

3 ARCHIVER

Overview

The VisualDSP++ archiver^{*}, `elfar.exe`, combines object files[†] into archive (library) files, which can serve as a reusable resource for code development. The linker can rapidly search the archive files for routines (archive members) referred to in other objects and link these routines into your executable program. You can run the archiver from a command line, or produce an archive file as the output of a VisualDSP++ project.

This chapter contains the following information on the archiver:

- [“Archiver Guide” on page 3-2](#): introduces the archiver’s functions
- [“Archiver Command-Line Reference” on page 3-6](#): reference information on archiver operations
- [“Archiver Glossary” on page 3-10](#): a glossary of archiver-related terms

^{*} Also called “librarian.”

[†] The archiver is general-purpose: it can combine (and extract) arbitrary files. This manual refers to DSP object files because they are relevant to DSP code development.

Archiver Guide

`elfar.exe` combines and indexes object (or any other) files, producing a searchable archive file. It can perform the following operations, as directed by options on its command line:

Append one or more object files to an existing archive file

- Create an archive file from a list of object files.
- Delete file(s) from an archive file.
- Extract file(s) from an archive file.
- List the filename contents of an existing archive file (to `stdout`)
- Replace file(s) in an existing archive file

The archiver can only run one of these operations at a time. However, for commands that can take a list of files as arguments, it can input a file containing the (whitespace-separated) names of object files, which makes long lists easily manageable.

Creating an Archive From VisualDSP++

Within the VisualDSP++ 2.0 IDDE, you can choose to create an archive file as your project's output. To do so, specify **Archive file** as the target type in the project's property page. That property page appears when you create a new project, or click `Property.Project Options` on an existing project.

VisualDSP++ writes its output to `<projectname>.DLB`. To modify or list the contents of an archive file, or perform any other operations on it, you must use the archiver from the command line.

Filename Conventions

To maintain consistency within your code, it is recommend you use the conventions in [Table 3-1](#). (Note that VisualDSP++ always writes out `<projectname>.DLB` when it creates an archive.)

Table 3-1. File Name Extension Conventions

Extension	File Description
.dlb	Archive file
.doj	Object file - input to archiver
.txt	Archiver - i switch input (list file)

Making Archived Functions Usable

In order to use the archiver effectively, you must know how to write archive files which make your DSP functions available to your code (via the linker), and how to write code that accesses these archives.

Archive usage consists of two tasks, namely:

- Writing *archive routines*, functions which can be called from other programs.
- Using archive routines: accessing archived functions in your code.

This section describes both tasks.

Writing Archive Routines: Creating Entry Points

An archive routine is simply a routine in your code that can be accessed by other programs. Each such routine must have a globally visible start label (*entry point*). Any code accessing that routine must know the entry point's name and declare it as an external variable in the calling code.

Archiver Guide

To create these entry points, use the following steps.

1. Declare the start label of each routine as a global symbol with the assembler's `.GLOBAL` directive. That is the entry point.

The following code fragment has two entry points, `dIriir` and `FAE`.

```
...  
.global dIriir;  
.var/dm/seg=seg_data1 FAE[2];  
.init FAE[2]: 0x1234,0x4321;  
  
.global FAE;  
dIriir:    CNTR=N-2;  
    I2 = ^FAE;
```

2. Assemble and archive the code containing these routines. You can do so in two ways.
 - Direct the VisualDSP++ IDDE to produce an archive (see above on how to do so). When you build the project, your object code containing the entry points is packaged in `<projectname>.DLB`. [You can extract the object (`.DOJ`) whenever you want, for example to incorporate it in another project.]
 - If you create executable or unlinked object files from the IDDE, you can archive them afterward from the command line. The result is the same.

Using Archive Routines

Programs that call an archive routine must define the routine's start label as an external label with the assembler's `.EXTERN` directive. When you link the program, you specify the archive file (`.DLB`) to the linker, along with the names of the object files to link. The linker then searches the library files to resolve symbols and links the appropriate routines into the executable file.

Any file containing a label referenced by your program is linked into the executable output file.

The advantage of linking archives over using individual object files is that the linker can search archives faster, and you do not need to enter all of the file names, just the archive name.

In the following example, the archiver creates the `filter.dlb` archive, containing the object files: `taps.doj`, `coeffs.doj`, and `go_input.doj`:

```
elfar -c filter.dlb taps.doj coeffs.doj go_input.doj
```

If you then run the linker with the following command line, the linker links the object files `main.doj`, `sum.doj`, and `graph.doj`; uses the default linker description file (for example, `ADSP-218x.ldf`;) and creates the executable file (`main.dxe`):

```
linker -DADSP-218x main.doj sum.doj graph.doj filter.dlb
```

Assuming that one or more library routines from `filter.dlb` are called from one or more of the object files, the linker searches the archive, extracts the required routines, and links the routines into the executable.

