nRF5x Command Line Tools v9.4.0

User Guide

v1.3

Doc. ID 4397_662 v1.3 2017-04-24

Contents

Revision history

Date	Version	Description
April 2017	1.3	Updated to match nRF5x Command Line Tools v9.4.0: • Installing the nRF5x Command Line Tools on page 5: Added guidance for installing SW on Linux and OS X • nRF5x Command Line Tools file structure on page 5: Added a new family, UNKNOWN, for automatic detection of device family • nrfjprog return codes on page 16: Added a new possible return value: RecoverFailed • nrfjprog commands on page 10: Added `--fast` modifier for `--verify` operations for devices of the nRF52 family to speed up programming verification times • DLL functions in nrfjprogdll.h on page 21: Added DLL functions `NRFJPROG_disconnect_from_device()` and `NRFJPROG_read_device_family()`
January 2017	1.2	Updated to match nRF5x Command Line Tools v9.3.1: • Updates in the following nrfjprog commands on page 10: `--eraseall, --qspieraseall, --program <hex_file> [--sectorerase | --chiperase | --sectoranduicrerase], --readuicr <path>, --readcode <path>`, and `--readram <path>` • New nrfjprog command: `--readqspi` • New exit codes in nrfjprog return codes on page 16: 29, 61, 70, 71, 72, 73, 104 • Functions added to DLL functions in nrfjprogdll.h on page 21: `NRFJPROG_is_dll_open, NRFJPROG_step, NRFJPROG_read_ram_sections_count, NRFJPROG_read_ram_sections_size, NRFJPROG_read_ram_sections_power_status, NRFJPROG_is_rtt_started, NRFJPROG_rtt_is_control_block_found, NRFJPROG_is_qspi_init`
December 2016	1.1	• Updated to match nRF5x Command Line Tools v9.2.0 • Editorial changes
July 2016	1.0	First release, based on nRF5x Command Line Tools v9.0.0

Chapter 1
Introduction

The nRF5x Command Line Tools are used for development, programming, and debugging of Nordic Semiconductor's nRF5x SoCs (System on Chip).

The nRF5x Command Line Tools consist of the following components:

- nrfjprog executable: The `nrfjprog` executable is a command line tool for programming nRF5x Series SoCs through SEGGER J-Link programmers and debuggers.
- mergehex executable: The `mergehex` executable is a command line utility that enables you to combine up to three HEX files into a single file.
- nrfjprog DLL: The nrfjprog DLL is a Dynamic-Link Library that exports functions for programming and controlling Nordic Semiconductor nRF5x Series SoCs. It lets developers create their own development tools for Nordic Semiconductor nRF5x SoCs using the DLLs API.
- SEGGER J-Link software and documentation pack (included only in the Windows installer).

The nRF5x Command Line Tools are available for the following operating systems:

- Windows
- Linux 64- and 32-bit
- Mac OS X

The nrfjprog utility is developed for use together with SEGGER debuggers, so the SEGGER software must also be installed. You should install the SEGGER version provided with this package (JLink_Windows_V612a), because this is the version that has been tested and verified to work. Using other versions might also work, but keep in mind that there might be major changes that could break compatibility. The SEGGER software is included in the Windows installer, but must be installed manually for Linux and Mac OS X. The SEGGER software is not documented here.

Chapter 2

Installing the nRF5x Command Line Tools

You can install the nRF5x Command Line Tools on Windows, Linux (64-bit and 32-bit), and Mac OS X®.

When installing on Mac OS X or Linux, the SEGGER software must be installed in its default location, or the shared library must be placed so that `dlopen()` can find it. The default location is:

- On Mac OS X: `/Applications/SEGGER/JLink`
- On Linux: `/opt/SEGGER/JLink`

The SEGGER software can be installed by downloading and running the installer from SEGGER Software.

When installing on Windows, the SEGGER software is automatically installed with the nRF5x Command Line Tools.

Complete the following steps to install the nRF5x Command Line Tools:

1. Download the software for your operating system:
   - Windows: nRF5x-Command-Line-Tools for Win32
   - Linux 32-bit: nRF5x-Command-Line-Tools-Linux-i386
   - Linux 64-bit: nRF5x-Command-Line-Tools-Linux-x86_64
   - Mac OS X: nRF5x-Command-Line-Tools-OSX
2. Install the software:
   - On Windows, run the installer and follow the given instructions.
   - On Linux and Mac OS X, extract the `.tar` archive anywhere on your filesystem. Define the path to the extracted directory to access the commands from anywhere in the command line.

After running the installer on Windows or extracting the `.tar` archive on Linux and Mac OS X, the nRF5x Command Line Tools are ready for use.

2.1 nRF5x Command Line Tools file structure

The file structure of the nRF5x Command Line Tools differs slightly depending on the operating system.

2.1.1 Windows file structure

The Windows file structure for the nRF5x Command Line Tools.

