ARM® Compiler

Version 6.01

armar User Guide

ARM®

ARM DUI 0806B (ID121814)

ARM Compiler
armar User Guide

Release Information

The following changes have been made to this document.

Change History

Date	Issue	Confidentiality	Change
14 March 2014	A	Non-Confidential	ARM Compiler v6.00 Release
15 December 2014	B	Non-Confidential	ARM Compiler v6.01 Release

Proprietary Notice

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Product Status

The information in this document is final, that is for a developed product.

Web Address

`http://www.arm.com`

Contents
ARM Compiler armar User Guide

Chapter 1
Conventions and feedback

The following describes the typographical conventions and how to give feedback:

Typographical conventions

The following typographical conventions are used:

`monospace` Denotes text that can be entered at the keyboard, such as commands, file and program names, and source code.

`mono`space Denotes a permitted abbreviation for a command or option. The underlined text can be entered instead of the full command or option name.

`monospace italic`

Denotes arguments to commands and functions where the argument is to be replaced by a specific value.

`monospace bold`

Denotes language keywords when used outside example code.

italic Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.

bold Highlights interface elements, such as menu names. Also used for emphasis in descriptive lists, where appropriate, and for ARM® processor signal names.

Feedback on this product

If you have any comments and suggestions about this product, contact your supplier and give:

- your name and company

- the serial number of the product
- details of the release you are using
- details of the platform you are using, such as the hardware platform, operating system type and version
- a small standalone sample of code that reproduces the problem
- a clear explanation of what you expected to happen, and what actually happened
- the commands you used, including any command-line options
- sample output illustrating the problem
- the version string of the tools, including the version number and build numbers.

Feedback on content

If you have comments on content then send an e-mail to `errata@arm.com`. Give:

- the title
- the number, ARM DUI 0806B
- if viewing online, the topic names to which your comments apply
- if viewing a PDF version of a document, the page numbers to which your comments apply
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

ARM periodically provides updates and corrections to its documentation on the ARM Information Center, together with knowledge articles and *Frequently Asked Questions* (FAQs).

Other information

- ARM Information Center `http://infocenter.arm.com/help/index.jsp`
- ARM Technical Support Knowledge Articles `http://infocenter.arm.com/help/topic/com.arm.doc.faqs/index.html`
- ARM Support and Maintenance `http://www.arm.com/support/services/support-maintenance.php`
- ARM Glossary `http://infocenter.arm.com/help/topic/com.arm.doc.aeg0014-/index.html`.

Chapter 2
Overview of the ARM Librarian

The following topics give an overview of the ARM® librarian, `armar`:

Tasks

Concepts

Reference

2.1 About the ARM librarian

The ARM librarian, `armar`, enables you to collect and maintain sets of ELF object files in standard format `ar` libraries. You can pass these libraries to the linker in place of several ELF object files.

With `armar` you can:

- create new libraries
- add files to a library
- replace individual files in a library
- replace all files in a library with specified files in a single operation
- control the placement of files in a library
- display information about a specified library. For example, list all members in a library.

A timestamp is also associated with each file that is added or replaced in a library.

Note

When you create, add, or replace object files in a library, `armar` creates a symbol table by default. However, debug symbols are not included by default.

2.1.1 See also

Tasks

- Chapter 3 *Creating Libraries*
- Chapter 4 *Managing libraries*
- Chapter 5 *Displaying information about libraries*.

Reference

- *--debug_symbols* on page 6-10.

armlink Reference Guide:

- *--library* on page 2-82
- *--libpath* on page 2-81
- *--userlibpath* on page 2-153.

2.2 Considerations when working with library files

Be aware of the following:

- A library differs from a shared object or *Dynamic Link Library* (DLL) in that:
  - symbols are imported from a shared object or DLL
  - code or data for symbols is extracted from an archive into the file being linked.
- Linking with an object library file might not produce the same results as linking with all the object files collected into the object library file. This is because the linker processes the input list and libraries differently:
  - Each object file in the input list appears in the output unconditionally, although unused areas are eliminated if the `armlink --remove` option is specified.
  - A member of a library file is only included in the output if it is referred to by an object file or a previously processed library file.

  The linker recognizes a collection of ELF files stored in an `ar` format file as a library. The contents of each ELF file form a single member in the library.

2.2.1 See also

Reference

armlink Reference Guide:

- *--remove, --no_remove* on page 2-118.

2.3 armar command-line syntax

The `armar` command-line syntax is:

`armar [options] archive [file_list]`

options armar command-line options.

archive The filename of the library. A library file must always be specified.

file_list The list of files to be processed.

2.3.1 See also

Reference

- *archive* on page 6-3
- *file_list* on page 6-17.

2.4 armar command-line options listed in groups

The `armar` command-line options are:

Controlling diagnostic messages

- *-c* on page 6-6
- *--diag_error=tag[,tag,...]* on page 6-11
- *--diag_remark=tag[,tag,...]* on page 6-12
- *--diag_style={arm|ide|gnu}* on page 6-13
- *--diag_suppress=tag[,tag,...]* on page 6-14
- *--diag_warning=tag[,tag,...]* on page 6-15
- *--show_cmdline* on page 6-26.

