Programming Guide

(Firmware Updates)

Pinnacle 100

Version 1.0

REVISION HISTORY

Version	Date	Notes	Contributors	Approver
1.0	24 Apr 20	Initial Release	Jamie Mccrae	Jonathan Kaye

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

CONTENTS

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

1 HARDWARE SETUP

1.1 Pinnacle 100 Development Board

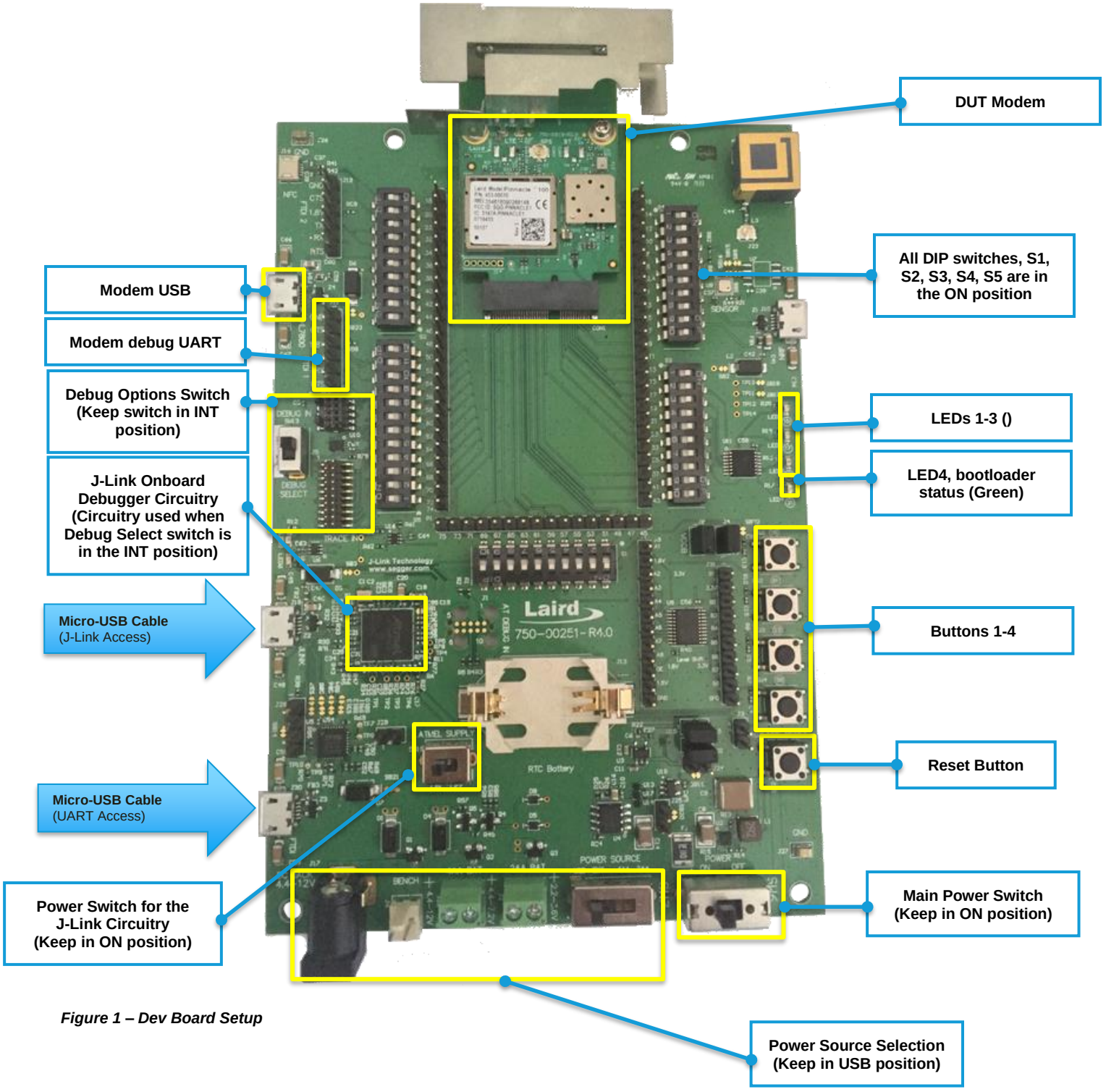
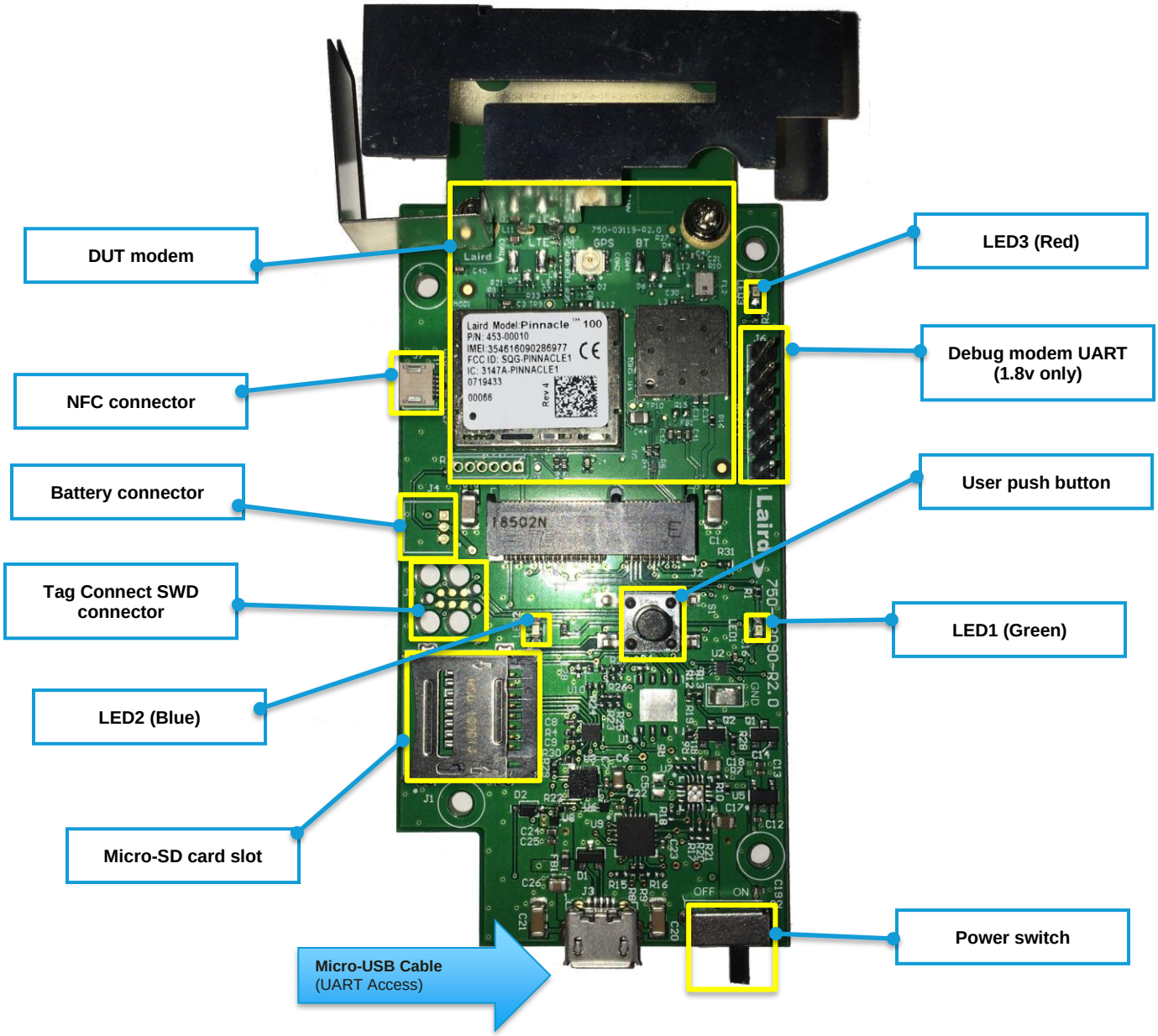


Figure 1 – Dev Board Setup

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

1.2 MG100 Micro Gateway Board



Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

1.3 Memory Map

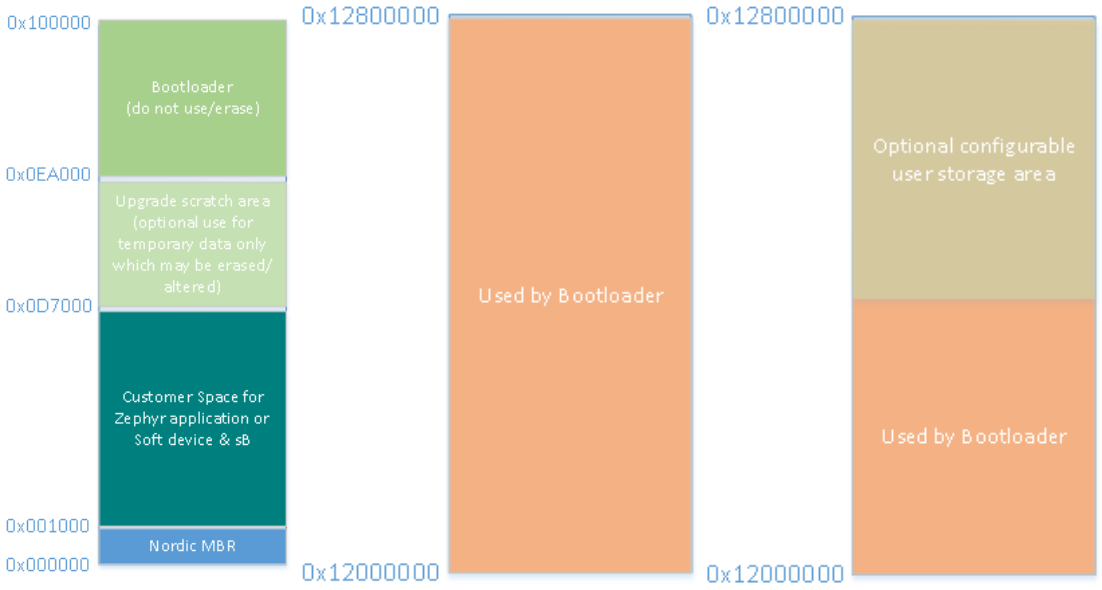
The Pinnacle 100 has a Nordic nRF52840 Cortex M4 MCU which stores the bootloader code and user application. Some of the space in this processor is reserved and cannot be used by the user application, which is as follows:

Table 1: Memory map

Start	End	Size	Purpose	For Customer Use	Description
0x00000	0x000FFF	0x1000	Nordic MBR (Master boot record)	x	Nordic library for setting up the module and loading the bootloader.
0x1000	0x0D6FFF	0xD6000	User application	✔	This is where your application goes.
0xD7000	0x0E9FFF	0x13000	Bootloader scratch area	✔ / x	The bootloader may (at any time) use this space which erases or alters its contents. User applications can therefore use this as temporary storage if required which, if erased, does not impact the user application. Please ensure you use a header and checksum to ensure that the data you store in the area is valid.
0xEA000	0xFFFFFF	0x16000	Bootloader	x	This contains the bootloader.
0x12000000	0x123FFFFF	0x400000	Bootloader Storage (via QSPI)	x	This area maps to the first half of the QSPI chip (once configured correctly) and is used for storing bootloader images. It can be used (via the included API) to store update images only
0x12400000	0x127FFFFF	0x400000	Optional user storage (via QSPI)	✔ / x	This area maps to the second half of the QSPI chip (once configured correctly) and can be used for user storage. Please note that this must be configured via the bootloader before using it

Total size of nRF52840 – 0x100000 (1 MB)
Total size of overheads (including MBR, bootloader and scratch area) – 0x02A000 (168 KB), 16%
Total size of space available for user application: 0x0D6000 (856 KB), 84%
Total size of QSPI: 0x800000 (8 MB)
Default size of QSPI reserved for bootloader: 0x800000 (8 MB), 100%
Default size of QSPI reserved for user storage: 0x0 (0 MB), 0%
Configurable minimum size of QSPI reserved for bootloader: 0x400000 (4 MB), 50%
Configurable maximum size of QSPI reserved for bootloader: 0x800000 (8 MB), 100%
Configurable minimum size of QSPI reserved for storage: 0x0 (0 MB), 0%
Configurable maximum size of QSPI reserved for user storage: 0x400000 (4 MB), 50%

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610



Note: Use of the bootloader and bootloader features is optional. If the bootloader is not needed, you can erase it from the module. However, doing so clears the license from the module and, should the bootloader be reloaded, it will not function until a license key is obtained from Laird Connectivity.

1.4 Upgrade Types

The Pinnacle 100 comes pre-loaded with a bootloader that supports secure firmware upgrades. It is designed for use in end-user applications with its own update system. It is not recommended that the secure bootloader update mechanism during development of firmware or applications due to the overhead of using the bootloader and increased time of programming when new firmware is available for testing for development purposes and leave using this mechanism until the testing stage of product design.

Table 2 gives an overview of the supported firmware upgrade methods and the feature set of them.

Table 2: Firmware upgrade methods

	UwFlashX	SWD (nRF52840 flash)	SWD (QSPI flash)
Use during application development	x	✔	x
Use during application testing	✔	✔	✔
Use during manufacturing	✔	✔	✔
Use for field upgrades	✔	Not recommended (does not work if readback protection is enabled)	
Required hardware	UART	SWD (J-Link)	
Transfer speed	Medium	Fast	Fast
Upgrade method	Bootloader	Direct	Bootloader
Upgrade time	Slow	Instant	Slow
Warnings	None	Can delete bootloader or license data	Can delete bootloader or license data
Transfer security	Supports UART unlock code	None	None
Downgrade prevention	✔	x	✔
Requires signed image	✔	x	✔
Image signing checks	✔	x	✔

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

	UwFlashX	SWD (nRF52840 flash)	SWD (QSPI flash)
Works with readback protection enabled	✔	x	x
Can update cellular firmware	✔	x	✔
Erases all existing QSPI data	✔	Potentially	✔
OS Support	Windows/Linux (x86 and ARM)/mac		
Utility description	Transfers full .uwf or .ubu update image to QSPI via UART and lets bootloader handle update process	Directly writes file data into nRF52840 or QSPI memory	Transfers full .uwf or .ubu update image to QSPI via SWD and lets bootloader handle update process
Use description	Recommended for use during production for setting up modules	Recommended for use during development only	Recommended for use during production only

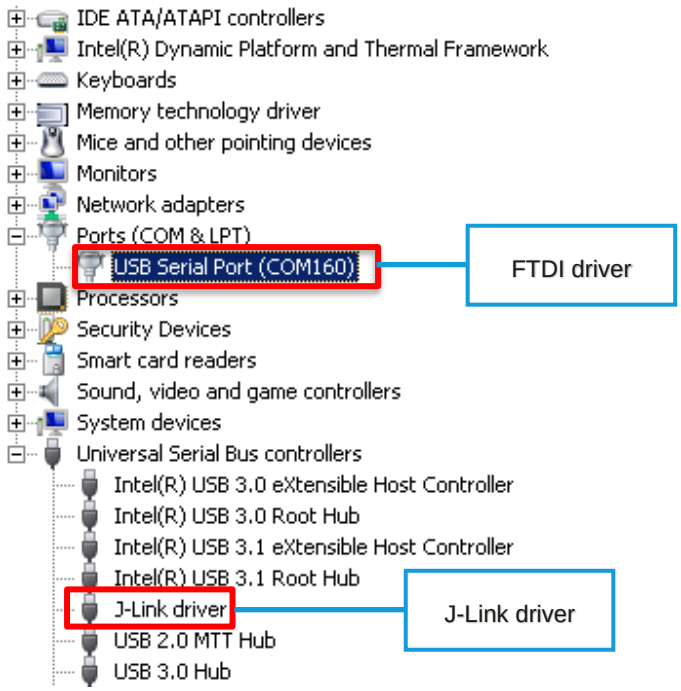
Note: Segger J-Link supports CLI programming operation only using nrfjprog. The Pinnacle 100 development board has a J-Link OB which allows for debugging and testing applications on the module present on the development board only. For further details, see the Segger website: https://www.segger.com/products/debug-probes/j-link/

For ease of development, we recommend SWD when developing your application. It can also be used in production to flash the required image to the modules. We recommend that you then disable SWD during mass production (after the flash programming stage) to prevent readback of code or malicious code injection. This can be performed by enabling readback protection directly via SWD or by downloaded a special configuration file to QSPI (whereby the bootloader enables readback protection) or via enabling it by using the UART bootloader interface.

1.5 Driver Verification

For driver verification, follow these steps:

1. Verify that the driver for the FTDI virtual serial port or Segger J-Link is installed from device manager.
2. For FTDI: Expand *Ports (COM & LPT)* and ensure that you see the FTDI device as *USB serial device*.
   For Segger J-Link: Expand *Universal Serial Bus Controllers* and ensure that you see the J-Link device as *J-Link driver*.



Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

Figure 2 – Driver identifification using Device Manager

2 SOFTWARE

2.1 FTDI UART Drivers

To download and install the FTDI UART drivers, follow these steps:

1. If UART access is required and drivers are not installed, visit the FTDI website: https://www.ftdichip.com/Drivers/VCP.htm and download the drivers for your operating system and architecture.
2. Once downloaded, run the installer and any attached FTDI devices should be automatically detected by the installer.

   Once installed, the FTDI ports can be used like they were a serial port from any supported applications such as UwTerminalX, available to download from: https://github.com/LairdCP/UwTerminalX

2.2 J-Link 'Segger' Drivers

To download and install the Segger J-Link drivers, follow these steps (**note that V6.62b or newer is mandatory/required**):

1. If Segger J-Link drivers are not installed or are outdated then visit the Segger download site: https://www.segger.com/downloads/jlink/ and download the J-Link Software and Documentation Pack for your operating system and architecture. At the time this document was written, the latest version was V6.64a.

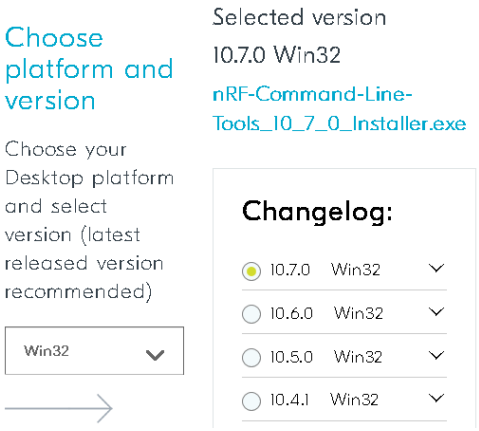
Figure 3 – J-Link driver package

2. Once downloaded, launch the installer which installs the drivers to your system and the corresponding Segger applications to your computer.

2.3 Nordic nRF Command Line Tools

To download and install the latest Nordic nRF command line tools, follow these steps (**note that version 10.7.0 or newer is mandatory/required**):

1. Download the latest Nordic nRF command line tools from https://www.nordicsemi.com/Software-and-Tools/Development-Tools/nRF-Command-Line-Tools for your operating system and architecture. At the time this document was written, the latest version was 10.7.0.



2. Once downloaded, launch the installer which installs the utilities to your system.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

2.4 UwTerminalX

UwTerminalX is a cross-platform utility for communicating with Laird Connectivity's modules via UART. To download and install the latest version, follow these steps:

1. Download the latest version from https://github.com/LairdCP/UwTerminalX/releases for your operating system and architecture.
2. If you are using Windows and have downloaded the SSL version, ensure you follow the instructions on the releases page for installing the visual studio 2015 redistributable

   If you are using Linux, ensure you follow the instructions available on the main Github project page.

2.5 UwFlashX

UwFlashX is a cross-platform utility for transferring upgrade images to the Pinnacle 100 module (and other Laird Connectivity modules) modules via UART, to download and install the latest version, follow these steps:

1. Download the latest version from https://github.com/LairdCP/UwFlashX/releases for your operating system and architecture.
2. If you are using Windows and have downloaded the SSL version, ensure you follow the instructions on the releases page for installing the visual studio 2015 redistributable.

   If you are using Linux, ensure you follow the instructions available on the main Github project page.

2.6 UBUtil

UBUtil is the Laird Connectivity Universal Bootloader Utility which is used to generate firmware upgrade packages for the Pinnacle 100 module. It is cross-platform and is required if using the Pinnacle 100 bootloader.

To download and install the latest version, follow these steps:

1. Download the latest version from the Laird Connectivity Pinnacle 100 website https://www.lairdconnect.com/wireless-modules/cellular-solutions/pinnacle-100-cellular-modem for your operating system and architecture

3 PROCESS

There are different methods of updating firmware on the Pinnacle 100 module as described in Table 2 located in *Upgrade Types*. We recommend using SWD to upload direct to nRF52840 flash during development as it does not require a signed image and is instantly executed once uploaded; unlike using the bootloader which requires time for the bootloader to verify and copy the image into place.

3.1 Direct to nRF52840 Flash Method

To begin, you need an application built using your desired toolchain in hex format. Please ensure that the application starts at 0x1000 and does not end at an address greater than 0xE9000. Refer to Table 1 in the *Memory Map* section for information on the memory map.

