nRF pynrfjprog

v10.x.x

User Guide

v1.4

NORDIC® SEMICONDUCTOR

Contents

Revision history

Date	Version	Description
November 2020	1.4	• Updated to match nRF pynrfjprog v10.x.x • Added tool version to the cover again • Added information about the high-level module • Updated Reference on page 8 • Editorial changes
April 2017	1.3	• Tool version removed from PDF cover. This document covers all versions of pynrfjprog starting from v9.0.0 • Updated Reference on page 8 • Editorial changes
January 2017	1.2	Updated to match nRF5x pynrfjprog v9.3.1
December 2016	1.1	• Updated to match nRF5x pynrfjprog v9.2.0 • Editorial changes
July 2016	1.0	First release, based on nRF5x pynrfjprog v9.0.0

1 Introduction

The nRF pynrfjprog utility is a simple Python interface for the nrfjprog DLL. The utility works with Python v2.7.x and v3.4.x, or later.

The pynrfjprog package provides Python bindings for the nrfjprog DLL and is an excellent tool if you use Python to test your application. It gives you all the functionality needed for properly testing the functionality of the Nordic Semiconductor SoCs as well as your own applications.

Module	Description
`HighLevel`	Provides Python bindings for the exported functions of the highlevelnrfjprog DLL.
`LowLevel`	Provides Python bindings for the exported functions of the nrfjprog DLL.
`API`	Deprecated module, redirects to LowLevel.
`MultiAPI`	Provides multiprocessing Python bindings for the exported functions of the nrfjprog DLL.
`Hex`	Implements an Intel HEX file parser that allows HEX files to be parsed and easily programmed using the API module.
`JLink`	Implements functions for finding the latest SEGGER DLL.
`examples`	Provides some examples on how the package can be used for programming and debugging SoCs.
`docs`	Provides the header files of the family and main `nrfjprogdll.h`, as well as the `DllCommonDefinitions.h`.
`lib_x86`	Provides the nrfjprog 32-bit DLLs.
`lib_x64`	Provides the nrfjprog 64-bit DLLs.

Table 1: pynrfjprog modules

2 Installing pynrfjprog

The pynrfjprog software package can be installed using the Python installer program `pip`, or from the zip file provided by Nordic Semiconductor.

As the pynrfjprog package provides access to the nrfjprog DLL, it relies on the SEGGER J-Link ARM DLL being installed on the system. The required nrfjprog DLL is part of the installation and is included in the `pynrfjprog` directory. The SEGGER DLL is not included, but nrfjprog must be able to find it.

In the following cases, the nrfjprog DLL can determine the location of the SEGGER DLL:

- If the SEGGER DLL is installed in the default location:
  - Windows: `C:\Program Files (x86)\SEGGER\JLink` or `C:\Program Files (x86)\SEGGER\JLink_`*`Version`*
  - Linux: `/opt/SEGGER/JLink`
  - macOS: `/Applications/SEGGER/JLink`
- Windows: If the DLL was installed through the SEGGER installer and the location is stored in the Windows registry.
- Linux and macOS: If **`dlopen`** can find the DLL.

If the location of the SEGGER DLL cannot be determined automatically, it is possible to pass the location of the DLL when instantiating the nrfjprog DLL object, i.e. when initializing the `API(..., jlink_arm_dll_path="/PATH", ...)`.

2.1 Installing pynrfjprog using pip (recommended)

The recommended way for installing the pynrfjprog utility is using `Python-pip`.

As a prerequisite, pip needs to be installed. For information on pip installation, see Python Packaging User Guide - Tool Recommendations. Pip will by default be included in the latest versions of Python (starting from v2.7.9) and can also be installed for older versions. For more information, see Python Documentation - Installing Python Modules.

The pynrfjprog package is published on the PyPI - the Python Package Index, and is therefore available through pip.

To install the pynrfjprog package, type `pip install pynrfjprog` at a Command Prompt window. The content of the package will be added to the Python defaults directory. Depending on the OS and version installed, the directory could be `C:\Python27\Lib\site-packages\pynrfjprog\`

After the installation, you can import the pynrfjprog modules from a Python interpreter or script by calling: `from pynrfjprog import HighLevel, Lowlevel, MultiAPI, Hex, examples`. This will import the modules HighLevel, LowLevel, MultiAPI, HEX, and examples.

2.2 Installing pynrfjprog via the zip file

The pynrfjprog utility can alternatively be installed from a zip file.

This can be useful if the intended workstation is without PyPI access. As a prerequisite, installation via the zip file requires the setup tools package to be present. For more information, see Python Setuptools.

1. Download the pynrfjprog zip file.
2. Extract the compressed zip file and open a Command Prompt window within that directory.

