NHS31xx one-time NFC program downloader

> NXP public
> Last modification date: 2021-09-01

About

The *NHS31xx one-time NFC program downloader* is the default firmware image flashed by NXP in all NHS31xx ICs during production. How to interface the on-chip one-time NFC program downloader of the NHS31xx is described in AN13041 *NFC program downloader interface*. This python3 script is one example how the host side can be implemented to communicate with that firmware.

Using this script, you can download over the NFC interface your final firmware image. This can only be done once: upon successful completion, the default firmware image will be at least partially overwritten with your firmware image, and your firmware image will be executed after a reset.

Setup

Python

This script is successfully tested on a Win10 PC running Python 3.7.4 32 bit. Later versions should work equally well.

Make sure `python.exe` and `pip.exe` are in your `PATH` environment variable.

Python module

This Python script relies on a 3rd-party python module for near field communication. This can either be the nfcpy_ or the pyscard module. Which one you can use depends on your choice of tag reader.

This script is successfully tested using these combinations [1]:

Running Python 3.7.4 32 bit on Win10 64 bit	ACS ACR122	Identiv uTrust 3700F	Sony RC-S380
nfcpy i.c.w. WinUSB driver [2]	yes		yes
pyscard i.c.w. product-specific driver	yes	yes	yes

[1] This is not an exhaustive list.

[2] A list of supported devices for use with *nfcpy* is available on nfcpy.readthedocs.io

nfcpy

The *nfcpy* module in turn uses the *libusb* driver to communicate with the tag. It is a low-level driver which requires one to know the exact commands each tag supports, but also allows to enable and disable the field programmatically.

When using *nfcpy*, follow the installation instructions on nfcpy.readthedocs.io.
In short:

- Install the libusb tool
- Install the `nfcpy` module: `pip3 install nfcpy`
  When experiencing problems with the installation, you can try `pip3 install --upgrade --force-reinstall nfcpy`

pyscard

The *pyscard* module in turn uses a PC/SC driver to communicate with the tag. The driver is tag reader-specific and provides a small abstraction by translating tag reader APDU commands into tag-specific commands.

pyscard is hosted on sourceforge.net. When using it, follow these installation steps:

- Install the SWIG (Simplified Wrapper and Interface Generator) tool and add it to your `PATH` environment variable.
- Install Microsoft Visual C++ Build Tools 2015
  Note: this is already installed together with Visual Studio.
- Install the *pyscard* module: `pip3 install pyscard`
  When experiencing problems with the installation, you can try `pip3 install --upgrade --`

`force-reinstall pyscard`

Tag reader

When having the choice between using the nfcpy and the pyscard module in combination with your USB NFC tag reader, keep in mind that it is not possible to switch instantly between two different script executions. Only one option - *nfcpy* or *pyscard* - is available for use at any given time.

- Use Zadig to change the associated driver to *WinUSB* - for use with nfcpy;
- use the tag reader-specific installer to change the driver back for use with *pyscard*.

ACS ACR122

- for *nfcpy*: The PlugAndPlay driver automatically selected and installed by Win10 should be sufficient. On some PCs, a wrong USB driver gets associated with the tag reader. If that is the case, Zadig (Win10 only) to ensure that the correct USB driver - *WinUSB* - is selected.
- for *pyscard*: download the *ACS Unified PC/SC driver* for all platforms from the ACR support site. The associated driver will be *ACR122U*.

Identiv uTrust 3700F

For use with the *pyscard* module only.

The PlugAndPlay driver automatically selected and installed by Win10 should be sufficient for most PCs. Alternatively, you can download driver installers for all platforms from the Identiv support site.

Sony RC-S380

- for *nfcpy*: use Zadig (Win10 only) to ensure the correct USB driver - *WinUSB* - is used with the tag reader.
- for *pyscard*: install the NFC Port Software from Sony (Win10 only). The associated driver will be *sonynfcport100c*.

macOS and Linux

Only Win10 is supported. That said, this script has been reported to work on macOS and Linux (Ubuntu) as well.