3.1.1 Flashing Using CLI J-Link Software

To flash using CLI J-Link software, follow these steps:

1. Open a terminal or console in the directory in which the output hex file resides.
2. Ensure that the Segger utilities are in your path. If they are not, add them.
3. Flash the hex file to the module and begin execution using the following command:

   `nrfjprog -f NRF52 --program <file.hex> --sectorerase --reset`

   The application outputs the progress of downloading the application to the module and resets the Pinnacle 100 after it is programmed.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

3.2 Bootloader Package Update

To update using the bootloader, a signed firmware package is required. This is the preferred method for once your application is stable and for mass production. The *Generating Ubu Firmware Updates* section and onwards describes the process of generating and using ubu firmware update packages.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

4 DEBUGGING

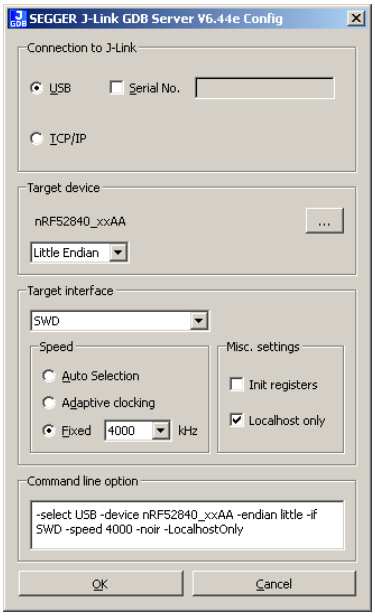
This presents a simple method of using GDB to debug an application on the Pinnacle 100 using SWD via the on-board Segger JLink.

Note that the on-board JLink differs from the full JLink units in the following ways:

- It is limited to six hardware breakpoints and zero software breakpoints (a trial is offered to test software breakpoints).
- It cannot be used with external modules and is only for use with the module on the development board.
- It has a maximum interface speed of 3.2 MHz (JLink base supports up to 15 MHz).
- It supports a maximum SWO speed of 12 MHz (JLink base supports up to 30 MHz).
- It supports a maximum download speed of 225 KBps (JLink base supports up to 1 MBps).

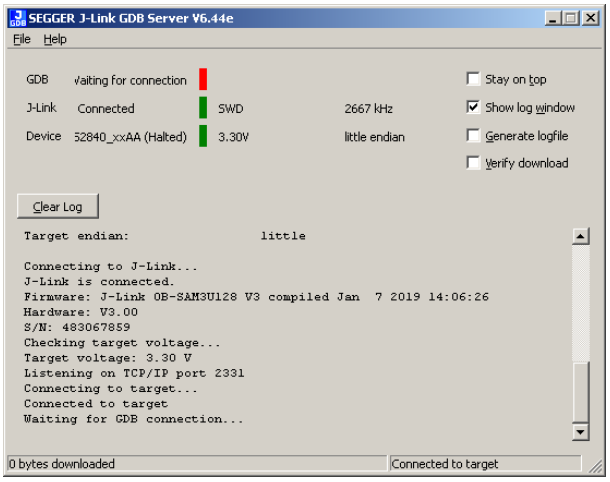
This process requires your application to be built with debugging enabled and have symbol information left in the executable (i.e. not stripped). Please refer to your development environment for details on how to do this. Once the debug output is generated, follow these steps:

1. Launch the Segger GDB server application which can be found in the Windows start menu.
2. Ensure that the connection to J-Link is set to USB.
3. Select the **…** button next to the target device.
4. Select the Nordic nRF52840_xxAA device and click **OK**.
5. Ensure that it is in little endian mode.
6. Set the target interface to *SWD.*
7. Change the speed to fixed and set it to 4000 KHz.



8. Click **OK** to start the GDB server and to begin the session.

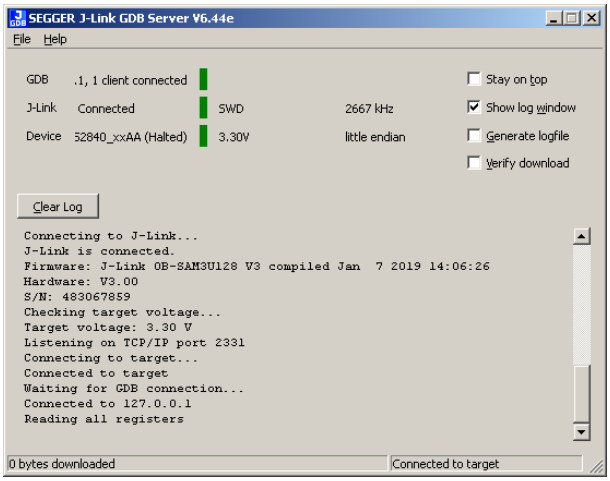
Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

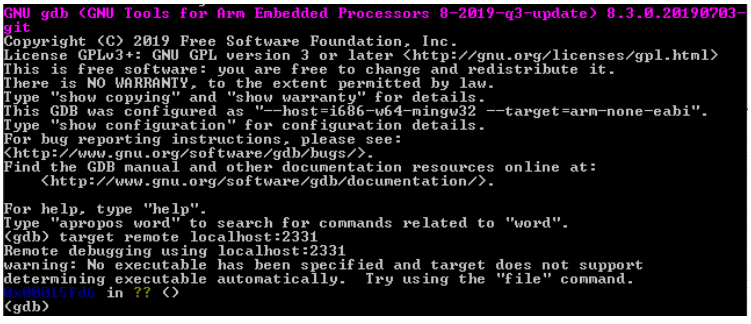


9. In the directory with your compiled application, launch a terminal or console.
10. Ensure that GCC utilities are in your path.
11. Run GDB for the target platform, arm-none-eabi-gdb.
12. Connect to the Segger JLink GDB server by entering the following command:

    `target remote localhost:2331`

13. Press enter which should then connect the debugger.





14. If you have not yet loaded your application, load it to the module using the load command. For Zephyr applications, the file to load is named *zephyr.hex*: load *zephyr.hex*
15. Load the file with debug symbols. For zephyr applications this is *zephyr.elf*, using the file command: *file zephyr.elf*.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

16. Press enter to load the details of the application. Debugging can now begin.

    For example, to set a breakpoint on a function, use the command **b** with the function name (e.g., b main)

17. Restart the module by using the following command: **`mon reset`**
18. Begin execution of the application using the continue command. Once the breakpoint is encountered, program execution stops and GDB commands can be used to query registers and other data on the processor.

```
(gdb) target remote localhost:2331
Remote debugging using localhost:2331
warning: No executable has been specified and target does not support
determining executable automatically.  Try using the "file" command.
0x00015fd6 in ?? ()
(gdb) file zephyr.elf
A program is being debugged already.
Are you sure you want to change the file? (y or n) y
Reading symbols from zephyr.elf...
(gdb) b main
Breakpoint 1 at 0x15c0: file ../src/main.c, line 65.
(gdb) mon reset
Resetting target
(gdb) continue
Continuing.

Breakpoint 1, main () at ../src/main.c:65
warning: Source file is more recent than executable.
65              LOG_INF("Pinnacle 100 LTE Console %d.%d.%d\r\n"
(gdb)
```

5 APPLICATIONS

Laird Connectivity provides the following two pre-built applications for use with the Pinnacle 100:

- **Zephyr out of box demo** – This application can be used to evaluate the Pinnacle 100 working with AWS. It simulates an end device sensor node which reads data from a Laird BL654-based BME280 sensor (via Bluetooth) and sends this data securely to the cloud (using Cat-M1 internet) to AWS which can then be viewed in a web browser or stored for analysis. A pre-compiled application is available for testing and the source code is available for a demonstration of how to create a Zephyr application for the Pinnacle 100. This is provided as an example application only, the source and pre-compiled application is available for download from https://github.com/LairdCP/Pinnacle_100_oob_demo
- **Hosted mode firmware** – This firmware includes *smart*BASIC and the AT interface application. It can be used by a host processor by sending AT commands over a UART to interact with the Cat-M1/NB-IoT modem, it also supports Bluetooth connectivity and GPIO/I2C/SPI/NFC functionality. It is provided as a firmware package which programs everything required to the module. There is an instruction guide available from the Laird Connectivity Pinnacle 100 website which explains how to install the hosted mode firmware image on https://www.lairdconnect.com/wireless-modules/cellular-solutions/pinnacle-100-cellular-modem

---

Note: To switch from this firmware back to Zephyr development requires a full erase from the bootloader. This is a production-grade firmware which is designed for use on systems with a separate processor controlling the system. For details on performing a full erase, see the Restoring to Factory Defaults (via UART) or Full-Chip Erase/Recovery (via SWD) sections.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

6 GENERATING UBU FIRMWARE UPDATES

UBU firmware update files are generated using the UBUtil application. This application takes in hex or binary files, a private key, and command line arguments which generates a single .ubu file which can update multiple images on a module. Ubu files can then be transferred to modules via UART (using UwFlashX) or SWD (using nrfjprog).

6.1 Generating a Private Key

A private key is required to sign firmware images, for security purposes all projects should have unique private keys (and to enhance the security further, albeit outside the scope of this guide, would be to have unique private keys per module) and these keys should be stored securely on a server or system, ideally one that has no internet access.

The UBUtil application can be used to generate a private key by using the **--create-key** argument followed by the output filename which stores the key. For example, using this command:

```
UBUtil --create-key Blinky.pem
```

An example output of this command is as follows:

```
Laird Connectivity Universal Bootloader Firmware Update Generation Utility
  v1.0
    Built Apr 24 2020

Successfully generated signing key file Blinky.pem.
Public Key (hex):
d24deb7ab6c3922d1fd561d0917304c00ece98a4b8bad9b0bd09ed4647edeb0eb5f938693a03ed52ac3f716b0b5
00cfc386bf8a77f4f49ff3b8b4eb4a4ca0470
```

6.2 Outputting a Private or Public Key

UBUtil can be used to output the private key or public key from a generated .pem file, to do this the --file-info argument is used along with --file-key to specify the type of key (0 will show the private key and 1 will show the public key).

To show the private key, use:

```
UBUtil --file-info Blinky.pem --file-key 0
```

```
Laird Connectivity Universal Bootloader Firmware Update Generation Utility
  v1.0
    Built Apr 24 2020

Private Key: 46cabe03a97c225ca5da8442d2cfc8191d9486c01e7143788d0924aea7ce1356
```

To show the public key, use:

```
UBUtil --file-info Blinky.pem --file-key 1
```

```
Laird Connectivity Universal Bootloader Firmware Update Generation Utility
  v1.0
    Built Apr 24 2020

Public Key:
d24deb7ab6c3922d1fd561d0917304c00ece98a4b8bad9b0bd09ed4647edeb0eb5f938693a03ed52ac3f716b0b5
00cfc386bf8a77f4f49ff3b8b4eb4a4ca0470
```

The public key can be loaded into the bootloader on a module so that it accepts the signed firmware updates you generate.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

6.3 Generating a Firmware Update Package

To generate a firmware update package, an application is required which runs on the Pinnacle 100 module. Creating such an application is outside the scope of this guide.

Once an application is ready and a private key has been generated as detailed in the Generating a Private Key step, then a firmware update package can be generated. UBUtil can generate a firmware update package which consists of multiple sections (named *partitions*) which correspond to different items.

For example, if an application consists of one part of executable code from 0x1000 – 0x3960, another part of executable code from 0x7000 – 0x71a0 and a static configuration area from 0x8000 – 0x9000, then this update consists of three sections/partitions.

UBUtil has many command line options. The following is a list of *global* command line arguments:

```
--application-key-file <file> (application private key)
--output <file>               (output hex file)
--output-headers              (created signed header files)
--help                        (display command listing)
--usage                       (display brief usage information)
--application-types           (display known application types)
--target-types                (display known target types)
--match-types                 (display match types)
--version                     (display version of utility)
--build-info                  (display build information of utility)
--license                     (show license information)
--create-key <file>           (create a new public/private key file)
--file-info <file>            (list information from a bin/hex/pem file)
--file-key <type>             (key to list; 0: private, 1: public, 2: both)
--file-no-verify              (will not check if files exist on the host)
--arg-file <file>             (read in arguments from a file, one per line)
--append-pub-key              (appends public key to the final section)
--prepend-pub-key             (prepends public key to the first section)
--ubu-platform <id>           (specifies target platform - must be 512A510F)
--ubu-flash-size <size>       (specifies target platform flash size - must be 8M)
--ubu-sector-size <size>      (specifies target platform flash sector size - must be 4K)
--ubu-base-address <address>  (specifies target platform base address - must be 0)
--ubu-align-length <size>     (specifies target platform align length - must be 4)
--ubu-output <file>           (output UART upgrade file)
```

The following is a list of command line arguments which can be used per section (where X is a number between 0 and 15 inclusive and R is a number between 0 and 3 inclusive):

```
--aX-version <version>        (version of section, 0-65534)
--aX-compressed               (compress section)
--aX-target <target>          (target of section, see --target-types)
--aX-startaddress <address>   (start address to place section on target)
--aX-endaddress <address>     (end address to place section on target)
--aX-size <size>              (size of section on target)
--aX-filetype <type>          (type of section, see --application-types)
--aX-filename <file>          (input file for section)
--aX-keytype <type>           (key to use; 0: bootloader, 1: application)
--aX-extradata <data>         (specify extra data for section in hex)
--aX-extradatatext <data>     (specify extra data for section in ASCII)
--aX-header <file>            (use pre-generated header section)
--aX-deleteafter              (mark section for delete after upgrade)
--aX-rR-match <match>         (match type, see --match-types)
--aX-rR-present <present>     (if section must be present; 0: no, 1: yes)
--aX-rR-filetype <type>       (type of section, see --application-types)
--aX-rR-version <version>     (version of section, 0-65534)
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

For information on application version requirements, refer to the Version Requirement Options section.

You can use the following command to create a firmware update for an application named Blinky.hex, version 1. It will be compressed using a private key named Blinky.pem which includes the public key in the output. It also creates a UBU file which can be used to upload it via UART using UwFlashX.

```
UBUtil --application-key-file Blinky.pem --a0-version 1 --a0-compressed --a0-target 0 --a0-
startaddress 0x1000 --a0-filetype 1 --a0-filename Blinky.hex --a0-keytype 1 --append-pub-key
--output BlinkyPackage.hex --ubu-output BlinkyPackage.ubu --ubu-platform 512A510F --ubu-
flash-size 8M --ubu-sector-size 4K --ubu-base-address 0 --ubu-align-length 4
```

The output will be similar to the following:

```
Laird Connectivity Universal Bootloader Firmware Update Generation Utility
  v1.0
    Built Apr 24 2020

Using input file Blinky.hex (232352 bytes)...

Header:
        Update header version: 1
        Sections present: 2
        Checksum: 0x56f6cb3c

Section 0:
        Section Start: 0x4000
        Section End: 0x306df
        Section Size: 0x2c6df
        Target: 0 (Internal flash)
        Target Start: 0x1000
        Target End: 0x39ba0
        Target Size: 0x38ba0
        Compressed: Yes
        Version: 1
        Checksum Type: 32-bit
        Image Checksum: 0x7bc86c9e
        Target Checksum: 0x1e5c79e0
        Signature Type: 1 (User key)
        Application Type: 1 (Main Application)
        Header Checksum: 0x947dc75a
        Filename: Blinky.hex
        Extra data (hex):
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000
        Signature (hex):
e448e2113de11a5fa80259cbcdcd2362a3d457ef83ad09fd594caf41d408109d272d05247ba7ffcb1332db8735a
f0a64ffdf1e7543129398b0b97300c62af32

Section 1:
        Section Start: 0x0
        Section End: 0x0
        Section Size: 0x0
        Target: 3 (Settings)
        Target Start: 0x0
        Target End: 0x0
        Target Size: 0x0
        Compressed: No
        Version: 1
        Checksum Type: 32-bit
        Image Checksum: 0xf2a52abf
        Target Checksum: 0x00000000
        Signature Type: 254 (Bypass)
        Application Type: 12 (User public key)
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

```
        Header Checksum: 0xb471376d
        Filename: Blinky.pem
        Extra data (hex):
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000
        Signature (hex):
d24deb7ab6c3922d1fd561d0917304c00ece98a4b8bad9b0bd09ed647edeb0eb5f938693a03ed52ac3f716b0b50
0cfc386bf8a77f4f49ff3b8b4eb4a4ca0470

Total uncompressed section size: 232352
Total compressed section size: 181983
Section compression: 22%

557991 bytes written to BlinkyPackage.hex successfully.
197912 bytes written to BlinkyPackage.ubu successfully.
```

For a full list of file types, requirements, and validities, refer to the Section Application Types section.

6.4 Manually Inputting a Public Key into the Bootloader

From the steps given in Outputting a Private or Public Key, you can obtain a public key from a key file. Once you obtain the key, you can program it into a Pinnacle 100 module which does not have a public key set.

Note: Once a key is set, it cannot be changed or removed.

With a Pinnacle 100 module connected to a computer, follow these steps to program the public key:

1. Enter bootloader mode on the pinnacle 100. You can achieve this by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up (on the development board, hold down SW1 and press the reset button).
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:
   - Hardware flow control enabled
   - Baud rate set to 115200
   - 1 stop bit
   - No parity

   The CTS status should be green to indicate that the module is ready to accept commands.
3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be *f* and a hex character. In UwTerminalX, this takes the form of a slash (\) followed by two numbers.
4. Type *y2* and press **Enter** to check if a public key is loaded to the module. If it responds with a 0, then no public key is programmed. If it responds with a 1, then a public key is programmed (and no public key can be set).
5. Press **x3** and paste the public key from the UBUtil listing step. The line will appear similar to the following:

   x30956845803d675ad448d4f9dec970abf06b4f64a6651661f01c9b1fc906f2760ea36eb0c43ee4305b59273cdf79bdc908aa4eb6d503c78303812040c7d3cbb9a
6. Press **Enter**.
7. The unit should respond with *a*.
   - If it responds with an *f*, then there was an error whilst setting the public key (most likely because it is already set or because the module is lacking a license).
   - If there is no response, then check to ensure you did not miss any characters of the key.

   Reset the module and put it back into bootloader mode before retrying or an invalid key could be programmed to the module.
8. Reset the module by pressing **z** and **Enter**. If a firmware update package is present on the module, then it will be updated assuming that the public key is correct. Otherwise, the package will be removed.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

6.5 Flashing a Firmware Update Package

Flashing a firmware first requires an update package to be generated. Follow the instructions in Generating a Firmware Update Package before continuing.

6.5.1 Using UwFlashX (UART)

Flashing a firmware using UwFlashX requires that a .ubu firmware package is created. It does not work with .hex or .bin files. Refer to Generating a Firmware Update Package to generate a compatible firmware update package. UwFlashX is a cross-platform utility for transferring firmware update images to wireless modules, it can be obtained from https://github.com/LairdCP/UwFlashX

After opening UwFlashX, you are presented with a dialogue showing multiple tabs and options from which to select (Figure 4).

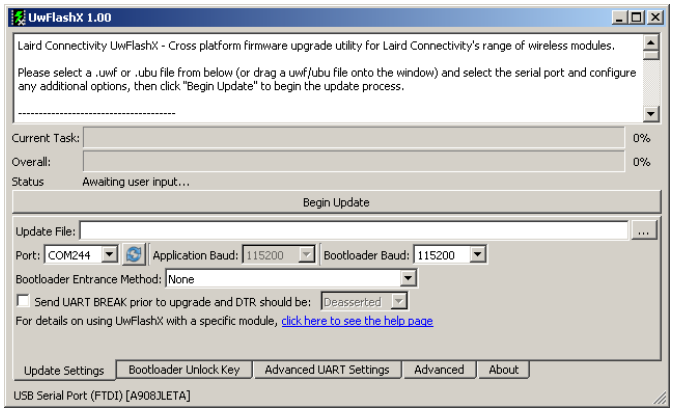


Figure 4: UwFlashX initial window

To program a Pinnacle 100 unit on a development board, follow these steps:

1. Change the Bootloader Entrance Method option from *None* to *FTDI reset (Pinnacle 100)*. This allows the bootloader to be entered before the update begins.

Note: MAC users are unable to use this option. Instead, these users must manually reboot the module into bootloader mode. You can do this by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up which, on the development board, is achieved by holding down SW1 and pressing the reset button).

---

2. A .ubu firmware file must now be selected. Click the **…** button and select the correct file to program to your module.

