

**nord stage 4**

**program file decoder**

# **ns4decode User Manual**

for MacOS and Windows  
(and probably Linux)

**edition 2**

2025/02/14

for ns4decode version 2.0 or later

## **Disclaimer**

The developers of `ns4decode` are not affiliated, associated, endorsed by, or in any way officially connected with Nord Keyboards / Clavia DMI AB, or any of its subsidiaries or its affiliates. The official Nord Keyboards website can be found at <https://www.nordkeyboards.com>. The names Nord and Clavia as well as related names, marks, emblems and images are registered trademarks of their respective owners.

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>Downloading and installing ns4decode</b>	<b>6</b>
2.1	Downloading ns4decode . . . . .	6
2.2	Installing ns4decode on MacOS . . . . .	6
2.3	Installing ns4decode on Windows . . . . .	6
<b>3</b>	<b>Do you need to download and install Python?</b>	<b>7</b>
<b>4</b>	<b>Two methods for running ns4decode</b>	<b>8</b>
<b>5</b>	<b>Using a command window</b>	<b>9</b>
5.1	Opening a command window . . . . .	9
5.2	Drag-and-drop . . . . .	9
5.3	Abandoning a command . . . . .	9
<b>6</b>	<b>Testing Python</b>	<b>10</b>
<b>7</b>	<b>Preparing a command window</b>	<b>12</b>
7.1	Instructions for MacOS . . . . .	12
7.2	Instructions for Windows . . . . .	14
<b>8</b>	<b>Verifying that it works</b>	<b>15</b>
<b>9</b>	<b>Copy-and-paste (for the IDLE method)</b>	<b>18</b>
<b>10</b>	<b>The IDLE method</b>	<b>19</b>
<b>11</b>	<b>Using ns4decode (basic)</b>	<b>22</b>
11.1	Decoding a single NS4 file, GUI output . . . . .	22
11.2	Decoding a single NS4 file, CSV output . . . . .	24
11.3	The CSV output files . . . . .	26
11.4	Formatting the CSV outputs for a plain text editor . . . . .	26
11.5	Decoding multiple NS4 files . . . . .	27
11.6	Decoding all NS4 files in a folder . . . . .	28

<b>12 Using ns4decode (advanced)</b>	<b>29</b>
12.1 Sending the CSV outputs to a specified folder . . . . .	29
12.2 Excluding parameters in inactive sections/layers/etc . . . . .	31
12.3 Combining options . . . . .	31
12.4 Getting the list of parameters, with codes . . . . .	31
12.5 Comparing selected parameters across multiple NS4 files . . . . .	32
12.6 Comparing selected parameters, part 2 . . . . .	33
12.7 Processing only selected NS4 file types in a folder . . . . .	35
<b>13 List of options</b>	<b>36</b>
13.1 Options when decoding NS4 files . . . . .	36
13.2 Options when not decoding anything . . . . .	39
<b>14 Customizing ns4decode</b>	<b>40</b>
14.1 The user preferences file . . . . .	40
14.2 Example: saving a <code>--writeto</code> preference . . . . .	41
<b>15 Understanding the outputs</b>	<b>42</b>
15.1 GUI output mode with Preset files . . . . .	42
15.2 Possible causes of “not decoded” . . . . .	42
15.3 Organ drawbars . . . . .	42
15.4 How samples are indexed . . . . .	42
15.5 Program transpose . . . . .	43
15.6 Piano timbre . . . . .	43
15.7 Organ vib/chorus type . . . . .	43
15.8 Exceptions to the one-row-per-parameter rule . . . . .	44
15.9 KB zones . . . . .	44
15.10 Arpeggiator accent, gate, and pan . . . . .	44
<b>16 Adding more sample names</b>	<b>45</b>
<b>17 Converting older NS4 files to the new version</b>	<b>46</b>
<b>18 Additional information for developers</b>	<b>47</b>
18.1 Regression testing on MacOS . . . . .	47
18.2 Regression testing on Windows . . . . .	48
18.3 Showing raw (uninterpreted) values . . . . .	49
18.4 Viewing ns4decode’s bitmaps . . . . .	50

<b>19 Troubleshooting</b>	<b>51</b>
19.1 Command not found . . . . .	51
19.2 Can't find ' <code>__main__</code> ' module . . . . .	51
19.3 Empty GUI . . . . .	51
19.4 Running <code>ns4decode</code> in verbose mode . . . . .	51
<b>20 Reporting problems</b>	<b>52</b>

# 1 Introduction

The Nord Stage 4 (NS4)<sup>1</sup> is a digital keyboard made by Clavia Digital Musical Instruments AB.<sup>2</sup> The associated Nord Sound Manager<sup>3</sup> software can be used to upload Program files (**ns4p** files), Live files (**ns4l** files), and Preset files (**ns4o**, **ns4n**, **ns4y**) from your NS4 to a desktop/laptop computer. These will be called **NS4 files** in the rest of this User Manual.

The purpose of **ns4decode** is to read those binary files and decode (most of) their content into a human-readable form. Two output modes are available:

- a graphical user interface (GUI),
- comma-separated value (CSV) files that can be opened with a spreadsheet app or with a plain text editor.

The GUI displays the content of a single NS4 file with a layout that roughly<sup>4</sup> follows the layout of the NS4 front panel, which makes it relatively intuitive. The CSV output mode can decode many NS4 files with a single command, and it can also produce a summary of selected parameters across many NS4 files.

This User Manual includes setup and usage instructions for both MacOS and Windows.<sup>5</sup> On both of those platforms, two different methods for running **ns4decode** are available:

- the command window method,
- the IDLE<sup>6</sup> method.

Each method has its own advantages. Section 4 compares them. This User Manual includes instructions for both.

---

<sup>1</sup><https://www.nordkeyboards.com/products/nord-stage-4/>

<sup>2</sup><https://www.nordkeyboards.com/about-us/contact/>

<sup>3</sup><https://www.nordkeyboards.com/software/nord-sound-manager/>

<sup>4</sup>The layout follows the NS4 front panel only roughly because the GUI displays more information simultaneously.

<sup>5</sup>One user has reported that **ns4decode** also works on Linux Mint 21.3. It may also work for other flavors of Linux. The usage instructions (command window method) are probably similar to those for MacOS because MacOS and Linux are both based on Unix.

<sup>6</sup>IDLE = Integrated Development and Learning Environment.

## 2 Downloading and installing ns4decode

### 2.1 Downloading ns4decode

ns4decode is free. It can be downloaded from this website:

<https://ns4decode.netlify.app/>

### 2.2 Installing ns4decode on MacOS

After you download `ns4decode_pkg.zip`, follow these steps:

1. Open the folder where your browser saved `ns4decode_pkg.zip`.
2. Double-click `ns4decode_pkg.zip`. This will create a new folder called `ns4decode_pkg`.
3. You may delete `ns4decode_pkg.zip` now, because that was just a temporary vessel for delivering the folder to your computer.
4. Use drag-and-drop to put the `ns4decode_pkg` folder wherever you want to keep it. (Example: you could put it on your Desktop.)

### 2.3 Installing ns4decode on Windows

After you download `ns4decode_pkg.zip`, follow these steps:

1. Open the folder where your browser saved `ns4decode_pkg.zip`.
2. Right-click on `ns4decode_pkg.zip` and select Extract All. This will create a new folder called `ns4decode_pkg`.
3. You may delete `ns4decode_pkg.zip` now, because that was just a temporary vessel for delivering the folder to your computer.
4. Use drag-and-drop to put the `ns4decode_pkg` folder wherever you want to keep it.<sup>7</sup> (Example: you could put it on your Desktop.)

---

<sup>7</sup>Some (maybe all) versions of Windows put the `ns4decode_pkg` folder inside another folder with the same name, so it's double-wrapped. You can move the inner `ns4decode_pkg` folder to wherever you want. You don't need to keep the extra outer wrapper.

### 3 Do you need to download and install Python?

To run `ns4decode`, your computer must have a recent enough version of Python installed.<sup>8</sup> If you are using MacOS and only want to use the CSV output mode, then you don't need to install Python.<sup>9</sup> If you are using Windows or if you want to use the GUI output mode, then you do need to install Python. This table summarizes the required downloads:

	MacOS	Windows
CSV output	<code>ns4decode</code>	<code>ns4decode</code> and Python
GUI output	<code>ns4decode</code> and Python	<code>ns4decode</code> and Python

To download and install Python, visit <https://www.python.org/> and follow the instructions given there.

---

<sup>8</sup>Professional app developers go to great lengths to make their apps more self-contained and cross-platform compatible, but the main developer of `ns4decode` (the author of this User Manual) is an amateur who does not have that much knowledge or time.

<sup>9</sup>`ns4decode` was originally developed only for MacOS and only for the CSV output mode. MacOS already includes a basic version of Python, so the CSV output mode works on MacOS without downloading anything other than `ns4decode` itself. However, the version of Python that comes with MacOS is not sufficient for the GUI output mode.

