Sample source code of a tool for Microsoft® Windows is available to demonstrate implementation of the interface described in this document.

> ⓘ This document is applicable only to Lassen SQ/iQ GPS receivers. It is not relevant to other products.

2 Overview

2.1 Software Architecture

The FLASH memory chip of the GPS receiver is divided into several functional sections as shown in Figure 1. The Boot ROM section is loaded during production and cannot changed or erased. The User Data section is maintained by the application. The Application Firmware section holds the main software application, and can be erased and loaded with a newer version through the GPS receiver's serial port.

Byte Address	***Software Component / Section***
0xC00000 to 0xC03FFF	Boot ROM
0xC04000 to 0xC0FFFF	User Data
0xC10000 to 0xC5FFFF	Application Firmware

Figure 1. Functional Software Components and Memory Map

2.2 Boot Monitor

The boot monitor module is a part of the Boot ROM section. It provides facilities to perform checksum verification and RAM tests, and to read/write data from/to a specified location in RAM or FLASH, thus allowing to update the firmware.

The GPS receiver will enter the boot monitor mode if either of the following conditions occurs:

- A special "force to monitor" pin is shorted to ground when the main power is applied to the unit;
- Application firmware checksum verification failed at power-up;
- RAM test failed at power-up;
- A special protocol packet is issued by the user.

Once the system is in the monitor mode, a special Monitor protocol is used to communicate with the target. The necessary details about this protocol are presented in Section 4.

To return from the monitor to the normal GPS operating mode (i.e. execute the application firmware), either cycle the main power, or toggle the reset pin.

The default settings for the GPS receiver's serial port in the monitor mode are 9600 baud, 8 data bits, 1 stop bit, and no parity.

2.3 Demon Code Format

The demon code contains software to erase and program the flash memory. It is not resided in the Boot ROM. Therefore, after the receiver is switched to the monitor mode, the demon code is loaded to the RAM of the receiver. Such code is then executed in RAM as well.

The demon code is in the format of the ASCII in the Motorola S-record format. It is provided and defined as a char string (pRRDemon) in sq_demon.h. The utility function *LoadSrecStr()* provided with the iQFlashLoader tool (see Appendix A) shows how to parse the S-record file, temporarily store the loadable data into a memory buffer in the correct order, and send to the target for proper loading into the receiver.

2.4 Firmware ROM File Format

The firmware is distributed as an ASCII file in the Motorola S-record format. The Monitor protocol requires that the actual loadable raw data bytes be sent to the target to program into FLASH. The loadable data is expected to be sent in a sequential manner, in the order from the lowest to the highest loading address. Data will be programmed starting at the base address specified when initiating firmware loading (See Section 4.3.2). Therefore, the S-records from the firmware ROM file must be parsed, and the loadable data extracted prior to sending it to the target. Appendix A provides a reference to example source code that shows how to extract data from the S-record file and convert to a binary format.

Please, note that the S-records in the firmware ROM file may not be sequential in memory. In other words, the records are not sorted from the lowest to the highest loading address. Some records at the start of the file may have loading addresses higher than some records that follow. The demon protocol requires that the loadable data from the S-record file be sent in the order from the lowest to the highest loading address. The utility function *LoadSrecFile()* provided with the iQFlashLoader tool (see Appendix A) shows how to parse the S-record file, temporarily store the loadable data into a memory buffer in the correct order, and send to the target for proper loading into FLASH.

Failure to program firmware in the correct manner may result in unexpected behavior of the target system.

3 Firmware Loading Procedure

This section describes the procedure for loading firmware into the FLASH chip of the GPS receiver (referred to as "target" throughout this document).

The following pseudo-code shows the general sequence of steps. The details of each step are provided later in this section. Appendix A includes a reference to the sample C source code that shows how to implement this pseudo-code.

```
Load Firmware to Target:
{
    Convert demon code from S-record to binary;
    Convert firmware ROM file from S-record to binary;

    Establish connection to target using TSIP protocol:
        Set local serial port settings to 9600-8-Odd-1;
        Force target into Monitor mode;

    Set local serial port settings to 9600-8-None-1;

    Establish connection to target using Monitor mode protocol:
        Send hand-shaking packet ENQ;
        Wait for response packet ACK;
        If ACK packet not received:
            Exit or Power-cycle target and repeat from beginning;

    Send "Download Memory" packet 0x81 to load the demon code;

    Wait for response packet ACK;
    If ACK packet not received:
        Exit or Power-cycle target and repeat the last packet;


    Send "Run Demon Code At" packet 0x82 to force the receiver to
    run code at 0x800;

    Send "Change Baud" packet 0x86 to set remote port settings to
    57600-8-None-1;

    Set local serial port settings to 57600-8-None-1;

    Send "Erase Firmware" packet 0x8F;
    Wait for response packet ACK;
    If ACK packet not received:
        Exit or Power-cycle target and repeat from beginning;

    Send "Program Firmware" packet 0x89 with 224 data bytes at a
    time;
    Wait for response packet ACK;
    If ACK packet not received:
        Exit or Power-cycle target and repeat the last packet;

    When all data bytes are sent, power cycle to restart target;
}
```

The following provides details about the steps shown in the above pseudo-code for the firmware loading procedure.