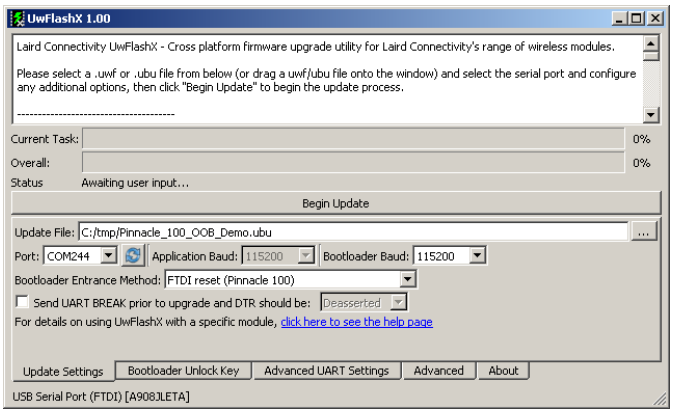


Figure 5: Select your .ubu firmware file

3. The Pinnacle 100 firmware update is now ready to proceed. Click **Begin Update**.

There are additional settings throughout UwFlashX which can be used to configure the update process.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

Bootloader Unlock Key Tab

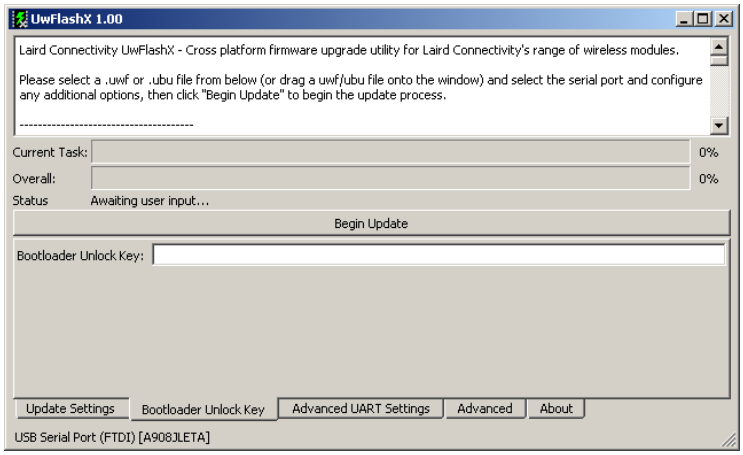


Figure 6: Bootloader Unlock Key tab

The Bootloader Unlock Key tab allows specifying a bootloader unlock key. This is a special key which can be programmed into the bootloader. It prevents read/write access to the data on the device unless the correct unlock key is provided to the module during the firmware update process. Details on setting a bootloader unlock key and how to use it are detailed in the Setting/Using a Bootloader Unlock Key section.

Advanced UART Settings Tab

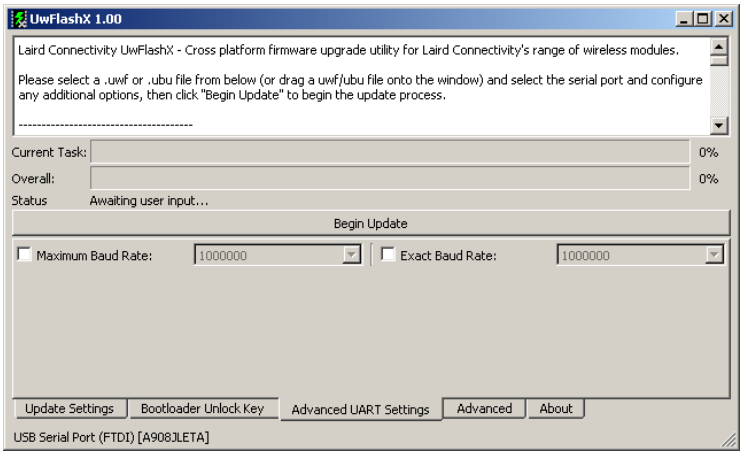


Figure 7: Advanced UART Settings tab

The Advanced UART Settings tab contains options for setting a maximum or exact baud rate:

- **Maximum Baud Rate** – This can be used to set a maximum baud rate for the bootloader to use whilst transferring an update image and is most useful when using an RS232 port or other hardware which does not support fast baud rates. UwFlashX will negotiate the fastest UART speed it can with the bootloader which is below this speed (which can be up to 1M baud if this option is not set). If a compatible baud rate is not found, then the update will fail and not be transferred to the module.
- **Exact Baud Rate** – This can be used to specify the baud rate which is used to communicate with the bootloader, if set then UwFlashX will negotiate this speed to be used for transferring the update image. If the specified baud rate is not supported by the module or is invalid, then the update will fail and not be transferred to the module.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

Advanced Tab

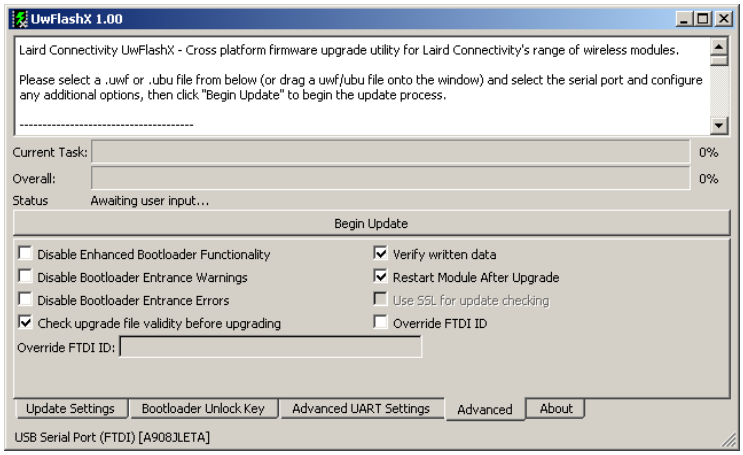


Figure 8: Advanced tab

The Advanced tab lists the following advanced options for the update process:

- **Disable Enhanced Bootloader Functionality** – Enabling this option disables newer bootloader functionality from UwFlashX and uses the legacy bootloader commands. Only use this option if there are issues updating firmware on legacy modules and if will cause issues with Pinnacle 100 modules if, for example, a Bootloader Unlock Key is set.
- **Disable Bootloader Entrance Warnings** – If this option is enabled, popup warning/question messages that open when using the FTDI reset functionality for entering the bootloader do not display and are automatically accepted. This is useful when running UwFlashX in an unattended or automated system and handling the error by a script or other application. Please ensure you use the correct serial port as Laird Connectivity cannot take responsibility for any possible damage or issues caused by use of the wrong serial port.
- **Disable Bootloader Entrance Errors** – If this option is enabled, popup error messages that can open when using the FTDI reset functionality for entering the bootloader do not display and are automatically accepted. This is useful when running UwFlashX in an unattended or automated system and handling the error by a script or other application.
- **Check upgrade file validity before upgrading** – Before the upgrade process starts, the .ubu file is checked to ensure it does not contain any errors. If it finds errors, then the upgrade does not proceed.
- **Verify written data** – If this option is enabled, data that is written to the module is verified to ensure that it was transferred correctly. This does not work if the Blocking UART Data Verification option is enabled on the bootloader.
- **Restart Module After Update** – If this option is enabled then, after the update process is complete, the module is reset so that the bootloader can update the software on the module. Otherwise, after finishing, it keeps the unit in bootloader mode.
- **Use SSL for update checking** – If this option is enabled, the upgrade check mechanism uses SSL. Otherwise it uses standard unencrypted HTTP.
- **Override FTDI ID** – This option should only be used if there is an issue with the FTDI serial number of the Pinnacle 100 development board. If enabled, the FTDI serial number placed in the ID box is used for entering the bootloader. Otherwise, UwFlashX attempt sto automatically detect the serial number.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

About Tab

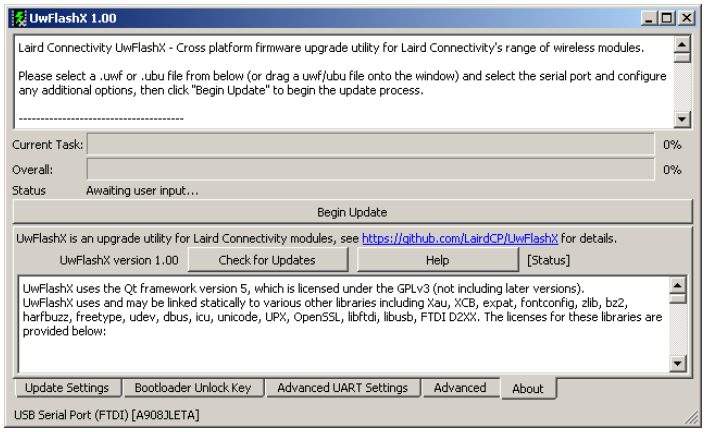


Figure 9: About tab

The About tab displays licenses of software used in the creation of UwFlashX. These are also displayed at the end of this document in the License Information section.

There is also an option to check if your version of UwFlashX is the latest and a link to the github page where a newer version of the application (if available) can be downloaded.

Note: By using the check for update functionality, some details are stored by server in a log file when the request is made. These details include IP address, time and date of request, web headers and URL, version number/name of the application, and the operating system of your computer. This information is held for security purposes only and is not used for any analytical purposes.

6.5.2 Using nrfjprog (SWD)

With the output hex file generated, open a command prompt window or terminal and flash the update image to QSPI using the following:

```
nrfjprog -f NRF52 --program <file.hex> --qspisectorerase --reset
```

The output will look similar to this the following:

```
Parsing hex file.
Reading flash area to program to guarantee it is erased.
Initializing the QSPI peripheral.
Erasing external memory pages.
Erasing external memory pages.
Erasing external memory pages.
Erasing external memory pages.
Uninitializing the QSPI peripheral.
Checking that the area to write is not protected.
Programming device.
Initializing the QSPI peripheral.
WARNING: An operation that can take up to several minutes is being executed.
WARNING: Please remain patient.
Uninitializing the QSPI peripheral.
Applying system reset.
Run.
```

Once the application has been successfully programmed, the module will reboot and the bootloader will update assuming that the public key has been programmed correctly or is present in the programmed update file.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

6.6 Listing Details About a Firmware Update Package

Similarly to how UBUtil can be used to list details about key files as seen in Outputting a Private or Public Key, it can also be used to display details about firmware updates by using the --file-info argument. This only works for output .hex files and does not work for output .uwf or .ubu files. This shows if there is any corruption in the output too. It can be used by running the following command:

`UBUtil --file-info BlinkyPackage.hex`

The output looks similar to the following:

```
Laird Connectivity Universal Bootloader Firmware Update Generation Utility
  v1.0
    Built Apr 24 2020

Header:
        Update header version: 1
        Sections present: 2
        Checksum: 0x56f6cb3c

Section 0:
        Section Start: 0x4000
        Section End: 0x306df
        Section Size: 0x2c6df
        Target: 0 (Internal flash)
        Target Start: 0x1000
        Target End: 0x39ba0
        Target Size: 0x38ba0
        Compressed: Yes
        Version: 1
        Checksum Type: 32-bit
        Image Checksum: 0x7bc86c9e
        Target Checksum: 0x1e5c79e0
        Signature Type: 1 (User key)
        Application Type: 1 (Main Application)
        Header Checksum: 0x947dc75a
        Filename: Blinky.hex
        Extra data (hex):
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000
        Image checksum valid: Yes
        Header checksum valid: Yes
        Signature (hex):
e448e2113de11a5fa80259cbcdcd2362a3d457ef83ad09fd594caf941d408109d272d05247ba7ffcb1332db8735
af0a64ffdf1e7543129398b0b97300c62af32

Section 1:
        Section Start: 0x0
        Section End: 0x0
        Section Size: 0x0
        Target: 3 (Settings)
        Target Start: 0x0
        Target End: 0x0
        Target Size: 0x0
        Compressed: No
        Version: 1
        Checksum Type: 32-bit
        Image Checksum: 0xf2a52abf
        Target Checksum: 0x00000000
        Signature Type: 254 (Bypass)
        Application Type: 12 (User public key)
        Header Checksum: 0xb471376d
        Filename: Blinky.pem
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

```
        Extra data (hex):
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000
        Image checksum valid: No
        Header checksum valid: Yes
        Signature (hex):
d24deb7ab6c3922d1fd561d0917304c00ece98a4b8bad9b0bd09ed4647edeb0eb5f938693a03ed52ac3f716b0b5
00cfc386bf8a77f4f49ff3b8b4eb4a4ca0470
```

6.7 Signed Image Header Files

Signed header files are header sections of files passed through UBUtil which have been signed with the private signing key. They can be used to integrate into various update packages containing different images and features. This allows for the signing process to take place on a secure build server which has the private signing key without needing to share the private key or compromise the security of it. Signed header files contain all the information required for input files to be processed by UBUtil on another PC without the need for the private key as they contain the pre-computed signature of the input file. To use signed image header files, the input .hex/.bin files processed by UBUtil and the output .hdr files must be presented to UBUtil at a later time to generate the final update package.

As an example, bootloader updates for the Pinnacle 100 module are distributed in paired .hex and .hdr files. These can be incorporated into user update packages along with application updates.

6.7.1 Generating Signed Image Header Files

Signed image header output can be applied to any UBUtil firmware generation command by appending:

`--output-headers`

When this command is used, all input sections with input files have a new file created with _header.hdr appended to the end of them (the format of the name output is not configurable). Note that each generated header file is unique and matches up with the input arguments which were used to generate the file. For example, if compression is used, then compression is mandatory for that section; it is automatically enabled when that header file is used in future.

6.7.2 Using Signed Image Header Files (e.g. Bootloader and Cell Modem Updates)

This section applies to signed image files generated by Laird Connectivity for bootloader and cellular modem firmware updates. It also applies to any user-created signed image header files.

When using a signed image header file with UBUtil, the only parameters allowed are the input filename and the input signed image header. **No other arguments for these sections are allowed**.

However, inputs to UBUtil can be of a mixed form: one section can be a user application and signed image header file, another section could be a Laird Connectivity bootloader update and signed image header file, and a third section could be a user configuration section with parameters supplied as normal and the private signing key supplied to sign this section. This will generate a valid update file. The argument **--aX-header <file >** must be used to supply a signed image header file along with the **--aX-filename <file>** argument to specify the input hex/binary file.

To use only signed image header files:

```
UBUtil --a0-filename Blinky.hex --a1-filename bootloader.hex --a0-header Blinky_header.hdr --
a1-header bootloader_header.hdr --output test_package.hex --ubu-platform 512A510F --ubu-
flash-size 8M --ubu-sector-size 4K --ubu-base-address 0 --ubu-align-length 4 --ubu-output
test_package.uwf
```

To use a mix of signed image header files and normal arguments:

```
UBUtil --application-key-file Blinky.pem --a0-version 1 --a0-compressed --a0-target 0 --a0-
startaddress 0x1000 --a0-filetype 1 --a0-filename Blinky.hex --a0-keytype 1 --a1-filename
bootloader.hex --a1-header bootloader_header.hdr --append-pub-key --output test_package.hex -
-ubu-platform 512A510F --ubu-flash-size 8M --ubu-sector-size 4K --ubu-base-address 0 --ubu-
align-length 4 --ubu-output test_package.uwf
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

Note: The --append-pub-key argument does not work if a key file is not provided as it extracts the public key from the pem file to append to the update file. However, the key can be manually appended if it is not already present on the module by following the Public Key Sections section. Alternatively, it can be manually set into the bootloader by following the Manually Inputting a Public Key into the Bootloader section.

The output looks similar to the following:

```
Laird Connectivity Universal Bootloader Firmware Update Generation Utility
  v1.0
    Built Apr 24 2020

Using input file Blinky.hex (232352 bytes)...
Using input file bootloader.hex (77824 bytes)...

Header:
        Update header version: 1
        Sections present: 3
        Checksum: 0x04152166

Section 0:
        Section Start: 0x4000
        Section End: 0x306df
        Section Size: 0x2c6df
        Target: 0 (Internal flash)
        Target Start: 0x1000
        Target End: 0x39ba0
        Target Size: 0x38ba0
        Compressed: Yes
        Version: 1
        Checksum Type: 32-bit
        Image Checksum: 0x7bc86c9e
        Target Checksum: 0x1e5c79e0
        Signature Type: 1 (User key)
        Application Type: 1 (Main Application)
        Header Checksum: 0xcc1012d5
        Filename: Blinky.hex
        Extra data (hex):
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000
        Signature (hex):
e7aae7a83a5c45f84504e1de45c66962beb4f7db93650e5377d87af585168d313eb89332267f831c4e713d61ce4
ccbebe9cd7e392fa89d5088ba69b01ea157e0

Section 1:
        Section Start: 0x31000
        Section End: 0x3f7ea
        Section Size: 0xe7ea
        Target: 0 (Internal flash)
        Target Start: 0xd7000
        Target End: 0xea000
        Target Size: 0x13000
        Compressed: Yes
        Version: 3
        Checksum Type: 32-bit
        Image Checksum: 0xd61ee013
        Target Checksum: 0x144735ae
        Signature Type: 0 (Bootloader key)
        Application Type: 2 (Bootloader Update)
        Header Checksum: 0x022ac000
        Filename: Bldr_Release.bin
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

```
        Extra data (hex):
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000
        Signature (hex):
ee67e762309b2b2281d4e92b3a30d1ca898a0dd7e4912eb768d80361862aa7878ec1f669b89f12728e1ee87166e
10c8772c81a4bd1da2f28b33352f4ca699a23

Section 2:
        Section Start: 0x0
        Section End: 0x0
        Section Size: 0x0
        Target: 3 (Settings)
        Target Start: 0x0
        Target End: 0x0
        Target Size: 0x0
        Compressed: No
        Version: 1
        Checksum Type: 32-bit
        Image Checksum: 0xf2a52abf
        Target Checksum: 0x00000000
        Signature Type: 254 (Bypass)
        Application Type: 12 (User public key)
        Header Checksum: 0xb471376d
        Filename: Blinky.pem
        Extra data (hex):
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000
        Signature (hex):
d24deb7ab6c3922d1fd561d0917304c00ece98a4b8bad9b0bd09ed4647edeb0eb5f938693a03ed52ac3f716b0b5
00cfc386bf8a77f4f49ff3b8b4eb4a4ca0470

Total uncompressed section size: 310176
Total compressed section size: 241353
Section compression: 23%

731556 bytes written to test_package.hex successfully.
257298 bytes written to test_package.uwf successfully.
```

6.8 Section Application Types

The following table (Table 3) lists the application types available for use on the Pinnacle 100 and specified by using the **--aX-filetype ?** argument. Bootloader updates and modem updates are available from Laird Connectivity only and should not be generated using the UBUtil application. They arrive as a .hex and .hdr file which can be supplied when generating a firmware update package to include them in it.

Table 3: Section application types

Type	Name	Description	Self-Purge	Section Size	Start Address	Allow Compression	Allow Multiple
0	***Reserved***	Reserved for future use					
1	Main application	Stores executable applications, these should start at 0x1000 if no Soft-Device is used or at an address corresponding to the Soft-Device address if one is used	x	Up to 0xD6000	0x1000 – 0xD6000	✔	x
2	Bootloader update	Stores bootloader updates which are provided by Laird Connectivity	x			✔	x
3	Soft-Device	Stores Soft-Devices which start at 0x1000, if one is used by the application	x	Up to 0x50000	0x1000	✔	x
4	User Configuration A	Can be used to store read-only user configuration	x	Up to 0xD5000	0x2000 – 0xD6000	x	x

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

Type	Name	Description	Self-Purge	Section Size	Start Address	Allow Compression	Allow Multiple
5	User Configuration B	Can be used to store read-only user configuration	x	Up to 0xD4000	0x3000 – 0xD6000	x	x
6	User Configuration C	Can be used to store read-only user configuration	x	Up to 0xD3000	0x4000 – 0xD6000	x	x
7	User Configuration D	Can be used to store read-only user configuration	x	Up to 0xD2000	0x5000 – 0xD6000	x	x
8	***Reserved***	Reserved for future use					
9	Erase Once	Can be used to erase a section of flash once (if not already erased)	✔		0x1000 – 0xD6000	x	x
10	Erase Always	Can be used to erase a section of flash every time the bootloader starts (if not already erased)	x		0x1000 – 0xD6000	x	x
11	QSPI user section (no verification)		x			x	x
12	Public Key	Can be used to program a public key into the bootloader without requiring UART access	x			x	x
13	Bootloader Settings					x	✔
14	Modem Update	Contains cellular modem update packages	✔			✔	✔
15	Code Section A		x	Up to 0xD5000	0x2000 – 0xD6000	x	x
16	Code Section B		x	Up to 0xD5000	0x2000 – 0xD6000	x	x
17	Code Section C		x	Up to 0xD5000	0x2000 – 0xD6000	x	x
18	Code Section D		x	Up to 0xD5000	0x2000 – 0xD6000	x	x
19	Bootloader Unlock Key	Can be used to program an unlock key into the bootloader without requiring UART access (note that for security purposes, this section will be removed once it has been programmed)	✔			x	x
20	QSPI user section (with verification)		x			x	x

6.8.1 User Configuration Sections

User configuration sections are sections that can reside in QSPI flash. They are read-only and can be used to set configuration for user applications or hold strings/data which can be used. Note that these sections are not for holding executable (eXecute-In-Place – XIP) code. These sections mandate a signed section and do not support being updated from the user application unless the entire section is replaced with a new section including a valid signature and header.

To add a user configuration section using UBUtil, the application type should be set to 5-7 depending upon the address of where the section is to be placed as shown in the *Section Application Types table* (Table 3).

The following is an example

```
UBUtil --application-key-file Blinky.pem --a0-version 1 --a0-filetype 5 --a0-target 0 --a0-
keytype 1 --a0-startaddress 0x18000 --a0-filename ConfigInput.hex --append-pub-key --output
configuration_section_example.hex --ubu-platform 512A510F --ubu-flash-size 8M --ubu-sector-
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

`size 4K --ubu-base-address 0 --ubu-align-length 4 --ubu-output configuration_section_example.uwf`