## 4 Two methods for running `ns4decode`

Two different methods for running `ns4decode` are available:

- The **command window method**.
  - With this method, you can drag-and-drop the files you want to decode into the command window. This doesn't move the files, but it tells `ns4decode` where to find them.
  - One drawback of this method is that the command window must be set up first.<sup>10</sup> (This only needs to be done once as long as the command window stays open.)
- The **IDLE method**, which should be available if you installed Python yourself (section 3).
  - With this method, the steps to set up a command window are not needed.
  - Drag-and-drop doesn't work with this method, but the paths to the NS4 files may be copy-and-pasted instead.
  - One drawback of this method is that the input dialog is small,<sup>11</sup> so longer commands (e.g. with multiple NS4 input files) won't fit in the view. The commands should still work if they're entered correctly, but checking the command visually can be difficult when it doesn't fit in the view.

Sections 5-8 explain how to set up the command window method. If you only want to use the IDLE method, you may skip to section 9. For simplicity, the usage instructions starting in section 11 are written for the command window method, but section 10 explains how to modify them for the IDLE method.

---

<sup>10</sup>The instructions for Windows in this edition of the User Manual are simpler than they were in the first edition, because the author learned a better way to do it.

<sup>11</sup>It does not appear to be resizable.



## 5 Using a command window

*This section is for the command window method only. You may skip this section if you only want to use the IDLE method.*

Using `ns4decode` is a matter of writing commands in a **command window** to tell `ns4decode` what you want to do. This section introduces a few basic command window skills, just enough to setup and use `ns4decode`.

### 5.1 Opening a command window

The MacOS command window is called **Terminal**. If you don't know where to find Terminal, you can press command-space to open the search tool, type *Terminal* in the search bar, and press return. This opens a Terminal window.

The Windows command window is called **Command Prompt**. To open one, you can go to the Windows start menu and type either *Command Prompt* or *Terminal*.<sup>12</sup>

### 5.2 Drag-and-drop

If you drag-and-drop a file or folder into a command window, the file/folder does not move, but the full path to that file or folder is appended to whatever command you started typing in the command window. This trick will be used repeatedly in this User Manual.

### 5.3 Abandoning a command

If you start typing a command and then change your mind, you can abandon the command without executing it by pressing `ctrl-C` (press `C` while holding down the `control` key). This discards the command and gives you a fresh prompt for another command. This can also be used to stop a long-running command before it finishes.

---

<sup>12</sup>Depending on your Windows computer's configuration, *Terminal* might open another flavor of command window called **PowerShell**. The instructions in this User Manual should work for either flavor of command window.

## 6 Testing Python

*This section is for the command window method only. You may skip this section if you only want to use the IDLE method.*

Section 7 will explain how to prepare a command window. First, you need to know the command that runs Python. To check that, follow these steps:

1. In a command window, type either

```
python3
```

or

```
python
```

and press return. If one of these doesn't work, then try the other one. If it works, then the prompt (the characters at the beginning of the new line) should look like this:

```
>>>
```

This is the Python prompt.

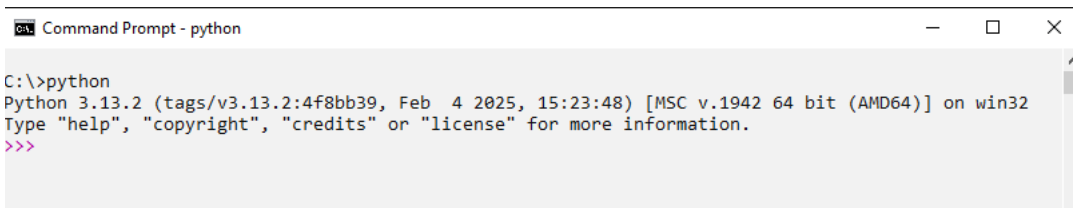
2. At the Python prompt, type

```
quit()
```

and press return. This exits Python but leaves the command window open.

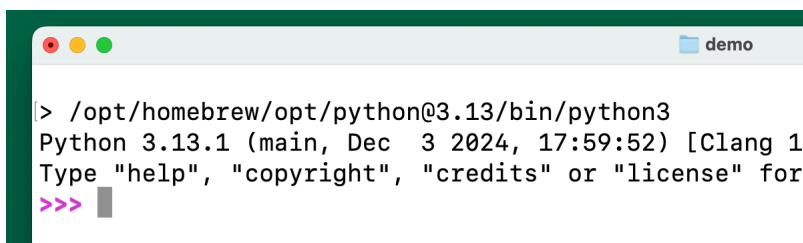
Those steps should work on Windows, and they should work on MacOS if you're using the version of Python that comes with MacOS (which does not support the GUI output mode).

If you're using MacOS and want to use the version of Python that you installed yourself, you'll need to find the command that runs that version of Python. One way to do this is to go to the folder (directory) where that version of Python is installed, look for a subfolder called `bin`, and in that subfolder look for a file called `python3`. Right-click while holding the option key and select *Copy "python3" as pathname*, then use command-V to paste that path into a command window, and press return. This should show you the command to run that version of Python. The command will be long because it includes the full path. Figure 6.2 shows an example.



```
Command Prompt - python
C:\>python
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb 4 2025, 15:23:48) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figure 6.1: Example of running Python on Windows. In this example, the command `python` (shown here on the first line) runs Python. For some installations, the command might be `python3`. The next two lines are the welcome message displayed by Python. The `>>>` is the Python command prompt.



```
demo
|> /opt/homebrew/opt/python@3.13/bin/python3
Python 3.13.1 (main, Dec 3 2024, 17:59:52) [Clang 14
Type "help", "copyright", "credits" or "license" for
>>>
```

Figure 6.2: Example of running a manually installed version of Python on MacOS (the last paragraph in section 6). In this example, the command `/opt/homebrew/opt/python3.13/bin/python3` runs the manually installed version instead of the built-in version (which would be just `python3`, without the long path). This is only an example – your path will probably be different. The next two lines are the welcome message displayed by Python. The `>>>` is the Python command prompt.

## 7 Preparing a command window

*This section is for the command window method only. You may skip this section if you only want to use the IDLE method.*

Section 7.1 (for MacOS) and 7.2 (for Windows) explain how to create an abbreviation (“`ns4decode`”) that tells the command window how to run `ns4decode`. As long as you keep that command window open, it will remember what the abbreviation “`ns4decode`” means, so you only need to do this once.

Sections 8 and 11-18 assume that you created this abbreviation.

To use `ns4decode` in another command window, you’ll need to repeat these steps in that window.

### 7.1 Instructions for MacOS

*If you only want to use the CSV output mode, then you may omit step 3 because `ns4decode` already knows where MacOS’s built-in version of Python is installed.*

1. **Open a command window** (section 5.1).
2. **Start forming an abbreviation.** In the command window, type

```
alias ns4decode="
```

with no space at the end. Don’t press return yet. The last two characters are the equals character and the double-quote character.

3. **Continue forming the abbreviation.** Section 6 explained how to determine the command that runs Python. Type (or paste) that command into the command window. (This should append the path to the line that you started typing in step 2.) Press the spacebar to add a space after the path.
4. **Finish forming the abbreviation.** Open the `ns4decode_pkg` folder (section 2). It contains a file called `ns4decode.py`. Drag-and-drop that one file into the command window (section 5.2). Click anywhere in the command window to bring it back into focus and type another double-quote character (to balance the one in step 2), and press return. This is illustrated in figures 7.1-7.2.

Now you can start using `ns4decode` (section 11) in this command window.

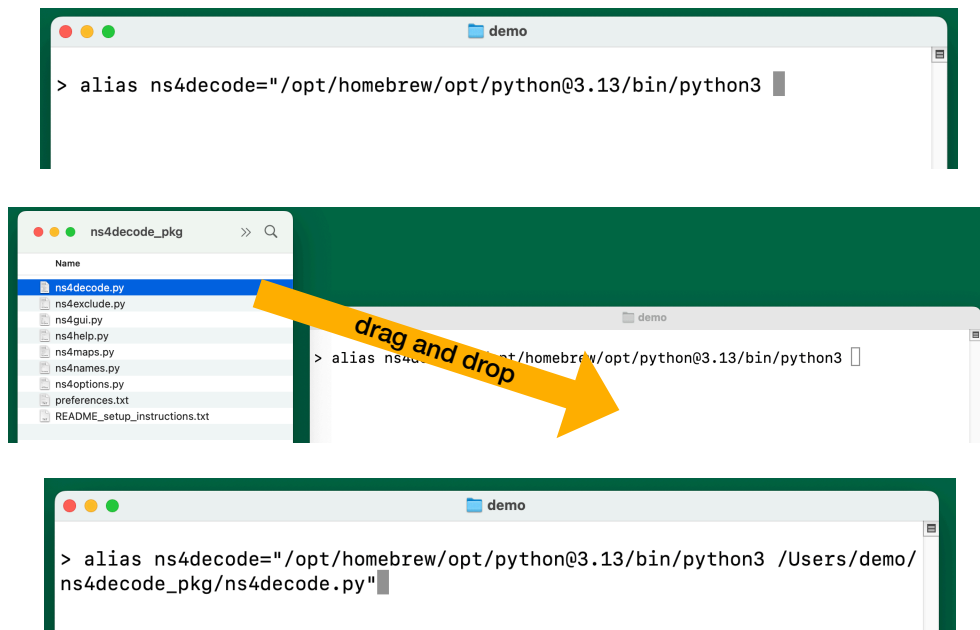


Figure 7.1: (This is for the command window method only. Figure 10.1 illustrates the IDLE method.) The first picture illustrates what the line might look like (depending on where you installed Python on your MacOS computer) after step 3 in section 7.1. The second picture shows how to do step 4, and the third picture shows the full abbreviation-creating command.

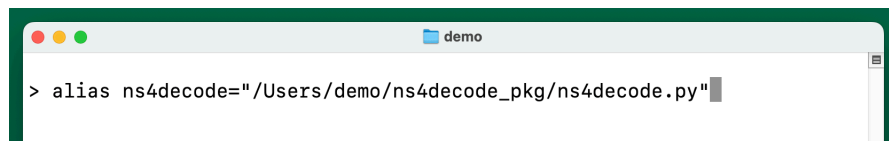


Figure 7.2: This is what the full abbreviation-creating command looks like for MacOS if you only need ns4decode's CSV output mode. The path to ns4decode on your computer will be different.

## 7.2 Instructions for Windows

*This section is for the command window method only. You may skip this section if you only want to use the IDLE method.*

1. **Open a command window** (section 5.1).
2. **Start forming an abbreviation.** In the command window, type

