



**Operating Instruction Manual**  
**Tag List Editor**  
**Viewing and Editing Tags in NXF/NXO/BSL Files**  
V1.1.x.x

**Hilscher Gesellschaft für Systemautomation mbH**  
**[www.hilscher.com](http://www.hilscher.com)**

DOC110306OI02EN | Revision 2 | English | 2011-04 | Released | Public

# Table of Contents

1	INTRODUCTION .....	4
1.1	About this Manual .....	4
1.1.1	List of Revisions .....	4
1.1.2	Conventions in this Manual .....	5
1.1.3	Terms, Abbreviations and Definitions.....	5
1.2	About Tag List Editor .....	6
1.2.1	System Requirements .....	6
1.2.2	Limitations .....	6
1.3	Legal Notes.....	7
1.3.1	Copyright .....	7
1.3.2	Important Notes .....	7
1.3.3	Exclusion of Liability .....	8
1.3.4	Warranty .....	8
1.3.5	Export Regulations .....	9
1.3.6	Registered Trademarks.....	9
2	INSTALLATION.....	10
2.1.1	Configuration Files.....	10
3	TAG LIST EDITOR.....	11
3.1	Loading and Saving .....	12
3.1.1	Loading an NXO Header Binary.....	13
3.1.2	Loading an NXO ELF File .....	13
3.1.3	Loading a Tag List.....	13
3.1.4	Loading an NXO/NXF File.....	14
3.1.5	Saving an NXO/NXF File.....	14
3.2	Editing Tags.....	15
3.2.1	Disabling Tags.....	15
3.3	Editing the Device Header .....	17
4	TAGTOOL .....	18
4.1	Usage .....	18
4.1.1	help, -h, /?: Print Usage Information.....	19
4.1.2	help_tags: Print known Tags .....	19
4.1.3	help_const: Print known Constants .....	19
4.1.4	-version: Print version Information.....	19
4.1.5	settags: Replace the Tag List.....	19
4.1.6	list: Print the Tag List and Device Header.....	19
4.1.7	diff: Compare Tag Lists and Device Headers of two Files .....	20
4.1.8	edit: Patch Tag List and Device Header.....	20
4.2	Patch files .....	21
4.2.1	File Format .....	21
4.2.2	How to generate a Patch File .....	22
4.2.3	Error Messages pertaining to the Patch File .....	23
5	MAKENXO .....	24
5.1	Usage .....	24

Introduction	3/28
6	LISTS .....25
6.1	List of Figures .....25
6.2	List of Tables .....25
7	APPENDIX .....26
7.1	Structure of an NXO/NXF File .....26
8	CONTACTS.....28

# 1 Introduction

## 1.1 About this Manual

This manual describes the

- Tag List Editor
- the command line tool makenxo
- the command line tool tagtool

The command line tools are included with the Tag List Editor.

### 1.1.1 List of Revisions

Index	Date	Version	Component	Chapter	Revision
1	2011-03-25	1.1.9855.0	Tag List Editor	all	created
2	2011-04-19	1.1.10032.0	Tag List Editor	1.2.2	removed remark in "Limitations" regarding the bug in the treatment of gap data

## 1.1.2 Conventions in this Manual

Operation instructions, a result of an operation step or notes are marked as follows:

### **Operation Instructions:**

➤ <instruction>

Or

1. <instruction>

2. <instruction>

### **Results:**

↻ <result>

### **Notes:**



**Important:** <important note>



**Note:** <note>



<note, were to find further information>

## 1.1.3 Terms, Abbreviations and Definitions

Term	Definition
NXF	netX Firmware file
NXO	netX option module
BSL	Bootstrap Loader, also called 2 <sup>nd</sup> Stage Loader. The newer versions from 1.3 and up are in NXF format.

*Table 1: Terms, Abbreviations and Definitions*

## 1.2 About Tag List Editor

The Tag List Editor provides a graphical user interface which allows the manipulation of tag lists and firmware files containing a tag list, which includes NXF and NXO files as well as the 2<sup>nd</sup> Stage Loader (BSL), which is in NXF format starting from version 1.3. The Tag List Editor allows you to

- edit the tag list
- edit the device header
- construct an NXO file from its basic components.

Also included are two command line tools:

- makenxo combines a header binary, ELF file and tag list to an NXO file.
- tagtool allows you to print, compare, replace and manipulate tag lists and to print, replace and manipulate the device header of NXF, NXO and BSL files.

The tag list is a list of configuration structures which has been compiled into an executable file for the netX chip in NXO or NXF format. It contains information which must be available before any configuration files are accessed (e.g. the location of the file system) and other configuration data.



**Note:** The tag list editor and the command line tools work on files with common header V3.0 and device header V1.0. Files containing a common header with a version below 3.0 are rejected. Files with a common header with a version above 3.0 are accepted, but a warning is shown.

### 1.2.1 System Requirements

- Windows® XP (32 bit) or Windows 7 (32 bit)
- 30 MB free hard disk space

### 1.2.2 Limitations

- The Tag List Editor handles only firmware files which contain the Common Header V3.
- It can only edit existing tag lists. It is not possible to insert or remove tags. This is a deliberate limitation.
- The tag list may contain tags which are unknown to the editor. These tags are not displayed, but they are kept and written back when the file is saved.
- There is no automatic consistency checking. The Tag List Editor does not ensure that the settings you make are reasonably applicable to the used hardware and firmware. For instance, when editing task priorities, you can only set values which are valid as task priorities, but the editor does not check if all tasks have distinct priorities.