File	Description
`docs`	Folder for documentation
`-- mergehex_release_notes.txt`	Release notes for mergehex
`-- nrfjprog_release_notes.txt`	Release notes for nRF5x Command Line Tools
`headers`	Folder for header files
`-- DllCommonDefinitions.h`	Header for common definitions used in the DLL
`-- nrfjprogdll.h`	Common nrfjprog DLL header file. Use family specific for more information

File	Description
`-- nrf51_nrfjprogdll.h`	nRF51 nrfjprog DLL header file
`-- nrf52_nrfjprogdll.h`	nRF52 nrfjprog DLL header file
`-- nrfjprog.h`	nrfjprog executable header file
`-- mergehex.h`	mergehex executable header file
`-- jlinkarm_unknown_nrfjprogdll.h`	Unknown family nrfjprog DLL header file
`nrfjprog.exe`	nrfjprog executable
`nrfjprog.ini`	Initialization file for nrfjprog executable
`nrfjprog.dll`	Top-level DLL
`QspiDefault.ini`	QSPI-connected external memory configuration file
`jlinkarm_nrf51_nrfjprog.dll`	DLL for nRF51
`jlinkarm_nrf52_nrfjprog.dll`	DLL for nRF52
`mergehex.exe`	mergehex executable
`jlinkarm_unknown_nrfjprog.dll`	DLL for automatic family detection
`msvcp100.dll`	Necessary Windows DLL
`msvcr100.dll`	Necessary Windows DLL

2.1.2 Linux file structure

The Linux file structure for the nRF5x Command Line Tools.

File	Description
`mergehex`	mergehex executable delivery
`-- mergehex`	mergehex executable
`-- mergehex_release_notes.txt`	Release notes for mergehex
`-- mergehex.h`	mergehex executable header file
`nrfjprog`	nrfjprog executable delivery
`-- DllCommonDefinitions.h`	Header for common definitions used in the DLL
`-- libjlinkarm_nrf51_nrfjprogdll.so`	Symbolic link to Major Version nRF51 DLL
`-- libjlinkarm_nrf51_nrfjprogdll.so.9`	Symbolic link to Patch Version nRF51 DLL
`-- libjlinkarm_nrf51_nrfjprogdll.so.9.0.0`	DLL for nRF51
`-- libjlinkarm_nrf52_nrfjprogdll.so`	Symbolic link to Major Version nRF52 DLL
`-- libjlinkarm_nrf52_nrfjprogdll.so.9`	Symbolic link to Patch Version nRF52 DLL
`-- libjlinkarm_nrf52_nrfjprogdll.so.9.0.0`	DLL for nRF52
`-- libnrfjprogdll.so`	Symbolic link to Major Version nRF5x DLL
`-- libnrfjprogdll.so.9`	Symbolic link to Patch Version nRF5x DLL
`-- libnrfjprogdll.so.9.0.0`	DLL for nRF5x

File	Description
`-- libjlinkarm_unknown_nrfjprogdll.so`	Symbolic link to Major Version unknown family DLL
`-- libjlinkarm_unknown_nrfjprogdll.so.9`	Symbolic link to Patch Version unknown family DLL
`-- libjlinkarm_unknown_nrfjprogdll.so.9.0.0`	DLL for unknown family
`-- nrf51_nrfjprogdll.h`	nRF51 nrfjprog DLL header file
`-- nrf52_nrfjprogdll.h`	nRF52 nrfjprog DLL header file
`-- unknown_nrfjprogdll.h`	Unknown family nrfjprog DLL header file
`-- nrfjprog`	nrfjprog executable
`-- nrfjprog.h`	nrfjprog executable header file
`-- nrfjprog.ini`	Initialization file for nrfjprog executable
`-- nrfjprogdll.h`	Common nrfjprog DLL header file. Use family specific for more information
`-- QspiDefault.ini`	QSPI-connected external memory configuration file
`-- nrfjprog_release_notes.txt`	Release notes for nrfjprog executable

2.1.3 Mac OS file structure

The Mac OS file structure for the nRF5x Command Line Tools.

File	Description
`mergehex`	mergehex executable delivery
`-- mergehex`	mergehex executable
`-- mergehex_release_notes.txt`	Release notes for mergehex
`-- mergehex.h`	mergehex executable header file
`nrfjprog`	nrfjprog executable delivery
`-- DllCommonDefinitions.h`	Header for common definitions used in the DLL
`-- libjlinkarm_nrf51_nrfjprogdll.9.0.0.dylib`	DLL for nRF51
`-- libjlinkarm_nrf51_nrfjprogdll.9.dylib`	Symbolic link to Patch Version nRF51 DLL
`-- libjlinkarm_nrf51_nrfjprogdll.dylib`	Symbolic link to Major Version nRF51 DLL
`-- libjlinkarm_nrf52_nrfjprogdll.9.0.0.dylib`	DLL for nRF52
`-- libjlinkarm_nrf52_nrfjprogdll.9.dylib`	Symbolic link to Patch Version nRF52 DLL
`-- libjlinkarm_nrf52_nrfjprogdll.dylib`	Symbolic link to Major Version nRF52 DLL
`-- libnrfjprogdll.9.0.0.dylib`	DLL for nRF5x