```
doskey ns4decode=
```

with no space at the end. Don't press return yet.

3. **Continue forming the abbreviation.** Section 6 explained how to determine the command that runs Python. Type (or paste) that command into the command window. (This should append the path to the line that you started typing in step 2.) Press the spacebar to add a space after the path.
4. **Finish forming the abbreviation.** Open the `ns4decode_pkg` folder (section 2). It contains a file called `ns4decode.py`. Drag-and-drop that file into the command window (section 5.2). Click anywhere in the command window to bring it back into focus, press the spacebar once to add a space at the end, type the two characters `$*`, and then press return. Figure 7.3 shows an example.

Now you can start using `ns4decode` (section 11) in this command window.



Figure 7.3: Example of the full abbreviation-creating command for Windows. The path to `ns4decode` on your computer will be different.

## 8 Verifying that it works

*This section is for the command window method only. You may skip this section if you only want to use the IDLE method.*

These instructions should work for both MacOS and Windows.

To test the abbreviation that you created in section 7, type

```
ns4decode
```

in the same command window and press return. It should say something that starts with “This is version ... of `ns4decode`,” like in figure 8.1. If it says something like “command not found” or “not recognized,” then try the steps in section 7 again, paying close attention to whitespace and special characters.

Windows users are advised to try one additional test: type

```
ns4decode --help
```

in the same command window and press return. It should display a long list of options, like in figure 8.2. If it says what running `ns4decode` by itself said in the first test, then something is wrong.<sup>13</sup>

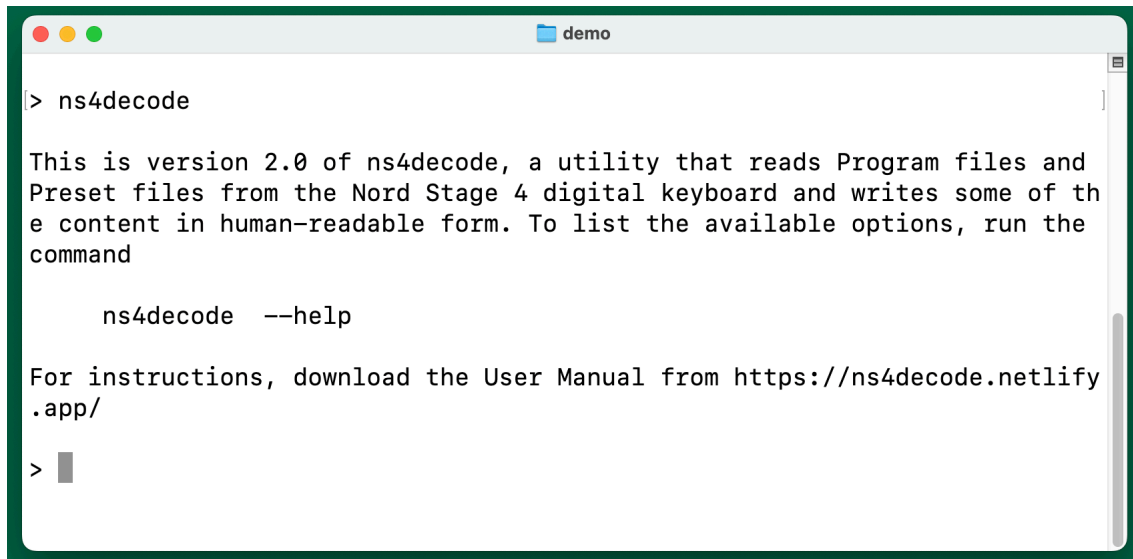
The rest of this User Manual assumes that the test(s) worked correctly.

Here’s a time-saving tip that works with MacOS and Linux.<sup>14</sup> In a command window, pressing the tab key should auto-complete whatever you started typing, if it’s unambiguous. Instead of typing the full word “`ns4decode`,” type just the first few letters (example: `ns4`) and then press the tab key. If `ns4decode` is the only recognized command that starts with those letters, then the command window should auto-complete it when you press the tab key.

---

<sup>13</sup>The purpose of this test is to make sure that the last two characters `$*` in step 4 in section 7.2 are doing their job: they should tell Windows to pay attention to whatever follows the word “`ns4decode`” on same line.

<sup>14</sup>This trick doesn’t work in Windows. Pressing the tab key in Windows causes it to auto-complete the name of a file or folder in the current working directory instead of causing it to auto-complete the name of a command.



```
> ns4decode

This is version 2.0 of ns4decode, a utility that reads Program files and
Preset files from the Nord Stage 4 digital keyboard and writes some of th
e content in human-readable form. To list the available options, run the
command

    ns4decode --help

For instructions, download the User Manual from https://ns4decode.netlify
.app/

> █
```

Figure 8.1: A successful test. This is what should happen when you type `ns4decode` by itself and press return, for both MacOS and Windows. This will only work after the abbreviation “`ns4decode`” is defined (in the same command window) as explained in section 8.



```
Command Prompt
C:\>ns4decode --help

Options when decoding
  --gui
    Abbreviation: -g
  --nogui
  --csv
  --nocsv
  --writeto (folder)
    Abbreviation: -w
  --columnwidth (number of characters)
    Abbreviation: -c
  --file (filename)
    Abbreviation: -f
  --find (list of file types separated by commas without spaces)
  --exclude
    Abbreviation: -x
  --includeall (on or off)
    (setting this to off is equivalent to the option -x)
  --askfirst (on or off)
  --warnversion (on or off)
  --verbosity (0, 1, 2, or 3)
    Abbreviation: -v
  --rawdecimal
  --rawhex
  --rawbinary
  --use (folder)

Options when not decoding
  --help
    Abbreviation: -h
  --settings
  --listparams
  --listparamsnocodes
  --version
  --bitmaps