## 1.3 Legal Notes

### 1.3.1 Copyright

© Hilscher, 2011, Hilscher Gesellschaft für Systemautomation mbH

All rights reserved.

The images, photographs and texts in the accompanying material (user manual, accompanying texts, documentation, etc.) are protected by German and international copyright law as well as international trade and protection provisions. You are not authorized to duplicate these in whole or in part using technical or mechanical methods (printing, photocopying or other methods), to manipulate or transfer using electronic systems without prior written consent. You are not permitted to make changes to copyright notices, markings, trademarks or ownership declarations. The included diagrams do not take the patent situation into account. The company names and product descriptions included in this document may be trademarks or brands of the respective owners and may be trademarked or patented. Any form of further use requires the explicit consent of the respective rights owner.

### 1.3.2 Important Notes

The user manual, accompanying texts and the documentation were created for the use of the products by qualified experts, however, errors cannot be ruled out. For this reason, no guarantee can be made and neither juristic responsibility for erroneous information nor any liability can be assumed. Descriptions, accompanying texts and documentation included in the user manual do not present a guarantee nor any information about proper use as stipulated in the contract or a warranted feature. It cannot be ruled out that the user manual, the accompanying texts and the documentation do not correspond exactly to the described features, standards or other data of the delivered product. No warranty or guarantee regarding the correctness or accuracy of the information is assumed.

We reserve the right to change our products and their specification as well as related user manuals, accompanying texts and documentation at all times and without advance notice, without obligation to report the change. Changes will be included in future manuals and do not constitute any obligations. There is no entitlement to revisions of delivered documents. The manual delivered with the product applies.

Hilscher Gesellschaft für Systemautomation mbH is not liable under any circumstances for direct, indirect, incidental or follow-on damage or loss of earnings resulting from the use of the information contained in this publication.

### 1.3.3 Exclusion of Liability

The software was produced and tested with utmost care by Hilscher Gesellschaft für Systemautomation mbH and is made available as is. No warranty can be assumed for the performance and flawlessness of the software for all usage conditions and cases and for the results produced when utilized by the user. Liability for any damages that may result from the use of the hardware or software or related documents, is limited to cases of intent or grossly negligent violation of significant contractual obligations. Indemnity claims for the violation of significant contractual obligations are limited to damages that are foreseeable and typical for this type of contract.

It is strictly prohibited to use the software in the following areas:

- for military purposes or in weapon systems;
- for the design, construction, maintenance or operation of nuclear facilities;
- in air traffic control systems, air traffic or air traffic communication systems;
- in life support systems;
- in systems in which failures in the software could lead to personal injury or injuries leading to death.

We inform you that the software was not developed for use in dangerous environments requiring fail-proof control mechanisms. Use of the software in such an environment occurs at your own risk. No liability is assumed for damages or losses due to unauthorized use.

### 1.3.4 Warranty

Although the hardware and software was developed with utmost care and tested intensively, Hilscher Gesellschaft für Systemautomation mbH does not guarantee its suitability for any purpose not confirmed in writing. It cannot be guaranteed that the hardware and software will meet your requirements, that the use of the software operates without interruption and that the software is free of errors. No guarantee is made regarding infringements, violations of patents, rights of ownership or the freedom from interference by third parties. No additional guarantees or assurances are made regarding marketability, freedom of defect of title, integration or usability for certain purposes unless they are required in accordance with the law and cannot be limited. Warranty claims are limited to the right to claim rectification.

### **1.3.5 Export Regulations**

The delivered product (including the technical data) is subject to export or import laws as well as the associated regulations of different countries, in particular those of Germany and the USA. The software may not be exported to countries where this is prohibited by the United States Export Administration Act and its additional provisions. You are obligated to comply with the regulations at your personal responsibility. We wish to inform you that you may require permission from state authorities to export, re-export or import the product.

### **1.3.6 Registered Trademarks**

Windows® 2000/Windows® XP are registered trademarks of Microsoft Corporation.

All other mentioned trademarks are property of their respective legal owners.

## 2 Installation

Under Windows XP, you must be logged in as an administrator in order to install the Tag List Editor. It will, however, run under a limited user account. Under Windows 7, it can be installed and run under a limited user account.

The installation is straightforward. The installer is a single executable file. Run it and follow the instructions:

1. On the welcome page, click on **Next**.
2. Read the license agreement. If you accept it, select **I accept the agreement** and click on **Next**. Otherwise, click on **Cancel** to abort the installation.
3. Select a directory for the installation, or accept the predefined one. Then, click on **Next**.
4. Accept the components to install. All components are necessary to use the tag list editor.  
Click on **Next** to continue.
5. Select whether you want to create a Start menu item, and what it should be called.  
Click on **Next** to continue.
6. On this page, you define whether you want to create a Start menu icon, a desktop icon and a quick launch icon.  
Also, select whether the environment variable "PATH\_NXOEDITOR" should be set. This variable is necessary if you want to use the command line tools, makenxo and tagtool.  
Click on **Next** to continue.
7. You are now ready to install. Review the installation settings on this page and click on **Install** or click on **Back** to go back and change the installation settings.
8. The change log is shown. Click on **Next**
9. Click on **Finish** to exit the installer.

### 2.1.1 Configuration Files

There is a default configuration file which is installed with the Tag List Editor. When you close the Tag List Editor, any changes you have made are written to a personal configuration file which is created in your user directory. The settings stored in the personal configuration file override those in the default configuration.