File	Description
`-- libnrfjprogdll.9.dylib`	Symbolic link to Patch Version nRF5x DLL
`-- libnrfjprogdll.dylib`	Symbolic link to Major Version nRF5x DLL
`-- libjlinkarm_unknown_nrfjprogdll.9.0.0.dylib`	DLL for unknown family
`-- libjlinkarm_unknown_nrfjprogdll.9.dylib`	Symbolic link to Patch Version unknown family DLL
`-- libjlinkarm_unknown_nrfjprogdll.dylib`	Symbolic link to Major Version unknown family DLL
`-- nrf51_nrfjprogdll.h`	nRF51 nrfjprog DLL header file
`-- nrf52_nrfjprogdll.h`	nRF52 nrfjprog DLL header file
`-- nrfjprog`	nrfjprog executable
`-- nrfjprog.h`	nrfjprog executable header file
`-- nrfjprog.ini`	Initialization file for nrfjprog executable
`-- nrfjprogdll.h`	Common nrfjprog DLL header file. Use family specific for more information
`-- QspiDefault.ini`	QSPI-connected external memory configuration file
`-- nrfjprog_release_notes.txt`	Release notes for nrfjprog executable

Chapter 3

Merging files with mergehex

To combine up to three HEX files into a single file, use the `mergehex` executable.

Since the Nordic SoftDevices come as precompiled HEX files, you will have at least two HEX files to program into the nRF5x SoC when adding your own application. `mergehex` allows you to combine the HEX files into a single file before programming it onto the SoC. The maximum supported number of HEX files to merge is currently three. Additional files can be added by invoking the tool multiple times.

The mergehex utility can make developing more efficient when flashing and testing applications. In production programming, it can significantly reduce the complexity of programming the firmware to nRF5x SoCs - especially when there is a bootloader, softdevice, and application.

Table 1: mergehex commands on page 9 shows the commands that are available for `mergehex`.

Table 1: mergehex commands

Shortcut	Command	Description
`-h`	`--help`	Displays the help.
`-v`	`--version`	Displays the `mergehex` version.
`-q`	`--quiet`	Reduces the stdout text info. Must be combined with another command.
`-m`	`--merge <hex.file> <hex.file> [<hex.file>]`	HEX files to be merged. Must be combined with the `--output` command.
`-o`	`--output <hex.file>`	HEX file with the result of the merge. Must be combined with the `--merge` command.

To see all the return codes that the mergehex executable can return, refer to the `mergehex.h` file that is included in the nRF5x Command Line Tools installation.

The following example shows how to use `mergehex` to merge three HEX files, `file1.hex`, `file2.hex`, `file3.hex`, into one, `output_file.hex`:

```
mergehex -m file1.hex file2.hex file3.hex -o output_file.hex
```

Chapter 4
Programming nRF5x SoCs with nrfjprog

To program nRF5x Series SoCs through SEGGER J-Link programmers and debuggers, use the `nrfjprog` executable.

> **Important:** This version of the `nrfjprog` executable has been developed and tested for SEGGER software, JLink_V612a. It will most likely work with other versions of the SEGGER software, but keep in mind that there could be major changes that break the compatibility.

See nrfjprog commands on page 10 for an overview of all available `nrfjprog` commands, and nrfjprog return codes on page 16 for a list of possible return codes.

To set up a standard configuration for using the `nrfjprog` utility, use the initialization file `nrfjprog.ini` (as listed in the nRF5x Command Line Tools file structure on page 5). The currently supported configuration parameters are `Family` and `Clockspeed`. For example, by setting `Family = NRF51`, the family `NRF51` will be chosen by default when calling nrjprog without providing the `--family` option.

The following example shows how to use `nrfjprog` to erase all available user flash (including UICR) and program the file `file.hex` to an nRF52 SoC:

```
nrfjprog -f NRF52 --program file.hex --chiperase
```

4.1 nrfjprog commands

`nrfjprog` offers a variety of commands for programming nRF5x SoCs with different options and executing other operations on the SoCs.

There are shorthand forms for the most commonly used commands. Some commands will only function together with other commands.

Table 2: nrfjprog commands

Shorthand form	Command	Description
	`--licenses`	Displays the licenses of the open source modules used in nrfjprog.
`-q`	`--quiet`	Reduces the stdout info. Must be combined with another command.
`-h`	`--help`	Displays this help.
`-v`	`--version`	Displays the nrfjprog and dll versions.
	`--jdll <file>`	Specifies the file path of the JLinkARM DLL that should be used. If this command is omitted, nrfjprog searches for the latest version of Segger's JLinkARM DLL. Must be combined with another command.