Archiver Command-Line Reference

The archiver processes object files into an archive file. Its output is an archive file with the file name extension `.DLB`^{*}. It can also append, delete, extract, or replace member files in an archive, as well as list them to `stdout`.

This section provides reference information on the archiver command line and linking.

Command-Line Syntax

Use the following syntax for the archiver command line. [Table 3-2](#) (below) describes each switch

```
elfar [-a|c|d|e|p|r][-v][-i filename] library_file object_file
...
```

This command line is subject to the following constraints:

- You may select exactly one action switch (a, c, d, e, p, r) in a single command.
- The verbose operation switch `-v`, must not be in a position where it can be mistaken for an object file, meaning it may not follow the `library_file` on append or create.
- The file include switch, `-i`, must immediately precede the include file name.
- Use the archive filename first, following the switches. [`-i` and `-v` are not operational switches, and can appear later.]

^{*} `.DLB` is the default extension for archive files, `.DOJ` is the default extension for object files. You can use any name.

- Enclose file names containing whitespace or colons within straight quotes.
- Append the appropriate file name extension to each file. The archiver assumes nothing, and will not do it for you.
- You cannot use wildcards. To perform an archive operation on a list of member files, write the list into a textfile and use it as input to the command line (with the `-i` switch).
- *object_file* — The name of an object file (.DOJ) to be added, removed, or replaced in the *library_file*.

Note that the archiver's `-i` switch lets you input a list of members from a text file, instead of listing all the members on the command line. Also note that when you use the archiver's `-p` switch, you do not need to identify any members on the command line.

The archiver's command line is case-insensitive. The following command line, for example:

```
elfar -v -c my_lib.dlb fft.doj sin.doj cos.doj tan.doj
```

runs the archiver as follows:

`-v` — Selects verbose mode for the archiver0

`-c my_lib.dlb` — Creates an archive file named `my_lib.dlb`

`fft.doj sin.doj cos.doj tan.doj` — Object files that the archiver puts in the archive file

The archiver takes file names as parameters. [Table 3-1 on page 3-3](#) lists the relevant types of files, names, and extensions normally used in VisualDSP++.

Archiver File Search

File searches are important in the archiver's process. The archiver supports relative and absolute directory names, default directories, and user-specified directories for file search paths. File searches occur as follows:

1. *Specified path*—If you include relative or absolute path information in a file name, the archiver only searches in that location for the file.
2. *Default directory*—If you do not include path information in the file name, the archiver searches for the file in the current working directory.

Command-Line Reference

[Table 3-2](#) describes each archiver parameter and switch*.

Table 3-2. Archiver Command-Line Switches

Switch	Description
<i>archive-file</i>	The archive that the archiver modifies. This parameter appears after the switch.
<i>member-file</i>	One or more object files that the archiver uses when modifying the archive. This parameter appears after <i>archive-file</i> . You can use the <i>-i</i> switch to input object file names as a list.
<i>-a</i>	Append one or more <i>member files</i> to the named archive file.
<i>-c</i>	Create a new <i>archive-file</i> containing the <i>member-files</i> on the command line.
<i>-d</i>	Delete one or more <i>member-files</i> from the selected <i>archive-file</i> .

Table 3-2. Archiver Command-Line Switches (Cont'd)

Switch	Description
<code>-i <list file></code>	Use <i>list-file</i> , containing member-file names, as an input. This file lists the <i>member-files</i> to add or modify in the selected <i>archive-file</i> (.DLB).
<code>-p</code>	The <code>-p</code> (print archive contents) switch directs the archiver to print to standard output a list of the <i>member-files</i> (.DDJ) in the selected <i>archive-file</i> (.DLB).
<code>-v</code>	The <code>-v</code> (verbose archiver messages) switch directs the archiver to output status information as the archiver processes your files.

* The switches in [Table 3-2](#) must appear before the *archive-file* name on the command line, except that the `-i` switch appears in place of the *member-files*. Items shown in [] are optional. Items shown in *italics* are user defined and are described with each switch in [Table 3-2](#).

Archiver Glossary

Archives—files containing one or more members. Archives are searched by the linker for routines that you call from your programs.

Entry point | Start label—the label in the archive member that begins the routine that you call from your program. Entry points are global labels in the archive member and must be external labels in your program.

Members—object files in an archive.