Default configuration: <install directory>\application\Modulator.cfg

e.g. C:\Program Files\Hilscher GmbH\Tag List Editor\application\Modulator.cfg

User configuration: <User local application data directory>\Modulator.cfg

e.g. C:\Documents and Settings\user\Local Settings\Application Data\Modulator.cfg

The user configuration file is not deleted when you uninstall the Tag List Editor. If you encounter any problems after an update, delete this file manually. The file is located in a hidden directory; in order to make it visible, open the Explorer's folder options and select "Show hidden files and folders".

### 3 Tag List Editor

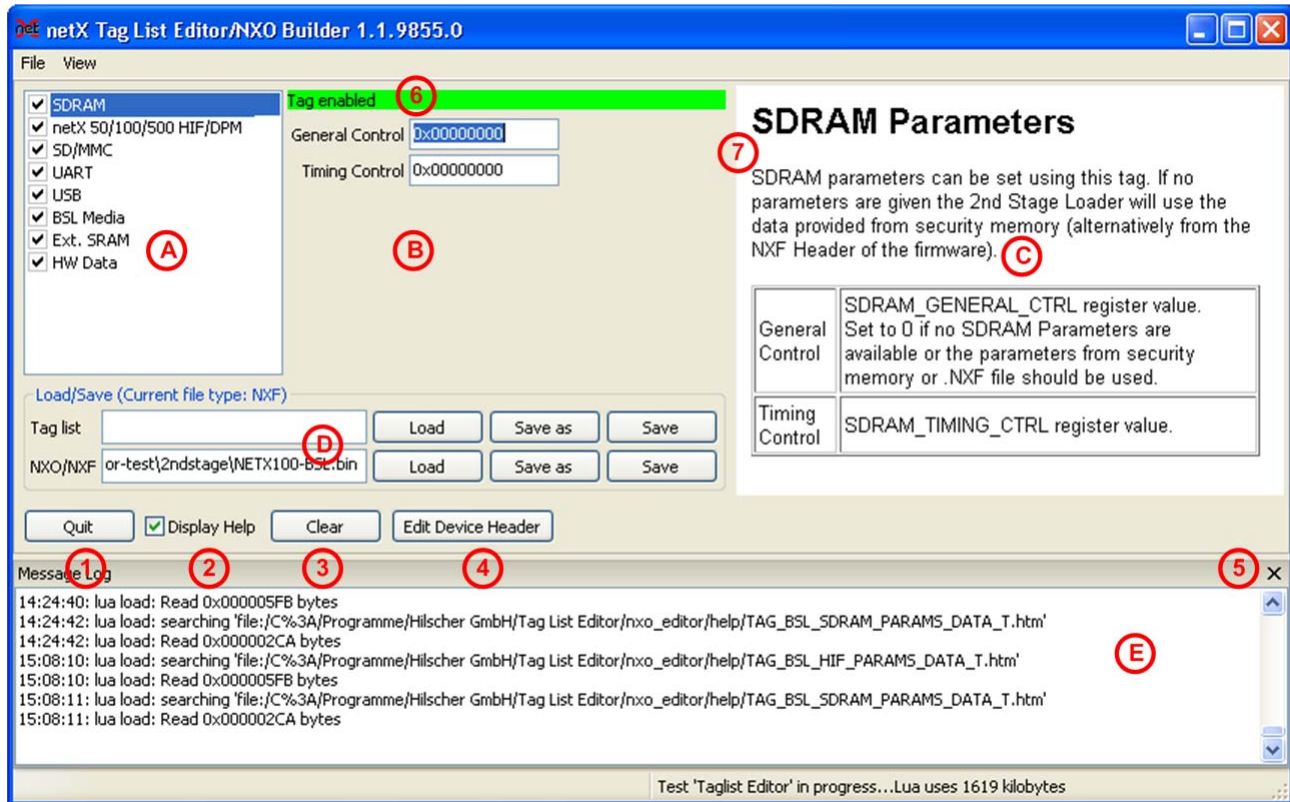


Figure 1: The Tag List Editor Window

Area	Description
(A)	The list of tags which are currently loaded and which are known by the editor. The check box next to each tag indicates if the tag is enabled.
(B)	The editing area for the currently selected tag.
(C)	The help page for the currently selected tag.
(D)	Load/save options.
(E)	The log area.

Table 2: Main Areas of the Editor Window

Control Element	Description
(1)	Exits the Tag List Editor.
(2)	Turns the help area on or off.
(3)	Resets the Tag List Editor to the initial state.
(4)	Opens a window which allows editing of the device header if the currently loaded file contains one.
(5)	Closes the log area. In order to display it again, select it in the View menu.
(6)	Indicates if the currently selected tag is enabled.
(7)	The edge between the editing area and the help area. Drag it to adjust the size of the help area.

Table 3: Control Elements of the Editor Window

### 3.1 Loading and Saving

The Load/Save area allows you to save and load complete NXO/NXF files, the tag list and the headers and/or ELF sections of an NXO file.