- On macOS 10.15 Catalina
  using *nfcpy*, after installing *nfcpy* and *libusb*.
- On Ubuntu 19.10 Eoan Ermine
  using *nfcpy*, after installing *nfcpy*, *ndeflib* and *libusb*.

  Add `import ndef.message` to `site-packages/nfc/tag/__init__.py`
  Follow the instructions given after executing `nfc/__main__.py`

Usage

- The tag to program (containing the NHS31xx IC) must be battery powered and never been flashed before (i.e. the default firmware image *NHS31xx one-time NFC program downloader* is still active).
- Place the tag over an NFC USB reader. Ensure a stable connection. A better connection may be obtained by maintaining a small distance of about 1cm between the antenna of the tag and the NFC reader. The optimal distance is dependent on your specific material.
- Launch the script. When providing insufficient or incorrect arguments, or when providing the `-h` or `--help` argument, compact execution instructions are shown.

```
nfcloader.py [-h] [-d {nfcpy,pyscard}] [-t TAGREADER] file

positional arguments:
  file                  The binary file to be loaded into the target

optional arguments:
  -h, --help            show this help message and exit
  -d {nfcpy,pyscard}, --driver {nfcpy,pyscard}
                        Module to be used for tag handling (default:
pyscard)
  -t TAGREADER, --tagreader TAGREADER
                        tag reader to connect to (default: None)
```

nfcpy

The *nfcpy* module will try to use the first USB NFC tag reader it finds. This is equal to providing `-t usb` at the command line.

It is possible to discriminate between different models by supplying vendor ID (*vid*) and a product ID (*pid*). The format for a correct port name then becomes `usb:<vid><pid>`.

Examples:

- to target the Sony RC-S380 reader, provide `-t usb:054c:06c1` at the command line
- to target the ACS ACR122 reader, provide `-t usb:072F:2200` at the command line
- to target the Identiv uTrust 3700F reader, provide `-t usb:04e6:5790` at the command line

(Aside: the vendor ID for NXP products is `1fc9`.)

pyscard

The *pyscard* module will try to use the first tag it can find. This is usually the NFC reader built-in in your PC.

A list of readers accessible through *pyscard* can be obtained by running

```
python3 -c "from smartcard.System import readers; print(readers())"
```

With as example output:

```
['Broadcom Corp Contacted SmartCard 0', 'Broadcom Corp Contactless SmartCard
0', 'Identiv CLOUD 3700 F Contactless Reader 0']
```

To target the Identiv reader, you can provide a string that matches case-insensitive with that reader alone. For example, by providing `-t cloud` at the command line.

Operation

Once the python3 script is launched, access to a tag reader is tried. When that succeeds, the script checks if an NFC tag is present. When that fails, it waits for an NFC tag to be touched.

When a tag is within reach of the NFC field of the tag reader, the NDEF data is read out. When an NDEF message is found, containing a MIME record with version information as payload, *and* the version numbers match, the download process starts.

The provided binary file is split up in smaller portions 'chunked' and fed one by one to the NHS31xx IC via the NFC shared memory. Each chunk is wrapped in an NDEF message and sent to the tag. A new chunk is sent when the tag has acknowledged the data by writing a response in the NFC shared memory.

The last chunk is sent with a CRC calculated over the entire file. This is used to check is the full file is received without errors. The last action of the *NHS31xx one-time NFC program downloader* running on the NHS31xx IC is to shut down the IC. After any reset the newly download firmware application gets control.

End state

The last response contains the overall error code, indicating whether the flash is successfully re-programmed. The IC remains awake for a few 100 milliseconds after providing the last response, to ensure the host has enough time to read the NDEF message. After that, the IC enters the power-off mode.

Points of failure

During the download process, it is possible that

- a chunk of data is not correctly written in the NFC shared memory.
- a response is not written by the NHS31xx.
- a response contains an error code indicating a handling problem by the NHS31xx IC.
- a response is not correctly read out by the tag reader.