Shorthand form	Command	Description
	`-qspiini <file>`	Specifies the file path of the QSPI settings file that should be used instead of searching for the default settings file. Must be combined with either `--memrd`, `--memwr`, `--program`, `--verify`, `--erasepage`, or `--qspieraseall`. **Limitations:** For nRF51 devices, the operation is not available. For nRF52 devices, the operation is available only for devices with a QSPI peripheral.
`-i`	`--ids`	Displays the serial numbers of all the debuggers connected to the computer.
`-f`	`--family <family>`	Selects the device family for the operation. Valid argument options are NRF51, NRF52, and UNKNOWN. If UNKNOWN family is given, an automatic detection of the device family is performed. Note that providing the actual family is faster than performing the automatic family detection. If the `--family` option is not given, the default is taken from `nrfjprog.ini`. Must be combined with another command.
`-s`	`--snr <serial_number>`	Selects the debugger with the given serial number among all debuggers connected to the computer for the operation. Must be combined with another command.
`-c`	`--clockspeed <speed>`	Sets the debugger SWD clock speed in kHz resolution for the operation. The valid clockspeed arguments go from 125 kHz to 50000 kHz. If the given clockspeed is above the maximum clockspeed supported by the emulator, its maximum will be used instead. If the `--clockspeed` option is not given, the default is taken from `nrfjprog.ini`. Must be combined with another command.
	`--recover`	Erases all user flash memory and disables the readback protection mechanism if enabled.
	`--rbp <level>`	Enables the readback protection mechanism. Valid argument options are CR0 and ALL. **Limitations:** For nRF52 devices, the CR0 argument option is invalid. **Important:** After an `--rbp` operation is performed, the available operations are reduced. For nRF51 devices, and if argument option ALL is used, `--pinreset` will not work on certain older devices. For nRF52 devices, only `--pinreset` or `--recover` operations are available after `--rbp`.

Shorthand form	Command	Description
	`--pinresetenable`	For nRF51 devices, the command is invalid. For nRF52 devices, pin reset will be enabled.
`-p`	`--pinreset`	Performs a pin reset. Core will run after the operation.
`-r`	`--reset`	Performs a soft reset by setting the SysResetReq bit of the AIRCR register of the core. The core will run after the operation. Can be combined with the `--program` operation. If combined with the `--program` operation, the reset will occur after the flashing has occurred to start execution.
`-d`	`--debugreset`	Performs a soft reset by use of the CTRL-AP. The core will run after the operation. Can be combined with the `--program` operation. If combined with the `--program` operation, the debug reset will occur after the flashing has occurred to start execution. **Limitations:** For nRF51 devices, the `--debugreset` operation is not available. For nRF52 Engineering A devices, the `--debugreset` operation is not available.
`-e`	`--eraseall`	Erases all user available program flash memory and the UICR page. Can be combined with the `--qspieraseall` operation. **Limitations:** For nRF51 devices with a pre-programmed SoftDevice, only the user available code flash and UICR will be erased.
	`--qspieraseall`	Erases all flash of the external memory device with help of the QSPI peripheral. Note that depending on the external memory device's erase speed, the operation might take several minutes. Can be combined with the `--eraseall` operation. **Limitations:** For nRF51 devices, the operation is not available. For nRF52 devices, the operation is available only for devices with a QSPI peripheral that are connected to an external memory device. To determine if an external memory device is present, nrfjprog checks the MemSize parameter from the `QspiDefault.ini` file or from the QSPI configuration ini file that is given with the `--qspiini` option.
	`--eraseuicr`	Erases the UICR page. **Limitations:**

Shorthand form	Command	Description
		This operation is only available for nRF51 devices with a pre-programmed SoftDevice.
	`--erasepage <start[-end]>`	Erases the flash pages starting at the given start address and ending at the given end address (not included in the erase). If no end address is given, only one flash page will be erased. If your device is equipped with a QSPI peripheral, the pages to erase belong to the XIP region of the device, and an external memory device is present, this command erases 4 kB pages from the external memory device. The first address of the region is considered as address 0 of the external memory device. To determine if an external memory device is present, nrfjprog checks the MemSize parameter from the `QspiDefault.ini` file or from the QSPI configuration ini file that is given with the `--qspiini` option. **Limitations:** For nRF51 devices, the page will not be erased if it belongs to region 0.
	`--program <hex_file> [--sectorerase | --chiperase | --sectoranduicrerase] [--qspisectorerase | --qspichiperase]`	Programs the specified HEX file into the device. If the target area to program is not erased, the `--program` operation will fail, unless an erase option is given. Valid erase operations for the internal flash memory are `--sectorerase`, `--sectoranduicrerase`, and `--chiperase`. If `--chiperase` is given, all the available user non-volatile memory, including UICR, will be erased before programming. If `--sectorerase` is given, only the targeted non-volatile memory pages, excluding UICR, is erased. If `--sectoranduicrerase` is given, only the targeted non-volatile memory pages, including UICR, will be erased. Note that the `--sectoranduicrerase` and `--sectorerase` operations normally take a significantly longer time compared to `--chiperase`, so use them with caution. If your device is equipped with a QSPI peripheral and an external memory device is present, data targeting the XIP region will be written to the external memory device. The first address of the region is considered as address 0 of the external memory device. To determine if an external memory device is present, nrfjprog checks the MemSize parameter from the `QspiDefault.ini` file or from the QSPI configuration ini file that is given with the `--qspiini` option. If the target area to program is not erased, the `--program` operation will fail, unless an erase option