The output looks similar to the following:

```
Laird Connectivity Universal Bootloader Firmware Update Generation Utility
  v1.0
    Built Apr 24 2020

Using input file ConfigInput.hex (108071 bytes)...

Header:
        Update header version: 1
        Sections present: 2
        Checksum: 0x56f6cb3c

Section 0:
        Section Start: 0x4000
        Section End: 0x1e627
        Section Size: 0x1a627
        Target: 0 (Internal flash)
        Target Start: 0x18000
        Target End: 0x32627
        Target Size: 0x1a627
        Compressed: No
        Version: 1
        Checksum Type: 32-bit
        Image Checksum: 0x42549f32
        Target Checksum: 0x42549f32
        Signature Type: 1 (User key)
        Application Type: 5 (User configuration B)
        Header Checksum: 0xf0c146f1
        Filename: ConfigInput.hex
        Extra data (hex):
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000
        Signature (hex):
f62d414c5d1c71821a97b40dac71bfb121f0d1cb3878fc40572de999138993097e46c3a34fd3af8ba1d02450409
5a634d693f93636135a4bec4ea8627f64879a

Section 1:
        Section Start: 0x0
        Section End: 0x0
        Section Size: 0x0
        Target: 3 (Settings)
        Target Start: 0x0
        Target End: 0x0
        Target Size: 0x0
        Compressed: No
        Version: 1
        Checksum Type: 32-bit
        Image Checksum: 0xf2a52abf
        Target Checksum: 0x00000000
        Signature Type: 254 (Bypass)
        Application Type: 12 (User public key)
        Header Checksum: 0xb471376d
        Filename: Blinky.pem
        Extra data (hex):
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000
        Signature (hex):
d24deb7ab6c3922d1fd561d0917304c00ece98a4b8bad9b0bd09ed4647edeb0eb5f938693a03ed52ac3f716b0b5
00cfc386bf8a77f4f49ff3b8b4eb4a4ca0470
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

```
Total uncompressed section size: 108071
Total compressed section size: 108071
Section compression: 0%

350102 bytes written to configuration_section_example.hex successfully.
124000 bytes written to configuration_section_example.uwf successfully.
```

6.8.2 Erase Sections

Erase sections can be used to erase a portion of internal nRF52840 flash space. There are two types of erase sections:

- **Erase Once** – These sections, when present, erase a section of nRF52840 flash and are then purged from the QSPI flash. They do not erase data again.
- **Erase Always** – These sections, when present, always erase a section of nRF52840 flash and remain present in the QSPI flash once the bootloader is finished. They erase the section every time the bootloader runs.

Note: If the area specified by an erase section is already erased (set to 0xff), then it skips the erase process. But if the erase file is an erase once type section, then that section is purged from QSPI regardless of whether or not the flash is erased.

In the following arguments, the version should be a number between 1-65535 and be incremented each time a section of that type is generated, the start address should be the sector-aligned address of the area to erase, the size (if used) should be a sector-aligned length to erase, the end address (if used) should be a sector-aligned address of the address to erase.

To add an erase once section using UBUtil, the following arguments should be used (where X is the section number):

```
--aX-version ? --aX-filetype 9 --aX-startaddress 0x? --aX-size 0x? --aX-target 0 --aX-keytype 1
```

Alternative, an end address can be specified instead of a size:

```
--aX-version ? --aX-filetype 9 --aX-startaddress 0x? --aX-endaddress 0x? --aX-target 0 --aX-keytype 1
```

To add an erase always section using UBUtil, the following arguments should be used (where X is the section number):

```
--aX-version ? --aX-filetype 10 --aX-startaddress 0x? --aX-size 0x? --aX-target 0 --aX-keytype 1
```

Alternative, an end address can be specified instead of a size:

```
--aX-version ? --aX-filetype 10 --aX-startaddress 0x? --aX-endaddress 0x? --aX-target 0 --aX-keytype 1
```

The following is an example:

```
UBUtil --application-key-file Blinky.pem --a0-version 1 --a0-filetype 9 --a0-startaddress 0xe000 --a0-size 0x1000 --a0-target 0 --a0-keytype 1 --append-pub-key --output erase_example.hex --ubu-platform 512A510F --ubu-flash-size 8M --ubu-sector-size 4K --ubu-base-address 0 --ubu-align-length 4 --ubu-output erase_example.uwf
```

Note: Start addresses, end addresses, and sizes must be sector aligned (0x1000 on the nRF52840) as sectors are erased in full.

The output looks similar to the following:

```
Laird Connectivity Universal Bootloader Firmware Update Generation Utility
  v1.0
    Built Apr 24 2020


Header:
        Update header version: 1
        Sections present: 2
        Checksum: 0x56f6cb3c
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

```
Section 0:
        Section Start: 0x0
        Section End: 0x0
        Section Size: 0x0
        Target: 0 (Internal flash)
        Target Start: 0xe000
        Target End: 0xf000
        Target Size: 0x1000
        Compressed: No
        Version: 1
        Checksum Type: 32-bit
        Image Checksum: 0x00000000
        Target Checksum: 0x3648671b
        Signature Type: 1 (User key)
        Application Type: 9 (Erase once)
        Header Checksum: 0xf47d0c31
        Filename: [Not Present]
        Extra data (hex):
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000
        Signature (hex):
6c9d85ab2babe7c728576d89520f4fe4b0bae676843ae0dea3ff2ca44efe1ebfc86f187106b0117e031776e74e1
8c9d76ff24021abf92a91fe6f4c1262160fd8

Section 1:
        Section Start: 0x0
        Section End: 0x0
        Section Size: 0x0
        Target: 3 (Settings)
        Target Start: 0x0
        Target End: 0x0
        Target Size: 0x0
        Compressed: No
        Version: 1
        Checksum Type: 32-bit
        Image Checksum: 0xf2a52abf
        Target Checksum: 0x00000000
        Signature Type: 254 (Bypass)
        Application Type: 12 (User public key)
        Header Checksum: 0xb471376d
        Filename: Blinky.pem
        Extra data (hex):
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000
        Signature (hex):
d24deb7ab6c3922d1fd561d0917304c00ece98a4b8bad9b0bd09ed4647edeb0eb5f938693a03ed52ac3f716b0b5
00cfc386bf8a77f4f49ff3b8b4eb4a4ca0470

Total uncompressed section size: 0
Total compressed section size: 0
46110 bytes written to erase_example.hex successfully.
15914 bytes written to erase_example.uwf successfully.
```

6.8.3 In-Place User Storage Sections

In-Place User Storage Sections can be used for holding data in QSPI which can then be used for reading and optionally writing data from the user application. Section sizes cannot be changed and are fixed to their initial size. There are two supported types:

- One with verification and a checksum which requires a valid signed header. This is recommended for uses where the data is not changed by the user application but can be upgraded;

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

- One without verification/checksum requirement. This can be changed by the user application at-will but a security system should be added to the user application to verify the data and ensure it is correct.

QSPI in-place sections stay in QSPI and are not transferred to the internal nRF52840 flash. The user application should query the QSPI header data to find this section and remain within the limits of the section. The QSPI chip supports single SPI mode, dual SPI mode, and quad SPI mode at clock frequencies of up to 32 MHz. It can erase 4 KB pages or 64 KB sectors.

6.8.3.1 Without Verification/Checksum

For sections without verification/checksum, the application type should be set to 11 and target set to 2. No private key needs to be supplied for this section type as it is unsigned and the underlying QSPI data can be changed by the user application at will; it is the responsibility of the user to add security to their application to ensure that the data is valid.

The following is an example:

`UBUtil --a0-version 1 --a0-target 2 --a0-startaddress 0x40000 --a0-filetype 11 --a0-filename App_Config.bin --output example_configuration_no_verification.hex`

The output looks similar to the following:

```
Laird Connectivity Universal Bootloader Firmware Update Generation Utility
  v1.0
    Built Apr 24 2020

Using input file App_Config.bin (108071 bytes)...

Header:
        Update header version: 1
        Sections present: 1
        Checksum: 0xc0b56ecc

Section 0:
        Section Start: 0x4000
        Section End: 0x1e627
        Section Size: 0x1a627
        Target: 2 (QSPI)
        Target Start: 0x40000
        Target End: 0x5a627
        Target Size: 0x1a627
        Compressed: No
        Version: 1
        Checksum Type: Bypass
        Image Checksum: 0x00000000
        Target Checksum: 0x00000000
        Signature Type: 254 (Bypass)
        Application Type: 11 (QSPI user section (no verification))
        Header Checksum: 0x06582ae5
        Filename: App_Config.bin
        Extra data (hex):
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000
        Signature (hex):
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000

Total uncompressed section size: 108071
Total compressed section size: 108071
Section compression: 0%

350102 bytes written to example_configuration_no_verification.hex successfully.
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

6.8.3.2 With Verification/Checksum

For sections with verification/checksum support, the application type should be set to 20 and target set to 2. The user private key must be supplied for this section type as it is signed and the underlying QSPI data cannot be changed by the user application at will. If the data is changed, then it is removed by the bootloader.

The following is an example:

```
UBUtil --application-key-file blinky.pem --a0-version 1 --a0-target 2 --a0-startaddress
0x40000 --a0-filetype 20 --a0-filename App_Config.hex --a0-keytype 1 --append-pub-key --
output example_configuration_with_verification.hex --ubu-platform 512A510F --ubu-flash-size
8M --ubu-sector-size 4K --ubu-base-address 0 --ubu-align-length 4 --ubu-output
example_configuration_with_verification.uwf
```

The output looks similar to the following:

```
Laird Connectivity Universal Bootloader Firmware Update Generation Utility
  v1.0
    Built Apr 24 2020

Using input file App_Config.hex (16384 bytes)...

Header:
        Update header version: 1
        Sections present: 2
        Checksum: 0x56f6cb3c

Section 0:
        Section Start: 0x4000
        Section End: 0x8000
        Section Size: 0x4000
        Target: 2 (QSPI)
        Target Start: 0x40000
        Target End: 0x44000
        Target Size: 0x4000
        Compressed: No
        Version: 1
        Checksum Type: 32-bit
        Image Checksum: 0x4bfe7b43
        Target Checksum: 0x4bfe7b43
        Signature Type: 1 (User key)
        Application Type: 20 (QSPI user section (with verification))
        Header Checksum: 0xce8347f3
        Filename: App_Config.hex
        Extra data (hex):
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000
        Signature (hex):
33cecae3a95099d3bef6257fb8bf53593e3b326112a99c91d8b9c5ddb4ab02bd6b49b00bdda168299b1d9af339
cc995035c87f1b79eadb6cf488722e720c4af4

Section 1:
        Section Start: 0x0
        Section End: 0x0
        Section Size: 0x0
        Target: 3 (Settings)
        Target Start: 0x0
        Target End: 0x0
        Target Size: 0x0
        Compressed: No
        Version: 1
        Checksum Type: 32-bit
        Image Checksum: 0xf2a52abf
        Target Checksum: 0x00000000
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

```
        Signature Type: 254 (Bypass)
        Application Type: 12 (User public key)
        Header Checksum: 0x5c907132
        Filename: blinky.pem
        Extra data (hex):
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000
        Signature (hex):
d24deb7ab6c3922d1fd561d0917304c00ece98a4b8bad9b0bd09ed4647edeb0eb5f938693a03ed52ac3f716b0b
500cfc386bf8a77f4f49ff3b8b4eb4a4ca0470

Total uncompressed section size: 16384
Total compressed section size: 16384
Section compression: 0%

92190 bytes written to example_configuration_with_verification.hex successfully.
32312 bytes written to example_configuration_with_verification.uwf successfully.
```

6.8.4 Public Key Sections

Public Key Sections allow the application public key to be programmed to the module by the bootloader without using the UART interface of the module. This is useful for mass production as it speeds up the process of programming modules.

There are two methods of adding public keys to update images:

- Automatically extracting the public key from the pem file and including it in the output image – either prepended to the start of the update sections or appended to the end of the update sections. Note that this does not work if UBUtil is working strictly with signed header image files without access to the private key file.
- Manually inputting the public key into a UBUtil command line argument. This generates a section without UBUtil needing access to the keys in the pem file

To prepend the public key from the pem file using UBUtil, use this argument:

--prepend-pub-key

To append the public key from the pem file using UBUtil, use this argument:

--append-pub-key

To specify a public key manually (which can be extracted from the pem file using UBUtil as described in the Outputting a Private or Public Key section), use these arguments (where X is the section number and where the extra data field contains the hexadecimal public key):

--aX-version 1 --aX-filetype 12 --aX-extradata ? --aX-target 3

The following is an example:

UBUtil --application-key-file Blinky.pem --a0-version 1 --a0-filetype 12 --a0-extradata 0956845803d675ad448d4f9dec970abf06b4f64a6651661f01c9b1fc906f2760ea36eb0c43ee4305b59273cdf79bdc908aa4eb6d503c78303812040c7d3cbb9a --a0-target 3 --a1-version 2 --a1-keytype 1 --a1-target 0 --a1-startaddress 0x1000 --a1-filetype 1 --a1-filename Blinky.hex --output public_key_and_application_example.hex --ubu-platform 512A510F --ubu-flash-size 8M --ubu-sector-size 4K --ubu-base-address 0 --ubu-align-length 4 --ubu-output public_key_and_application_example.uwf

The output looks similar to the following:

```
Laird Connectivity Universal Bootloader Firmware Update Generation Utility
  v1.0
    Built Apr 24 2020

Using input file Blinky.hex (232352 bytes)...

Header:
        Update header version: 1
        Sections present: 2
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

```
        Checksum: 0x56f6cb3c

Section 0:
        Section Start: 0x0
        Section End: 0x0
        Section Size: 0x0
        Target: 3 (Settings)
        Target Start: 0x0
        Target End: 0x0
        Target Size: 0x0
        Compressed: No
        Version: 1
        Checksum Type: 32-bit
        Image Checksum: 0x06d81eb6
        Target Checksum: 0x00000000
        Signature Type: 254 (Bypass)
        Application Type: 12 (User public key)
        Header Checksum: 0x8d30c6f3
        Filename: [Not Present]
        Extra data (hex):
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000
        Signature (hex):
0956845803d675ad448d4f9dec970abf06b4f64a6651661f01c9b1fc906f2760ea36eb0c43ee4305b59273cdf7
9bdc908aa4eb6d503c78303812040c7d3cbb9a

Section 1:
        Section Start: 0x4000
        Section End: 0x3cba0
        Section Size: 0x38ba0
        Target: 0 (Internal flash)
        Target Start: 0x1000
        Target End: 0x39ba0
        Target Size: 0x38ba0
        Compressed: No
        Version: 2
        Checksum Type: 32-bit
        Image Checksum: 0x1e5c79e0
        Target Checksum: 0x1e5c79e0
        Signature Type: 1 (User key)
        Application Type: 1 (Main Application)
        Header Checksum: 0x9435dee6
        Filename: Blinky.hex
        Extra data (hex):
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000
        Signature (hex):
8e80c934e6f29ad90b6611352c250946e071c61bc85d58c4b23206f556503d90cf1d9d94581575326f581d05f9
00b1919c67e00c9b3b827d259e8e9606e00abd

Total uncompressed section size: 232352
Total compressed section size: 232352
Section compression: 0%

699651 bytes written to public_key_and_application_example.hex successfully.
248280 bytes written to public_key_and_application_example.uwf successfully.
```

6.8.5 Bootloader Settings Sections

Bootloader settings sections allow various configuration options of the bootloader to be set without mandating UART access to enable them. This is designed for mass production so that a single image can be deployed to modules for faster deployment

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

times. A .ubu file can have a bootloader settings section to set the module up alongside other partitions like an application. Details on how to use the bootloader settings section are described in the Setting Bootloader Options via UBUtil section which also includes examples showing how to include these sections in firmware upgrade files.

6.8.6 Bootloader Unlock Key Sections

A bootloader unlock key is a 64-byte (512-bit) key which is used to prevent access to the read/write/verify UART bootloader commands until the correct key has been entered. This can help with bolstering security. Details on this can be found in the Setting/Using a Bootloader Unlock Key section.

A bootloader unlock key can be programmed to a module with a section in a firmware upgrade package. This allows for keys to be set during production. For security reasons, once the bootloader processes a bootloader unlock key section, it purges the data from QSPI to prevent unauthorised reading of the key by possible malicious users. Another possible method of reducing risk is to generate and store a unique key for every device produced. This is a decision for the user to make based upon their security requirements.

For a bootloader unlock key section, the application type is 19 and the 64-byte key can be provided in hex format using this argument:

`--aX-extradata ?`

With a 128-character argument, or can be provided in ASCII using this argument:

`--aX-extradatatext ?`

With a 64-characters argument, X must be set to a free section. The target type must be set to bootloader storage which is type 3.

Note: Using the ASCII input option restricts the range of allowable characters to those in the ASCII range.

Note: At the time of writing this guide, UwFlashX only supports ASCII/UTF-8 input for the bootloader unlock key.

The following is an example of providing the key in hex:

```
UBUtil --application-key-file Blinky.pem --a0-version 1 --a0-filetype 19 --a0-extradata
74657374756e6c6f636b6b65796f6e7468654c616972642d436f6e6e6563746976974795f556e6976657273616c2
d426f6f746c6f616465725f53797374656d --a0-keytype 1 --a0-target 3 --output
unlock_key_example.hex --ubu-platform 512A510F --ubu-flash-size 8M --ubu-sector-size 4K --
ubu-base-address 0 --ubu-align-length 4 --ubu-output unlock_key_example.uwf
```

The following is an example of providing the key in ASCII:

```
UBUtil --application-key-file Blinky.pem --a0-version 1 --a0-filetype 19 --a0-extradatatext
testunlockkeyontheLaird-Connectivity_Universal-Bootloader_System --a0-keytype 1 --a0-target 3
--output unlock_key_example.hex --ubu-platform 512A510F --ubu-flash-size 8M --ubu-sector-size
4K --ubu-base-address 0 --ubu-align-length 4 --ubu-output unlock_key_example.uwf
```

The output looks similar to the following:

```
Laird Connectivity Universal Bootloader Firmware Update Generation Utility
  v1.0
    Built Apr 24 2020


Header:
        Update header version: 1
        Sections present: 1
        Checksum: 0xc0b56ecc

Section 0:
        Section Start: 0x0
        Section End: 0x0
        Section Size: 0x0
        Target: 3 (Settings)
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

```
        Target Start: 0x0
        Target End: 0x0
        Target Size: 0x0
        Compressed: No
        Version: 1
        Checksum Type: Bypass
        Image Checksum: 0x00000000
        Target Checksum: 0x00000000
        Signature Type: 1 (User key)
        Application Type: 19 (Bootloader unlock code)
        Header Checksum: 0xee19eeca
        Filename: [Not Present]
        Extra data (hex):
e565713db51ddd39656125ede171eded09e4e1253d396579a5e955ddb571a579a5712d65e9e9ed0de46139a525
c165a171e9edb565adad2dede1e975713d6571
        Signature (hex):
9dd39a99d17106c839e690fba430fe9f92de63351fcd72fbdb387d62fccc6247a7de0108cf20bffc5dd2d7ac21
35f0186ce912f9f82b7a9bc807f9d84655f2fc

Total uncompressed section size: 0
Total compressed section size: 0
46110 bytes written to unlock_key_example.hex successfully.
15914 bytes written to unlock_key_example.uwf successfully.
```

6.9 Version Requirement Options

Some firmware packages might contain multiple components which are only compatible with certain versions of other components. Because firmware files are transferred as a whole, if there is an error in one of the sections then that section may be removed when the bootloader starts. One of the other sections may be upgraded which could cause an incompatibility with the software loaded on the module.

One example of this is Nordic SDK projects which utilise the Nordic soft-device. If a previous major version of the soft-device is loaded and the application is built against a newer one, then it causes the module to hard-fault. This feature ensures that these components are only programmed if they are all present and valid in the upgrade image.

Each upgrade section can have four version requirements which link to other sections. The following (Table 4) is a list of supported comparison modes.

Table 4: Supported comparison modes

#	Match	Description	Command
0	=	Equal to	e
1	!=	Not equal to	ne
2	>	Greater than	gt
3	<	Less than	lt
4	>=	Greater than or equal to	gtet
5	=<	Less than or equal to	ltet
6	!>	Not greater than	ngt
7	!<	Not less than	nlt
8	!>=	Not greater than or equal to	ngtet
9	!=<	Not less than or equal to	nltet

Either the number (#) or command can be specified on the command line. Symbols cannot be used because they have special meaning in command prompt and terminals.

Version requirements can be specified with the following command line arguments:

`--aX-rY-match =<match> --aX-rY-filetype <type> --aX-rY-version <version> --aX-rY-present <present>`

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

Where X is the section number, Y is the requirement number, <match> is the match # or command from the above table, <type> is the application type to check against (see the Section Application Types section for details on application types), <version> is the version to match against and <present> can be 0 if the section does not have to be present or 1 if the section does have to be present.