For instructions, download the User Manual from https://ns4decode.netlify.app/

C:\>
```

Figure 8.2: Another successful test. This test verifies that the `ns4decode` command that you defined in section 7.2 pays attention to everything on the command line (`--help` in this case).

## 9 Copy-and-paste (for the IDLE method)

*This section is for the IDLE method only. You may skip this section if you only want to use the command window method.*

Using `ns4decode` is a matter of writing inputs in an **input dialog** to tell `ns4decode` what you want to do. Section 10 will explain how to open an input dialog.

The last part of a typical set of inputs is the name(s) of one or more NS4 files that you want to decode. These names must include the full path so that `ns4decode` knows where to find the files. This is called the **fully qualified name** of the NS4 file. Typing the fully qualified name manually can be tedious, but it's not necessary. You can copy-and-paste the fully qualified name by following these steps:

1. Find an NS4 file (or folder)<sup>15</sup> that you want to decode.
2. Copy the fully qualified name by following these steps:
  - On MacOS, right-click on the file while holding the option key, and select *Copy ... as pathname*.
  - On Windows, right-click on the file while holding the shift key, and select *Copy ... as path*.
3. Paste the fully qualified name into the input dialog, like this:
  - On MacOS, use `command-V` to paste it into the input dialog.
  - On Windows, use `ctrl-V` to paste it into the input dialog.

---

<sup>15</sup>Instead of decoding an individual NS4 file, you can decode all of the NS4 files in a folder by giving `ns4decode` the fully qualified name of the folder. Section 11.6 explains how to do this with the command line method, and it can be adapted to the IDLE method as explained at the end of section 10.

## 10 The IDLE method

*You may skip this section if you only want to use the command window method.*

These instructions should work for both MacOS and Windows. With this method, the inputs for `ns4decode` are typed or pasted into an **input dialog**. To open the input dialog, follow these steps, which are illustrated in figures 10.1-10.3:

1. Go to the `ns4decode_pkg` folder that you installed in section 2.
2. That folder contains a file called `ns4decode.py`. Right-click on that file and select Open With > IDLE.<sup>16</sup>
  - If the Open With does not show IDLE as an option, then you can try this instead: find where Python is installed and look for a file with a name that starts with the letters “IDLE” or “idle.” Drag-and-drop `ns4decode.py` onto that file.

This might open two windows, one with a title like “IDLE Shell” and one with the title `ns4decode.py`, or it might open only one window.

3. If it opened two windows, click in the one titled `ns4decode.py` to bring it into focus.
4. Find the Run menu<sup>17</sup> and select Run...Customized. This opens the input dialog, a dialog window with a title like “Customize `ns4decode.py` Run.”
  - Instead of using the Run menu, you can press `shift-F5`. This also opens the input dialog.
  - If the Run menu isn’t visible or if `shift-F5` doesn’t work, then make sure the correct window is in focus (step 3) and try again.

The usage instructions in the rest of this User Manual are written for the command window method, but you can adjust them for the IDLE method like this:

- Replace “command window” with “input dialog.”
- Omit the word `ns4decode` at the beginning of the command.
- Use copy-and-paste (section 9) instead of drag-and-drop.

---

<sup>16</sup>The details of the name may vary, but it should start with the letters “IDLE” or “idle.”

<sup>17</sup>On MacOS, the Run menu will be at the top of the screen. On Windows, it will be at the top of the window.

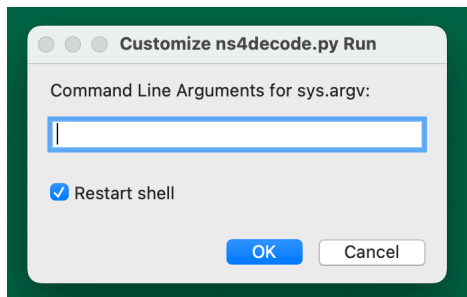
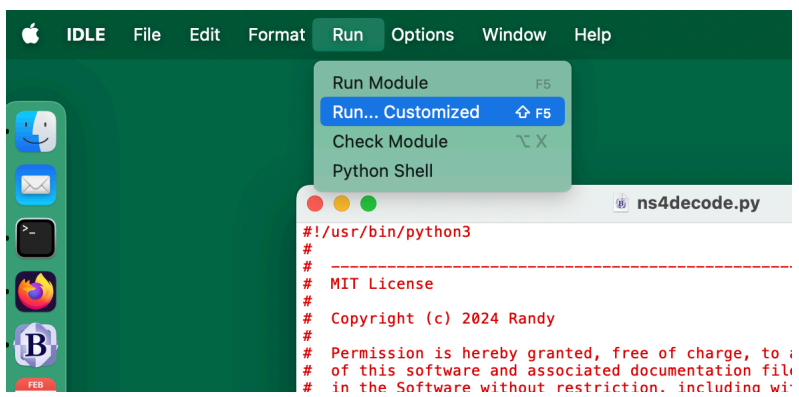
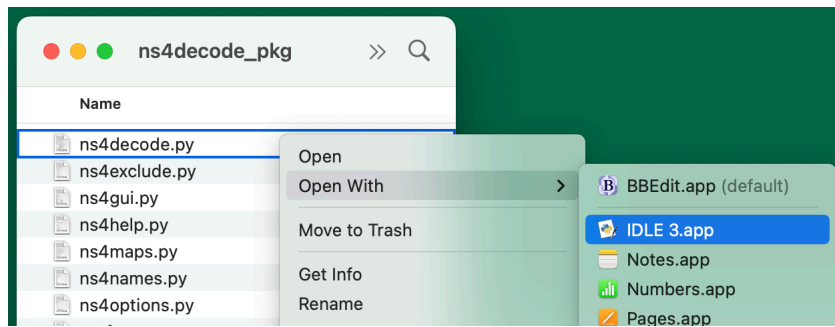


Figure 10.1: This sequence of pictures illustrates the instructions in section 10 for the MacOS case. The first picture shows how to open ns4decode with IDLE after right-clicking on the file ns4decode.py. The second picture shows how to open the input dialog. (Figure 10.3 shows the Windows version of the second picture.) The third picture shows that the input dialog looks like.

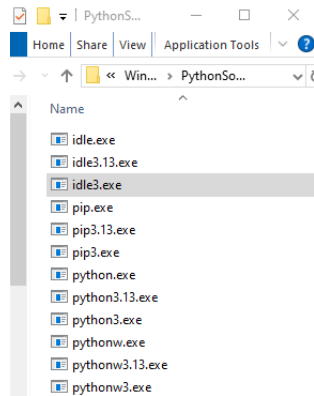


Figure 10.2: If IDLE is not listed in the Open With menu (the first picture in figure 10.1), then you can look for it on your computer. This picture illustrates what it might look like after you find it. To open `ns4decode` with IDLE, you can drag-and-drop the file `ns4decode.py` onto one of the `idle*.exe` files shown here.

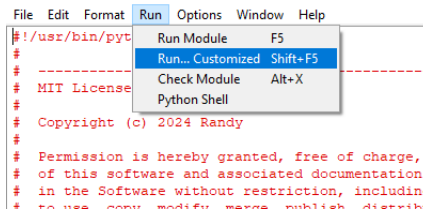


Figure 10.3: This is the Windows version of the second picture in figure 10.1. The resulting input dialog looks the same as the third picture in figure 10.1.

## 11 Using ns4decode (basic)

These instructions should work for both MacOS and Windows. The end of section 10 explained how to adjust these instructions for the IDLE method.

### 11.1 Decoding a single NS4 file, GUI output

1. In the prepared command window (section 7), type

```
ns4decode --gui
```

followed by a space, but don't press return yet. You can use `-g` as an abbreviation for `--gui`.

2. Find the NS4 file<sup>18</sup> that you want to decode, and drag-and-drop the file into the command window (section 5.2). Figure 11.1 illustrates the fully-formed command, and figure 11.2 illustrates the IDLE version.
3. Click anywhere in the command window to bring it back into focus, and press return to run the command.

This will open a GUI window that displays the outputs. By default, the outputs are also written to CSV files (section 11.2). If you don't want to write CSV files, then replace the option `--gui` in step 1 with `--gui --nocsv`.

For Program files (`ns4p`) and Live files (`ns4l`), the GUI window has these pages:

Global OrganA OrganB PianoA PianoB SynthA SynthB SynthC

To view a page, click its button at the top of the GUI window. The letters A/B/C refer to individual layers within the Organ, Piano, and Synth sections. For a Preset file (`ns4o`, `ns4n`, `ns4y`), only the pages appropriate for that file type are shown. Tips:

- After using the mouse to select a page, you can use the arrow keys or the mouse wheel to quickly flip between pages.
- Section 12.2 explains how to tell `ns4decode` to mask inactive values. When that option is used, inactive values are shown in a darker shade.

---

<sup>18</sup>Section 1 defined "NS4 file."

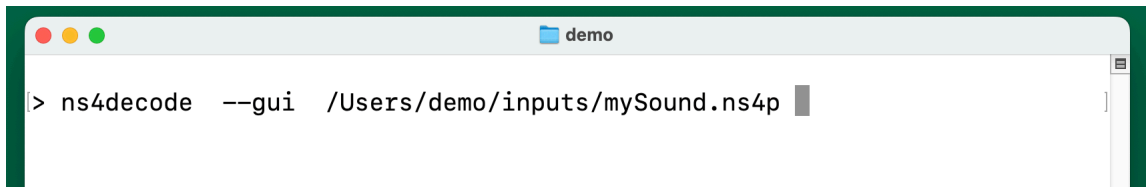
A terminal window titled "demo" with a light gray background. The command prompt shows the command `ns4decode --gui /Users/demo/inputs/mySound.ns4p` entered. The cursor is at the end of the command. The window has standard macOS window controls (red, yellow, green buttons) in the top-left corner.

Figure 11.1: This is the command formed in steps 1 and 2 in section 11.1 to run `ns4decode` on a single NS4 file, using the GUI output mode. Your NS4 filename will be different.

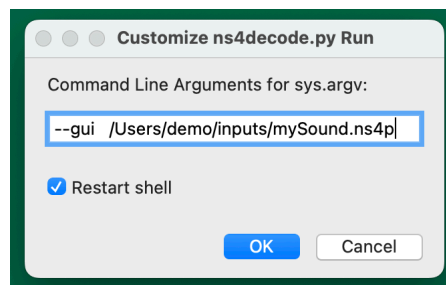


Figure 11.2: This is the IDLE version of figure 11.1. Figure 10.1 showed how to open this input dialog.

## 11.2 Decoding a single NS4 file, CSV output

1. In the prepared command window (section 7), type