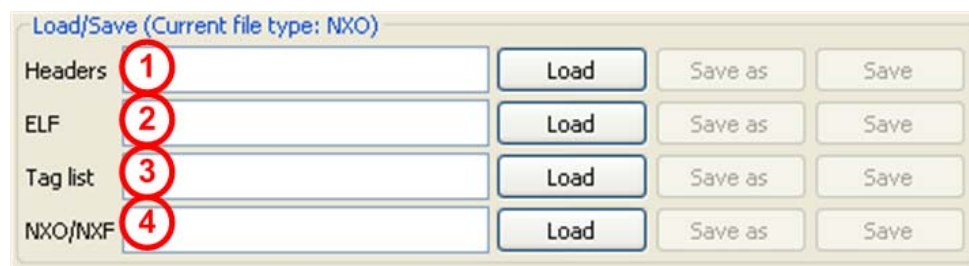


Figure 2: Load/Save Area configured for NXO Files

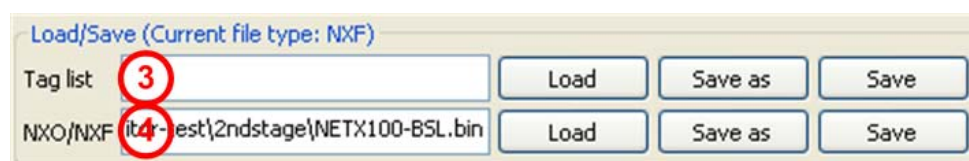


Figure 3: Load/Save Area configured for NXF Files

Control Element	Description
①	Load or save the header section of an NXO file.
②	Load or save the ELF section of an NXO file.
③	Load or save the tag list of an NXF or NXO file.
④	Load or save an NXF or NXO file.

Table 4: Control Elements for Loading and Saving

Options ① and ② are shown initially, after the Clear button has been used to return to the initial state or after an NXO file has been loaded. They are hidden when an NXF file has been loaded.

Loading files is always possible, saving only when the respective data is in memory. That is, for instance, after loading an NXF file containing a tag list you can save the NXF file or the tag list itself.

This allows for a number of use cases:

- Editing a tag list which is embedded in a file: Load an NXO/NXF file which contains a tag list, edit tags and save the file.
- Editing a tag list which is in a separate file: Load a tag list from a separate file, edit it and save the modified tag list.
- Extracting a tag list which is embedded in a file: Load an NXO/NXF file containing a tag list and save the tag list to a separate file.
- Inserting/replacing a tag list: Load an NXO/NXF file, then load a tag list from a separate file and save the modified NXO/NXF file.
- Constructing an NXO file: Load the Headers, ELF file and tag list from separate files and save them as an NXO file.

The loading functions perform some checks on the files. In order to understand the meaning and extent of the checks, some knowledge of the

file format is required. See Appendix 7.1, Structure of an NXO/NXF File on page 26 for a short description.



**Note:** Gap data following the sections of the file (headers, data/ELF, tag list) is kept as long as these sections are not replaced. If a section is replaced by loading a new header or ELF file, the gap data is discarded and replaced with 0 to 3 padding bytes.

Exception: If the tag list is located between the header and data sections as it is the case in the 2<sup>nd</sup> stage loader, the gap data is retained when the tag list is replaced, and shortened or extended to ensure that the data section following it stays at the same absolute location in the file.

### 3.1.1 Loading an NXO Header Binary

This function loads an NXO header binary. The file must contain the common header V3 and any following headers but not the initial default header.

The following checks are performed after loading the file. If any of them fails, the file is rejected:

- The common header must have major version 3
- The file size, ulHeaderLength and bNumModuleInfo must be consistent
- If a tag list is in memory and ulTagListSizeMax is set, the tag list must not be longer than ulTagListSizeMax.

### 3.1.2 Loading an NXO ELF File

This function loads the ELF section of an NXO file. If the file does not have the ELF signature, it is rejected.

### 3.1.3 Loading a Tag List

In the following cases, the file is rejected:

- The tag list ends within a tag
- The tag list has a 4-byte or 8-byte end marker followed by additional data
- The size information of a known tag is inconsistent with the structure definition in the editor
- A common header with the field ulTagListSizeMax greater than 0 is in memory and the tag list is longer than the maximum size
- A file with the tag list located before the data section is in memory and the tag list does not fit between the headers and the data section.

In the following case, a warning is shown:

- The tag list ends does not have an end marker or only a 4-byte end marker. If the maximum tag list size allows it, the editor offers to replace the end-marker with an 8-byte end marker

### 3.1.4 Loading an NXO/NXF File

In the following cases, the file is rejected:

- the file type as indicated by the magic cookie at the start of the file is not known
- the common header version is below 3
- any offset or size entry in the common header exceeds the file size, or any of the sections overlap

In the following cases, the file is accepted but a warning message is shown:

- any of the checksums in the boot header/common header are incorrect
- the common header version is higher than 3.0
- the file contains data in-between the headers, data and tag list sections

If the checks on the file succeed, the checks for tag lists described in the previous section are performed.

### 3.1.5 Saving an NXO/NXF File

The tag list, the common header fields pertaining to the tag list and the checksums are updated.

## 3.2 Editing Tags

Figure 4: Editing Controls for Values in Tags

Each field in a tag is presented as a numeric field <sup>③</sup>, a text field <sup>①</sup>, a selection box <sup>②</sup> or a checkbox <sup>④</sup>.

Entries with a grey background <sup>①</sup> are read-only. For instance, task priority tags contain pre-defined name entries which are used by the firmware to retrieve the tags.

Numeric values <sup>③</sup> are displayed either in decimal or hex notation, but may be entered in either notation.

The editor will prevent you from entering numbers which are larger than the bit width of the field. For some fields, a more limited value range is defined. You can always enter the value 0, because it is part of the prefix of hexadecimal numbers.



### **Note:** Priorities and priority ranges

If you change task priority and task token settings, always set both to the same value.

Priority base and priority range values must fulfill the following condition:

$\text{base} + \text{range} - 1 \leq \text{max. value}$

For instance, if the task priority range is 2, the base priority must be set to a value less than 55, since 55 is the highest value.