As an example, if a firmware consists of three components: a soft-device, user application, and user configuration section, whereby all components must be of the same version then such an upgrade file could be generated using:

```
UBUtil --application-key-file Blinky.pem --a0-version 2 --a0-compressed --a0-target 0 --a0-
startaddress 0x1000 --a0-filetype 2 --a0-filename SD7.hex --a0-keytype 1 --a0-r0-match e --
a0-r0-filetype 1 --a0-r0-version 2 --a0-r1-match e --a0-r1-filetype 4 --a0-r1-version 2 --a1-
version 2 --a1-compressed --a1-target 0 --a1-startaddress 0x27000 --a1-filetype 2 --a1-
filename App.hex --a1-keytype 1 --a1-r0-match e --a1-r0-filetype 2 --a1-r0-version 2 --a1-r1-
match e --a1-r1-filetype 4 --a1-r1-version 2 --a2-version 2 --a2-target 0 --a2-startaddress
0x38000 --a2-filetype 4 --a2-filename Cfg.hex --a2-keytype 1 --a2-r0-match e --a2-r0-filetype
1 --a2-r0-version 2 --a2-r1-match e --a2-r1-filetype 2 --a2-r1-version 2 --output
UpgradePackage.hex --ubu-output UpgradePackage.ubu --ubu-platform 512A510F --ubu-flash-size
8M --ubu-sector-size 4K --ubu-base-address 0 --ubu-align-length 4
```

The output is similar to the following:

```
Laird Connectivity Universal Bootloader Firmware Update Generation Utility
  v1.0
    Built Apr 24 2020

Using input file SD7.hex (152984 bytes)...
Using input file App.hex (236060 bytes)...
Using input file Cfg.hex (124464 bytes)...

Header:
        Update header version: 1
        Sections present: 3
        Checksum: 0x04152166

Section 0:
        Section Start: 0x4000
        Section End: 0x23eed
        Section Size: 0x1feed
        Target: 0 (Internal flash)
        Target Start: 0x1000
        Target End: 0x26598
        Target Size: 0x25598
        Compressed: Yes
        Version: 2
        Checksum Type: 32-bit
        Image Checksum: 0x5d1a3120
        Target Checksum: 0xa5230a59
        Signature Type: 1 (User key)
        Application Type: 2 (Bootloader Update)
        Header Checksum: 0xa82f2048
        Filename: SD7.hex
        Extra data (hex):
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000
        Signature (hex):
0dce5694b48da98cc4c9cd0ab590a63c281486992fcf06e9bac9efb8fd2eda9ceb12c91d4daabb802ea2b2bef3
7d525218801b4ab5f9744e62502a44dc333400
        Version requirements: 2
        Version requirement 0: Application Type 1 (Main Application) must = [e]version 2
        Version requirement 1: Application Type 4 (User configuration A) must =[e] version
2

Section 1:
        Section Start: 0x24000
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

```
        Section End: 0x5078a
        Section Size: 0x2c78a
        Target: 0 (Internal flash)
        Target Start: 0x27000
        Target End: 0x60a1c
        Target Size: 0x39a1c
        Compressed: Yes
        Version: 2
        Checksum Type: 32-bit
        Image Checksum: 0x5468df78
        Target Checksum: 0xb35ff51f
        Signature Type: 1 (User key)
        Application Type: 2 (Bootloader Update)
        Header Checksum: 0xb745c372
        Filename: App.hex
        Extra data (hex):
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000
        Signature (hex):
930193d2262a35f2519e5cb415d6c122f0104b7f68ead0b1f1691ddc04cc12ca97eb9ca96803a31dfeda4d056f
80085a28877c312a95e3a9b97a3249b4fffd50
        Version requirements: 2
        Version requirement 0: Application Type 2 (Bootloader Update) must = [e] version 2
        Version requirement 1: Application Type 4 (User configuration A) must =[e] version
2

Section 2:
        Section Start: 0x51000
        Section End: 0x6f630
        Section Size: 0x1e630
        Target: 0 (Internal flash)
        Target Start: 0x98000
        Target End: 0x96630
        Target Size: 0x1e630
        Compressed: No
        Version: 2
        Checksum Type: 32-bit
        Image Checksum: 0x6213619c
        Target Checksum: 0x6213619c
        Signature Type: 1 (User key)
        Application Type: 4 (User configuration A)
        Header Checksum: 0x96f6a66a
        Filename: Cfg.hex
        Extra data (hex):
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000
        Signature (hex):
6fbc061af76d2b06e652ea360b0a746001c3a34f715ed001cf4f64b74b66bbda7cfc38d57e7c278d852b859cc5
4e1b1735b223f872f794fc157f490a5f65aa08
        Version requirements: 2
        Version requirement 0: Application Type 1 (Main Application) must = [e]version 2
        Version requirement 1: Application Type 2 (Bootloader Update) must = [e] version 2

Total uncompressed section size: 513508
Total compressed section size: 437415
Section compression: 15%

1283307 bytes written to UpgradePackage.hex successfully.
453376 bytes written to UpgradePackage.ubu successfully.
```

When the above firmware package is programmed to a module, it only upgrades the sections if all three sections are present in the upgrade file. If one of the sections is missing, then the upgrade data remains on the module, but the upgrade is not

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

processed. This prevents the module from getting into a possible erroneous situation whereby a configuration file for a specific firmware version is not present or possibly causing a boot-loop with different versions of executable code.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

7 List of Indexes

The following tables lists the index which can be used with the yX and xX commands to query and set values for the bootloader respectively (where X is an item in the Value column):

Table 5: Index list

Value	Name	Configuration Details
2	User application public key	Manually Inputting a Public Key into the Bootloader
3	User application public key set	
4	Readback protection	Enabling Readback Protection
5	CPU debug protection	Enabling CPU Debug Protection
6	Block UART readback	Blocking UART Data Readback
7	QSPI mode	Configuring QSPI Power/Mode
8	UART unlock key set	Setting/Using a Bootloader Unlock Key
9	UART unlock key	
A	Full erase mode	Changing Full-Erase Security Level
B	QSPI size limit	Enabling QSPI Usage For User Applications
C	QSPI user start address	
D	Block UART verification	Blocking UART Data Verification
E	UART verification minimum size	Limiting UART Data Verification
F	Boot verification	Boot Verification
G	Main application/Soft-device section protection	Main Application/Soft-Device Section Protection
I	Erase section mode	Setting Erase Section Mode

8 Setting/Using a Bootloader Unlock Key

A bootloader unlock key can be used to restrict access to a device via the UART when in bootloader mode. It can also require that a code be sent before any data reading/writing can take place; this increases the security of IP and application integrity. An unlock code consists of 64 bytes (512 bits) of data.

To set a bootloader unlock key, follow these steps:

1. Enter bootloader mode on the pinnacle 100, this can be achieved by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up (on the development board, hold down SW1 and press the reset button).
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:
   - Hardware flow control enabled
   - Baud rate set to 115200
   - 1 stop bit
   - No parity

   The CTS status should be green to indicate that the module is ready to accept commands.
3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. In UwTerminalX this takes the form of a slash (\)followed by two numbers.
4. Right-click on the UwTerminalX window and select the automation option.
5. Check the Un-Escape Strings box.
6. In the top field, add the following data: *p\0f\51\2a\51*
7. In the second field, add the following data: x9 and add 64 bytes of data to the end of it. This becomes the bootloader unlock key.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

Note: ASCII hex-escape sequences can be used by adding a slash (\) followed by two hex characters. For example: **\00** is a 0x00 null character and **\ab** is a 0xab character. These are not ASCII-printable but can be used in the unlock codes.

8. Press the top send button. The module should respond with an ***a*** which indicates that it is now unlocked.
9. Press the second send button. The module should respond with an ***a*** which means that the key is set.
   - If it responds with an ***f***, then there was an error whilst setting the unlock key (most likely because a key is already set or because the module is lacking a license).
   - If there is no response, then check to ensure you included 32 characters

   Reset the module and put it back into bootloader mode before retrying or an invalid unlock key could be programmed to the module.
10. Reset the module by pressing **Z** and **Enter**.
11. Press the top send button. The module should respond with an ***a*** indicating that it is now unlocked.
12. In the automation window in the third field, add the following data:

    u00000000000000000000000000000000000000000000000000000000000000000
13. In the automation window in the fourth field, add the following data: u and append the key you have in the second field (minus the x9).
14. Press the third send button. The module should return an ***f*** to indicate that the unlock code is incorrect.
15. Press the fourth send button. The module should return an ***a*** to indicate that the unlock code is successful.
16. Ensure that the bootloader unlock key is saved so that it can be unlocked in future.

Once a bootloader unlock key has been set, it cannot be changed or disabled unless a full erase is performed as described in Restoring to Factory Defaults (via UART) or Full-Chip Erase/Recovery (via SWD). Any time that an update image is used in UwFlashX, the bootloader unlock key must be provided in the utility under the Bootloader Unlock Code tab.

9 Enabling Readback Protection

Readback Protection is a hardware feature of the nRF52840 chip which prevents SWD access to the chip. This prevents the application code of the MCU from being read out which keeps the IP safe and increases security.

Note: Once readback protection is enabled, writing data to the QSPI memory using *nrfjprog* no longer functions.

To enable readback protection, follow these steps:

1. Enter bootloader mode on the pinnacle 100. Do this by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up (on the development board, hold down SW1 and press the reset button).
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:
   - Hardware flow control enabled
   - Baud rate set to 115200
   - 1 stop bit
   - No parity

   The CTS status should be green to indicate that the module is ready to accept commands.
3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. in IwTerminalX, this takes the form of a slash (\) followed by two numbers.
4. Right-click on the UwTerminalX window and select the automation option.
5. Check the Un-Escape Strings box.
6. In the top field, add the following data: ***p\0f\51\2a\51***
7. In the second field, add the following data: ***y4***

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

8. In the third field, add the following data: ***x4\01***
9. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
10. Press the second send button. The module should respond with ***y4\00***. The *00* indicates that readback protection is currently disabled.
11. Press the third send button. The module should respond with an ***a*** which means that the option is set and that the protection will be enabled at next bootup.
    - If it responds with an ***f***, then there was an error whilst setting the functionality (most likely because the module is lacking a license).
12. Reset the module by pressing **Z** and **Enter**.

    The module enables readback protection and boots to the bootloader.
13. Press the top send button. The module should respond with an ***a*** which indicates that it is unlocked.
14. Press the second send button. The module should respond with ***y4\01***. The 01 indicates that readback protection is currently enabled.
15. Readback protection can be verified by using *nrfjprog* to read data from the module.

    For example: ***nrfjprog -f NRF52 --memrd 0x1000 --n 8*** should result in an error that readback protection is active and that the data could not be read.

    Once readback protection is set, it can only be removed by performing a recover operation via SWD. See the Full-Chip Erase/Recovery (via SWD) section for details on how to do this.

10 Enabling CPU Debug Protection

CPU Debug Protection is a hardware feature of the nRF52840 chip which prevents the FPB and ETM functionalities on the chip. This helps keeps algorithms safe and increases security.

To enable CPU debug protection, follow these steps:

1. Enter bootloader mode on the pinnacle 100. Do this by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up. On the development board, hold down SW1 and press the reset button.
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:
    - Hardware flow control enabled
    - Baud rate set to 115200
    - 1 stop bit
    - No parity

    The CTS status should be green to indicate that the module is ready to accept commands.
3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. In UwTerminalX, this takes the form of a slash (\) followed by two numbers.
4. Right click on the UwTerminalX window and select **Automation**.
5. Check the Un-Escape Strings box.
6. In the top field, add the following data: ***p\0f\51\2a\51***
7. In the second field, add the following data: ***y5***
8. In the third field, add the following data: ***x5\01***
9. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
10. Press the second send button. The module should respond with **y5\00**. The *00* indicates that CPU debug protection is currently disabled.
11. Press the third send button. The module should respond with an ***a*** which indicates that the option is set and that the protection will be enabled at next bootup.

    If it responds with an ***f***, then there was an error whilst setting the functionality (most likely because the module is lacking a license).

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

12. Reset the module by pressing **Z** and **Enter**.

    The module enables CPU debug protection and boots to the bootloader.

13. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
14. Press the second send button. The module should respond with ***y5\01***. The *01* indicates that CPU debug protection is currently enabled.

    Once CPU debug protection is set, it can only be removed by performing a recover operation via SWD. See the Full-Chip Erase/Recovery (via SWD) section for details on how to do this.

11 Configuring QSPI Power/Mode

The QSPI flash on the Pinnacle 100 has two modes of operation:

- **Ultra-low power mode** – Operations take considerably longer than in high performance mode, but the flash consumes much less power. This makes this mode ideal for battery-operated devices.
  - Maximum of 24uA when in standby mode
  - Maximum of 4mA when in read mode
  - Maximum of 7mA in programming/erase mode
- **High performance mode** – Operations complete more quickly than in ultra-low power mode, but the flash consumes much more power. This makes this mode ideal for applications where the speed is important or lower power consumption is not a concern.
  - Maximum of 50uA when in standby mode
  - Maximum of 9mA in read mode
  - Maximum of 10mA in programming/erase mode.

The operating mode of the QSPI flash can be changed from the bootloader. To view the current operating mode of QSPI, follow these steps:

1. Enter bootloader mode on the pinnacle 100. Do this by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up. On the development board, hold down SW1 and press the reset button.
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:
   - Hardware flow control enabled
   - Baud rate set to 115200
   - 1 stop bit
   - No parity

   The CTS status should be green to indicate that the module is ready to accept commands.
3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. In UwTerminalX, this takes the form of a slash (\) followed by two numbers.
4. Right-click on the UwTerminalX window and select the **Automation**.
5. Check the Un-Escape Strings box.
6. In the top field, add the following data: ***p\0f\51\2a\51***
7. In the second field, add the following data: **y7**
8. Press the top send button. The module should respond with an ***a*** which indicates that it is unlocked.
9. Press the second send button. The module should respond with ***y7\00*** or ***y7\01.*** If *00*, then the QSPI is in ultra-low power mode. If *01*, then the QSPI is in high performance mode.

To change the operating mode of QSPI, follow these steps:

1. Enter bootloader mode on the pinnacle 100. Do this by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up. On the development board, hold down SW1 and press the reset button.
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:
   - Hardware flow control enabled
   - Baud rate set to 115200

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

- 1 stop bit
- No parity

The CTS status should be green to indicate that the module is ready to accept commands.

3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. In UwTerminalX, this takes the form of a slash (\) followed by two numbers.
4. Right-click on the UwTerminalX window and select the **Automation**.
5. Check the Un-Escape Strings box.
6. In the top field, add the following data: ***p\0f\51\2a\51***
7. In the second field, add the following data: ***y7***
8. In the third field:
   - Add the following data if you want to switch to ultra-low power mode: ***x7\00***
   - Add the following if you want to switch to high performance mode: ***x7\01***
9. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
10. Press the third send button. The module should respond with an ***a*** indicating that the QSPI operating mode set and will be enabled at next bootup.

    If it responds with an ***f***, then there was an error whilst setting the functionality (most likely because the module is lacking a license).
11. Reset the module by pressing **Z** and **Enter**.

    The module changes the QSPI operating mode and boots to the bootloader.
12. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
13. Press the second send button. The module should respond with:
    - ***y7\00*** if ultra-low power mode is set
    - ***y7\01*** if high performance mode is set

12 Blocking UART Data Readback

By default, the bootloader on the Pinnacle 100 has operations for reading, writing, and erasing data on the QSPI. The functionality can be limited by setting a bootloader unlock key (see the Enabling Readback Protection section for details) or by enabling UART Data Readback protection which prevents the UART read commands from working.

To enable UART Data Readback protection, follow these steps:

1. Enter bootloader mode on the pinnacle 100. Do this by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up. On the development board, hold down SW1 and press the reset button.
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:
   - Hardware flow control enabled
   - Baud rate set to 115200
   - 1 stop bit
   - No parity

   The CTS status should be green to indicate that the module is ready to accept commands.
3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. In UwTerminalX, this takes the form of a slash (\) followed by two numbers.
4. Right-click on the UwTerminalX window and select **Automation**.
5. Check the Un-Escape Strings box.
6. In the top field, add the following data: ***p\0f\51\2a\51***
7. In the second field, add the following data: ***y6***
8. In the third field, add the following data: ***x6\01***
9. In the fourth field, add the following data: ***r\00\01\00\00\04***

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

10. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
11. Press the third send button. The module should respond with an ***a*** indicating that UART data readback is set and will be enabled at next bootup.

    If it responds with an ***f***, then there was an error whilst setting the functionality (most likely because the module is lacking a license).
12. Reset the module by pressing **Z** and **Enter**.

    The module enables UART data readback protection and boots to the bootloader.
13. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
14. Press the second send button. The module should respond with ***y6\01*** indicating that UART data readback protection is enabled.
15. To confirm that UART data readback protection is active, click the fourth send button.

    An ***f*** response confirms that UART data readback is no longer allowed.

    A ***d*** response indicates that UART data readback protection was not successfully applied.

13 Blocking/Limiting UART Data Verification

From the bootloader, data stored on the QSPI chip can be verified. This ensures that data written from an update file is correctly transferred to the module without errors. Whilst this feature is useful for data verification, it presents a potential attack point which can be used to create a duplicate of the data stored in the module. Because of this, there are options to disable this functionality (if it is not required) or limit the minimum size of data verification.

Note: The verification command is subject to the Setting/Using a Bootloader Unlock Key functionality. If set, it requires that the bootloader unlock code is issued before it can be used.

Because you can specify the size of data to be verified, a malicious attacker could try to verify every byte on the module by guessing the checksum. Once the attacker knows the checksum of the byte, it would also know the contents of that single byte and could then proceed onto the next byte. This is a very slow process but is a possible attack vector; limiting the UART data verification can make it almost impossible for a malicious attacker to do this.

13.1 Limiting UART Data Verification

The checksum used by the bootloader is 32-bits. The minimum size of data on the QSPI which can be verified can be set to a multiple of the flash write size (4 bytes) up to a value of 16,384 (16 KB). Because the checksum represents a very finite value in comparison to what the minimum UART data verification size can be, it becomes increasingly difficulty to work out what the underlying data actually is.

For example, if a size of 32 bytes is chosen, there are 256-bits which can have 1.15e+77 possible combinations. The 32-bit checksum can only represent around 4.3 billion combinations and so there many permutations of the underlying data that can give the same checksum value. This is unlikely to happen with the use-case of verifying that data is written correctly but is very likely to arise when guessing what the underlying data consists of.

To check the current UART data verification minimum size, follow these steps:

1. Enter bootloader mode on the pinnacle 100. You can achieve this by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up (on the development board, hold down SW1 and press the reset button).
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:
   - Hardware flow control enabled
   - Baud rate set to 115200
   - 1 stop bit
   - No parity

   The CTS status should be green to indicate that the module is ready to accept commands.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. In UwTerminalX, this takes the form of a slash (\) followed by two numbers.
4. Right click on the UwTerminalX window and select **Automation**.
5. Check the Un-Escape Strings box.
6. In the top field, add the following data: ***p\0f\51\2a\51***
7. In the second field, add the following data: ***yE***
8. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
9. Press the second send button. The module should respond with ***yE*** followed by a 16-bit little-endian hex value. This indicates the current minimum UART data verification size. Move the first two hex values to the right-hand side and that is the size in hex. For example: yE\00\01 is 0x0100 = 256 bytes

To change the current UART data verification minimum size follow these steps:

1. Enter bootloader mode on the pinnacle 100. You can achieve this by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up (on the development board, hold down SW1 and press the reset button).
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:
   - Hardware flow control enabled
   - Baud rate set to 115200
   - 1 stop bit
   - No parity

   The CTS status should be green to indicate that the module is ready to accept commands.
3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. In UwTerminalX, this takes the form of a slash (\) followed by two numbers.
4. Right click on the UwTerminalX window and select **Automation**.
5. Check the Un-Escape Strings box.
6. In the top field, add the following data: ***p\0f\51\2a\51***
7. In the second field, add the following data: xE and append a 16-bit little-endian hex value of the minimum UART verification size which is escaped with back-slashes

   For example, if you want a value of 256, 256 = 0x0100; in little endian without the 0x prefix it is 0001, then add two backslashes \00\01 – final command is xE\00\01
8. In the third field, add the following data: **yE**
9. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
10. Press the second send button. The module should respond with an ***a*** indicating that the minimum UART verification size option is set and will be enabled at next bootup.

    If it responds with an ***f***, then there was an error whilst setting the functionality (most likely because the module is lacking a license or because the minimum UART verification size is larger or equal to the value trying to be set).
11. Reset the module by pressing **Z** and **Enter**.
12. The module boots to the bootloader.
13. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
14. Press the third send button. The module should respond with ***yE*** and the 16-bit little hex value of the minimum UART verification size.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

13.2 Blocking UART Data Verification

By blocking UART data verification, this command no longer functions. It should be used alongside the Blocking UART Data Readback feature to block being able to read back data from the bootloader.

To check the if UART data verification is blocked, follow these steps:

1. Enter bootloader mode on the pinnacle 100. You can achieve this by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up (on the development board, hold down SW1 and press the reset button).
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:
   - Hardware flow control enabled
   - Baud rate set to 115200
   - 1 stop bit
   - No parity

   The CTS status should be green to indicate that the module is ready to accept commands.
3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. In UwTerminalX, this takes the form of a slash (\) followed by two numbers.
4. Right click on the UwTerminalX window and select **Automation**.
5. Check the Un-Escape Strings box.
6. In the top field, add the following data: ***p\0f\51\2a\51***
7. In the second field, add the following data: ***yD***
8. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
9. Press the second send button. The module should respond with ***yA\0*** followed by a 0 if UART data verification is enabled or a 1 if UART data verification is disabled.

To disable UART data verification, follow these steps:

1. Enter bootloader mode on the pinnacle 100. You can achieve this by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up (on the development board, hold down SW1 and press the reset button).
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:
   - Hardware flow control enabled
   - Baud rate set to 115200
   - 1 stop bit
   - No parity

   The CTS status should be green to indicate that the module is ready to accept commands.
3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. In UwTerminalX, this takes the form of a slash (\) followed by two numbers.
4. Right click on the UwTerminalX window and select **Automation**.
5. Check the Un-Escape Strings box.
6. In the top field, add the following data: ***p\0f\51\2a\51***
7. In the second field, add the following data: ***xD\01***
8. In the third field, add the following data: ***yD***
9. In the fourth field, add the follow data: ***v\00\00\00\00\00\00\00\00\00\00\00\00***
10. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
11. Press the second send button. The module should respond with an ***a*** which means that UART data verification value is set and will be disabled from next bootup.

    If it responds with an ***f***, then there was an error whilst setting the functionality (most likely because the module is lacking a license or because UART data verification is already blocked).

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

12. Reset the module by pressing **Z** and **Enter**.

    The module disables UART data verification and boots to the bootloader.
13. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
14. Press the third send button. The module should respond with ***yD\01***.
15. Press the fourth send button. The module should respond with an ***f*** indicating that UART data verification is disabled.

14 Setting Erase Section Mode

As shown in the Erase Sections section, there are application types with IDs 9 and 10 which are *erase sections*. These sections contain no data but erase sectors of flash on the nRF52840 either once or every time the bootloader starts. By default, there is no restriction placed on these sections although it can be applied. There are four types of erase section mode (Table 6).

Table 6: Erase section mode types

Bitmask	Name	Description
0	None	All erase sections allowed without restrictions (default)
1	Block lower	Block using erase sections with an older version that the last one used
2	Block equal	Block using erase sections with an qual version that the last one used (makes always erase sections behave as erase once sections)
4	Block user-signed	Blocks using erase sections signed with the user signing key
8	Block Laird Connectivity-signed	Blocks using erase sections signed with the Laird Connectivity signing key

You can add options together whereby all options being set entirely block erase sections. If the erase section mode is already configured on a module, the security level can be increased by adding in additional bitmask values but cannot be decreased.

To check the current erase section mode:

1. Enter bootloader mode on the pinnacle 100. You can achieve this by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up (on the development board, hold down SW1 and press the reset button).
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:
   - Hardware flow control enabled
   - Baud rate set to 115200
   - 1 stop bit
   - No parity

   The CTS status should be green to indicate that the module is ready to accept commands.
3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. In UwTerminalX, this takes the form of a slash (\) followed by two numbers.
4. Right click on the UwTerminalX window and select **Automation**.
5. Check the Un-Escape Strings box.
6. In the top field, add the following data: ***p\0f\51\2a\51***
7. In the second field, add the following data: ***yI***
8. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
9. Press the second send button. The module should respond with ***yI\0*** followed by a hex value which indicates the current bitmask of boot verification options listed above.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

To change the current erase section mode, follow these steps:

1. Enter bootloader mode on the pinnacle 100. You can achieve this by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up (on the development board, hold down SW1 and press the reset button).
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:
   - Hardware flow control enabled
   - Baud rate set to 115200
   - 1 stop bit
   - No parity

   The CTS status should be green to indicate that the module is ready to accept commands.
3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. In UwTerminalX, this takes the form of a slash (\) followed by two numbers.
4. Right click on the UwTerminalX window and select **Automation**.
5. Check the Un-Escape Strings box.
6. In the top field, add the following data: ***p\0f\51\2a\51***
7. In the second field, add ***x\0*** and append a hex value of the bitmask boot verification level which corresponds to the table above.
8. In the third field, add the following data: ***y***
9. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
10. Press the second send button. The module should respond with an ***a*** which means that the erase section mode option is set and will be enabled at next bootup.

    If it responds with an ***f***, then there was an error whilst setting the functionality (most likely because the module is lacking a license or because the current erase section mode is higher or equal to the level you are attempting to set).
11. Reset the module by pressing **Z** and **Enter**.

    The module boots to the bootloader.
12. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
13. Press the third send button. The module should respond with ***y\0*** and the hex value of the erase section mode set.

15 Main Application/Soft-Device Section Protection

Protection can be applied to the main application and soft-device sections so that other upgrade sections cannot interact with the data on the module from these sections. For example, if an application is present from 0x1000 – 0x10000 and there is an erase section partition which erases a sector at 0x9000 then, by default, this sector is erased. This allows for applications that include specific sectors with configuration data to be cleared, if necessary. There are two types of section protection (Table 7).

Table 7: Section protection types

Bitmask	Name	Description
0	None	A (default)
1	Main application	Prevent other sections from modifying data in the main application section
2	Soft-device	Prevent other sections from modifying data in the soft-device section

To check the current main application/soft-device section protection level, follow these steps:

1. Enter bootloader mode on the pinnacle 100. You can achieve this by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up (on the development board, hold down SW1 and press the reset button).
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:
   - Hardware flow control enabled
   - Baud rate set to 115200
   - 1 stop bit

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

- No parity

The CTS status should be green to indicate that the module is ready to accept commands.

3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. In UwTerminalX, this takes the form of a slash (\) followed by two numbers.
4. Right click on the UwTerminalX window and select **Automation**.
5. Check the Un-Escape Strings box.
6. In the top field, add the following data: ***p\0f\51\2a\51***
7. In the second field, add the following data: ***yG***
8. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
9. Press the second send button. The module should respond with ***yG\0*** followed by a hex value which indicates the current bitmask of main application/soft-device section protection levels listed above.

To change the current main application/soft-device section protection level, follow these steps:

1. Enter bootloader mode on the pinnacle 100. You can achieve this by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up (on the development board, hold down SW1 and press the reset button).
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:
   - Hardware flow control enabled
   - Baud rate set to 115200
   - 1 stop bit
   - No parity

   The CTS status should be green to indicate that the module is ready to accept commands.
3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. In UwTerminalX, this takes the form of a slash (\) followed by two numbers.
4. Right click on the UwTerminalX window and select **Automation**.
5. Check the Un-Escape Strings box.
6. In the top field, add the following data: ***p\0f\51\2a\51***
7. In the second field, add ***xG\0*** and append a hex value of the bitmask boot verification level which corresponds to the previous table (Table 7).
8. In the third field, add the following data: ***yG***
9. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
10. Press the second send button. The module should respond with an ***a*** which means that the main application/soft-device section protection level option is set and will be enabled at next bootup.

    If it responds with an ***f***, then there was an error whilst setting the functionality (most likely because the module is lacking a license or because the current main application/soft-device section protection level is higher or equal to the level you are attempting to set).
11. Reset the module by pressing **Z** and **Enter**.

    The module boots to the bootloader.
12. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
13. Press the third send button, the module should respond with ‘yG\0’ and the hex value bitmask main application/soft-device section protection level options which were set

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

16 Boot Verification

Boot verification is a security enhancement option that prevents the module from booting code which is not correctly signed or which has been modified. There are eight options for boot verification (see Table 8).

Table 8: Boot verification options

Bitmask	Name	Description
0	None	No boot verification (default)
1	MBR	Checks that the Nordic MBR is valid
2	Bootloader	Checks that the bootloader is valid
4	External function	Checks that the external function is valid
8	Main application	Checks that the main user application is valid
16	Soft-device	Checks that the soft-device (if present) is valid
32	Compromised boot chain	Entirely prevents boot if MBR/bootloader are compromised
64	Sections required	Only boots the user application if there is details for it (from a .ubu file). Does not boot user application, even if present, if this information is missing
128	Prevent boot	If boot verification fails, does not boot to the UART bootloader, prevents boot, and enters a low power sleep mode.

Options can be added together with all options being enabled providing the most secure environment. If boot verification is already configured on a module, the security level can be increased by adding in more bitmask values but cannot be decreased.

To check the current boot verification level, follow these steps:

1. Enter bootloader mode on the pinnacle 100. You can achieve this by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up (on the development board, hold down SW1 and press the reset button).
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:
   - Hardware flow control enabled
   - Baud rate set to 115200
   - 1 stop bit
   - No parity

   The CTS status should be green to indicate that the module is ready to accept commands.
3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. In UwTerminalX, this takes the form of a slash (\) followed by two numbers.
4. Right click on the UwTerminalX window and select **Automation**.
5. Check the Un-Escape Strings box.
6. In the top field, add the following data: ***p\0f\51\2a\51***
7. In the second field, add the following data: ***yF***
8. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
9. Press the second send button. The module should respond with ***yF*** followed by a hex value which indicates the current bitmask of boot verification options listed above.

To change the current boot verification level, follow these steps:

1. Enter bootloader mode on the pinnacle 100. You can achieve this by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up (on the development board, hold down SW1 and press the reset button).
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:
   - Hardware flow control enabled
   - Baud rate set to 115200
   - 1 stop bit

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

- No parity

The CTS status should be green to indicate that the module is ready to accept commands.

3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. In UwTerminalX, this takes the form of a slash (\) followed by two numbers.
4. Right click on the UwTerminalX window and select **Automation**.
5. Check the Un-Escape Strings box.
6. In the top field, add the following data: ***p\0f\51\2a\51***
7. In the second field, add ***xF*** and append a hex value of the bitmask boot verification level which corresponds to the previous table (Table 8).
8. In the third field, add the following data: ***yF***
9. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
10. Press the second send button, the module should respond with an ***a*** which means that the boot verification option is set and will be enabled at next bootup.

    f it responds with an ***f***, then there was an error whilst setting the functionality (most likely because the module is lacking a license or because the current boot verification level is higher or equal to the level you are attempting to set).
11. Reset the module by pressing **Z** and **Enter**.
12. The module boots to the bootloader.
13. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
14. Press the third send button. The module should respond with ***yF*** and the hex value bitmask of the boot verification options which were set.

17 CHANGING FULL-ERASE SECURITY LEVEL

The Pinnacle 100 bootloader has a full-erase command which can be issued from the UART. This erases all data on the module and restores it to factory defaults (except for the readback protection and CPU debug security bits). For details on this feature, refer to the Restoring to Factory Defaults (via UART) section.

To improve security, this functionality can be disabled (which leaves SWD as the only way to erase the module, as detailed in Full-Chip Erase/Recovery (via SWD)) or protected (which prevents accidental erasure of modules). Security can be tightened at any time by going to a higher security level but it cannot be decreased.

The following table () displays the security levels of the Full-Erase command.

Table 9: Full-erase command security levels

Level	Value	Description
0	Always Allow (default)	Allows the full-erase command to be used at any time without any restrictions
1	Allow if no Bootloader Unlock Key or only allow if Bootloader Unlock Key is set and entered correctly	Allows the full-erase command to be used if there is no bootloader unlock key present on the module, or if there is one then it must have been entered correctly before the full erase command can be used
2	Allow only if Bootloader Unlock Key is set and entered correctly	Allows the full-erase command to be used only if a bootloader unlock key is set and if it has been entered correctly
3	Always Deny	Prevents the full-erase command from being used

To check the current security level of the Full-Erase command, follow these steps:

1. Enter bootloader mode on the pinnacle 100. You can achieve this by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up (on the development board, hold down SW1 and press the reset button).
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

- Hardware flow control enabled
- Baud rate set to 115200
- 1 stop bit
- No parity

The CTS status should be green to indicate that the module is ready to accept commands.

3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. In UwTerminalX, this takes the form of a slash (\) followed by two numbers.
4. Right click on the UwTerminalX window and select **Automation**.
5. Check the Un-Escape Strings box.
6. In the top field, add the following data: ***p\0f\51\2a\51***
7. In the second field, add the following data: ***yA***
8. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
9. Press the second send button. The module should respond with ***yA\0*** followed by a number which indicates the current security mode as listed above.

To change the current security level of the Full-Erase command, follow these steps:

1. Enter bootloader mode on the pinnacle 100. You can achieve this by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up (on the development board, hold down SW1 and press the reset button).
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:
   - Hardware flow control enabled
   - Baud rate set to 115200
   - 1 stop bit
   - No parity

   The CTS status should be green to indicate that the module is ready to accept commands.
3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. In UwTerminalX, this takes the form of a slash (\) followed by two numbers.
4. Right click on the UwTerminalX window and select **Automation**.
5. Check the Un-Escape Strings box.
6. In the top field, add the following data: ***p\0f\51\2a\51***
7. In the second field, add ***xA\0*** and append a security level which corresponds to one of the levels displayed in Table 9.
8. In the third field, add the following data: ***yA***
9. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
10. Press the second send button. The module should respond with an ***a*** which means that the full-erase security level is set and will be enabled at next bootup.

    f it responds with an ***f***, then there was an error whilst setting the functionality (most likely because the module is lacking a license or because the current full-erase security level is higher or equal to the level you are attempting to set).
11. Reset the module by pressing **Z** and **Enter**.

    The module will enable full-erase security protection and boot to the bootloader
12. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
13. Press the third send button. The module should respond with ***yA\0*** and the level of full-erase security which was set.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

18 Enabling QSPI Usage For User Applications

The 8MB QSPI flash chip on the Pinnacle 100 module is, by default, set for exclusive use by the bootloader although you can optionally set it up that up to half of the memory (4MB) can be used by the user application for any purpose (this data is entirely ignored by the bootloader).

Note: Regardless of what split of memory is set for use by the bootloader and user application, when performing a full-erase as detailed in the Restoring to Factory Defaults (via UART) section, the entire contents of the QSPI memory is erased.

There are two values which can be retrieved from the bootloader which specify the details of how much of the QSPI data can be used by a user application. If both of these values are set to 0 (default), then no QSPI flash space is reserved for use by the user application.

There is a restriction when updating the amount of QSPI flash space that can be used by the user application – the size can always be increased in sectors up to half of the flash size but cannot be reduced once set.

To view the user application QSPI flash size limit and QSPI flash start address, follow these steps:

1. Enter bootloader mode on the pinnacle 100. You can achieve this by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up (on the development board, hold down SW1 and press the reset button).
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:
   - Hardware flow control enabled
   - Baud rate set to 115200
   - 1 stop bit
   - No parity

   The CTS status should be green to indicate that the module is ready to accept commands.
3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. In UwTerminalX, this takes the form of a slash (\) followed by two numbers.
4. Right click on the UwTerminalX window and select **Automation**.
5. Check the Un-Escape Strings box.
6. In the top field, add the following data: ***p\0f\51\2a\51***
7. In the second field, add the following data: ***yB***
8. In the third field, add the following data: ***yC***
9. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
10. Press the second send button. The module should respond with ***yB*** followed by two sets of hexadecimal numbers which indicates the user application QSPI flash size sectors.

    Note: This is in little endian so a response of ***\10\00*** means 0x0010 which is 16 sectors.

11. Press the third send button. The module should respond with ***yC*** followed by four sets of hexadecimal numbers which indicates the user application QSPI flash start address.

    Note: This is in little endian so a response of ***\00\00\60\00*** means 0x00600000.

To set the user application QSPI flash size limit, follow these steps:

1. Enter bootloader mode on the pinnacle 100. You can achieve this by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up (on the development board, hold down SW1 and press the reset button).
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

- Hardware flow control enabled
- Baud rate set to 115200
- 1 stop bit
- No parity

The CTS status should be green to indicate that the module is ready to accept commands.

3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. In UwTerminalX, this takes the form of a slash (\) followed by two numbers.
4. Right click on the UwTerminalX window and select **Automation**.
5. Check the Un-Escape Strings box.
6. In the top field, add the following data: ***p\0f\51\2a\51***
7. In the second field, add the following data: ***yB***
8. In the third field, add ***xB*** and add two little endian bytes escaped with backslaches (\) which indicate the number of sectors to assign to the user application. For example: ***xB\10\00*** is 0x0010 which is 16 sectors; each sector is 4KB.
9. Press the top send button. The module should respond with an ***a*** indicating that it is unlocked.
10. Press the third send button. The module should respond with ***a*** indicating success.

    If you get an ***f***, then it means there was an error either with the value supplied (not valid or less than the existing value), there is a hardware issue, or the module lacks a license.
11. Press the second send button. The module should respond with ***yB*** followed by the two sets of hexadecimal numbers from the third field (which indicates the user application QSPI flash size sectors).

    The QSPI user section can now be used from the user application

19 Viewing Bootloader Version Information

The bootloader on the Pinnacle 100 module supports firmware updates. There may be many versions of the bootloader; if support is requested for the bootloader, then the version information of the bootloader you are using is required.

To retrieve the bootloader version information, follow these steps:

1. Enter bootloader mode on the pinnacle 100. You can achieve this by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up (on the development board, hold down SW1 and press the reset button).
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:
   - Hardware flow control enabled
   - Baud rate set to 115200
   - 1 stop bit
   - No parity

   The CTS status should be green to indicate that the module is ready to accept commands.
3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. In UwTerminalX, this takes the form of a slash (\) followed by two numbers.
4. Press **M** and **Enter**.

   The module emits version information which can be used to check if the bootloader is the latest version or not included in any request for support. It will resemble the following:

