nRF5x Command Line Tools

v1.0

2016-07-05

Contents

Revision history

Date	Version	Description
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 Nordic Semiconductor's nRF5x SoCs (System on Chip) and consists of the following components:

- nrfjprog executable

  The nrfjprog is a simple command line utility.
- mergehex executable

  The mergehex is a simple command line utility.
- nrfjprog DLL

  The nrfjprog DLL (Dynamic-Link Library) lets developers create their own development tools for Nordic Semiconductor nRF5x SoCs using the DLLs API.
- SEGGER J-Link software and documentation pack (only included in the Windows installer)

The nRF5x Command Line Tools package is supported for Windows, Linux, and Mac OS X. Nordic provides separate installers or packages for all of these operating systems.

The nrfjprog utility is developed for use together with SEGGER debuggers, so the SEGGER software must also be installed. We always suggest installing the SEGGER version provided with this package (JLink_V512g) as that is the one which has been tested and verified to work. Using other version will possibly also work, but keep in mind there might be some Major changes that could break compatibility. The SEGGER software is included in the Windows installer, but has to be installed manually for Linux and Mac OS X. The SEGGER software is not documented here.

Chapter 2
Installation

This section describes the installation procedure for nRF5x Command Line Tools.

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

- Windows
- Linux 64- and 32-bit
- Mac OS X on page 8

2.1 Windows

Windows installer:

- nRF5x-Command-Line-Tools for Win32

After the installer has been downloaded, run it and follow the given instructions.

> **Important:** As the nrfjprog utility is based on SEGGER debuggers and their software, the SEGGER software installer is bundled with our Windows installer. This means that the SEGGER software is installed at the same time as our software.

Table 1: nRF5x Command Line Tools structure, Windows on page 6 lists the installed files of nRF5x Command Line Tools. The default installation path of nRF5x Command Line Tools on Windows is: `C:\Program Files (x86)\Nordic Semiconductor\nrf5x\bin`.

Table 1: nRF5x Command Line Tools structure, Windows

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.
`-- 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.
`nrfjprog.exe`	nrfjprog executable.
`nrfjprog.ini`	Initialization file for nrfjprog executable.
`nrfjprog.dll`	Top-level DLL.
`jlinkarm_nrf51_nrfjprog.dll`	DLL for nRF51.

File	Description
jlinkarm_nrf52_nrfjprog.dll	DLL for nRF52.
mergehex.exe	mergehex executable.
msvcp100.dll	Necessary Windows DLL.
msvcr100.dll	Necessary Windows DLL.

2.2 Linux

Nordic currently provides the following .tar packages containing the nRF5x Command Line Tools.

- Linux installer for 32-bit nRF5x-Command-Line-Tools-Linux-i386
- Linux installer for 64-bit nRF5x-Command-Line-Tools-Linux-x86_64

To use the tools on Linux, the SEGGER software also needs to be installed to its default location (/opt/SEGGER/JLink), or their shared library must be placed so that `dlopen()` can find it. The SEGGER software can be installed on Ubuntu by downloading and running their .deb installer from SEGGER Software.

Once the nRF5x Command Line Tools .tar archive has been downloaded, extract it anywhere on your filesystem and it will be ready for use.

Table 2: nRF5x Command Line Tools structure, Linux

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_nrf51_nrfjprogdll.so.9	Symbolic link to Patch Version nRF52 DLL.
-- libjlinkarm_nrf51_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.
-- nrf51_nrfjprogdll.h	nRF51 nrfjprog DLL header file.
-- nrf52_nrfjprogdll.h	nRF52 nrfjprog DLL header file.
-- nrfjprog	nrfjprog executable.

File	Description
-- 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.
-- nrfjprog_release_notes.txt	Release notes for nrfjprog executable.

2.3 Mac OS X

For Mac OS X, Nordic currently provides the following .tar package containing the nRF5x Command Line Tools.

- Mac OS X installer nRF5x-Command-Line-Tools-OSX

To use the tools on Mac OS X, the SEGGER software also needs to be installed to its default location (/Applications/SEGGER/JLink), or their shared library must be placed so that `dlopen()` can find it. The SEGGER software can be installed on Mac OS X by downloading and running their .pkg installer from SEGGER Software.