```
ns4decode
```

followed by a space, but don't press return yet.

2. Find the NS4 file<sup>19</sup> that you want to decode, and drag-and-drop the file into the command window (section 5.2).
3. Click anywhere in the command window to bring it back into focus, and press return.

This is illustrated by the first picture in figure 11.3. If the input file is a Program (**ns4p**) file, say `mySound.ns4p`, then these output files should appear in the same folder that holds that file:

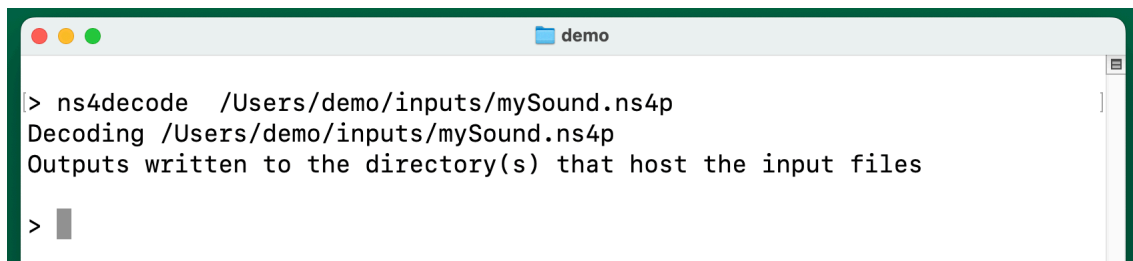
```
mySound_master.csv  
mySound_organ.csv  
mySound_piano.csv  
mySound_synth.csv
```

This is illustrated in figure 11.3. If the input file is a Preset file (**ns4o**, **ns4n**, **ns4y**), then only two output files are written (the `_master.csv` file and one of the others, depending on the type of Preset file). Section 11.3 will explain what's in these files and how to open them.

---

<sup>19</sup>Section 1 defined "NS4 file."





```
> ns4decode /Users/demo/inputs/mySound.ns4p
Decoding /Users/demo/inputs/mySound.ns4p
Outputs written to the directory(s) that host the input files
>
```

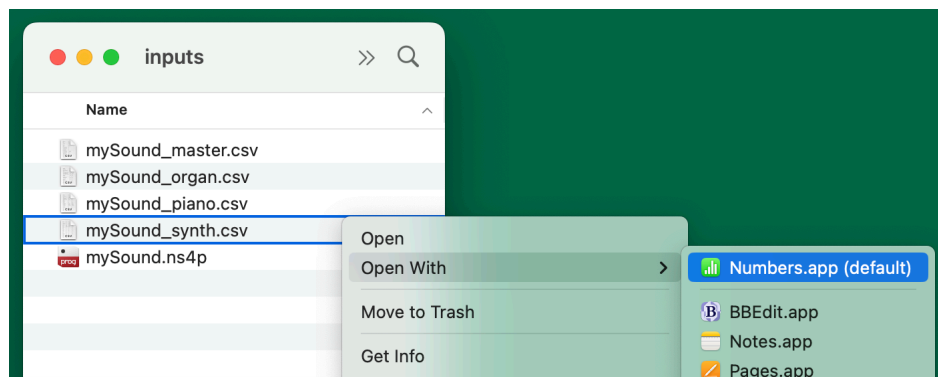
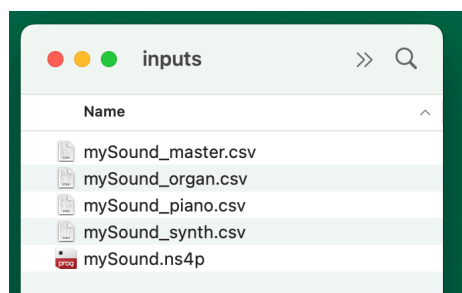


Figure 11.3: The first picture shows an example of running the command described in section 11.2. (If you’re using the IDLE method, you will omit the word “ns4decode” at the beginning of the command.) The second picture shows the resulting list of CSV output files in the same directory as the input file (mySound.ns4p in this example). The third picture illustrates how to open one of the CSV files (section 11.3). Depending on how your computer is configured, you might be able to open a CSV file just by double-clicking it.

### 11.3 The CSV output files

The `_organ.csv`, `_piano.csv`, and `_synth.csv` output files show the parameter values for each organ, piano, and synth layer. The `_master.csv` output file shows parameters that are not tied to any individual layer. This includes the organ FX parameters, because they're the same for both organ layers.

The output files are written in the same folder that holds the original NS4 file.<sup>20</sup> You might be able to open these files in your computer's default spreadsheet app by double-clicking on them. If that doesn't work, or if you prefer to open them with a plain text editor, then you can try this:

**MacOS:** Right-click on the file and select either Open With > Numbers (spreadsheet app) or Open With > TextEdit (a plain text editor).

**Windows:** Right-click on the file and select either Open With > Excel (spreadsheet app) or Open With > Notepad (a plain text editor).

The contents are formatted as a table in which each row represents one parameter, and the parameter values for each layer are in separate columns.

### 11.4 Formatting the CSV outputs for a plain text editor

If you choose to view the output files using a plain text editor, then you can align the columns by replacing

```
ns4decode
```

in step 1 in section 11.2 with this:

```
ns4decode --columnwidth 12
```

You may replace the “12” with whatever you want the minimum column width to be.<sup>21</sup> This abbreviation is equivalent:

```
ns4decode -c 12
```

The other steps are the same as in section 11.2.

---

<sup>20</sup>This is the default behavior. You can tell `ns4decode` where to write them instead (section 12.1).

<sup>21</sup>If the specified number of characters is greater than the width of the output string, then the column will be padded with whitespace up to the specified number of characters. If the specified number of characters is less than the width of the output string, then the full output string will still be shown but the column might not be aligned nicely.

## 11.5 Decoding multiple NS4 files

1. In the prepared command window (section 7), type

```
ns4decode
```

followed by a space, but don't press return yet.

2. Drag-and-drop one NS4 file into the command window. Click anywhere in the command window to bring it back into focus, and add a space at the end.<sup>22</sup>
3. Repeat step 2 for more NS4 files.
4. Click anywhere in the command window to bring it back into focus, and press return.

This will write a set of CSV files for each of the NS4 files, in the same folder(s) that hold those NS4 files. This is equivalent to running `ns4decode` separately on each individual NS4 file (section 11.2).

On MacOS, instead of using drag-and-drop for each individual NS4 file as in steps 2-3, you can select multiple NS4 files and then drag-and-drop all of them into the command window at once. The result is the same.

When decoding multiple NS4 files with the `--gui` option, the GUI opens a window that displays the results for the first file, and when you close that window it opens a second window with the results for the second file, and so on. If you use the option `--gui` with a large number of input files and want to stop before viewing all of them, use `ctrl-C` in the command window (section 5.3).

---

<sup>22</sup>MacOS adds the space automatically.

## 11.6 Decoding all NS4 files in a folder

1. In the prepared command window (section 7), type

```
ns4decode
```

followed by a space, but don't press return yet.

2. Find a folder that contains one or more NS4 files that you want to decode, and drag-and-drop the whole folder into the command window.
3. Click anywhere in the command window to bring it back into focus, and press return.

This is equivalent to running `ns4decode` individually on all of the NS4 files in the selected and its subfolders, recursively. The number of CSV output files will be proportional to the number of NS4 files. The selected folder and its subfolders could contain a large number of NS4 files, so `ns4decode` will ask you to confirm that you really want to decode them all.

## 12 Using ns4decode (advanced)

These instructions should work for both MacOS and Windows. The end of section 10 explained how to adjust these instructions for the IDLE method. The available options are summarized in section 13.

### 12.1 Sending the CSV outputs to a specified folder

If you don't want the output files to be written in the same folder(s) as the original NS4 files, then you can tell `ns4decode` where to write them:

1. Create a folder to hold the outputs that `ns4decode` will write. You may create the folder wherever you want, and you may give the folder whatever name you want. This example will pretend that you named it `outputFolder`.
2. In the prepared command window, type either one of the equivalent commands

```
ns4decode --writeto  
ns4decode -w
```

followed by a space, but don't press return yet.

3. Drag-and-drop `outputFolder` from step 1 into the command window. Click in the command window to bring it back into focus and add a space at the end, but don't press return yet.
4. Use drag-and-drop to tell `ns4decode` which NS4 files you want to decode, like in sections 11.2, 11.5, or 11.6.

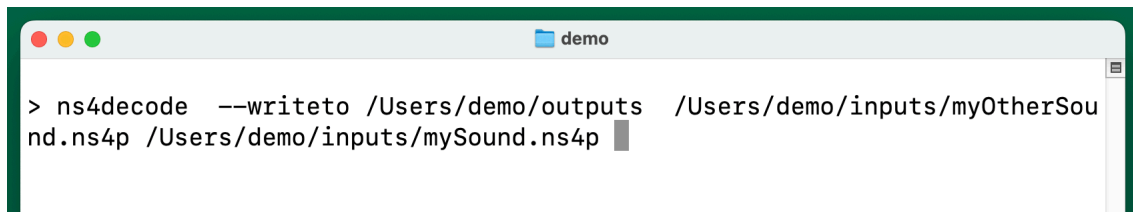
Figures 12.1-12.2 show an example.<sup>23</sup>

---

<sup>23</sup>Computer-savvy users who prefer working at the command line (instead of using a file browser) might want to make `ns4decode` write the CSV output files to the current working directory. The command