Shorthand form	Command	Description
		is given. Valid erase operations for the external memory device are `--qspichiperase` and `--qspisectorerase`. If `--qspichiperase` is given, the external memory device will be erased. If the `--qspisectorerase` is given, only 4 kB pages from the targeted external memory device will be erased. Note that the `--qspichiperase` operation may take several minutes. The `--program` command can be combined with the `--verify` option. It can also be combined with either the `--reset` or the `--debugreset` operations. The reset will occur after the flashing operation to start execution. **Limitations:** For nRF51 devices, the `--sectoranduicrerase` operation is not available. For nRF51 devices, if the `hex_file` provided contains sectors belonging to region 0, the `--program` operation will fail. For nRF52 devices, the `--qspisectorerase` and `--qspichiperase` operations are not available unless the device is equipped with a QSPI peripheral, and an external memory is connected. To determine if an external memory device is present, nrfjprog checks the MemSize parameter from the `QspiDefault.ini` file or from the QSPI configuration ini file that is given with the `--qspiini` option.
	`--memwr <addr> --val <val> [--verify]`	Writes to the provided address in memory with help of the NVM Controller or, if your device is equipped with a QSPI peripheral and the address to write belongs to the XIP region, with the help of the QSPI peripheral to an external memory device. To determine if an external memory device is present, nrfjprog checks the MemSize parameter from the `QspiDefault.ini` file or from the QSPI configuration ini file that is given with the `--qspiini` option. The first address of the region is considered as address 0 of the external memory device. If the target address is flash (either internal or in the external memory device) and not erased, the operation will fail. This command can be combined with the `--verify` operation.
	`--ramwr <addr> --val <val> [--verify]`	Writes to memory without help of the NVM Controller to the provided address. Can be combined with the `--verify` operation.
	`--verify [<hex_file>][--fast]`	The provided `hex_file` contents are compared with the contents in the device code flash, RAM, UICR, and XIP regions (for devices that are equipped with a QSPI

Shorthand form	Command	Description
		peripheral and connected to an external memory device) and fail if there is a mismatch. To determine if an external memory device is present, nrfjprog checks the MemSize parameter from the `QspiDefault.ini` file or from the QSPI configuration ini file that is given with the `--qspiini` option. If the optional `--fast` parameter is given, nrfjprog will calculate a hash of the flash target area using a SHA-256 algorithm and compare it to the expected hash instead of reading back the actual contents of the device flash in order to speed the operation. This command can be combined with the `--program`, `--memwr`, and `--ramwr` operations if provided without the `hex_file` parameter. **Limitations:** The `--fast` verifying option is not available for nRF51 devices.
	`--memrd <addr> [--w <width>] [--n <n>]`	Reads n bytes from the provided address. If `width` is not given, 32-bit words are read if `addr` is word aligned, 16-bit words if `addr` is half word aligned, and 8-bit words otherwise. If `n` is not given, one word of size `width` is read. The address and n must be aligned to the width parameter. The maximum number of bytes that can be read is 1 MB. The width must be 8, 16, or 32. If your device is equipped with a QSPI peripheral and the addresses to read belong to the XIP region, the QSPI peripheral is used to read from the external memory device if present. To determine if an external memory device is present, nrfjprog checks the MemSize parameter from the `QspiDefault.ini` file or from the QSPI configuration ini file that is given with the `--qspiini` option. The first address of the region is considered as address 0 of the external memory device. **Limitations:** A single `--memrd` instruction cannot be used to read addresses from both the external memory device and the nRF device.
	`--halt`	Halts the CPU core.
	`--run [--pc <pc_addr> --sp <sp_addr>]`	Starts the CPU. If `--pc` and `--sp` options are given, the `pc_addr` and `sp_addr` are used as initial PC and stack pointer. For `pc_addr` to be valid, its last bit must be one. For `sp_addr` to be valid, it must be word aligned.
	`--readuicr <path>`	Reads the device UICR and stores it in the given file `path`. Can be combined with `--readcode`, `--readram`, and `--readqspi`. If combined, only one instruction can provide a path.
	`--readcode <path>`	Reads the device flash and stores it in the given file path. Can be combined with `--readuicr`, `--readram`, and

Shorthand form	Command	Description
		`--readqspi`. If combined, only one instruction can provide a path.
	`--readram <path>`	Reads the device RAM and stores it in the given file path. Can be combined with `--readuicr`, `--readcode`, and `--readqspi`. If combined, only one instruction can provide a path.
	`--readqspi <path>`	Reads the QSPI-connected external memory and stores it in the given file name. Can be combined with `--readuicr`, `--readcode`, and `--readram`. If combined, only one instruction can provide a path.
	`--readregs`	Reads the CPU registers.