```
Model: 124, Variant: 0, Name: PINNACLE100, Bootloader version: 3 (FUP: 6.011, Ext.
struct: 1, Ext. function: 2)
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

20 Restoring to Factory Defaults (via UART)

Once a Pinnacle 100 module is programmed with settings such as application and public key, this information cannot be changed by programming another key. The module *can* be restored to factory default which erases all data on the module except the license key, allowing it to be re-used if a wrong key is programmed or if the programmed application is not valid.

Note: If the full-erase block has been enabled as described in the Blocking Full-Erase Command section, then issuing the full-erase command does not work and an error is returned. The only way to erase the module in this instance is to perform an erase using SWD as described in the Full-Chip Erase/Recovery (via SWD) section.

Note: Restoring to factory defaults does not erase or reset the readback protection security option or the CPU debug protection. The only way to remove this protection is to perform a recovery operation via SWD, which is detailed in the Full-Chip Erase/Recovery (via SWD) section.

Returning a unit to factory default settings can take up to approximately three minutes but is usually quicker. It depends upon the age and utilization of the device.

To perform a restore process, follow these steps:

1. Enter bootloader mode on the pinnacle 100. You can achieve this by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up (on the development board, hold down SW1 and press the reset button).
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:
   - Hardware flow control enabled
   - Baud rate set to 115200
   - 1 stop bit
   - No parity

   The CTS status should be green to indicate that the module is ready to accept commands.
3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. In UwTerminalX, this takes the form of a slash (\) followed by two numbers.
4. Right click on the UwTerminalX window and select **Automation**.
5. Check the Un-Escape Strings box.
6. In the top field, add the following data: ***p\0f\51\2a\51***
7. In the second field, add the following data: ***\7f\7f***
8. If the automation window is in the way of the terminal window, move it so that both are visible.
9. Click **Send** next to the top field to unlock the bootloader. The module should respond with an ***a***. If it does not, then there is an issue with your setup or configuration, or you are using a different (incompatible) module.
10. Click **Send** next to the second field to begin the restore process.The module should not emit a response for some time. Once complete, the module should output ***a***. If it emits an ***f***, then an error occurred during the erase process. The erase process usually takes two minutes but may take up to five minutes under certain circumstances.

    The module is now restored to factory defaults, excluding resetting any security bits. It can now be used or programmed as desired.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

21 Full-Chip Erase/Recovery (via SWD)

Performing a full-chip erase via SWD erases all data on the Pinnacle 100 module including bootloader, settings, and update images (this includes security bits which include readback protection and CPU debug protection).

Note: A full-chip erase does not erase modem settings or firmware.

To perform a full-chip erase, follow these steps:

1. Enter bootloader mode on the pinnacle 100. You can achieve this by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up (on the development board, hold down SW1 and press the reset button).
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:
   - Hardware flow control enabled
   - Baud rate set to 115200
   - 1 stop bit
   - No parity

   The CTS status should be green to indicate that the module is ready to accept commands.
3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. In UwTerminalX, this takes the form of a slash (\) followed by two numbers.
4. Type y0 and press enter to retrieve the license from the module. It should respond with something similar to the following: ***y0B297902D39C8D34E2A0A***
5. Remove the ***y0*** from the front of the string to extract the license. Keep this safe until the unit is fully erased and bootloader functionality is restored. With the above response, the license is: ***B297902D39C8D34E2A0A***
6. Connect the Pinnacle 100 module to your PC with the Segger J-Link.
7. Open a terminal or command prompt window and issue the following command: ***nrfjprog -f NRF52 –recover***
8. Once complete, issue the following command: ***nrfjprog -f NRF52 –qspieraseall***

   The Pinnacle 100 is now blank and not running any software. If you wish to reprogram the bootloader, follow the remaining steps.
9. Download the latest version of the bootloader from the Laird Connectivity Pinnacle 100 site.
10. Program the bootloader to the module using the following command: ***nrfjprog -f NRF52 --program Pinnacle_100_Bootloader.hex –reset***
11. Once the programming has finished, open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:
    - Hardware flow control enabled
    - Baud rate set to 115200
    - 1 stop bit
    - No parity

    The CTS status should be green to indicate that the module is ready to accept commands.
12. Send a new-line character (by pressing Enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. In UwTerminalX, this takes the form of a slash (\) followed by two numbers.
13. Type ***x0*** and paste the license code you backed up earlier, then press **Enter**. The command should resemble the following: ***x0B297902D39C8D34E2A0A***

    The module should respond with ***a*** which means the code is accepted.
14. Reboot the module by sending **Z** and **Enter**. There should be no error after this about a license being missing.

    The module is been fully restored to factory default settings, including removing any security bit settings. It can now be used/programmed as desired.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

22 Querying Bootloader Information From a User Application

The bootloader has structure located at a specific location on the nRF52840's flash (with 4-byte packing) which can be used to retrieve information on the bootloader and software.