Step 1. Convert demon string and firmware ROM file from S-record to binary.
As described in Section 2.3 and 2.4, the S-record demon and firmware ROM file must be converted to binary. Individual S-records may not be sorted by the loading address. However, when sending this data in the binary format, it must send in the order from the lowest to the highest loading address. Refer to Appendix A for an example function that shows how this is achieved.

Step 2. Establish a serial port connection to the target in the TSIP mode.
Communication with the target over its serial port must be established first. Change the local host's port settings to match those of the target. Refer to the GPS receiver's user manual for details.

> ⓘ In some cases, the target may enter the monitor mode automatically when power is applied. For example, if the previous firmware loading process has not been finished, the firmware checksum won't match, and the target will automatically start up in the monitor mode. In such cases, Step 2 will fail, and the loading procedure should continue at Step 4 as described below.

Step 3. Force the target into the monitor mode.
Assuming the receiver is running in the normal TSIP output mode, and the communication has been established, the following TSIP byte string (hex values) must be sent to the target to force it into the monitor mode:

```
10 1E 4D 10 03
```

Step 4. Establish a serial port connection to the target in the Monitor mode.
Once the target enters the monitor mode, it changes the GPS receiver's serial port settings to 9600 baud, 8 data bits, 1 stop bit, and no parity. To establish communication to the target in the monitor mode, the local host's settings must be changed to the same value, and the ENQ packet sent to the target. The target will respond with ACK to indicate the communication has been established. Refer to Section 4.3.1 for details on this packet.

Step 5. Load Demon.
As described in Section 2.3, the demon code needs to be loaded with packet 0x81 so that the receiver would be able to erase and program the flash while running code in RAM. Refer to Section 4.3.2 for details on the packets.

Step 6. Run Demon Code at.
This packet 0x82 is sent to force the receiver to run the demon code at the specified memory location. Refer to Section 4.3.3 for details on the packets.

Step 7. Establish a serial port connection to the target in the Demon mode.
Once the target enters the demon mode, the GPS receiver's serial port settings should switch to 57600 baud, 8 data bits, 1 stop bit, and no parity by packet 0x86. To establish communication to the target in the demon mode, the local host's settings must be changed to the same value, and the ENQ packet sent to the target. The target will respond with ACK to indicate the communication has been established. Refer to Section 4.3.1 and 4.3.4 for details on these packets.

Step 8. Erase FLASH sectors.
Before the firmware can be programmed, the firmware section in FLASH must be erased. The "Erase Firmware Section" packet 0x8F must be sent to the target. Refer to Section 0 for details on this packet.

Step 9. Send firmware data.
Once the flash is erased, the firmware data can be sent with packet 0x89. The target will respond with ACK to indicate the reception of the packet. Refer to Section 4.3.5 for details on this packet.

Error Recovery

The GPS receiver is designed in such way that the system will not be damaged during a firmware update. When there is an unexpected error while loading firmware, the target can always be restarted by cycling the main power. At power-up, the target will automatically enter the monitor mode if the firmware loading process has not completed successfully. In such a case, the host will be able to repeat the firmware loading procedure described above.

4 Monitor Interface Protocol

4.1 Protocol Format

The following packet structure is used by the monitor protocol:

BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTES 4 … N	BYTE N+1	BYTE N+2
STX 0x02	NULL_C 0x00	ID	LEN	DATA	CHKSM	ETX 0x03

Byte 0 – start of new packet (value: 0x02)
Byte 1 – delimiter byte (value: 0x00)
Byte 2 – packet ID
Byte 3 – size (in bytes) of packet data (DATA field only)
Bytes 4 … N – packet data
Byte N+1 – packet checksum [NOTE 1].
Byte N+2 – end of packet (value: 0x03)

Note 1: The checksum is computed as the sum of all bytes from the packet ID to the end of the packet data, i.e.:

CHKSM = (unsigned char)(ID + LEN + DATA[0] + … + DATA[N-1]);

4.2 Data Transmission

Data values are transmitted with the most significant byte of the value sent first. For example, transmitting a 4-byte memory address 0x004101F0 means sending byte 0x00 first, 0x41 second, 0x01 third, and 0xF0 last.

4.3 Packet Descriptions

4.3.1 ENQ, ACK, NAK

ENQ, ACK, and NAK are special bytes that are sent out without being formatted as described in Section 4.1.

The target responds to a formatted packet with either ACK (hex byte: 0x06) or NAK (hex byte: 0x15) unless specified otherwise. ACK indicates a successful operation. NAK indicates a failure in executing the command.