```
ns4decode --writeto . (list of NS4 files)
```

does that. Notice the period, which means "put the files *right here*."

A terminal window titled "demo" with a dark green border. The terminal text is: 

```
> ns4decode --writeto /Users/demo/outputs /Users/demo/inputs/myOtherSound.ns4p /Users/demo/inputs/mySound.ns4p
```

Figure 12.1: This is an example of a command that decodes two specific NS4 files (mySound.ns4p and myOtherSound.ns4p) and writes the CSV output files in a specified folder (/Users/demo/outputs) instead of in the folder that holds the input files (/Users/demo/inputs).

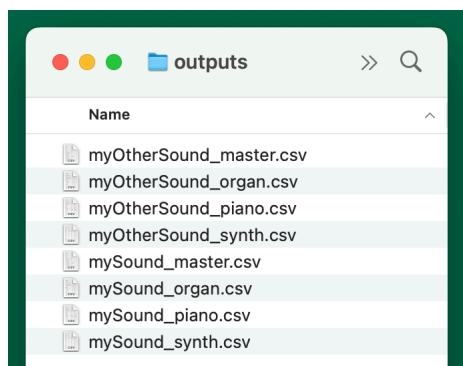


Figure 12.2: This shows the list of CSV output files after the command in figure 12.1 is done running.

## 12.2 Excluding parameters in inactive sections/layers/etc

By default, the output files show the values of all of the parameters that `ns4decode` knows how to decode, even if those parameters don't affect the sound, like the parameters in a layer that is turned off. You can tell `ns4decode` to omit the values of most of those parameters. This feature doesn't omit all unused parameters, but it does omit most of the major groupings of unused parameters (sections, layers, and some smaller groupings). To use this feature, start with this step:

1. In the prepared command window, type either one of the equivalent commands

```
ns4decode --exclude  
ns4decode -x
```

followed by a space, but don't press return yet.

The other steps are the same as before.

## 12.3 Combining options

Options may be combined. Examples:

```
ns4decode -c 12 -x (NS4 files)  
ns4decode -c 12 -x -w (output folder) (NS4 files)
```

The NS4 file(s) must always be listed after the options. In the second example, whatever comes right after the `-w` will be treated as the folder where you want the output files to be written. Everything after that will be treated as an NS4 file to be decoded.

## 12.4 Getting the list of parameters, with codes

In the prepared command window, type either of the equivalent commands

```
ns4decode --listparams  
ns4decode -l
```

and press return. This gives you a list of available parameters, each with a concise code next to its name.<sup>24</sup> You'll need this for the feature described in section 12.5.

---

<sup>24</sup>The list is long. You can send it to a file by appending `>(filename)` to the end of the command.

## 12.5 Comparing selected parameters across multiple NS4 files

Instead of using `ns4decode` to decode everything in an NS4 file, you may also use it to quickly compare a small number of parameters across multiple NS4 files:

1. After getting the list of parameters as described in section 12.4, decide which parameters you want to compare, and type

```
ns4decode (list of codes)
```

but don't press return yet. Example: if the parameters of interest are `-m 17`, `-y 2`, and `-y 3`, then the command will look like this:

```
ns4decode -m 17 -y 2 -y 3
```

or you can abbreviate it like this:

```
ns4decode -m 17 -y 2,3
```

Make sure there are spaces between the letters and numbers but no spaces in the comma-delineated list of numbers. Make sure there is a space at the end. Don't press return yet.

2. To finish forming the command, use drag-and-drop to append the list of NS4 files (or a folder with lots of NS4 files), and then press return.

That command doesn't write a file, but it does display CSV-formatted output in the command window. Section 12.6 explains how to make it write that output to a file and explains what the columns mean, and figure 12.3 shows an example.



## 12.6 Comparing selected parameters, part 2

Section 12.5 explained how to form a command that compares selected parameters across multiple NS4 files. The output from that command is written into the command window instead of to a file. If you want the output to go to a file, then you need to specify the location and the filename. To do that, insert these steps between steps 1 and 2:

1b. Type

```
--file
```

Make sure there is a space at the end. Don't press return yet.

- 1c. Drag-and-drop the desired location (a folder) into the command window. Click in the command window to bring it back into focus. Don't add a space at the end (if there is a space at the end, delete it), and don't press return yet.
- 1d. On MacOS, type a forward-slash. On Windows, type a backslash. Don't add a space, and don't press return yet.
- 1e. Type the name that you want the file to have. The name should end in `.csv` so that double-clicking on the file will open it in a spreadsheet app. Make sure there is a space at the end. Don't press return yet.

Finish forming the command using step 2 in section 12.5. Figure 12.3 shows an example. The resulting CSV output file has one row for each NS4 file, and it has these columns:

- one column showing the location of the Program/Preset in the NS4,
- 1, 2, or 3 columns (one per layer) for each specified parameter,
- one column showing the filename without the `.ns4p/o/n/y` extension.

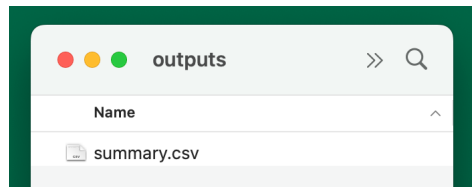
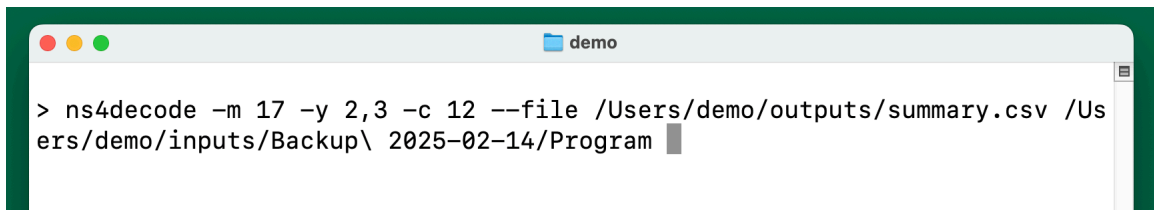
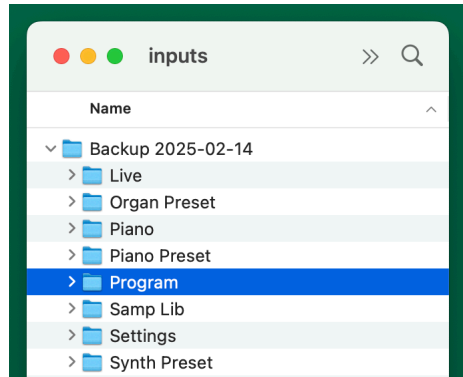


Figure 12.3: The first picture shows a Backup folder with all of the Program files, Live files, and Preset files from the NS4, obtained using the Nord Sound Manager. The subfolder with all of the Program files is highlighted. The second picture shows an example of a command that will decode the specified parameters (`-m 17 -y 2,3`) in all of the Program files and write the results in a file with the specified location and name (`/Users/demo/outputs/summary.csv`). The NS4 file(s)/folder(s) must be the last thing on the command line. The third picture shows the resulting output file.

## 12.7 Processing only selected NS4 file types in a folder

When the input to `ns4decode` is a folder instead of individual NS4 files, it will process all NS4 files that it finds in that folder and its subfolders (section 11.6). If you only want to process a subset of those file types, you can use the `--find` option.<sup>25</sup> Example: the command

```
ns4decode --find ns4p,ns4l,ns4y (folder)
```

will only process Program files, Live files, and synth Preset files, but not organ or piano Preset files (`ns4o` or `ns4n`). The list of file types must be comma-separated with no spaces on either side of the commas.

The `--find` option becomes important when comparing selected parameters across multiple NS4 files (section 12.5), because `ns4decode` tries to read the selected parameters from each one of those NS4 files. If the folder contains a Preset file (`ns4o`, `ns4n`, or `ns4y`) and the Preset file does not contain one of the selected parameters, `ns4decode` will exit and report an error. Example: the command

```
ns4decode -y 2,3 (folder)
```

will run with no errors if the specified folder does not contain any organ or piano Preset files, but it will exit with an error if it finds either of those file types, because they do not contain the specified synth parameters. That error can be avoided by using this command instead:

```
ns4decode -y 2,3 --find ns4p,ns4l,ns4y (folder)
```

---

<sup>25</sup>This option was added in version 2.0 of `ns4decode`.

## 13 List of options

### 13.1 Options when decoding NS4 files

`--csv`

Write CSV output files. This is the default behavior,<sup>26</sup> but this option can still be useful if `--nocsv` is in your preferences file (section 14).

`--nocsv`

Don't write CSV output files.

`--gui`

Show the outputs in a GUI (section 11.1).

`--nogui`

Don't show the outputs in a GUI. This is the default behavior,<sup>26</sup> but this option can still be useful if `--gui` is in your preferences file (section 14).

`--writeto` (folder)

Tells `ns4decode` where to write CSV output files (section 12.1). The default is to write them in the same folder(s) as the original NS4 file(s). If a `--writeto` location is specified in your preferences file (section 14) and you want to restore the default behavior, use `--writeto ""` (double-quotes with nothing between them).

`-w` (folder)

Abbreviation for `--writeto`

`--columnwidth` (number of characters)

This option is explained in section 11.4.

`-c`

Abbreviation for `--columnwidth`

---

<sup>26</sup>This is for the benefit of MacOS users who don't want to install Python.

`--file` (filename)

This option is explained in section 12.6.

`-f`

Abbreviation for `--file`

`--find` (filename)

This option is explained in section 12.7.

`--exclude`

This option is explained in section 12.2.

`-x`

Abbreviation for `--exclude`

`--includeall` (on or off)

The option `--includeall off` is equivalent to `--exclude`. The default behavior is `--includeall on`. If `--exclude` or `--includeall off` is specified in your preferences file (section 14) and you want to restore the default behavior, you can use `--includeall on`.

`--askfirst` (on or off)

By default, `ns4decode` warns before overwriting existing files. It also warns before decoding files found recursively under a folder. The option `--askfirst off` silences those warnings.

`--warnversion` (on or off)

By default, `ns4decode` warns when trying to decode a file that is not among the NS4 file versions on which it was tested, because then the results might not always be accurate. The option `--warnversion off` silences those warnings.

(continued on next page...)

`--verbosity` (0, 1, 2, or 3)

Used to specify how much or how little will be written to the screen while `ns4decode` is running. The default verbosity level is 1. The allowed verbosity levels are:

- 0 = silent except to announce where outputs were written
- 1 = also announce which files are being decoded
- 2 = also announce every time a file is opened for reading or writing
- 3 = also include extra announcements for troubleshooting (if any)

`-v`

Abbreviation for `--verbosity`

Some additional options for developers are listed in section 18.

## 13.2 Options when not decoding anything

`--help`

Shows the list of available options (a more concise version of the list you're reading now) and then gives the URL to the `ns4decode` website, where you can download the latest edition of the User Manual that you're reading now.

`-h`

Abbreviation for `--help`

`--settings`

Shows the current settings for each option that affects `ns4decode`'s behavior when decoding NS4 files (the options listed in section 13.1).

`--listparams`

This option is explained in section 12.5.

`-l`

Abbreviation for `--listparams`

`--listparamsnocodes`

Like `--listparams` but without the codes

`--version`

Shows the version of `ns4decode`

## 14 Customizing ns4decode

### 14.1 The user preferences file

The folder `ns4decode_pkg` that you got when you downloaded `ns4decode` (section 2) includes a file called `preferences.txt` (figure 14.1). This is your preferences file. You can customize `ns4decode`'s behavior by using a plain text editor to edit this file. Any of the options listed in section 13.1 may be included in your preferences file, and `ns4decode` will use those settings each time you run it unless you override them as part of the command.

In your preferences file, each option must be on a line by itself, with nothing else on that line. Lines that start with the hashtag character (`#`) are ignored, so you can use this to write notes in your preferences file. If an option is specified more than once, the last occurrence overrides the others. (This allows you to override any of the settings in your preferences file by including them in the command line.) If an option is not specified in your preferences file or as part of the command, then `ns4decode` will use its built-in (default) setting.

Running the command<sup>27</sup>

```
ns4decode --settings
```

shows you the current settings of each of the options in section 13.1. These are the settings that `ns4decode` will use if you don't override them as part of the command. Any settings specified in your preferences file will be shown here.<sup>28</sup>

---

<sup>27</sup>For the IDLE method, just type `--settings` in the input dialog.

<sup>28</sup>For options that are not in your preferences file, the default setting will be shown.



## 14.2 Example: saving a `--writeto` preference

Suppose that you always want the CSV output files to be written to a folder called `allCsvFiles`. You could type the option `--writeto allCsvFiles` every time you run `ns4decode`, but that would be tedious. Instead of doing that, you can include the line

```
--writeto allCsvFiles
```

on a line by itself in your preferences file. Then `ns4decode` will use that setting every time, except when you override it.

If you want to override that setting without changing your preferences file, then you can specify `--writeto` as part of the command, say

```
ns4decode --writeto differentFolder (list of NS4 files)
```

then the CSV output files will be written to `differentFolder` instead, like section 12.1 explained. If you want `ns4decode` write the CSV output files to the same folder with the NS4 input file(s), like it would if you had never specified `--writeto` at all, then you can use `--writeto ""`, like this:

```
ns4decode --writeto "" (list of NS4 files)
```

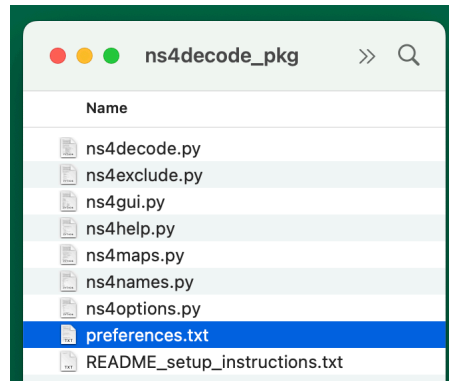


Figure 14.1: The user preferences file (highlighted) is included in the package `ns4decode_pkg` that you downloaded from <https://ns4decode.netlify.app/>. You can edit it with a plain text editor (open by double-clicking or by right-click > Open With) to customize the settings. The command `ns4decode --settings` confirms the current settings. If an option on the command line conflicts with any of those settings, the option on the command line takes precedence.

## 15 Understanding the outputs

Most of the items in the output files should be easy to understand for anybody familiar with the NS4's parameters, but some of them are less obvious.

### 15.1 GUI output mode with Preset files

When decoding a Preset file (`ns4o`, `ns4n`, `ns4y`), the GUI only shows the pages appropriate for that file type, and some of the values may be empty because they do not exist in that file type.

### 15.2 Possible causes of “not decoded”

Sometimes a value may say “not decoded.” This can happen for two reasons:

- If a piano model name or synth-library sample name is not in the list of samples that `ns4decode` currently recognizes, then “not decoded” will be shown in place of the name. To fix this, follow the instructions in section 16.
- The result “not decoded” may also occur when processing older `ns4p` files. Version 3.13 of the `ns4p` format should be reliably decoded by `ns4decode`. Section 17 explains how to make the Nord Stage 4 update the version of a Program file.

### 15.3 Organ drawbars

Some organ drawbars are inactive for some organ models, but `ns4decode` still shows the drawbar value stored in the file even if it is not used or displayed by the instrument.

### 15.4 How samples are indexed

Each category in the synth library can hold up to 100 samples, and each piano-model category in the piano library can hold up to 20 samples. Programs reference samples by the memory-slot where the sample is stored. A single index runs continuously through all memory slots in all sample-categories. This index is stored in each `ns4p` file, and `ns4decode` always reports it. In contrast, the NS4 display only shows populated categories. It indexes the populated samples in each category consecutively, as though the unpopulated slots did not exist. The library information needed to

deduce this index is apparently not stored in an NS4 file, so `ns4decode` cannot report it.

## 15.5 Program transpose

If program transpose is `on` and the transpose amount is zero, the NS4 automatically changes the amount to `+1` when program transpose is turned `off`.<sup>29</sup> This is why you never see a zero transpose amount in `ns4decode`'s output files when program transpose is `off`.

## 15.6 Piano timbre

The interpretation of the piano timbre value normally depends only on the piano type, but it can also depend on the piano model. (Example: if you choose a timbre setting for one model in the CLAV category and then switch to a different model in the CLAV category, this may automatically change the displayed timbre setting.) Since `ns4decode` was developed using a limited list of piano models, the way it interprets the timbre value might not be reliable for piano models that are added later. If you notice any errors in the interpretation of the piano timbre value, you are welcome to report the error.

## 15.7 Organ vib/chorus type

In the NS4, some vib/chorus types might not be compatible with some organ models. To prevent incompatible combinations, the NS4 file does not have separate vib/chorus types for the two organ layers. Instead, it has a single vib/chorus type parameter that is not tied to any layer. That parameter specifies a map from organ models to vib/chorus types. The GUI output mode and the CSV output mode handle this in different ways:

- The GUI displays the vib/chorus type like the NS4 front panel does.
- The CSV output mode organizes the parameters like they are organized in the NS4 file, so the vib/chorus type is presented in the `_master.csv` file as a map from organ models to vib/chorus types.

---

<sup>29</sup>The NS4 designers presumably did this so that turning program transpose back on will immediately have an effect.

## 15.8 Exceptions to the one-row-per-parameter rule

In the CSV output mode, the column-widths for a few parameters are much wider than most of the others. To help alleviate this nuisance, `ns4decode` breaks some of the wider-column parameters into two rows, even though the data comes from a single parameter in the `ns4p` file. This is an exception to the one-row-per-parameter rule (section 11.3).

## 15.9 KB zones

The value of each “KB zones” parameter is reported as a string of four characters, where 1 and 0 mean that zone is or isn’t active, respectively.

## 15.10 Arpeggiator accent, gate, and pan

The arpeggiator accent, gate, and pan values are presented as strings with one character for each of the 16 positions. Groups of four positions are separated by spaces to improve readability. For arpeggiator accent, the characters are:

- > (accent at this position)
- .

For arpeggiator gate, the characters are:

- Q (full Quarter note at this position; blends into next position)
- q (short quarter note at this position; does not blend into next position)
- E (Eighth note at this position)
- .

For arpeggiator pan, the characters are:

- C (Center)
- L (Left)
- R (Right)
- .

## 16 Adding more sample names

The piano layers in an NS4 file use samples, and the synth layers can also use samples from the sample library. An NS4 file apparently doesn't refer to any samples that it uses by name. It seems to refer to them only by an ID number.

A partial list of ID-to-name conversions is built into `ns4decode`. The output from `ns4decode` will show the names of those samples next to their ID numbers in the output files, but for other samples, it will only show the ID number.

You can add to `ns4decode`'s built-in list of ID-to-name conversions. The folder `ns4decode_pkg` (section 2) has a file called `ns4names.py`, which you can open with a plain text editor (section 11.3). Near the bottom of that file are functions called `getPianoModels` and `getSampleNames`. The function `getPianoModels` has the ID-to-name map for piano-section samples, and the function `getSampleNames` has the ID-to-name map for the synth-section samples. You may add items to those lists by following the instructions in the comments above those functions.

## 17 Converting older NS4 files to the new version

`ns4decode` was developed based on version 3.13 of the Nord Stage 4 Program file format, so those `ns4p` files should be reliably decoded by `ns4decode`, but its outputs for a few parameters might not be accurate when used on older NS4 files. You can make the NS4 update the version of an NS4 file by temporarily changing the value of a parameter and immediately changing it back (example: turning a layer off and then back on) and then saving the file (by pressing the Save button twice). The Program should still look and sound the same in the NS4, but the Nord Sound Manager should show the updated version number for that Program. If it shows 3.13, then `ns4decode` should work correctly on that file.<sup>30</sup>

If the resulting format is newer than the one(s) used to develop `ns4decode`, then `ns4decode` will issue a generic warning that some of its outputs might not be accurate, but it won't know which ones (if any).<sup>31</sup> The `ns4decode` developers are not affiliated with Nord, so we can't predict how Nord might change their file format in future releases of the Nord Stage 4 OS.

---

<sup>30</sup>The NS4 file version is also shown in the corresponding `_master.csv` file.

<sup>31</sup>You can silence this warning by using the option `--warnversion off` (section 13.1).

## 18 Additional information for developers

### 18.1 Regression testing on MacOS

To execute the regression test, first navigate to the `regressionTest` folder within a prepared<sup>32</sup> command window by following these steps:

1. In the command window, type