### 3.2.1 Disabling Tags

The checkbox to the left of each tag determines whether the tag is enabled, that is, visible to the firmware:

Figure 5: The selected Tag is enabled.

Figure 6: The selected Tag is disabled.



---

**Important:**

Disabling a tag makes it invisible to the firmware. It does NOT necessarily disable the feature which the tag configures. The firmware may have a built-in default configuration in which the feature is enabled. In this case, the feature will be active and using the default configuration.

If a tag contains an on/off option, the secure way to disable the feature is to enable the tag and set the option to "off".

**Example:**

In Figure 5, the EDD configuration tag is enabled and the DLR feature is disabled.

In Figure 6, the EDD configuration tag is disabled. Whether the DLR feature is active or not depends on the default behaviour of the firmware.

---

### 3.3 Editing the Device Header

When a file containing a device header is in memory, the **Edit Device Header** button is enabled. It opens a window which allows you to view and edit the device header.

Figure 7: The Device Header Editor Window

Manufacturer code, device class, hardware options, device product code and device serial number can be edited. All other fields are read-only.

Control Element	Description
①	Writes the device header with the currently displayed values to a binary file.
②	Loads the device header from a binary file. The file is only accepted if the size and version entry match those of the device header V1. If the file is accepted, the contents are displayed in the editing window, but not replaced in the loaded NXF/NXO file until you exit the window using <b>Confirm Changes</b> .
③	Stores any changes you have made and closes the window
④	Discards any changes you have made and closes the window. This is equivalent to clicking on the close icon.

Table 5: Control Elements of the Device Header Editor Window.

## 4 Tagtool

Tagtool is a command line tool to perform operations on the tag list and device header of an NXO/NXF file. It can print, compare and modify the tag list and device header and replace the tag list.

### Path

Tagtool is located in

```
<Install directory>\nxo_editor\tagtool.bat
```

The install directory is stored in the variable PATH\_NXOEDITOR, unless you disable this option in the installer. In order to call the tool, add %PATH\_NXOEDITOR%\nxo\_editor to the search path.

### 4.1 Usage

The general usage is

```
tagtool <command> [flags] <arguments>
```

with the following commands, flags and argument types:

<command>	<arguments>	Function
settags	input_file taglist_file output_file input_file = input NXO/NXF file taglist_file = binary file containing a tag list output_file = output NXO/NXF file	Replace the tag list
list	input_file input_file = input NXO/NXF file	Print tag list and device header
diff	input_file1 input_file2 input_file1/2 = input NXO/NXF files	Compare tag lists and device headers
edit	input_file patch_file output_file input_file = input NXO/NXF file patch_file = text file describing modifications output_file = output NXO/NXF file	Apply patch to tag list and device header
help -h		Print usage information
help_tags		Print the known tags
help_const		Print the known value constants
-version		Print version information

Table 6: Tagtool Commands and Arguments

Flag	Function
-v	enables verbose output
-debug	enables debug output

Table 7: Tagtool Flags

Since the tool is wrapped in a batch file, you must use following syntax to call it from another batch file:

```
call tagtool <command> [flags] <arguments>
```

### 4.1.1 **help, -h, /?: Print Usage Information**

```
tagtool help
tagtool -h
tagtool /?
tagtool
```

This command prints the help page.

### 4.1.2 **help\_tags: Print known Tags**

```
tagtool help_tags
```

This command prints a list of the known tags.

### 4.1.3 **help\_const: Print known Constants**

```
tagtool help_const
```

This command prints a list of the known value constants. These constants may be used in a patch file instead of values.

### 4.1.4 **-version: Print version Information**

```
tagtool -version
```

This command prints version information.

### 4.1.5 **settags: Replace the Tag List**

```
tagtool settags input_file taglist_file output_file
```

This command reads `input_file`, replaces the tag list with the contents of `taglist_file`, updates the offset/size fields of the common header and the checksums and writes the result to `output_file`.

#### **Example:**

Assume you have saved a tag list as `tags_xc3.bin` using the editor. The following command will put this tag list into an NXO file:

```
>tagtool settags nx100mpi.nxo tags_xc3.bin nx100mpi_xc3.nxo
```

### 4.1.6 **list: Print the Tag List and Device Header**

```
tagtool list input_file
```

This command reads `input_file` and prints the tag list and device header. For unknown tags, only the tag ID is listed.

#### **Example:**

```
>tagtool list nx100mpi.nxo
Tag 1: RCX_TAG_LED (0x00001040)
ENABLED
.szIdentifier      = FB_STA_RED
.ulUsesResourceType = 0x00000002
.ulPinNumber       = 0x00000005
.ulPolarity        = 0x00000000

Tag 2: RCX_TAG_LED (0x00001040)
ENABLED
.szIdentifier      = FB_STA_GREEN
.ulUsesResourceType = 0x00000002
.ulPinNumber       = 0x00000004
.ulPolarity        = 0x00000000

Tag 3: RCX_TAG_XC (0x00001050)
ENABLED
.szIdentifier      = PB_XC
.ulXcId            = 0x00000002
```

```
...
# Tag 7: unknown (0x00000806)

DEVICE_HEADER_V1_T
.ulStructVersion      = 0x00010000
.usManufacturer       = 0x0001
.usDeviceClass        = 0x0007
.bHwCompatibility     = 0x00
.bChipType            = 0x00
.usReserved           = 0x0000
.usHwOptions_1        = 0x0000
.usHwOptions_2        = 0x0000
.usHwOptions_3        = 0x0050
.usHwOptions_4        = 0x0000
.ulLicenseFlags1      = 0x00000000
.ulLicenseFlags2      = 0x00000000
.usNetXLicenseID      = 0x0000
.usNetXLicenseFlags   = 0x0000
.ulFwVersion_Major    = 0x0002
.ulFwVersion_Minor    = 0x0004
.ulFwVersion_Build    = 0x0001
.ulFwVersion_Revision = 0x0000
.ulFwNumber           = 0x00000000
.ulDeviceNumber       = 0x00000000
.ulSerialNumber       = 0x00000000
.ulReserved1          = 0x00000000
.ulReserved2          = 0x00000000
.ulReserved3          = 0x00000000
```

