

B UTILITIES

Overview

Your Analog Devices development software comes with several file conversion utilities, which only run from a command line. Some of these utilities provide support for legacy code, and others are intended for a group of users who prefer to use the command line version of the tools instead of using them through the VisualDSP++ environment. This appendix describes these utilities and their command lines.

Dumper — ELF File Dumper

The ELF file dumper (`elfdump.exe`) extracts data from ELF format executable files (`.DXE`) and provides a text output file that describes the ELF file's contents. The ELF file dumper has the following command line:

```
C:\Program Files\Analog Devices\VisualDSP elfdump
```

```
Usage: elfdump {option} {objectfile}
```

Table B-1. ELF File Dumper Command-Line Switches

Switch	Description
-fh	Print file header.
-arsym	Print the archive symbol table.
-arall	Print every archive member.

Table B-1. ELF File Dumper Command-Line Switches (Cont'd)

Switch	Description
-ph	Print program header table.
-sh	Print section header table. The default is -sh if no options are specified.
-n <i>name</i>	Print contents of the named section(s). Name may be a simple 'glob'-style pattern, using ? and * as wild card characters. Each section's name and type determines its output format, unless overridden by a modifier (see below).
-i x0[-x1]	Print contents of the sections numbered x0 through x1, where x0 and x1 are decimal integers, and x1 defaults to x0 if omitted. Formatting rules as are for -n.
-all	Print everything. Same as -fh -ph -sh -notes -n '*'
-ost	Omit string table sections.
<i>objectfile</i>	File whose contents are to be printed. It can be a core file, executable, shared library, or relocatable object file. If the name is in the form A(B), A is assumed to be an archive and B is an ELF element in the archive. B can be a pattern like the one accepted by -n. The -n and -i options can have a modifier letter after the main option character which forces section contents to be formatted in the following ways: a Dump contents in hex and ASCII format, 16 bytes per line. x Dump contents in hex format, 32 bytes per line. xN Dump contents in hex format, N bytes per group (default is N=4). t Dump contents in hex format, N bytes per line, where N is the section's table entry size. If N is not in the range 1..32, 32 is used. i Print contents as list of disassembled machine instructions.

Using the Archiver and Dumper For Disassembly

The file utilities can become much more effective when you combine their capabilities. One interesting application of these utilities is to disassemble a library member, converting it to source code. This application can be used when your source for a particularly useful routine is missing and is only available as a library routine.

The following procedure lists the objects in a library, extracts an object, and converts the object to a listing file. Using the following archiver command line, list the objects in the library and write the output to a text file:

```
elfar -p libc.dlb > libc.txt
```



Assuming the current directory is:

```
C:\Program Files\Analog Devices\VisualDSP++\219x\lib>
```

Open the text file, scroll through it, and find the object file that you need. Then, use the following archiver command line to extract the object from the library:

```
elfar -e libc.dlb fir.doj
```

To convert the object file to an assembly listing file with labels (almost source, just needs the line numbers and opcodes removed), use the following `elfdump` command line:

```
elfdump -ns * fir.doj > fir.asm
```

Using disassembly, you get a listing file with symbols. Assemble source with symbols can be useful if you are familiar with the code and hopefully have some documentation on what the code does. If symbols were stripped during linking, there are no symbols in the dumped file.



Using disassembly on a third party's library may violate the license for the third party's software. Check on copyright and license issues with the code's owner before using this disassembly technique.

Dumping Overlay Archive Files

Use the `elfar` and `elfdump` commands to extract and view the contents of the overlay archive file (*.OVL).

For example,

```
elfar -P CLONE2.OVL
```

will print the contents to `CLONE2.ELF` which will be an *.ELF file that can be viewed with `elfdump`.

Use `elfdump` to view `CLONE2.ELF`:

```
elfdump -all CLONE2.OVL(CLONE2.elf)
```

or extract `CLONE2.ELF` and dump:

```
elfar -e CLONE2.elf
```

```
elfdump -all CLONE2.elf
```

(or use whatever `elfdump` options one wants).

These commands are case sensitive.

AEXE2ELF and ELF2AEXE Converters

The linker supports legacy object files and libraries created by the previous release of VisualDSP Development Tools. The linker recognizes AEXE-formatted object files (`.obj`) and libraries (`.o` or `.lib`), and translates them into ELF format automatically before linking.

If you are using many object or library files in your project builds or find you are rebuilding frequently, consider converting these files first. You can do this using the `aexe2elf` utility. Doing this conversion first can potentially save time in your project development cycles.



Neither the linker nor the `aexe2elf` conversion utility program preserve debug information during this translation. As a result, the AEXE files must be reassembled or recompiled to be debugged.

Use the following command syntax and switches to perform AEXE-to-ELF file conversion.

```
aexe2elf <switches> input_filename <output_filename>
```

Example:

```
aexe2elf -x App61 App70
```

The result is `App70.dxe`.

The *input_filename* is the base name of the files to translate, the *output_filename* is the output base filename, if specified. If the *output_filename* is not specified, the default is the input base filename plus `.ELF` extension.

The following is the list of switches available for use with the `aexe2elf` command.

AEXE2ELF and ELF2AEXE Converters

Table B-2. AEXE2ELF Switches

Switch	Description
-o	Use -o to translate AEXE object files to an ELF object file (default)
-x	Use -x to translate AEXE executable files to an ELF executable file
-a	Use -a to translate AEXE archive file to an ELF 'ar' archive file
-s source_filename	Enter the original source file name
-source source_filename	Enter the original source file name (also allowed)
-h	Use -h to display the help screen with the list of switches
-help	Use -help to display this help screen with the list of switches (also allowed)

Use the following command syntax and switches to perform ELF-to-AEXE file conversion.

```
elf2aexe <switches> input_filename <output_filename>
```

The *input_filename* is the base name of the files to translate, the *output_filename* is the output base filename, if specified. If the *output_filename* is not specified, the default is the input base filename plus .AEXE extension. There are two AEXE outfiles: *base.sym* and *base.exe*.

Table B-3. ELF2AEXE Switches

Switch	Description
-x	Use -x to translate the ELF executable files to AEXE executable files (default)
-h or -help	Use -h (or -help) to display the help screen with the list of switches
-v or -version	Use -v or -version to display version information only.