```
cd
```

followed by a space, but don't press return yet.

2. Drag-and-drop the `regressionTest` folder into the command window, and then press return.

Then run these commands:

```
ns4decode -c 20 regressionTest.ns4p
diff regressionTest_organ.csv previousOutputs/.
diff regressionTest_piano.csv previousOutputs/.
diff regressionTest_synth.csv previousOutputs/.
diff regressionTest_master.csv previousOutputs/.
```

The “/.” at the end tells the `diff` command to compare the specified file to the file with the same name in `previousOutputs`. Any differences compared to those previous outputs will be displayed in the command window.

---

<sup>32</sup>Section 7 explains how to prepare the command window.

## 18.2 Regression testing on Windows

To execute the regression test, first navigate to the `regressionTest` folder within a prepared<sup>33</sup> command window by following these steps:

1. In the command window, type

```
cd
```

followed by a space, but don't press return yet.

2. Drag-and-drop the `regressionTest` folder into the command window, and then press return.

Then run this command:

```
ns4decode -c 20 regressionTest.ns4p
```

Windows doesn't have the `diff` utility that Unix-based systems (MacOS and Linux) have,<sup>34</sup> so you'll need to find another efficient way to compare the new CSV files (the ones written by the preceding command) to the previous CSV files (the ones in the `previousOutputs` folder).

---

<sup>33</sup>Section 7 explains how to prepare the command window.

<sup>34</sup>PowerShell has something called "diff," but it's not the same: it can fail to show differences in plain text files that are clearly different.



## 18.3 Showing raw (uninterpreted) values

To convert an NS4 file to human-readable form, `ns4decode` needs to know two things:

- It needs a **bitmap** that tells it where to find each parameter in the binary NS4 file.
- It needs to know how to **interpret** the raw integer values in the NS4 file into the form displayed on the NS4 front panel.

Section 18.4 will explain how to access and modify `ns4decode`'s built-in bitmaps. To deactivate `ns4decode`'s interpretations and show each parameter's uninterpreted value instead, these options are available:

`--rawdecimal`

Write each parameter value as the raw integer (decimal) value that is actually stored in the NS4 file instead of trying to convert it to the value displayed on the keyboard's front panel.

`--rawhex`

Like `--rawdecimal`, but hexadecimal instead of decimal.

`--rawbinary`

Like `--rawdecimal`, but binary instead of decimal.

## 18.4 Viewing ns4decode's bitmaps

This command

```
ns4decode --bitmaps
```

writes these files to the current working folder:

```
ns4p_bitmap_master.txt
ns4p_bitmap_organ.txt
ns4p_bitmap_piano.txt
ns4p_bitmap_synth.txt
```

These files describe where each parameter lives in the `ns4p` format. If you want these files to be written to another folder (say `auxFolder`), then use this command instead:

```
ns4decode --writeto auxFolder --bitmaps
```

The bitmaps files express the location of a bit using the notation XXX-Y:

XXX is a 3-digit decimal number giving the 1-based index of the byte

Y is a 1-digit decimal number giving the 1-based index of the bit within the byte

In the column headers, “beg” and “end” refer to the first and last bit, respectively, in the sequence of bits representing the given parameter.

You can edit these bitmap files and then tell `ns4decode` to use them. If the bitmap files are saved in `auxFolder` (for example), then you can do that using this command:

```
ns4decode --use auxFolder *.ns4p
```

If an entry in a bitmap file conflicts with what `ns4decode` already knows, then `ns4decode` will use the one in the file. You may delete rows from those files if you want `ns4decode` to use its built-in knowledge of those parameters.

Beware that changing the name of a parameter may interfere with the `--exclude` option, because that option uses the first part of some parameter names to infer how they are grouped into collections.

## 19 Troubleshooting

This section lists a few ideas for things to try if something isn't working. If you suspect that the problem is with `ns4decode` itself, you can use the contact information in section 20 to report the suspected problem. You may also use that contact information to report errors or deficiencies in this User Manual.

### 19.1 Command not found

If you are using the command window method and you get an error that says something like “command not found,” this could mean either of two things:

- It could mean that the command window does not recognize “`ns4decode`.” In that case, you can try revisiting the instructions in section 7. Remember that those steps need to be done once in each new command window.
- It could mean that Python is not found. In that case, you can try revisiting the instructions in sections 6 and 7, or you might need to search for help online.

### 19.2 Can't find ‘`__main__`’ module

If you get this error, revisit the command window preparation steps in section 7 and make sure to drag-and-drop the file `ns4decode.py`, not the package `ns4decode_pkg` that contains that file.

### 19.3 Empty GUI

The version of Python that comes with MacOS does not support the GUI feature. If the GUI is empty, this could mean that a suitable version of Python is not installed, or it could mean that `nsdecode` is trying to use the wrong version of Python. You can try revisiting the instructions at the end of section 6.

### 19.4 Running `ns4decode` in verbose mode

If you include the option `--verbosity 3` as part of the command that you're trying to run, then `ns4decode` might write some additional clues to the screen... or it might not, depending on the problem.

## 20 Reporting problems

You can visit

<https://ns4decode.netlify.app/revisions>

to check for recent bug-fixes, new features, and updates to this User Manual. To see what version of `ns4decode` you have, type

```
ns4decode --version
```

in a prepared command window and press return. In the IDLE method, just type `--version` in the input dialog and press return.

Even though `ns4decode` is free and does not come with any customer support,<sup>35</sup> the developers would like fix things that don't work as intended. You may report problems with `ns4decode` or with this User Manual by sending a polite email to `ns4decode@gmail.com`. The email will be forwarded (if needed) to the appropriate developer.<sup>36</sup> This email is not always frequently monitored, so please be patient.

Thank you!

– Randy

---

<sup>35</sup>The developers of `ns4decode` are amateurs.

<sup>36</sup><https://ns4decode.netlify.app/history>