This can occur when

- The tag's position is not optimal.
- The tag's position is not kept stable.
- The file size is too big.

If anything goes wrong during download, the process will be aborted. The script will write a corresponding error message and exit. The host is to check the error and make adjustments to the setup. After waiting 5 seconds, the download can be retried.

One critical point of failure is the moment when the *NHS31xx one-time NFC program downloader* gets swapped out for the new firmware. This is done immediately after receiving the last chunk of data (which also includes the file CRC). The flash sector is erased, and written again. Sector `0` contains among others the interrupt vector table.

- After erasing sector `0`, the downloader firmware can no longer run after a reset.
- Before sector `0` is written again, the new firmware is not yet able to run after a reset.

A too severe voltage drop or a reset pulse will reset the IC. There is no valid firmware image, so the target will hang and a retry is not possible. The only way to overcome this issue is to program the IC wired via the HW pins (SWD pins).

Erasing a sector is more taxing to the supply than programming a sector. Programming or erasing a flash sector necessitates the presence of a battery or an external supply, to guarantee the load can be provided during the erase operation. When running passively, using the NFC harvested energy, this guarantee is not possible. The tag must be battery or externally powered.

Last: the write operation in flash is not read back to verify the correct contents have been written. Testing from our part of the on-chip flash driver has not raised a concern.

Examples

Identiv uTrust 3700F & app_demo_dp_tlogger

Using the Identiv uTrust 3700F tag reader and pyscard to flash the NHS3100 temperature logger demo from SDK 12.3.

Launching the script

```
C:\sdk\tools\nfcloader\python>python nfcloader.py -t identiv
C:\sdk\sw\nss\app_demo_dp_tlogger\Release\app_demo_dp_tlogger.bin
```

produces the following output:

```
NHS31xx one-time NFC program downloader
Searching for a tag
TAG connected
Firmware version 1.2
Tag present, application version OK
[ 0.00] Sending 486 bytes of data ...
        ... data sent. Waiting for response ...
        ... response received.
[ 0.79] Sending 486 bytes of data ...
        ... data sent. Waiting for response ...
        ... response received.
[ 1.58] Sending 486 bytes of data ...
        ... data sent. Waiting for response ...
        ... response received.
.
.
.
[28.56] Sending 486 bytes of data ...
        ... data sent. Waiting for response ...
        ... response received.
[29.35] Sending 486 bytes of data ...
        ... data sent. Waiting for response ...
        ... response received.
[30.15] Sending the last 484 bytes of data ...
        ... data sent. Waiting for response ...
        ... response received.
Download successful
Transmitted (bytes): 18952
Speed (bytes/sec): 609.20
```

Sony RC-S380 & app_demo_dp_blinky

Using the Sony RC-S380 tag reader and nfcpy to flash the 'Hello World' blinky demo.

Launching the script

```
C:\sdk\tools\nfcloader\python>python nfcloader.py -d nfcpy
C:\sdk\sw\nss\app_demo_dp_blinky\Release\app_demo_dp_blinky.bin
```

produces the following output:

```
NHS31xx one-time NFC program downloader
Now accessing tag reader SONY RC-S380/S NFC Port-100 v1.11 at usb:001:011.
Searching for a tag
TAG connected: Type2Tag ID=04CA0E00131221
Firmware version 1.2
Tag present, application version OK
[ 0.00] Sending 486 bytes of data ...
        ... data sent. Waiting for response ...
        ... response received.
[ 1.26] Sending 486 bytes of data ...
        ... data sent. Waiting for response ...
        ... response received.
[ 2.47] Sending 486 bytes of data ...
        ... data sent. Waiting for response ...
        ... response received.
[ 3.69] Sending 486 bytes of data ...
        ... data sent. Waiting for response ...
        ... response received.
[ 4.93] Sending the last 336 bytes of data ...
        ... data sent. Waiting for response ...
Download successful
Transmitted (bytes): 2280
Speed (bytes/sec): 378.01
```