ENQ (hex byte: 0x05) provides a simple hand-shaking mechanism to verify that the target is alive and running in the monitor mode. The target sends ACK for every ENQ received.

4.3.2 Packet ID – 0x81 (Program Memory)

This packet programs the specified memory location with the provided data bytes. The target returns either ACK or NAK indicating the result of the operation.

BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4 to 7	BYTE 8 to (3+Size)	BYTE (4+Size)	BYTE (5+Size)
0x02	0x00	0x81	Size	Address	Data	CHKSUM	0x03

Parameter	Data Type	Description
Size	unsigned long	Size of loadable data in bytes. (number of data bytes + 4 bytes for Address)
Address	unsigned long	Starting physical address where data will be written.

4.3.3 Packet ID – 0x82 (Run Demon Code At)

This packet forces the target to run code at the specified memory location

BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 5 to 8	BYTE 9	BYTE 10
0x02	0x00	0x82	0x04	Address	CHKSUM	0x03

Parameter	Data Type	Description
Address	unsigned long	Starting physical address where data will be written.

4.3.4 Packet ID – 0x86 (Change Baud)

This packet configures the baud rate of the serial port. The target returns either ACK or NAK indicating the result of the operation.

BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6
0x02	0x00	0x86	0x01	Baud	CHKSUM	0x03

Parameter	Data Type	Description
Baud	unsigned char	0x0E – 115200 0x0D – 57600 0x0C – 38400 0x0B – 9600 Otherwise – Reserved

4.3.5 Packet ID – 0x89 (Program Firmware)

This packet programs the specified memory location with the provided data bytes. The target returns either ACK or NAK indicating the result of the operation.

BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4 to 7	BYTE 8 to (3+Size)	BYTE (4+Size)	BYTE (5+Size)
0x02	0x00	0x81	Size	Address	Data	CHKSUM	0x03

Parameter	Data Type	Description
Size	unsigned long	Size of loadable data in bytes. (number of data bytes + 4 bytes for Address)
Address	unsigned long	Starting physical address where data will be written.

4.3.6 Packet ID – 0x8F (Erase Firmware Section)

This packet initiates the erase operation on the target. It only erases the firmware portion of the FLASH chip. The target returns either ACK or NAK indicating the result of the operation.

BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5
0x02	0x00	0x8F	0x00	0x8F	0x03

5 Appendix A: iQFlashLoader Tool Reference Guide

5.1 Introduction

iQFlashLoader is a tool for Microsoft Windows that loads firmware into the FLASH chip of the GPS receiver over a serial RS-232 communication port on the host PC. The source code of the tool is documented to provide an example of how to develop a custom application to perform firmware updates. It shows how to use the Monitor protocol to implement the firmware loading procedure described in Section 3. It can be used, for example, to develop a program to update firmware remotely over a network connection.

iQFlashLoader has been created using the Microsoft Visual C++® development environment. It uses the MFC framework to implement the graphical user interface. While the compiled executable of the tool is provided together with the source code, Microsoft Visual C++ v6.0 or .NET is required to re-compile the source files and generate a fresh executable if desired.

5.2 File and Folder Structure

The iQFlashLoader tool directory contains the following 3 sub-directories:

- *bin* – contains the iQFlashLoader binary executable file;
- *mak* – contains the project files for Microsoft Visual C++ v6.0 and .NET development environments;
- *src* – contains the C++ source and header files

5.3 Source Code Reference

All source code files referenced in this section are located in the *src* directory of the iQFlashLoader tool distribution. The source files are fully commented throughout.

Parsing Demon ROM Character String (S-record) and Converting to Binary

Function *LoadSrecStr()* defined in Util.cpp shows how to parse the firmware S-record ROM character string (NULL-terminated), extract the loadable data, and store into a local buffer for sending to the target.

Parsing Firmware ROM File (S-record) and Converting to Binary

Function *LoadSrecFile()* defined in Util.cpp shows how to parse the firmware S-record ROM file, extract the loadable data, and store into a local buffer for sending to the target.

Creating Packets in the Monitor Protocol Format

Functions *GetXxxxxPkt()* defined in Util.cpp show how to format various packets using the monitor interface protocol described in Section 4.

Loading Firmware to the Target

Function *FlashProgrammingThread()* defined in FlashLoaderDlg.cpp shows how to implement the firmware loading procedure described in Section 3.

5.4 Compiling and Generating Executable

The iQFlashLoader tool can be re-compiled using the provided project make files.

If using Microsoft Visual C++ v6.0, open the workspace file *iQFlashLoader.dsw* located in the *mak* directory of the tool distribution. From the main menu, select *Build → Rebuild All*. This will compile the source files, generate the executable, and place it in the *bin* directory.

If using Microsoft Visual C++ .NET, open the solution file *iQFlashLoader.sln* located in the *mak* directory of the tool distribution. From the main menu, *select Build → Rebuild Solution*. This will compile the source files, generate the executable, and place it in the *bin* directory.