Firmware Update Protocol

VectorNav Encrypted Bootloader

Application Note

Abstract

This document describes the protocol used by the VectorNav Encrypted Bootloader used in the VN-100 hardware versions 5 and above, as well as all releases of the VN-200. Example programming utilities are provided in C, C#, and Python.

Document Information	
Title	Firmware Update Protocol
Subtitle	VectorNav Encrypted Bootloader
Document Type	Application Note
Document Number	AN013
Document Status	Released

Contents

1 Introduction

This document describes the protocol used by the VectorNav Encrypted Bootloader used in the VN-100 hardware versions 5 and above, as well as all releases of the VN-200. Example programming utilities are provided in C, C#, and Python.

The goal of this application note and the accompanying source code is to present the general workings of the firmware upgrade protocol to the extent required for implementation. The intended audience is the portion of customers developing a custom programming tool which integrates into their own operating environment. For the remainder of customers, the Firmware Update Tool provided by VectorNav is sufficient for performing firmware upgrades on a Windows PC.

2 System Overview

The current bootloader versions support firmware updates only on the primary sensor UART. Support for firmware updates on the secondary UART and SPI bus is planned for future releases.

Unlike previous hardware revisions which used a different firmware update method, the calibration parameters remain intact on the sensor and do not need to be saved and restored by the user as part of the update process.

There are four components which are required for the firmware update process. The following sections describe the components in further detail.

2.1 Firmware

The firmware is the program code on the VectorNav sensor which is run under normal conditions. This can be updated in the field to add new features or to fix bugs. The firmware is protected by an internal checksum which the bootloader uses to determine if the firmware is valid before the firmware is executed.

2.2 Bootloader

The bootloader is the code on the VectorNav sensor which provides the ability to update the firmware and other parameters as well as monitor the integrity of the firmware. The bootloader is run immediately on any reset or power cycle. If a valid firmware is found in the sensor's memory, the bootloader will run the firmware. If no firmware is found, or the firmware is incomplete or corrupt, the bootloader will enter its detection phase automatically (described

below) and wait for a programmer to connect to it for firmware update. During normal operation, the bootloader can be entered by sending the Firmware Update command.

2.3 Programmer

The programmer is the hardware/software combination which communicates with the bootloader to provide commands and data necessary to perform a firmware update. For the rest of the document, we will consider only the software side of the programmer, and the term programmer will be used to reference the client code which talks to the bootloader.

2.4 Encrypted Firmware File

The encrypted firmware file, denoted by a VNX file extension, is an encrypted representation of a specific version of firmware. Firmware updates are distributed by VectorNav as VNX files.

The VectorNav VNX file format is similar to the standard Intel HEX format, with extensions for encryption and program validation. Each line is called a "record," and each record consists of various internal field data. An understanding of the different fields and their meaning is not required when performing a firmware update, so they are not discussed here.

3 Firmware Update Process

In terms of the above components, the general firmware update process consists of relaying the encrypted program data contained in the VNX file to the bootloader through the programmer, updating the firmware in the sensor memory.

The firmware update process consists of three steps, discussed in the following sections.

3.1 Entering the Bootloader

Since the sensor will normally run the firmware on startup, the bootloader must be entered explicitly. This is done by sending the Firmware Update Command, VNFWU, in the standard VectorNav VN-100/200 ASCII protocol. The command string should be sent with the baudrate and checksum settings that are currently configured for the sensor. Correct commands for 8-bit checksum and 16-bit CRC are:

$VNFWU*5C
$VNFWU*2604

Upon successful receipt of the firmware update command, the sensor will echo back a confirmation of the command and then immediately reboot into the bootloader detection phase. After sending the Firmware Update Command, a one second delay is recommended to allow the firmware to receive and process the command and reboot into the bootloader before continuing.

Note: If a previous firmware update attempt failed or was incomplete, the bootloader may instead start up immediately in the detection phase instead of running the firmware. In this case, sending the firmware update command described above should be skipped, as sending it will interfere with the bootloader's baudrate detection.

During the bootloader's detection phase, the sensor waits for a sequence of 8 or more "space" characters (ASCII 0x20) which are used for automatic baud rate detection. More space characters are allowed, but do not send non-space characters during this phase. The bootloader can use a different baudrate than the firmware, and will communicate using the baudrate at which the space characters are sent. The bootloader supports the same baudrates as the main firmware (between 9600 and 921600). Baudrates of 115200 or lower are recommended in any environment where communication may be unreliable.