4.2 nrfjprog return codes

`nrfjprog` returns the exit code 0 if the requested operation was completed successfully. Otherwise, an error code is returned.

Table 3: nrfjprog return codes

Exit code	Definition	Description
0	Success	Requested operation (operations) were successfully completed.
1	NrfjprogError	An error condition that should not occur has happened.
2	NrfjprogOutdatedError	Nrfjprog version is too old for the device.
3	MemoryAllocationError	Memory allocation for nrfjprog failed.
11	InvalidArgumentError	Invalid arguments passed to the application.
12	InsufficientArgumentsError	Needed arguments not passed to the application.
13	IncompatibleArgumentsError	Incompatible arguments passed to the application.
14	DuplicatedArgumentsError	The same argument has been provided twice.
15	NoOperationError	The arguments passed do not perform a valid operation.
16	UnavailableOperationBecauseProtectionError	The operation attempted cannot be performed because either the main-ap or the ctrl-ap is not available.
17	UnavailableOperationInFamilyError	The operation attempted cannot be performed in the device because the feature is lacking in the device family.

Exit code	Definition	Description
18	WrongFamilyForDeviceError	The `--family` option given with the command (or the default from nrfjprog.ini) does not match the device connected.
19	UnavailableOperationBecauseMpuConfiguration	For nRF51, `--eraseuicr` is unavailable unless the device came with an ANT SoftDevice programmed at Nordic factory.
20	NrfjprogDllNotFoundError	Unable to find nrfjprog.dll in the installation folder. Reinstall nrfjprog.
21	NrfjprogDllLoadFailedError	Failed to load nrfjprog.dll.
22	NrfjprogDllFunctionLoadFailedError	Failed to load the functions from nrfjprog.dll.
23	NrfjprogDllNotImplementedError	DLL does not implement this function for your device.
25	NrfjprogIniNotFoundError	Unable to find nrfjprog.ini in the installation folder. Reinstall nrfjprog.
26	NrfjprogIniCannotBeOpenedError	Opening the nrfjprog.ini file for read failed.
27	NrfjprogIniFamilyMissingError	Family parameter cannot be parsed from ini file. Line might be deleted or invalid format.
28	NrfjprogIniClockspeedMissingError	Clockspeed parameter cannot be parsed from ini file. Line might be deleted or invalid format.
29	NrfjprogIniQspiIniFileMissingError	DefaultQspiIni parameter cannot be parsed from ini file. Line might be deleted or invalid format.
30	JLinkARMDllNotFoundError	Unable to find install path for JLink software.
31	JLinkARMDllInvalidError	DLL found does not seem a valid DLL.
32	JLinkARMDllFailedToOpenError	DLL could not be opened.
33	JLinkARMDllError	DLL reported error.
34	JLinkARMDllTooOldError	DLL is too old for functionality. Install a newer version of JLinkARM.dll
40	InvalidSerialNumberError	Serial number provided is not among those connected.
41	NoDebuggersError	There are no debuggers connected to the PC.
42	NotPossibleToConnectError	Not possible to connect to the nRF device.
43	LowVoltageError	Low voltage detected at target device.
51	FileNotFoundError	Unable to find the given file.
52	InvalidHexFileError	File specified does not seem a valid HEX file.
53	FicrReadError	FICR read failed.
54	WrongArgumentError	One of the arguments is wrong. Path does not exist, memory access is not aligned.

Exit code	Definition	Description
55	VerifyError	The write verify operation failed.
56	NoWritePermissionError	Unable to create file in the current working directory.
57	NVMCOperationError	The flash operation in the device failed.
58	FlashNotErasedError	A program operation failed because the area to write was not erased.
59	RamIsOffError	The RAM area to read or write is unpowered.
60	NoReadPermissionError	Unable to open file for read.
61	NoExternalMemoryConfiguredError	A QSPI operation is attempted without an external memory configured.
62	RecoverFailed	`--recover` operation failed for an unknown reason. Has the proper family been provided?
70	NrfjprogQspiIniNotFoundError	Unable to find QSPI ini file given as default or given with option `--qspiini`.
71	NrfjprogQspiIniCannotBeOpenedError	Opening the QSPI ini file for read failed.
72	NrfjprogQspiSyntaxError	The QSPI ini file has a syntax error.
73	NrfjprogQspiIniParsingError	The QSPI ini file parsed has one or more missing keys.
100	FicrOperationWarning	FICR operation. It is important to be certain of what you do.
101	UnalignedPageEraseWarning	Address provided with page erase is not aligned to first address of page.
102	NoLogWarning	No log is possible because the program has no write permission in the current directory.
103	UicrWriteOperationWithoutEraseWarning	A UICR write operation is requested but there has been no UICR erase.
104	VeryLongOperationWarning	An operation that might take several minutes is being executed. Please wait.