Once the nRF5x Command Line Tools .tar archive has been downloaded, extract it anywhere on your filesystem and it will be ready for use.

Table 3: nRF5x Command Line Tools structure, OS X

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.
-- libnrfjprogdll.9.dylib	Symbolic link to Patch Version nRF5x DLL.
-- libnrfjprogdll.dylib	Symbolic link to Major Version nRF5x DLL.
-- nrf51_nrfjprogdll.h	nRF51 nrfjprog DLL header file.
-- nrf52_nrfjprogdll.h	nRF52 nrfjprog DLL header file.

File	Description
-- 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.
-- nrfjprog_release_notes.txt	Release notes for nrfjprog executable.

Chapter 3
mergehex executable

The mergehex executable is a command line utility enabling you to combine up to three hex files into a single file.

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.

This is a simple example of a typical mergehex use case in a Windows command prompt:

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

The mergehex utility merges three hex files, `file1.hex`, `file2.hex`, `file3.hex`, into one, `output_file.hex`.

3.1 Available commands for mergehex executable

This section describes the available commands for the mergehex executable.

Table 4: mergehex commands on page 10 shows the available commands for the mergehex executable and their descriptions. There is a shortcut for all mergehex commands.

Table 4: 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 which the mergehex executable can return, please refer to the `mergehex.h` that is included in the nRF5x-Command-Line-Tools installation.

Chapter 4
nrfjprog executable

The nrfjprog executable is a command line tool for programming nRF5x Series SoCs through SEGGER J-Link programmers and debuggers.

This is a simple example of a typical nrfjprog use case in a Windows command prompt:

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

Family type nRF52 is chosen and `file.hex` is programmed to the SoC. All available user flash (including UICR) will be erased before the programming.

> **Important:** This version of nrfjprog executable has been developed and tested for SEGGER software, JLink_V512g. It will most likely work with other versions of the SEGGER software, but keep in mind that there could be major changes that breaks the compatibility.

4.1 nrfjprog structure and commands

This section describes the nrfjprog executable's structure and commands.

nrfjprog.ini

The initialization file for nrfjprog executable, `nrfjprog.ini`, as listed in the nRF5x Command Line Tools structure, can be used for setting up a standard configuration when using the nrfjprog utility. The current supported configuration parameters are `Family` and `Clockspeed`. For example, by setting the `Family = NRF51`, when calling nrjprog without providing the `--family` option, the family `NRF51` will be chosen by default.

Table 5: nrfjprog commands on page 11 shows the available commands for the nrfjprog executable and their explanations. For some commonly used commands there is also a shortcut. Some commands will only function together with other certain commands.

Table 5: nrfjprog commands

Shortcut	Command	Description
`-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.
`-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 and NRF52. If `--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.

Shortcut	Command	Description
-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 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.
	--pinresetenable	For nRF51 devices, 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_FP1_EngA devices, the --debugreset operation is not available.
-e	--eraseall	Erases all user available program flash memory and the UICR page. **Limitations:**

Shortcut	Command	Description
		For nRF51 devices with a pre-programmed SoftDevice, only the user available code flash and UICR will be erased.
	`--eraseuicr`	Erases the UICR page. **Limitations:** This operation is only available to nRF51 devices with a pre-programmed SoftDevice.
	`--erasepage <start[-end]>`	Erases the flash pages starting at start address and ending at end address (not included in the erase). If end address is not given, only one flash page will be erased. **Limitations:** For nRF51 devices, the page will not be erased if it belongs to region 0.
	`--program <hex_file> [--sectorerase | --chiperase | --sectoranduicrerase]`	Programs the specified hex file into the nRF SoC. If the target area to program is not erased, the `--program` operation will fail, unless an erase option is given. Valid erase operations are `--sectorerase`, `--sectoranduicrerase` and `--chiperase`. If `--chiperase` is given, all the available user flash (including UICR) will be erased before programming. If `--sectorerase` is given, the target sectors (excluding UICR) will be erased. If `--sectoranduicrerase` is given, the target sectors (including UICR) will be erased. Note that the `--sectoranduicrerase` and `--sectorerase` operations normally take significantly longer time compared to `--chiperase`, so use them with caution. Can be combined with the `--verify` operation. Can 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, a `--sectorerase` operation will fail.
	`--memwr <addr> --val <val> [--verify]`	Writes to memory with the help of the NVM Controller to the provided address. If the target address is flash and is not erased, the operation will fail. Can be combined with the `--verify` operation.