3. Type `python setup.py install` at the Command Prompt.
   The content of the package will be added to the Python defaults directory. Depending on the OS and version installed, the directory could be `C:\Python27\Lib\site-packages\pynrfjprog\`

After the installation, you can import the pynrfjprog modules from a Python interpreter or script by calling: `from pynrfjprog import HighLevel, Lowlevel, MultiAPI, Hex, examples`. This will import the modules HighLevel, LowLevel, MultiAPI, HEX, and examples.

3 Example

This section shows some simple examples of a typical pynrfjprog use case.

To initialize the API module to work with nRF52 Series SoCs, establish a connection to a SEGGER debug emulator, erase all user flash of the nRF52 SoC, and finally disconnect the emulator and close the connection, type the following commands in a Python interpreter:

```
from pynrfjprog import HighLevel
api = HighLevel.API('NRF52')
api.open()
api.connect_to_emu_without_snr()
api.erase_all()
api.close()
```

The API object includes `enter` and `exit` methods so the `with` construct can be used. When using `with`, then `open()` and `close()` are called automatically.

Pynrfjprog also allows you to work with multiple devices at once. To do this, type the following:

```
from pynrfjprog import MultiAPI
api = MultiAPI.MultiAPI('NRF52')
api.open()

api2 = MultiAPI.MultiAPI('NRF51')
api2.open()

api1.close()
api2.close()
```

The two MultiAPI modules are initialized to work with different nRF families. Their APIs are initialized through `.open()`, and closed without calling any of the API functions.

Other examples on how to use pynrfjprog can be found in the examples module provided with the package. The examples can be found in the *`installed pynrfjprog directory`*`/examples` folder. These examples can be run either from within the folder, for example calling `python memory_read_write.py`, or they can be run from an interpreter by importing the examples module `>>> import examples`, and calling `>>> examples.memory_read_write.run()`.

4 Reference

This section summarizes some of the central pynrfjprog files. For more information, see comments and header files provided with the module itself.

The following table lists some of the central files contained in the pynrfjprog package and describes their purpose.

File name	Description
`Hex.py`	The Python file `Hex.py` is included in pynrfjprog to allow you to parse an Intel HEX file from a Python script and then program it using the `API.py` write function.
`JLink.py`	The Python file `JLink.py` defines the functions for finding the latest JLinkARM DLL.
`HighLevel.py`	The Python file `HighLevel.py` contains the high-level API for pynrfjprog. This API will give you access to the functions from the high-level DLL.
`LowLevel.py`	The Python file `LowLevel.py` contains the low-level API for pynrfjprog. This API will give you access to the functions from the nrfjprog DLL.
`MultiAPI.py`	The Python file `MultiAPI.py` contains the API for multi-access pynrfjprog. This MultiAPI will give you access to the functions from the nrfjprog DLL for handling multiple devices at the same time.

Table 2: Central pynrfjprog files

For a description of all DLL functions that are exported through `HighLevel.py` and `LowLevel.py`, see the comments in the files. Also see the `highlevelnrfjprogdll.h` and `nrfjprogdll.h` header files that are provided in the `docs` folder.

There are three modules that can be used to interact with an nRF target: HighLevel, LowLevel, and MultiAPI. LowLevel.API gives the program access to one nRF target, while the MultiAPI.API class can be used for accessing multiple nRF targets at the same time. When instantiating a MultiAPI.API object, it creates a subprocess for each LowLevel.API instance and runs it. This subprocess can then be communicated to via the MultiAPI.API instance, allowing you to work with multiple instances at the same time.

Note:

- Limitation: When using MultiAPI on Windows, be sure to run your script only through `if __name__ == '__main__':`. In Linux and macOS, an API object cannot be instantiated before instantiating a MultiAPI object.
- This version of pynrfjprog has been developed and tested for SEGGER software, JLink_V6.86f. 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.

Legal notices

By using this documentation you agree to our terms and conditions of use. Nordic Semiconductor may change these terms and conditions at any time without notice.

Liability disclaimer

Nordic Semiconductor ASA reserves the right to make changes without further notice to the product to improve reliability, function, or design. Nordic Semiconductor ASA does not assume any liability arising out of the application or use of any product or circuits described herein.

Nordic Semiconductor ASA does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. If there are any discrepancies, ambiguities or conflicts in Nordic Semiconductor's documentation, the Product Specification prevails.

Nordic Semiconductor ASA reserves the right to make corrections, enhancements, and other changes to this document without notice.

Life support applications

Nordic Semiconductor products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury.

Nordic Semiconductor ASA customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Nordic Semiconductor ASA for any damages resulting from such improper use or sale.

RoHS and REACH statement

Complete hazardous substance reports, material composition reports and latest version of Nordic's REACH statement can be found on our website www.nordicsemi.com.

Trademarks

All trademarks, service marks, trade names, product names, and logos appearing in this documentation are the property of their respective owners.

Copyright notice

COMPANY WITH
QUALITY SYSTEM
CERTIFIED BY DNV GL
= ISO 9001 =