The format of the structure is as follows:

```
typedef struct
{
    uint32_t       Checksum;
    uint32_t       ExternalFunctionAddress;
    uint16_t       HeaderVersion;
    uint16_t       ExternalFunctionVersion;
    uint16_t       HeaderSize;
    char           BuildDate[12];
    uint8_t        ChecksumType;
    uint8_t        Areas;
    AreaInfoStruct AreaInfo[3];
    uint8_t        PADDING[52];
} ExternalSettingsInfoStruct;
```

The values of which are as follows:

- **Checksum** – A 32-bit checksum of the whole structure, excluding the checksum bytes. The checksum generation formula is given in the Checksum Generation Code section.
- **External function address –** The address of the external function (details on the external function and how to query it are shown in the Querying Bootloader Parameters from a User Application section).
- **Header version –** The version of this header (it should be *1*). If it is newer than *1*, then please obtain the latest documentation for the module from the Laird Connectivity website.
- **External function version –** The total size (in bytes) of the bootloader query function.
- **Header size –** The size of the data in this header excluding the padding bytes.
- **Build date –** A string which contains the build date of the bootloader firmware.
- **Checksum type –**Indicates the type of checksum (it should be *3*).
- **Areas –** Indicates how many areas there are (described below).
- **Area info –** Hold information on sections which are part of the base Laird Connectivity firmware image. You can use this to ensure that your software is running on a genuine unaltered firmware.
- **Padding –** Reserved for future use.

The following displays the area info array structure:

```
typedef struct
{
    uint32_t       StartAddress;
    uint32_t       Size;
    uint32_t       Checksum;
    uint16_t       Version;
    uint8_t        Type : 3;
    uint8_t        Access : 3;
    uint8_t        SignatureType : 2;
    uint8_t        Signature[64];
} AreaInfoStruct;
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

The values of which are as follows:

- **Start address** – The start address of the section, internal to the nRF52840.
- **Size** – The size of the section.
- **Checksum** – A 32-bit checksum of the whole data section. The checksum generation formula is given in the Checksum Generation Code section.
- **Version** – Section version.
- **Type** – Section type (described in Table 10).
- **Access** – The access rights to the section when the user application is running, for verification purposes. Described in Table 11.
- **Signature type** – Signature type which should always be *1* to indicate the Laird Connectivity key.
- **Signature** – A signature of the SHA256 hash of the data section using the Laird Connectivity private key, using ECDSA

The following table displays the type value:

Table 10: Type values

Type	Description
0	Nordic MBR (Master boot record)
1	Bootloader
2	External function

The following table describes the access rights:

Table 11: Access rights

Access Right	Description
0	No access
1	Read only access *(not used)*
2	Write only access *(not used*)
3	Read and write access *(not used)*
4	Execute only access *(not used)*
5	Read and execute access
6	Write and execute access *(not used)*
7	Read, write and execute access (full) *(not used)*

The Bootloader blocks read access after it has booted. Attempting to read the bootloader data from flash by a user application causes a hard-fault on the module.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

23 Querying Bootloader Parameters from a User Application

There is an API available in the Pinnacle 100 bootloader which allows user applications to query information from the bootloader. The format of this function is as follows.

Note: For Zephyr users, this is already included in the Pinnacle 100 Zephyr project source code and can be enabled by selecting the Blr Laird Connectivity HAL module.

```
uint8_t
GetBootloaderSetting(
    uint32_t  nIndex,
    uint8_t   nPartition,
    uint8_t   nSubKey,
    uint8_t  *pBuffer,
    uint32_t  nBufferSize,
    uint32_t *pFullSettingSize,
    uint16_t *pFlags
    );
```

The following are parameter descriptions:

- **Return value** – Returns the status of the function call indicating if it was successful or if an error occurred.
- **nIndex** – Contains the index of the information to return. See Table 12.
- **nPartition** – Contains the partition to query (for partition-based queries).
- **nSubKey** – Contains the sub-key to query (for sub-key based queries). This is not currently used.
- **pBuffer** – A pointer to a buffer which is updated with the information requested. If querying without returning data, then set this to NULL.
- **pBufferSize** – Indicates the size of the buffer and limits the amount of data returned to the buffer. Set to 0 to return no data.
- **pFullSettingSize** – Can optionally contain a pointer to a 32-bit unsigned number which, if successful, contains the full size (in bytes) of the query response excluding the provided buffer size value.
- **pFlags** – Can optionally contain a pointer to a 16-bit unsigned number which, if successful, contains information flags about the requested information.

The address of the function can be extracted from the bootloader information struct as described in the Querying Bootloader Information from a User Application section. Once the address of the function is retrieved, it can be called as required.

Return values are displayed in Table 12.

Table 12: Return values

Return Value	Description
0	Successfully returned data
5	Specified index is not valid
6	No data present for specified index
7	Discrepancy detected (possible hardware issue or unauthorised code). This is returned if readback protection or CPU debug protection should be enabled but isn't
8	Specified partition is not valid
9	Specified sub-key is not valid

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

Index values are displayed in Table 13.

Table 13: Index values

Index	Partition	Field	Description
0x00000001		Bluetooth Address	An optional field which can be set in the bootloader which contains a 7-byte address for a module's Bluetooth address
0x00000002		Public Key Set	Returns 0 if the bootloader does not have a public key set or 1 if a public key is set
0x00000003		Public Key	Returns the customer public key
0x00000004		Readback Protection	Returns 0 if readback protection is disabled or 1 if it is enabled
0x00000005		CPU Debug Protection	Returns 0 if CPU debug protection is disabled or 1 if it is enabled
0x00000006		UART Data Readback Protection	Returns 0 if UART data readback protection is disabled or 1 if it is enabled
0x00000007		QSPI Mode	Returns 0 if the QSPI is configured to use ultra-low performance mode or 1 if it is configured to use high performance mode **Note:** This bit is volatile on the QSPI flash chip.
0x0000000A		QSPI Checked	Returns 0 if QSPI contents have not been checked (and therefore should not be trusted), 1 if only Laird Connectivity signed components have been checked (i.e. because a public key has not been set) or 2 if all components have been checked
0x0000000B		QSPI Header Checksum	Contains a checksum of the QSPI header which can be used to check that it has not been altered since the bootloader last started
0x0000000C		QSPI Header SHA256	Contains a SHA256 hash of the QSPI header which can be used to check that it has not been altered since the bootloader last started
0x0000000D		QSPI User Application Start Offset	Contains the start offset of the area in the QSPI flash which can be used by user applications. If the returned value is 0, then no QSPI flash can be used by the user application
0x0000000E		QSPI User Application Start Sector	Contains the start sector number of the area in the QSPI flash which can be used by user applications. If the returned value is 0, then no QSPI flash can be used by the user application
0x0000000F		QSPI User Application Sectors	Contains the number of sectors of the area in the QSPI flash which can be used by user applications. If the returned value is 0 then no QSPI flash can be used by the user application
0x00000010		QSPI User Application Sector Size	Contains the sector size of pages in the QSPI flash

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

Index	Partition	Field	Description
0x00000020	Application type, 0 – 20, see the Section Application Types section for details	Version Information for Components	Contains version information about updated/installed components on the module
0x00000041	Field number, 0 – 5	User Application 32-byte Registers	Contains optional 32-byte user-data fields which can be set in the bootloader and retrieved by user applications
0x00000042	Field number, 0 – 3	User Application 64-byte Registers	Contains optional 64-byte user-data fields which can be set in the bootloader and retrieved by user applications
0x00000043	Field number, 0 – 1	User Application 128-byte Registers	Contains optional 128-byte user-data fields which can be set in the bootloader and retrieved by user applications

Flag values are displayed in Table 14.

Table 14: Flag values

Flag (Bitmask)	Flag (Integer)	Description
0b1	1	Little endian number
0b100	4	Byte array

The Pinnacle 100 Zephyr out of box demo application with source available on github demonstrates how to query information from the bootloader.

24 Checksum Generation Code

Checksums used in update packages and for verification of sections can be calculated using the following C code.

Note: For Zephyr users, this is already included in the Pinnacle 100 Zephyr project source code and can be enabled by selecting the Msc Laird Connectivity HAL Module):

```
/*****************************************************************************
**              Copyright (C) 2020 Laird Connectivity
**
** Project:     Zephyr
**
** Module:      MscCRC32.c
**
** Mnemonic:    Msc
**
** Notes:
**
** License:     This code is for use only in projects based around the Laird
**              Connectivity Pinnacle 100 module (including PC or embedded
**              device applications). All other use of this code is prohibited.
**
**              All rights reserved.
**
**              It is provided "as is", without warranty/guarantee of any kind,
**              express of implied, including but not limited to the warranties
**              of merchantability, fitness for a particular purpose and
**              non-infringement.
**
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

```
*****************************************************************************/

/*****************************************************************************/
/* Local Functions or Private Members*/
/*****************************************************************************/

/*===========================================================================*/
#define CRC32_POLYNOMIAL 0xEDB88320
/*===========================================================================*/
static uint32_t
MscPubCalc32bitForByte(
    uint8_t nByte
    )
{
    uint8_t nBitCount=8;
    uint32_t nCrc32 = nByte;
    while(nBitCount--)
    {
        if(nCrc32 & 1)
        {
            nCrc32 = (nCrc32 >> 1)^CRC32_POLYNOMIAL;
        }
        else
        {
            nCrc32 >>= 1;
        }
    }
    return nCrc32;
}

/*===========================================================================*/
/* Given an array of bytes, a new 32 bit CRC is calculated.                  */
/*===========================================================================*/
uint32_t
MscPubCalc32bitCrcNonTableMethod(
    uint32_t nCrc32,
    const uint8_t *pSrcStr,
    uint32_t nSrcLen /* in bytes */
    )
{
    uint32_t nTemp1,nTemp2;
    while(nSrcLen--)
    {
        nTemp1 = (nCrc32 >> 8) & 0x00FFFFFFL;
        nTemp2 = MscPubCalc32bitForByte( (uint8_t)((nCrc32 ^ *pSrcStr++) & 0xFF) );
        nCrc32 = nTemp1 ^ nTemp2;
    }
    return nCrc32;
}

/*****************************************************************************/
/* END OF FILE*/
/*****************************************************************************/
```

The MscPubCalc32bitCrcNonTableMethod() function contains the following three parameters:

- nCrc32 – Input to CRC function. This is used for chained checksum code and should be 0 for the first input to the function.
- pSrcStr – The pointer to the data for which a checksum is to be calculated.
- nSrcLen – The length, in bytes, of the input data

The return value is the 32-bit checksum of the input data. If the function must run multiple times for each block of data, then the checksum value should be initialised to 0 and provided to the function in the nCrc32 parameter.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

25 SETTING BOOTLOADER OPTIONS VIA UBUTIL

As mentioned in the Bootloader Settings Sections section, there is an application type of ID 13 which can be used to set security configuration options on the bootloader without requiring sending UART commands to the bootloader. This is especially useful during production of systems. Various bootloader configuration options can be set by using this type which are shown in the following table (Table 15).

Table 15: Bootlegger configuration options

Offset	Name	Configuration Details (UART)	Configuration Details (Partition)
0	Readback protection	Enabling Readback Protection	Readback protection
1	CPU debug protection	Enabling CPU Debug Protection	CPU debug protection
2	Block UART readback	Blocking UART Data Readback	Block UART readback
3	QSPI mode	Configuring QSPI Power/Mode	QSPI mode
4	Full erase mode	Changing Full-Erase Security Level	Full erase mode
5-6	QSPI user sectors	Enabling QSPI Usage For User Applications	QSPI user sections
7	Block UART verification	Blocking UART Data Verification	Block UART verification
8-9	UART verification minimum size	Limiting UART Data Verification	UART verification minimum size
10	Boot verification	Boot Verification	Boot verification
11	Main application/Soft-device section protection	Main Application/Soft-Device Section Protection	Main application/Soft-device section protection
12	Erase section mode	Setting Erase Section Mode	Erase section mode

To create a partition in a .ubu file which sets bootloader settings, the **extradata** parameter is used. This parameter ontains 13 bytes of data, each byte offset corresponds to the feature in Table 15.

Any options which are to be left at their default values should be supplied at 00 to the extradata parameter. Any setting with a non-00 value is configured on the module, assuming that the configuration option set is valid and does not lower security.

For example, if a module has boot verification set to a value of 0x01 and a partition section is present which sets it to 0x02, then the additional setting is ignored and the setting remains at 0x01. However, if a partition section is present which sets it to 0x03, then the setting is updated to 0x03.

When using the extradata command, be sure that the configuration value is placed at the correct offset, the values are entered with the MSB being the first offset. For example: If ***--aX-extradata 010203*** is used, then 01 is offset 0 (readback protection), 02 is offset 1 (CPU debug protection), and 03 is offset 2 (block UART readback). Any missing offset values are set to 0. Values which span more than 1 byte (8-bits) such as the QSPI user sectors or UART verification minimum size are stored in little-endian format with the LSB being the first byte. For example: If ***--aX-extradata 00000000003412*** is used, then the QSPI user sectors value is entered as 3412 and is read as 0x1234 when the bootloader processes it.

25.1 Configuration Options

25.1.1 Readback Protection

Offset 0: --aX-extradata YY where YY is as shown in the following table (Table 16).

Table 16: Readback protection

Value (hex)	Description
00	No change
01	Enable Readback Protection

This configures readback protection for the module.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

25.1.2 CPU Debug Protection

Offset 1: --aX-extradata XXYY where YY is as shown in the following table (Table 17).

Table 17: CPU debug protection

Value (hex)	Description
00	No change
01	Enable CPU Debug Protection

This configures CPU debug protection for the module.

25.1.3 Block UART Readback

Offset 2: --aX-extradata XXXXYY where YY is as shown in the following table (Table 18).

Table 18: Block UART readback values

Value (hex)	Description
00	No change
01	Block UART readback

This configures blocking the UART read data command for the module.

25.1.4 QSPI mode

Offset 3: --aX-extradata XXXXXXYY where YY is as shown in the following table (Table 19).

Table 19: QSPI mode values

Value (hex)	Description
00	No change
01	Ultra-low power mode
02	High performance mode

This configures the default QSPI mode for the module. Please see the Configuring QSPI Power/Mode section for details on what these modes do.

25.1.5 Full Erase Mode

Offset 4: --aX-extradata XXXXXXXXYY where YY is as shown in the following table (Table 20).

Table 20: Full erase mode values

Value (hex)	Description
00	No change
01	Allow if no Bootloader Unlock Key or only allow if Bootloader Unlock Key is set and entered correctly
02	Allow only if Bootloader Unlock Key is set and entered correctly
03	Always Deny

This configures the if the full erase command is allowed from the bootloader.

25.1.6 QSPI User Sections

Offsets 5-6: --aX-extradata XXXXXXXXXXYYYY where YYYY is a little-endian number containing the number of QSPI flash sectors from the end of the device which are available for use by the user application. The QSPI flash has a 4-KB sector size and up to 50% of the QSPI flash is available for the user application. A value of 0 ignores the setting and a value of 1-1024 sets the number of sectors to the desired value. For further details, see the Enabling QSPI Usage For User Applications section.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

Note: This is a two-byte (16-bit) little-endian number and is supplied as an argument with the LSB byte first. For example: 0x5678 would be supplied on the command line as 7856.

This configures the QSPI user sectors for the module.

25.1.7 Block UART Verification

Offset 7: --aX-extradata XXXXXXXXXXXXXXYY where YY is as shown in the following table (Table 21).

Table 21: Block UART verification values

Value (hex)	Description
00	No change
01	Block UART verification

This configures blocking the UART verify data command for the module.

25.1.8 UART Verification Minimum Size

Offsets 8-9: --aX-extradata XXXXXXXXXXXXXXXXYYYY where YYYY is a little-endian number containing the minimum UART verification size in bytes. A value of 0 ignores the setting and a value of 4-16384 (must be divisible by the flash write size - 4) sets the minimum UART verification size to the desired value. For further details, see the Limiting UART Data Verification section.

Note: This is a two-byte (16-bit) little-endian number and is supplied as an argument with the LSB byte first. For example: 0x5678 would be supplied on the command line as 7856.

25.1.9 Boot Verification

Offset 10: --aX-extradata XXXXXXXXXXXXXXXXXXXXYY where YY is as shown in the following table (Table 22).

Table 22: Boot verification values

Value (hex)	Description
00	No change
01	MBR
02	Bootloader
04	External function
08	Main application
10	Soft-device
20	Compromised boot chain
40	Sections required
80	Prevent boot

This list contains the different bitmask options for all the individual features. Multiple features can be combined into a single value which enables all of those options. For example: adding MBR, external function, compromised boot chain and prevent boot = 0x01 + 0x04 + 0x20 + 0x80 = 0xA5.

This command configures the boot verification level for the module.

25.1.10 Main Application/Soft-Device Section Protection

Offset 11: --aX-extradata XXXXXXXXXXXXXXXXXXXXXXYY where YY is as shown in the following table (Table 23).

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

Table 23: Main application/soft-device section protection values

Value (hex)	Description
00	No change
01	Enable for main application
02	Enable for soft-device
03	Enable for main application and soft-device

This command configures the main application and/or soft-device section for the module.

25.1.11 Erase Section Mode

Offset 12: --aX-extradata XXXXXXXXXXXXXXXXXXXXXXXXYY where YY is as shown in the following table (Table 24).

Table 24: Erase section mode values

Value (hex)	Description
00	No change
01	Block lower versions
02	Block equal versions
03	Block lower or equal versions
04	Block user-signed sections
05	Block user-signed sections and lower version Laird Connectivity-signed sections
06	Block user-signed sections and equal version Laird Connectivity-signed sections
07	Block user-signed sections and lower or equal version Laird Connectivity-signed sections
08	Block Laird Connectivity-signed sections
09	Block Laird Connectivity-signed sections and lower version user-signed sections
0a	Block Laird Connectivity-signed sections and equal version user-signed sections
0b	Block Laird Connectivity-signed sections and lower or equal version user-signed sections
0f	Block all erase sections

Italic indicates an option is a mix of other options, non-italic indicates an option is a single option.

This configures the erase section mode of the module.

25.2 Usage

Bootloader configuration options must be in their own self-contained partition in an upgrade image. They cannot be added to another partition's data (e.g. an application update). A version must be specified, the filetype must be specified as 13, the target must be specified as 3 and the keytype must be specified as 1, the options are then placed in the extradata field, like so:

```
--aX-version 1 --aX-filetype 13 --aX-target 3 --aX-keytype 1 --aX-extradata X
```

These sections can be placed before or after upgrade data. The options are updated prior to any section updates being processed.

Once a .ubu or .hex file with bootloader settings is programmed to the module, it should be reset and not interrupted until either the user application boots or UART bootloader boots (if no user application is present). Before using a file in production, the settings on the module should be queried from the bootloader to ensure that they are all valid and were accepted. It is best to do this on a fresh module or on one which is restored to factory defaults (if readback protection and CPU debug protection were never previously used on the module) or by doing a full chip erase and reprogramming the bootloader. Guides for both actions are available in the Restoring to Factory Defaults (via UART) and Full-Chip Erase/Recovery (via SWD) sections respectively.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

25.3 Examples

These examples are provided for reference only to demonstrate the format of the options field.