Shortcut	Command	Description
	`--ramwr <addr> --val <val> [--verify]`	Writes to memory without the help of the NVM Controller to the provided address. Can be combined with the `--verify` operation.
	`--verify [<hex_file>]`	The provided `hex_file` contents are compared with the contents in the device code flash, RAM and UICR, and fail if there is a mismatch. It can be combined with the `--program`, `--memwr` and `--ramwr` operations if provided without the `hex_file` parameter.
	`--memrd <addr> [--w <width>] [--n <n>]`	Reads n bytes from the provided address. If the width is not given, 32-bit words will be read if addr is word aligned, 16-bit words if the addr is half word aligned, and 8-bit words otherwise. If n is not given, one word of size width will be 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 w must be 8, 16, or 32.
	`--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` and `--readram`. 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` and `--readram`. 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` and `--readcode`. If combined, only one instruction can provide a path.
	`--readregs`	Reads the CPU registers.

4.2 nrfjprog return codes

This section describes the nrfjprog executable's return codes.

Table 6: nrfjprog return codes on page 15 shows the return codes for the nrfjprog executable and their explanations.

Table 6: 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 can not be performed because either the main-ap or the ctrl-ap is not available.
17	UnavailableOperationInFamilyError	The operation attempted can not be performed in the device because the feature is lacking in the device family.
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 still 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.

Exit code	Definition	Description
28	NrfjprogIniClockspeedMissingError	Family 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.
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.
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 reuqested but there has not been a UICR erase.

Chapter 5
nrfjprog DLL

The nrfjprog DLL is a Dynamic-Link Library which exports functions for programming and controlling Nordic Semiconductor nRF5x series SoCs.

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_V512g. It will most likely work with other versions of the SEGGER software, but keep in mind that there could be major changes that breaks compatibility.

5.1 How to load DLL function

This section describes how to use the nrfjprog DLL from a C/C++ application.

As the nrfjprog DLL is provided for multiple platforms, two approaches for loading the DLL in Windows and Linux/Mac OS X will be described in this section. 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.

Loading the DLL and its functions requires platform-specific calls. The following code snippets will 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.

Windows:

1. Include the necessary header files:

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

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

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

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

5. Call the function:

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

6. Free the DLL:

```
FreeLibrary(dll);
```

Linux and Mac OS X:

1. Include the necessary header files:

```
#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:

   a. Linux:

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

   b. Mac OS X:

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

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

```
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:

```
dlclose(dll);
```

5.2 Recommended DLL function calling sequence

This section describes the recommended calling sequence of the nrfjprog DLL functions.

Calling the different nrfjprog DLL functions has be to done in a specific order. The following list describes the recommended sequence of calling the nrfjprog DLL functions.

1. `NRFJPROG_open_dll()`
2. Connect with or without specifying the serial number:
   a. `NRFJPROG_connect_to_emu_with_snr()`
   b. `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 reference of the nrfjprog DLL, please refer to the `nrfjprogdll.h` header file provided as part of the nRF5x Command Line Tools installation.

Table 7: DLL functions in nrfjprogdll.h on page 19 lists all the DLL functions of the nrfjprog DLL. Please refer to the `nrfjprogdll.h` for detailed description of the API itself. The file `DllCommonDefinitions.h` provided with the installation defines all the return codes of the DLL functions as well as other necessary type definitions.

Table 7: DLL functions in nrfjprogdll.h

Function	Description
`NRFJPROG_dll_version`	Returns the JLinkARM.dll version.
`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_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.

Function	Description
NRFJPROG_connect_to_device	Connects to the nRF SoC and halts it.
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.
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_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_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.

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_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.