Chapter 5

nrfjprog DLL

The nrfjprog DLL is a Dynamic-Link Library that exports functions for programming and controlling Nordic Semiconductor nRF5x Series SoCs. It lets developers create their own development tools for Nordic Semiconductor nRF5x SoCs using the DLLs API.

The nrfjprog DLL is a 32-bit Dynamic-Link Library on Windows and Mac OS X, and for Linux it has been compiled as a shared library for both 32- and 64-bit. The DLL exports functions for programming and controlling nRF5x SoCs through SEGGER J-Link programmers and debuggers.

> **Important:** This version of the nrfjprog DLL has been developed and tested for SEGGER software, JLink_V612a. It will most likely work with other versions of the SEGGER software, but keep in mind that there could be major changes that break compatibility.

5.1 Loading the DLL

To use the nrfjprog DLL from a C/C++ application, you must load it first.

The following platform-specific code snippets describe how to load and call one function of the nrfjprog DLL. Remember that certain functions can only be called after certain other functions of the DLL have been called (see Calling DLL functions on page 20).

The nrfjprog DLL is provided for multiple platforms, and some of the required steps for loading the DLL differ for Windows and Linux/Mac OS X. Remember that error checking should be done in each step of the code, but for simplicity this is not illustrated in the following code snippets.

1. Include the necessary header files:
   - On Windows:

     ```
     #include "nrfjprogdll.h"
     #include <windows.h>
     ```

   - On Linux or Mac OS X:

     ```
     #include "nrfjprogdll.h"
     #include <dlfcn.h>
     ```

2. Declare a function pointer type to store the address of the DLL function:

   ```
   typedef nrfjprogdll_err_t (*Dll_NRFJPROG_is_halted_t)(bool *
    is_device_halted);
   ```

3. Load the DLL:
   - On Windows:

     ```
     HMODULE dll = LoadLibrary("nrfjprog.dll");
     ```

- On Linux:

```
    void * dll = dlopen("libnrfjprogdll.so", RTLD_LAZY);
```

- On Mac OS X:

```
    void * dll = dlopen("libnrfjprogdll.dylib", RTLD_LAZY);
```

4. Define a function pointer and load into it the DLL function address:
   - On Windows:

```
Dll_NRFJPROG_is_halted_t NRFJPROG_is_halted =
    (Dll_NRFJPROG_is_halted_t)GetProcAddress(dll,
 "NRFJPROG_is_halted");
```

   - On Linux or Mac OS X:

```
Dll_NRFJPROG_is_halted_t NRFJPROG_is_halted =
    (Dll_NRFJPROG_is_halted_t)dlsym(dll, "NRFJPROG_is_halted");
```

5. Call the function:

```
bool halted;
NRFJPROG_is_halted(&halted);
```

6. Free the DLL:
   - On Windows:

```
FreeLibrary(dll);
```

   - On Linux or Mac OS X:

```
dlclose(dll);
```

5.2 Calling DLL functions

The nrfjprog DLL functions must be called in a specific order.

This is the recommended sequence of calling the nrfjprog DLL functions:

1. `NRFJPROG_open_dll()`
2. Connect with or without specifying the serial number:
   - `NRFJPROG_connect_to_emu_with_snr()`
   - `NRFJPROG_connect_to_emu_without_snr()`
3. `NRFJPROG_connect_to_device()`

4. `NRFJPROG_halt()`
5. Other desired functions such as `NRFJPROG_read()` or `NRFJPROG_write()`
6. `NRFJPROG_close()`

5.3 DLL functions in nrfjprogdll.h

For a complete reference of the nrfjprog DLL and a description of the API, refer to the `nrfjprogdll.h` header file provided as part of the nRF5x Command Line Tools installation.

Table 4: DLL functions in nrfjprogdll.h on page 21 lists all the DLL functions of the nrfjprog DLL. The file `DllCommonDefinitions.h` provided with the installation defines all return codes of the DLL functions as well as other necessary type definitions.