#### 4.1.7 diff: Compare Tag Lists and Device Headers of two Files

```
tagtool diff input_file1 input_file2
```

This command reads `input_file1` and `input_file2`, compares the tag lists and device headers and prints output in the patch file format which can be read by the `edit` command.



**Note:** The tag lists of the two files must contain exactly the same tags in the same order.

For any tags whose values differ, the tag from the second file is printed as an edit record which can be used as input for the "edit" function.

##### Example:

The file `nx100mpi.nxo` was edited to use xC unit 3 and saved as `nx100mpi_xc3.nxo`.

```
>tagtool diff nx100mpi.nxo nx100mpi_xc3.nxo
Tag 3: RCX_TAG_XC (0x00001050)
  ENABLED
  .szIdentifier = PB_XC
SET .ulXcId     = 0x00000003
```

#### 4.1.8 edit: Patch Tag List and Device Header

```
tagtool edit input_file patch_file output_file
```

This command reads `input_file`, applies the modifications listed in `patch_file` to the tag list and device header, updates the checksums and writes the result to `output_file`.

### Example:

Assume you have edited an NXF/NXO file in the editor and saved the result to a different file.

Using the `diff` and `edit` commands, it is possible to extract the changes and apply them to another file, for example, a new version of the firmware.

```
>tagtool diff nx100mpi.nxo nx100mpi_xc3.nxo >diff_xc3.txt
>tagtool edit nx100mpi_new.nxo diff_xc3.txt nx100mpi_new_xc3.nxo
```

## 4.2 Patch files

### 4.2.1 File Format

The patch file is a text file generated by the "diff" command and used as input by the "edit" command. It contains one or more edit records (patch records) which describe a patch to a data structure, either a tag or the device header. Each edit record is matched against all tags and the device header. If it matches exactly one data structure, the patch is applied. Otherwise, an error is raised.

The first line of an edit record specifies the type of data structure which this record should be matched to, either a particular tag type or the device header:

Structure type constraints	Function
TAG_NAME	Matches a tag of type TAG_NAME
Tag <ignored until colon>: TAG_NAME <The rest of the line is ignored>	Matches a tag of type TAG_NAME
DEVICE_HEADER_V1_T	Matches the device header

Table 8: Data Type Selection in an Edit Record

The following lines specify further constraints on the selected data structure:

Member constraints	Function
ENABLED	Matches a tag which is enabled
DISABLED	Matches a tag which is disabled
member_name = value	Matches a structure whose member has the given value

Table 9: Value Constraints in an Edit Record

Any lines which start with the SET keyword specify the actual patch. They are applied if a matching data structure has been found:

Member modifiers	Function
SET ENABLED	Enables the selected tag
SET DISABLED	Disables the selected tag
SET member_name = value	Assigns a value to a member

Table 10: Value Assignment in an Edit Record

String values may be written with or without double quotes.

Numeric values may be written in decimal or hexadecimal notation or using the symbolic names shown by the `help_const` command.

Empty lines are ignored, and everything following a # until the end of the line is ignored.

### Example:

```
Tag 15: RCX_MOD_TAG_IT_XC (0x00001050)
.szIdentifier = "RTE_XC1"
DISABLED
SET .ulXcId = 2
SET ENABLED
```

This selects a tag with type RCX\_MOD\_TAG\_IT\_XC which has the identifier string "RTE\_XC1" and is currently disabled. If the tag list contains exactly one tag which matches this description, it is enabled and its ulXcId field is set to 2.



**Note:** Some tags contain nested structures. To access these values, the full path from the root of the structure must be specified:

```
Tag 2: TAG_BSL_HIF_PARAMS (0x40000001)
.ulBusType = 0x00000000
.tDpmIsaAuto.ulIfConf0 = 0x00000000
.tDpmIsaAuto.ulIfConf1 = 0x00000000
.tDpmIsaAuto.ulIoRegMode0 = 0x00000001
.tDpmIsaAuto.ulIoRegMode1 = 0x00000002
.tPci.bEnablePin = 0x01
.tPci.bPinType = 0x01
.tPci.bInvert = 0x80
.tPci.usPinNumber = 0x000f
```



**Note:** In some tags, two values are combined in one byte, word or dword. In the textual representation used by "list", "diff" and "edit", these values appear as separate values.

For instance, in the following tag:

```
Tag 5: TAG_BSL_USB_PARAMS (0x40000004)
.bEnable = 0x01
.bPullupPinType = 0x01
.bInvert = 0x80
.usPullupPinIdx = 0x0001
```

bPullupPinType and bInvert share one byte in the binary encoding. bPullupPinType occupies bits 0-6 (value range 0x00-0x7f) and bInvert occupies bit 7 (value is either 0x00 or 0x80).