25.3.1 Enabling Readback Protection CPU Debug Protection with an Application

Enabled readback protection and CPU debug protection, with an application named Blinky.hex where the application is placed after the bootloader settings section, UBUtil command:

```
UBUtil --application-key-file Blinky.pem --a0-version 1 --a0-filetype 13 --a0-extradata 0101
--a0-target 3 --a0-keytype 1 --a1-version 1 --a1-compressed --a1-target 0 --a1-startaddress
0x1000 --a1-filetype 1 --a1-filename Blinky.hex --a1-keytype 1 --append-pub-key --output
Example_1.hex --ubu-platform 512A510F --ubu-flash-size 8M --ubu-sector-size 4K --ubu-base-
address 0 --ubu-align-length 4 --ubu-output Example_1.uwf
```

The output from UBUtil is similar to the following:

```
Laird Connectivity Universal Bootloader Firmware Update Generation Utility
  v1.0
    Built Apr 24 2020

Using input file Blinky.hex (232352 bytes)...

Header:
        Update header version: 1
        Sections present: 3
        Checksum: 0x04152166

Section 0:
        Section Start: 0x0
        Section End: 0x0
        Section Size: 0x0
        Target: 3 (Settings)
        Target Start: 0x0
        Target End: 0x0
        Target Size: 0x0
        Compressed: No
        Version: 1
        Checksum Type: Bypass
        Image Checksum: 0x00000000
        Target Checksum: 0x00000000
        Signature Type: 1 (User key)
        Application Type: 13 (Bootloader settings)
        Header Checksum: 0x6e9766ca
        Filename: [Not Present]
        Extra data (hex):
010100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000
        Signature (hex):
c843202ec48534ff4f30b5497ffc863a5f81888549b9663ad6e646fdcb1a280a04eb0c7c71f1c89ed3b71610b54
d097a8e8aae114276979ee942185386c8fdfb

Section 1:
        Section Start: 0x4000
        Section End: 0x306df
        Section Size: 0x2c6df
        Target: 0 (Internal flash)
        Target Start: 0x1000
        Target End: 0x39ba0
        Target Size: 0x38ba0
        Compressed: Yes
        Version: 1
        Checksum Type: 32-bit
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

```
        Image Checksum: 0x7bc86c9e
        Target Checksum: 0x1e5c79e0
        Signature Type: 1 (User key)
        Application Type: 1 (Main Application)
        Header Checksum: 0x4a5160a3
        Filename: Blinky.hex
        Extra data (hex):
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000
        Signature (hex):
380d7fd40a1f9f009ff123de24fb40b3b6882d2268f715e8f80e76ce5e0a6b3f3f39feb46f75b368c22e9de1d09
413c08830e282ad3950451cf190dc2527c62d

Section 2:
        Section Start: 0x0
        Section End: 0x0
        Section Size: 0x0
        Target: 3 (Settings)
        Target Start: 0x0
        Target End: 0x0
        Target Size: 0x0
        Compressed: No
        Version: 1
        Checksum Type: 32-bit
        Image Checksum: 0xf2a52abf
        Target Checksum: 0x00000000
        Signature Type: 254 (Bypass)
        Application Type: 12 (User public key)
        Header Checksum: 0xb471376d
        Filename: Blinky.pem
        Extra data (hex):
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000
        Signature (hex):
d24deb7ab6c3922d1fd561d0917304c00ece98a4b8bad9b0bd09ed4647edeb0eb5f938693a03ed52ac3f716b0b5
00cfc386bf8a77f4f49ff3b8b4eb4a4ca0470

Total uncompressed section size: 232352
Total compressed section size: 181983
Section compression: 22%

557991 bytes written to Example_1.hex successfully.
197912 bytes written to Example_1.uwf successfully.
```

25.3.2 Setting Boot Verification and Minimum UART Verification Size With an Application

Sets boot verification to full and minimum UART verification size to 32 bytes with an application named Blinky.hex where the application is placed before the bootloader settings section, UBUtil command:

UBUtil --application-key-file Blinky.pem --a0-version 1 --a0-compressed --a0-target 0 --a0-startaddress 0x1000 --a0-filetype 1 --a0-filename Blinky.hex --a0-keytype 1 --a1-version 1 --a1-filetype 13 --a1-extradata 0000000000000002000FF --a1-target 3 --a1-keytype 1 --append-pub-key --output Example_2.hex --ubu-platform 512A510F --ubu-flash-size 8M --ubu-sector-size 4K --ubu-base-address 0 --ubu-align-length 4 --ubu-output Example_2.uwf

The output from UBUtil is similar to the following:

```
Laird Connectivity Universal Bootloader Firmware Update Generation Utility
  v1.0
    Built Apr 24 2020
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

```
Using input file Blinky.hex (232352 bytes)...

Header:
        Update header version: 1
        Sections present: 3
        Checksum: 0x04152166

Section 0:
        Section Start: 0x4000
        Section End: 0x306df
        Section Size: 0x2c6df
        Target: 0 (Internal flash)
        Target Start: 0x1000
        Target End: 0x39ba0
        Target Size: 0x38ba0
        Compressed: Yes
        Version: 1
        Checksum Type: 32-bit
        Image Checksum: 0x7bc86c9e
        Target Checksum: 0x1e5c79e0
        Signature Type: 1 (User key)
        Application Type: 1 (Main Application)
        Header Checksum: 0x3140d90f
        Filename: Blinky.hex
        Extra data (hex):
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000
        Signature (hex):
32ffdc3c7bf454906253ee24b60f71015ce1be4ab23e583b76d105145317de4a6439da1141d057b355582d7ffa7
41bcd12369c5bda0cd5722781784c2278a3c6

Section 1:
        Section Start: 0x0
        Section End: 0x0
        Section Size: 0x0
        Target: 3 (Settings)
        Target Start: 0x0
        Target End: 0x0
        Target Size: 0x0
        Compressed: No
        Version: 1
        Checksum Type: Bypass
        Image Checksum: 0x00000000
        Target Checksum: 0x00000000
        Signature Type: 1 (User key)
        Application Type: 13 (Bootloader settings)
        Header Checksum: 0xc56fc292
        Filename: [Not Present]
        Extra data (hex):
00000000000000002000ff0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000
        Signature (hex):
2d7091c8e55cdaf925ab1a2d71cbbe5644b30f2df2f391539c1e8fed4ecc4b97d33a562d2722c291414d2e53049
f14d4b908912785e543742eec9a8f3088f424

Section 2:
        Section Start: 0x0
        Section End: 0x0
        Section Size: 0x0
        Target: 3 (Settings)
        Target Start: 0x0
        Target End: 0x0
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

```
        Target Size: 0x0
        Compressed: No
        Version: 1
        Checksum Type: 32-bit
        Image Checksum: 0xf2a52abf
        Target Checksum: 0x00000000
        Signature Type: 254 (Bypass)
        Application Type: 12 (User public key)
        Header Checksum: 0xb471376d
        Filename: Blinky.pem
        Extra data (hex):
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000
        Signature (hex):
d24deb7ab6c3922d1fd561d0917304c00ece98a4b8bad9b0bd09ed4647edeb0eb5f938693a03ed52ac3f716b0b5
00cfc386bf8a77f4f49ff3b8b4eb4a4ca0470

Total uncompressed section size: 232352
Total compressed section size: 181983
Section compression: 22%

557991 bytes written to Example_2.hex successfully.
197912 bytes written to Example_2.uwf successfully.
```

25.3.3 Setting QSPI Mode to High Performance and Enabling 8 QSPI Flash Sectors for Customer Use

Sets QSPI mode to high performance and enables 8 QSPI flash sectors to be used by the user application, UBUtil command:

UBUtil --application-key-file Blinky.pem --a0-version 1 --a0-filetype 13 --a0-extradata 00000002000800 --a0-target 3 --a0-keytype 1 --append-pub-key --output Example_3.hex --ubu-platform 512A510F --ubu-flash-size 8M --ubu-sector-size 4K --ubu-base-address 0 --ubu-align-length 4 --ubu-output Example_3.uwf

The output from UBUtil looks similar to the following:

```
Laird Connectivity Universal Bootloader Firmware Update Generation Utility
  v1.0
    Built Apr 24 2020


Header:
        Update header version: 1
        Sections present: 2
        Checksum: 0x56f6cb3c

Section 0:
        Section Start: 0x0
        Section End: 0x0
        Section Size: 0x0
        Target: 3 (Settings)
        Target Start: 0x0
        Target End: 0x0
        Target Size: 0x0
        Compressed: No
        Version: 1
        Checksum Type: Bypass
        Image Checksum: 0x00000000
        Target Checksum: 0x00000000
        Signature Type: 1 (User key)
        Application Type: 13 (Bootloader settings)
        Header Checksum: 0x22749910
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

```
        Filename: [Not Present]
        Extra data (hex):
00000002000800000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000
        Signature (hex):
5dc3ef547e747452c519f9b49254cb0e330afff890b158b7f8d9ea6f778fa7e9b2d4f60bad607d6f2ed352aef1e
9bf2d4b32891fd63c5e3d90065b91f0c6af16

Section 1:
        Section Start: 0x0
        Section End: 0x0
        Section Size: 0x0
        Target: 3 (Settings)
        Target Start: 0x0
        Target End: 0x0
        Target Size: 0x0
        Compressed: No
        Version: 1
        Checksum Type: 32-bit
        Image Checksum: 0xf2a52abf
        Target Checksum: 0x00000000
        Signature Type: 254 (Bypass)
        Application Type: 12 (User public key)
        Header Checksum: 0xb471376d
        Filename: Blinky.pem
        Extra data (hex):
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000
        Signature (hex):
d24deb7ab6c3922d1fd561d0917304c00ece98a4b8bad9b0bd09ed4647edeb0eb5f938693a03ed52ac3f716b0b5
00cfc386bf8a77f4f49ff3b8b4eb4a4ca0470

Total uncompressed section size: 0
Total compressed section size: 0
46110 bytes written to Example_3.hex successfully.
15914 bytes written to Example_3.uwf successfully.
```

26 Pinnacle 100 Module Bootloader Status LED

The Pinnacle 100 features a bootloader status LED output on P1.07 (connected to LED4 on the Pinnacle 100 development board). This line is used by the bootloader to indicate the status of the bootloader which is displayed in the following table (active high) (Table 25).

Table 25: Bootloader status LED

Pattern	Meaning
Fading in/out (PWM)	In UART bootloader, awaiting handshake
3 blinks	In UART bootloader, awaiting data
5 blinks	Initialising
7 blinks	Updating
9 blinks	Modem updating
11 blinks	Full erase in progress
Always on	Fatal bootloader error, module in low power mode

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

27 UBUTIL EXIT CODES

As of UBUtil version 1.00, the exit codes for the utility are as described in the following table (Table 26).

Table 26: UBUtil exit codes

Code	Name	Description
0	OK	Finished successfully
-1	INV_ARG	A provided argument is not valid
-2	STR_TOO_LONG	A provided string is too long
-3	KEY_INVALID	A key file was not provided, or does not exist or is not valid
-4	FILE_NOT_FOUND	A provided file was not found
-5	COMP_KEY_FAIL	An error has occurred whilst computing the public key from the private key
-6	HASH_SIGN_FAIL	The hash signing process has failed
-7	SIG_CHECK_FAIL	A signature check has failed and the signature is not valid or does not match
-8	FILE_OPEN_FAIL	An file could not be opened for reading/writing
-9	COMP_SYS_FAIL	An error occurred with the compression system
-10	PUB_KEY_MISSING	A public key was required which was not supplied
-11	END_MISSING	An end address was expected but was not supplied
-12	END_INVALID	A supplied end address is not valid
-13	KEN_GEN_FAILED	An error occurred whilst trying to generate a keypair
-14	CERT_GEN_FAILED	An error occurred whilst trying to generate a certificate
-15	BASE64_FAILED	An error occurred whilst trying to base64 encode/decode data
-16	HDR_FILE_INVALID	A supplied header file is not valid
-17	ARG_FILE_INVALID	The supplied argument list file is not valid
-18	DUPLICATE_ARG	A duplicate argument was supplied
-19	BOOTLOADER_KEY_MISSING	The bootloader private key was expected but not provided
-20	APPLICATION_KEY_MISSING	The application private key was expected but not provided
-21	KEY_TYPE_INVALID	The key type selected is not valid
-22	BLDR_SETTINGS_MISSING	Bootloader settings were expected but were not provided
-23	BLDR_UNLOCK_MISSING	A bootloader unlock code was expected but was not provided
-24	CONFLICT	Conflicting arguments were passed to the application
-25	HDR_WITH_SUBOPTS	Options cannot be combined with a header file but both were supplied
-26	HDR_NOT_ALLOWED	A header file is not allowed to be used with a particular section
-27	MALLOC_FAIL	A memory allocation has failed – does your system have sufficient free RAM?
-28	FILE_INFO_TYPE_INVALID	The suppled file type is not valid or supported
-29	FILE_INFO_BAD_FILE	The supplied file is not a valid upgrade file
-30	INV_SUFFIX	A provided suffix is not valid
-31	UBU_PLATFORM_ID_INVALID	The supplied UBU platform ID is not valid
-32	UBU_FLASH_SIZE_INVALID	The supplied UBU flash size is not valid
-33	UBU_SECTOR_SIZE_INVALID	The supplied UBU sector size is not valid

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

Code	Name	Description
-34	UBU_BASE_ADDR_INVALID	The supplied UBU base address is not valid
-35	UBU_ALIGN_LEN_INVALID	The supplied UBU align length is not valid
-36	UBU_GEN_FAIL	An error occurred whilst generating the UBU file

28 PROGRAMMING THE MODEM USING DEBUG UART

The modem on the Pinnacle 100 can be upgraded using differential patch files which can be included in hex/ubu firmware update packages, these packages will be distributed on the Pinnacle 100 website and work in the same way bootloader update files work, see the Using Signed Image Header Files section for details on how to include these files in your update packages.

There is also a debug UART connector on the Pinnacle 100 development board connected to the modem which allows for the firmware to be loaded directly to the modem without using differential patch files and can be loaded much faster. Note that modem firmware updates using the debug UART on Windows 10 (or newer).

28.1 UART

To update the modem firmware using UART, a 1.8v USB cable (FTDI or compatible) must be connected to the 'HL7800 FTDI 1' port on the Pinnacle 100 development board which has the following pinout:

Pin	Function
1	Ground
2	Clear to send (CTS)
3	1.8v (no-connect)
4	Transmit (TX)
5	Receive (RX)
6	Ready to send (RTS)

To upgrade the modem firmware using UART:

1. Enter bootloader mode on the pinnacle 100, this can be achieved by holding P0.31 (pin 16 on the M.2 connector) low and rebooting the module or powering it up (on the development board, hold down SW1 and press the reset button).
2. Open a serial utility such as UwTerminalX and select the correct serial port connected to the Pinnacle 100 module with the following settings:
   - Hardware flow control enabled
   - Baud rate set to 115200
   - 1 stop bit
   - No parity

   The CTS status should be green to indicate that the module is ready to accept commands.
3. Send a new-line character (by pressing enter on the terminal) to confirm that it is in bootloader mode. The response should be ***f*** and a hex character. In UwTerminalX this takes the form of a slash (\) followed by two numbers.
4. Right-click on the UwTerminalX window and select the automation option.
5. Check the Un-Escape Strings box.
6. In the top field, add the following data: *p\0f\51\2a\51*
7. In the second field, add the following data: *~\01\06\01\06*
8. In the third field, add the following data: *AT+SWITRACEMODE=RnD\r\n*

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

9. In the forth field, add the following data: *at+cfun=1,1\r\n*
10. Press the top send button, the module should respond with ‘a’
11. Press the second send button, the modem will boot and should emit responses within 15 seconds. If it does not, type ‘ati3’ and press enter and ensure the module response has HL7800 in it (the module will not emit any messages at startup if unsolicited responses are disabled)
12. Press the third send button, the module should respond with ‘OK’
13. Press the forth send button, the module will reboot
14. Open the modem firmware package, when prompted select UART upgrade, put in the serial port of the cable you connected to the ‘HL7800 FTDI 1’ port, not the serial port of the bootloader, and put 921600 as the baudrate

```
Enter Installation COM Port Type-(1.UART, 2.USB(Ymodem), 3.USB(Kermit)):1
Installation port type can be set using Environment Variables: SFT_COM_TYPE
SFT v2.3.2002.0 Feb 21 2020 08:37:41
Available ports:

Enter COM Port ID:COM3
Com port can be set using Environment Variables: SFT_COM_PORT
Enter COM Speed (9600, 57600, 115200, 460800, 921600[default], 1843200, 2400000)
:921600
```

15. The modem firmware upgrade process will begin and will take 10-25 minutes to complete if no errors are encountered
16. Once complete, remove the serial cable from the ‘HL7800 FTDI 1’ port

29 License Information

The bootloader on the Pinnacle 100 and the PC utilities used for generating firmware or flashing firmware to Pinnacle 100 modules utilises code from other software authors whose licenses are as follows:

29.1 b64.c

Uses b64.c from https://github.com/littlstar/b64.c (MIT License) Copyright (c) 2014 Little Star Media, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

29.2 Tinycrypt

Uses sha256.c from tinycrypt (c) 2017, Intel Corporation. All Rights Reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Intel Corporation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

29.3 uECC

Uses uECC from https://github.com/kmackay/micro-ecc/ Copyright (c) 2014, Kenneth MacKay. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
 * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
 * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

29.4 Heatshrink

Uses heatshrink from https://github.com/atomicobject/heatshrink Copyright (c) 2013-2015, Scott Vokes <vokes.s@gmail.com> All rights reserved.

Permission to use, copy, modify, and/or distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

29.5 ARM Object Code

ARM Object Code and Header Files License Version 1.0

Redistribution and use of object code, header files, and documentation, without modification, are permitted provided that the following conditions are met:
1) Redistributions must reproduce the above copyright notice and the following disclaimer in the documentation and/or other materials provided with the distribution.
2) Unless to the extent explicitly permitted by law, no reverse engineering, decompilation, or disassembly of is permitted.
3) Redistribution and use is permitted solely for the purpose of developing or executing applications that are targeted for use on an ARM-based product.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

DISCLAIMER. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS." ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

29.6 Lib Xau

Copyright 1988, 1993, 1994, 1998 The Open Group

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation.

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE OPEN GROUP BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the name of The Open Group shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from The Open Group.

29.7 Xcb

Copyright (C) 2001-2006 Bart Massey, Jamey Sharp, and Josh Triplett.
All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the 'Software'), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the names of the authors or their institutions shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from the authors.

29.8 expat

Copyright (C) 1998, 1999, 2000 Thai Open Source Software Center Ltd and Clark Cooper
Copyright (C) 2001, 2002, 2003, 2004, 2005, 2006 Expat maintainers.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the 'Software'), to deal in the Software without

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

29.9 fontconfig

Copyright (C) 2001,2003 Keith Packard

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Keith Packard not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. Keith Packard makes no representations about the suitability of this software for any purpose. It is provided 'as is' without express or implied warranty.

KEITH PACKARD DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL KEITH PACKARD BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

29.10 z

(C) 1995-2013 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly Mark Adler
jloup@gzip.org madler@alumni.caltech.edu

29.11 bz2

This program, 'bzip2', the associated library 'libbzip2', and all documentation, are copyright (C) 1996-2010 Julian R Seward. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

3. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
4. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE

ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Julian Seward, jseward@bzip.org
bzip2/libbzip2 version 1.0.6 of 6 September 2010

29.12 harfbuzz

HarfBuzz is licensed under the so-called 'Old MIT' license. Details follow.

Copyright (C) 2010,2011,2012 Google, Inc.
Copyright (C) 2012 Mozilla Foundation
Copyright (C) 2011 Codethink Limited
Copyright (C) 2008,2010 Nokia Corporation and/or its subsidiary(-ies)
Copyright (C) 2009 Keith Stribley
Copyright (C) 2009 Martin Hosken and SIL International
Copyright (C) 2007 Chris Wilson
Copyright (C) 2006 Behdad Esfahbod
Copyright (C) 2005 David Turner
Copyright (C) 2004,2007,2008,2009,2010 Red Hat, Inc.
Copyright (C) 1998-2004 David Turner and Werner Lemberg

For full copyright notices consult the individual files in the package.

Permission is hereby granted, without written agreement and without license or royalty fees, to use, copy, modify, and distribute this software and its documentation for any purpose, provided that the above copyright notice and the following two paragraphs appear in all copies of this software.

IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE COPYRIGHT HOLDER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE COPYRIGHT HOLDER SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN 'AS IS' BASIS, AND THE COPYRIGHT HOLDER HAS NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

29.13 freetype

The FreeType 2 font engine is copyrighted work and cannot be used legally without a software license. In order to make this project usable to a vast majority of developers, we distribute it under two mutually exclusive open-source licenses.

This means that *you* must choose *one* of the two licenses described below, then obey all its terms and conditions when using FreeType 2 in any of your projects or products.

- The FreeType License, found in the file `FTL.TXT', which is similar to the original BSD license *with* an advertising clause that forces you to explicitly cite the FreeType project in your product's documentation. All details are in the license file. This license is suited to products which don't use the GNU General Public License.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

Note that this license is compatible to the GNU General Public License version 3, but not version 2.

- The GNU General Public License version 2, found in `GPLv2.TXT' (any later version can be used also), for programs which already use the GPL. Note that the FTL is incompatible with GPLv2 due to its advertisement clause.

The contributed BDF and PCF drivers come with a license similar to that of the X Window System. It is compatible to the above two licenses (see file src/bdf/README and src/pcf/README).

The gzip module uses the zlib license (see src/gzip/zlib.h) which too is compatible to the above two licenses.

The MD5 checksum support (only used for debugging in development builds) is in the public domain.

29.14 udev

Copyright (C) 2003 Greg Kroah-Hartman <greg@kroah.com>
Copyright (C) 2003-2010 Kay Sievers <kay@vrfy.org>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

29.15 dbus

D-Bus is licensed to you under your choice of the Academic Free License version 2.1, or the GNU General Public License version 2 (or, at your option any later version).

29.16 icu

ICU License - ICU 1.8.1 and later Copyright (c) 1995-2015 International Business Machines Corporation and others. All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the 'Software'), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

29.17 Unicode

29.18 Qt

29.19 UPX

29.20 OpenSSL

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

3. All advertising materials mentioning features or use of this software must display the following acknowledgment: 'This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)'4. The names 'OpenSSL Toolkit' and 'OpenSSL Project' must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org.

5. Products derived from this software may not be called 'OpenSSL' nor may 'OpenSSL' appear in their names without prior written permission of the OpenSSL Project.
6. Redistributions of any form whatsoever must retain the following acknowledgment: 'This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/)'

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR").append(" CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
==========
This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com).

Original SSLeay License

Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
All rights reserved.

This package is an SSL implementation written by Eric Young (eay@cryptsoft.com).
The implementation was written so as to conform with Netscapes SSL.

This library is free for commercial and non-commercial use as long as the following conditions are adhered to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, lhash, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@cryptsoft.com).

Copyright remains Eric Young's, and as such any Copyright notices in the code are not to be removed.
If this package is used in a product, Eric Young should be given attribution as the author of the parts of the library used.
This can be in the form of a textual message at program startup or
in documentation (online or textual) provided with the package.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement: 'This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)' The word 'cryptographic' can be left out if the routines from the library being used are not cryptographic related :-).
4. If you include any Windows specific code (or a derivative thereof) from the apps directory (application code) you must include an acknowledgement: 'This product includes software written by Tim Hudson (tjh@cryptsoft.com)'

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The licence and distribution terms for any publicly available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution licence
[including the GNU Public Licence.]

29.21 libftdi

The C library "libftdi" is distributed under the GNU Library General Public License version 2.

29.22 libusb

The C library "libusb" is distributed under the GNU Library Lesser General Public License version 2.1.

29.23 FTDI D2XX

This software is provided by Future Technology Devices International Limited ``as is'' and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall future technology devices international limited be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.
FTDI drivers may be used only in conjunction with products based on FTDI parts.
FTDI drivers may be distributed in any form as long as license information is not modified.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610