Upon receipt of the space characters and successful baudrate detection, the bootloader will respond with an identification string representing the version of the bootloader. As extra space characters will be discarded by the bootloader, the programmer can send space characters one at a time until the identification string is received, or send a sufficient number of space characters (16-20 recommended) and then check for the identification string. An example identification string is presented below, and will always consist of the same format, although the four version number identifiers can be any value from 0 to 255:

VectorNav Bootloader v.1.2.1.4

Once the identification string has been transmitted, the bootloader is now in programming mode and awaiting commands. If no valid commands are received for a timeout period of 10 seconds, the bootloader will reset the sensor, causing the main firmware to run if it is valid and present.

3.2 Programming the Firmware

Programming the firmware consists of sending VNX records one at a time and waiting for an acknowledgement. Each record is sent over, one at a time, as the payload of a Bootloader Command, VNBLD, in the standard VectorNav ASCII protocol. The leading colon character that exists for each line in the VNX file is not considered part of the data payload and should not be included in the command.

Like the main sensor firmware, commands to the bootloader must begin with '$' and terminated with a carriage return (CR) or linefeed (LF), and responses from the bootloader are terminated with a LF. The command must use a 16-bit CRC to validate all messages. See the VN-100 User Manual for more details.

Here is an example record from a line in a VNX file and the corresponding command string:

:020000040802F0

$VNBLD,020000040802F0*B846

After each line is sent and processed, the sensor will respond with its own VNBLD message containing a success or error code. The programmer must wait and receive this response before sending the next record. During programming of certain areas of the flash, the delay in the bootloader between command and response may be as long as 6 seconds, so any communication timeout settings need to be configured to handle this accordingly. For reference, an example success response to a command is:

$VNBLD,00*BBD4

The following table enumerates the possible error codes and appropriate response actions.

Error	Type	Details
0	NoError	Success, send next record
1	InvalidCommand	Problem with VNX record, abort
2	InvalidRecordType	Problem with VNX record, abort
3	InvalidByteCount	Problem with VNX record, abort
4	InvalidMemoryAddress	Problem with VNX record, abort
5	CommError	Checksum error, resend record
6	InvalidHexFile	Problem with VNX file, abort
7	DecryptionError	Invalid VNX file or records sent out of order, abort
8	InvalidBlockCRC	Data verification failed, abort
9	InvalidProgramCRC	Problem with firmware on device
10	InvalidProgramSize	Problem with firmware on device
11	MaxRetryCount	Too many errors, bootloader aborts
12	Timeout	Timeout expired, bootloader resets
13	Reserved	Contact VectorNav, abort

Note that the only error which can be recovered from during the programming process is the CommError, which signifies a communication error between the programmer and the bootloader. In this case, try sending the record again. Most other errors are caused by a corrupt or incorrect VNX file or by the programmer not sending the records over in the correct format and order. The InvalidProgramCRC and InvalidProgramSize errors signal that the firmware onboard the device is invalid, and will not be executed at next boot.

3.3 Exiting the Bootloader

There are two ways to exit the bootloader after programming is complete, or after encountering an error. First, as mentioned before, if no valid commands are received in a 10 second period, the bootloader will automatically exit and reset, running a valid firmware if one is present. Therefore, the bootloader will eventually exit and reset at the end of programming if no further action is taken.

The recommended method is to issue a standard Reset Command, VNRST, to force an immediate reset. The reset command should only be sent before or after the programming process. If the reset command is issued in the middle of a firmware update, it may leave the firmware in an invalid state and the sensor will not operate normally until the next successful firmware update.

$VNRST*6542

In either case, once the bootloader exits and performs a reset, it will run the firmware if it is considered valid. If no valid firmware is found, the bootloader will once again enter its detection phase.

4 Example Implementation

Example programs are provided to demonstrate the implementation of basic command-line firmware update utilities. Arguments supplied to the programs are the serial port, baud rate, and firmware (VNX) file. Run the utilities at the command line with no arguments for example arguments. In the interest of illustrating the basic program logic as clearly as possible, minimal error checking has been implemented. Additionally, the program assumes it is run while a valid firmware is running. The same baudrate is used in both the firmware and the bootloader.

Implementation source code is provided in the Python, C#, and C programming languages. Microsoft Visual Studio 2010 projects are provided for the C and C# versions and were tested on Windows 7. The Python script was tested on Windows with Python 2.7.2 and PySerial 2.5. The various source codes may work with other compilers and operating systems with little to no modification, but that has not been tested and is beyond the scope of this Application Note.

5 Summary

This concludes the description of the firmware update protocol.

Please check the Application Note section of http://www.vectornav.com/support to download this Application Note and the accompanying software. Contact support@vectornav.com with any questions.