This is currently the case with the following tags:

```
TAG_BSL_HIF_PARAMS_DATA_T
TAG_BSL_SDMMC_PARAMS_DATA_T
TAG_BSL_USB_PARAMS_DATA_T
TAG_BSL_FSU_PARAMS_DATA_T
```

## 4.2.2 How to generate a Patch File

Method 1: If you have an original NXO/NXF and a copy of the file with a modified tag list/device header, use the "diff" command to extract the changes.

Method 2: The output of the "list" function can be parsed by the "edit" function.

Redirect the output to a text file and open this file in an editor. Change the values as desired and add the keyword "SET" at the beginning of every line which contains a change.

Method 3: Write it manually according to the format described above.

### 4.2.3 Error Messages pertaining to the Patch File

Error message	Meaning
parse error	The format of a line could not be recognized.
parse error (no tag specified)	The file must start with a line which selects a tag or the device header
unknown type name <type name>	You have misspelt the tag name, or the program does not know this tag.
type <type name> has no member <member name>	The type is known, but the type does not have a member with the given name.
tried to get member of primitive type <type name>	The path specified in a value condition or assignment is too deeply nested.
Failed to parse value in match/assignment: type=<type name> value=<value>	The value does not have the correct type or cannot be parsed. If you specified a value constant, it may be spelt incorrectly.
edit record matches no tag	The tag list contains no tag which has the correct type and satisfies the conditions specified in the edit record.
edit record matches multiple tags	There is more than one tag which has the correct type and satisfies the conditions specified in the edit record.
Error while deserializing device header	
Device header has the wrong version: <32 bit version number>	Device header has the wrong version: <32 bit version number>
device header does not match	An edit record for the device header does not match
The file does not contain a tag list.	
BUG...	Any error messages starting with "BUG" indicate bugs in the program or the tag structure definitions. If you have made any changes of your own, check them. If you have not made any such changes, contact netX support.

Table 11: Error Messages which can occur when processing a Patch File

## 5 MakeNXO

MakeNXO is a command line tool to create an rcX loadable module in NXO format.

### Path

MakeNXO is located in

<Install directory>\nxo\_editor\makenxo.bat

The install directory is stored in the variable PATH\_NXOEDITOR, unless you disable this option in the installer. In order to call the tool, add %PATH\_NXOEDITOR%\nxo\_editor to the search path.

### 5.1 Usage

The call syntax is

```
makenxo -o output.nxo -H header_file [-t taglist_file] [-v] elf_file
```

- header\_file is a binary file starting with a common header V3, usually followed by a device info block and a module info block.
- taglist\_file is a binary file containing the tag list. The tag list is optional.
- elf\_file is the ELF file of the loadable module
- output.nxo is the output file

The -v flag enables verbose output.

Example:

If the firmware build process generates the following files:

- Fileheader.bin (HIL\_FILE\_STRIPPEDFIRMWARE\_HEADER)
- Firmware.elf
- DefaultTagList.bin

the following command line will generate the nxo file:

```
makenxo -o Firmware.nxo -h Fileheader.bin -t DefaultTagList.bin  
Firmware.elf
```

or, from a batch file:

```
call makenxo.bat -o Firmware.nxo -h Fileheader.bin -t  
DefaultTagList.bin Firmware.elf
```

## 6 Lists

### 6.1 List of Figures

Figure 1: The Tag List Editor Window	11
Figure 2: Load/Save Area configured for NXO Files	12
Figure 3: Load/Save Area configured for NXF Files	12
Figure 4: Editing Controls for Values in Tags	15
Figure 5: The selected Tag is enabled.	15
Figure 6: The selectedTag is disabled.	15
Figure 7: The Device Header Editor Window	17
Figure 8: NXF/NXO file layout	26

### 6.2 List of Tables

Table 1: Terms, Abbreviations and Definitions	5
Table 2: Main Areas of the Editor Window	11
Table 3: Control Elements of the Editor Window	11
Table 4: Control Elements for Loading and Saving	12
Table 5: Control Elements of the Device Header Editor Window.	17
Table 6: Tagtool Commands and Arguments	18
Table 7: Tagtool Flags	18
Table 8: Data Type Selection in an Edit Record	21
Table 9: Value Constraints in an Edit Record	21
Table 10: Value Assignment in an Edit Record	21
Table 11: Error Messages which can occur when processing a Patch File	23

## 7 Appendix

### 7.1 Structure of an NXO/NXF File

Files which use the Common Header V3 have the following basic layout:

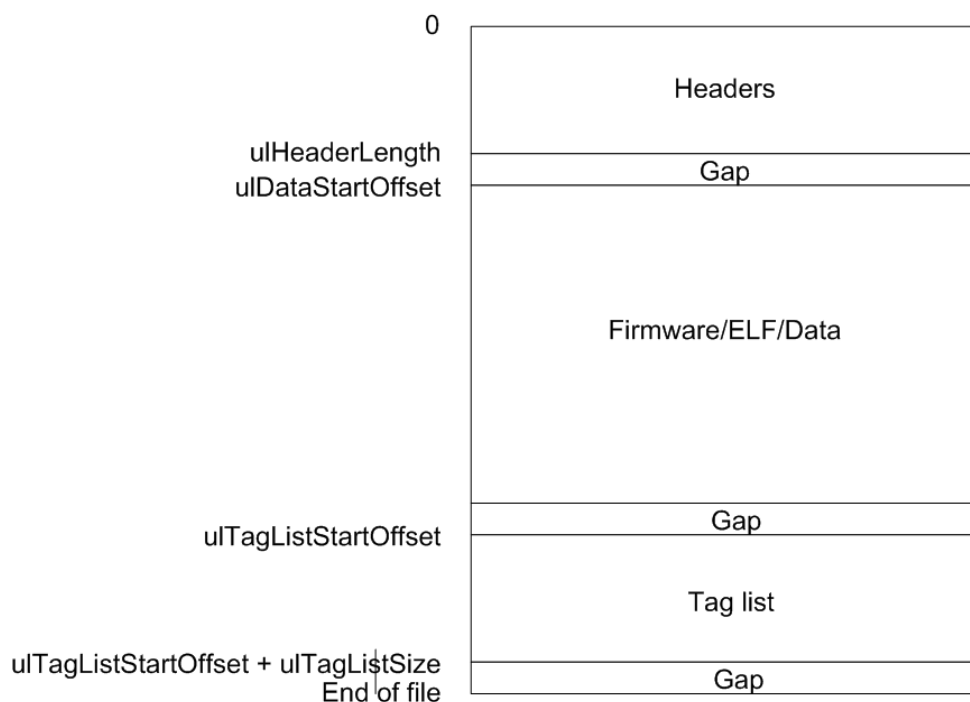


Figure 8: NXF/NXO file layout

- The header section consists of a default or boot header, the common header, a device header and a number of module info headers.
- The data section: In an NXF file, this is the executable firmware, in an NXO file, it is an ELF.
- The tag list is optional and may be located at the end of the file or between headers and data.
- There may be gaps between these sections, which are either padding or other data.

The headers contain, among others, offset and size information for the header, data and tag list sections and several checksums.

- Magic Cookie – the first four bytes of the file identify the file type.
- Common Header version. This editor was written for header version 3.0. It rejects any file with a lower version and warns if it encounters a higher version.
- ulHeaderLength – the length of the header section
- bNumModuleInfo – the number of module info blocks in the header section
- offset/size information giving the location and size of the data and tag list
- ulTagListSizeMax – the maximum size of the tag list allowed for the current file.
- Checksums – these are checked when an NXF/NXO/BSL file is loaded, and updated when it is saved.

The offset/size information allows the editor to recognize the layout of a file and to extract or replace sections of the file. The following cases are recognized:

- headers + data + tag list
- headers + data
- headers + tag list + data (the 2nd stage loader is an NXF file using this layout)



---

**Note:** There may be additional data or "gaps", between the end of the header, data and tag list sections and the beginning of the following section or the end of file. This may either be 0-3 padding bytes to align the end of the section to a dword boundary, or "real" data.

---

## 8 Contacts

### Headquarters

#### Germany

Hilscher Gesellschaft für  
Systemautomation mbH  
Rheinstrasse 15  
65795 Hattersheim  
Phone: +49 (0) 6190 9907-0  
Fax: +49 (0) 6190 9907-50  
E-Mail: [info@hilscher.com](mailto:info@hilscher.com)

#### Support

Phone: +49 (0) 6190 9907-99  
E-Mail: [de.support@hilscher.com](mailto:de.support@hilscher.com)

### Subsidiaries

#### China

Hilscher Systemautomation (Shanghai) Co. Ltd.  
200010 Shanghai  
Phone: +86 (0) 21-6355-5161  
E-Mail: [info@hilscher.cn](mailto:info@hilscher.cn)

#### Support

Phone: +86 (0) 21-6355-5161  
E-Mail: [cn.support@hilscher.com](mailto:cn.support@hilscher.com)

#### France

Hilscher France S.a.r.l.  
69500 Bron  
Phone: +33 (0) 4 72 37 98 40  
E-Mail: [info@hilscher.fr](mailto:info@hilscher.fr)

#### Support

Phone: +33 (0) 4 72 37 98 40  
E-Mail: [fr.support@hilscher.com](mailto:fr.support@hilscher.com)

#### India

Hilscher India Pvt. Ltd.  
New Delhi - 110 025  
Phone: +91 11 40515640  
E-Mail: [info@hilscher.in](mailto:info@hilscher.in)

#### Italy

Hilscher Italia srl  
20090 Vimodrone (MI)  
Phone: +39 02 25007068  
E-Mail: [info@hilscher.it](mailto:info@hilscher.it)

#### Support

Phone: +39 02 25007068  
E-Mail: [it.support@hilscher.com](mailto:it.support@hilscher.com)

#### Japan

Hilscher Japan KK  
Tokyo, 160-0022  
Phone: +81 (0) 3-5362-0521  
E-Mail: [info@hilscher.jp](mailto:info@hilscher.jp)

#### Support

Phone: +81 (0) 3-5362-0521  
E-Mail: [jp.support@hilscher.com](mailto:jp.support@hilscher.com)

#### Korea

Hilscher Korea Inc.  
Suwon, 443-810  
Phone: +82-31-204-6190  
E-Mail: [info@hilscher.kr](mailto:info@hilscher.kr)

#### Switzerland

Hilscher Swiss GmbH  
4500 Solothurn  
Phone: +41 (0) 32 623 6633  
E-Mail: [info@hilscher.ch](mailto:info@hilscher.ch)

#### Support

Phone: +49 (0) 6190 9907-99  
E-Mail: [ch.support@hilscher.com](mailto:ch.support@hilscher.com)

#### USA

Hilscher North America, Inc.  
Lisle, IL 60532  
Phone: +1 630-505-5301  
E-Mail: [info@hilscher.us](mailto:info@hilscher.us)

#### Support

Phone: +1 630-505-5301  
E-Mail: [us.support@hilscher.com](mailto:us.support@hilscher.com)