Table 4: DLL functions in nrfjprogdll.h

Function	Description
`NRFJPROG_dll_version`	Returns the JLinkARM.dll version.
`NRFJPROG_is_dll_open`	Checks if the JLinkARM DLL is open.
`NRFJPROG_open_dll`	Opens the JLinkARM DLL and sets the log callback. Prepares the DLL for work with a specific nRF Series.
`NRFJPROG_close_dll`	Closes and frees the JLinkARM DLL.
`NRFJPROG_enum_emu_snr`	Enumerates the serial numbers of connected USB SEGGER J-Link emulators.
`NRFJPROG_read_connected_emu_fwstr`	Reads the firmware identification string of the emulator connected to.
`NRFJPROG_is_connected_to_emu`	Checks if the emulator has an established connection with SEGGER emulator/debugger.
`NRFJPROG_connect_to_emu_with_snr`	Connects to a given emulator/debugger.
`NRFJPROG_connect_to_emu_without_snr`	Connects to an emulator/debugger.
`NRFJPROG_read_connected_emu_snr`	Reads the serial number of the emulator connected to.
`NRFJPROG_disconnect_from_emu`	Disconnects from an emulator.
`NRFJPROG_recover`	Recovers the device.
`NRFJPROG_is_connected_to_device`	Checks if the emulator has an established connection with an nRF SoC.
`NRFJPROG_connect_to_device`	Connects to the nRF SoC.
`NRFJPROG_disconnect_from_device`	Disconnects from the nRF SoC.
`NRFJPROG_readback_protect`	Protects the SoC against read or debug.
`NRFJPROG_readback_status`	Returns the status of the readback protection.
`NRFJPROG_read_region_0_size_and_source`	Returns the region 0 size and source of protection if any.
`NRFJPROG_debug_reset`	Executes a reset using the CTRL-AP.

Function	Description
`NRFJPROG_sys_reset`	Executes a system reset request.
`NRFJPROG_pin_reset`	Executes a pin reset.
`NRFJPROG_disable_bprot`	Disables BPROT.
`NRFJPROG_erase_all`	Erases all flash.
`NRFJPROG_erase_page`	Erases a page of code flash.
`NRFJPROG_erase_uicr`	Erases UICR.
`NRFJPROG_write_u32`	Writes one uint32_t data at the given address.
`NRFJPROG_read_u32`	Reads one uint32_t address.
`NRFJPROG_write`	Writes data from the array starting at the given address.
`NRFJPROG_read`	Reads data_len bytes starting at address addr.
`NRFJPROG_is_halted`	Checks if the nRF SoC CPU is halted.
`NRFJPROG_halt`	Halts the nRF SoC CPU.
`NRFJPROG_run`	Starts the nRF SoC CPU with the given pc and sp.
`NRFJPROG_go`	Starts the nRF SoC CPU.
`NRFJPROG_step`	Runs the device CPU for one instruction.
`NRFJPROG_read_ram_sections_count`	Reads the number of RAM sections in the device.
`NRFJPROG_read_ram_sections_size`	Reads the size of the RAM sections in the device in bytes.
`NRFJPROG_read_ram_sections_power_status`	Reads the RAM section power status.
`NRFJPROG_is_ram_powered`	Reads the RAM power status.
`NRFJPROG_power_ram_all`	Powers up all RAM sections of the device.
`NRFJPROG_unpower_ram_section`	Powers down a RAM section of the device.
`NRFJPROG_read_cpu_register`	Reads a CPU register.
`NRFJPROG_write_cpu_register`	Writes a CPU register.
`NRFJPROG_read_device_version`	Reads the device version connected to the device.
`NRFJPROG_read_device_family`	Reads the family of the device connected to the emulator. Can only be called if `NRFJPROG_open_dll()` was called with `UNKNOWN_FAMILY` as family parameter.
`NRFJPROG_read_debug_port_register`	Reads a debugger debug port register.
`NRFJPROG_write_debug_port_register`	Writes a debugger debug port register.
`NRFJPROG_read_access_port_register`	Reads a debugger access port register.
`NRFJPROG_write_access_port_register`	Writes a debugger access port register.
`NRFJPROG_is_rtt_started`	Checks if the RTT is started.

Function	Description
`NRFJPROG_rtt_set_control_block_address`	Indicates to the DLL the location of the RTT control block in the nRF SoC memory.
`NRFJPROG_rtt_start`	Starts RTT.
`NRFJPROG_rtt_is_control_block_found`	Checks if an RTT control block has been found.
`NRFJPROG_rtt_stop`	Stops RTT.
`NRFJPROG_rtt_read`	Reads from an RTT channel.
`NRFJPROG_rtt_write`	Writes to an RTT channel.
`NRFJPROG_rtt_read_channel_count`	Gets the number of RTT channels.
`NRFJPROG_rtt_read_channel_info`	Reads the info from one RTT channel.
`NRFJPROG_is_qspi_init`	Checks if the QSPI peripheral is initialized.
`NRFJPROG_qspi_init`	Initializes the QSPI peripheral.
`NRFJPROG_qspi_uninit`	Uninitializes the QSPI peripheral.
`NRFJPROG_qspi_read`	Reads from the external QSPI-connected memory.
`NRFJPROG_qspi_write`	Writes to the external QSPI-connected memory.
`NRFJPROG_qspi_erase`	Erases the external QSPI-connected memory.
`NRFJPROG_qspi_custom`	Sends a custom instruction to the external QSPI-connected memory.

Liability disclaimer