Getting command-line help

- *--help* on page 6-18
- *--version_number* on page 6-32
- *--vsn* on page 6-34.

Getting command-line arguments from a file

- *--via=filename* on page 6-33.

Managing libraries

- *-C* on page 6-7
- *--create* on page 6-8
- *-d* on page 6-9
- *--debug_symbols* on page 6-10
- *--entries* on page 6-16
- *-r* on page 6-24
- *-T* on page 6-29
- *-u* on page 6-30.

Controlling the placement of files in a library

- *-a pos_name* on page 6-4
- *-b pos_name* on page 6-5
- *-i pos_name* on page 6-19
- *-m pos_name* on page 6-20

Controlling librarian output

- *-n* on page 6-21
- *-p* on page 6-23
- *-s* on page 6-25
- *--sizes* on page 6-27
- *-t* on page 6-28
- *-v* on page 6-31
- *-x* on page 6-35
- *--zs* on page 6-36
- *--zt* on page 6-37.

2.5 Getting help on the armar command

Use the `--help` option to display a summary of the main command-line options.

This is the default if you do not specify any options or source files.

2.5.1 Example

To display the help information, enter:

```
armar --help
```

2.5.2 See also

Reference

- *armar command-line syntax* on page 2-4
- *--help* on page 6-18.

Chapter 3
Creating Libraries

The following topic describes how to use the ARM® librarian, `armar`, to create libraries:

Tasks

- *Creating a new object library* on page 3-2.

3.1 Creating a new object library

Use the `--create` option to create a new object library, and either specify the list of object files:

- directly on the command-line
- in a via file.

Note

If the library already exists, the previous contents are deleted.

To create a new library containing only the files listed on the command-line, enter:

```
armar --create mylib.a *.o
```

To create a new library containing the files listed in a via file, enter:

```
armar --create mylib.a --via myobject.via
```

You can use this option in conjunction with the following compatible command-line options:

- `-c`
- `--diag_style`
- `-n`
- `-v`
- `--via`.

Note

Other options can also create a new library in some circumstances. For example, using the `-r` option with a library that does not exist.

3.1.1 See also

Reference

- *-c* on page 6-6
- *--create* on page 6-8
- *--entries* on page 6-16
- *-n* on page 6-21
- *-r* on page 6-24
- *-v* on page 6-31
- *--via=filename* on page 6-33.

Chapter 4
Managing libraries

The following topics describe how to manage libraries with the ARM® librarian, `armar`:

Tasks

4.1 Replacing individual files in an object library

Use the `-r` option to replace existing files in the library. If the library does not exist, a new library file is created and a diagnostic message is written to standard error. You can use this option in conjunction with other compatible command-line options.

`-q` is an alias for `-r`.

If no files are specified and the library exists, the results are undefined. Files that replace existing files do not change the order of the library.

If the `-u` option is used, then only those files with modification dates later than the library files are replaced.

If the `-a`, `-b`, or `-i` option is used, then *`pos_name`* must be present and specifies that new files are to be placed after (`-a`) or before (`-b` or `-i`) *`pos_name`*. Otherwise the new files are placed at the end.

4.1.1 Example

To add or replace files `obj1.o`, `obj2.o`, and `obj3.o` in a library `mylib.a`, enter:

```
armar -r mylib.a obj1.o obj2.o obj3.o
```

To replace files in a library, and only if the file in the library is older than the specified file, enter:

```
armar -ru mylib.a k*.o
```

4.1.2 See also

Tasks

- *Replacing all files in a library with specific files* on page 4-3
- *Controlling the placement of files when adding to a library* on page 4-4.

Reference

- *armar command-line syntax* on page 2-4.
- *-a pos_name* on page 6-4
- *-b pos_name* on page 6-5
- *-i pos_name* on page 6-19
- *-r* on page 6-24.
- *-u* on page 6-30.

4.2 Replacing all files in a library with specific files

Use the `-r` option to replace all files in a library with the files specified in the *`file_list`* parameter. You can use this option in conjunction with other compatible command-line options.

Note

If the library specified on the command line does not exist, then a new library will be created.

4.2.1 Example

To replace all files in the `mylib.a` library by all object files in the current directory, enter:

```
armar -r mylib.a *.o
```

4.2.2 See also

Tasks

- *Replacing individual files in an object library* on page 4-2
- *Controlling the placement of files when adding to a library* on page 4-4.

Reference

- *armar command-line syntax* on page 2-4
- *-r* on page 6-24.

4.3 Controlling the placement of files when adding to a library

These options enable you to specify the location of files that are added to the library:

`-a pos_name` Place new files in the library after the file *pos_name*.

`-b pos_name` Place new files in the library before the file *pos_name*.

`-i pos_name` Place new files in the library before the file *pos_name*.

You can use these options in conjunction with other compatible command-line options.

4.3.1 Example

To add files `obj3.o` and `obj4.o` immediately after `obj2.o` in `mylib.a`, enter:

```
armar -a obj2.o mylib.a obj3.o obj4.o
```

4.3.2 See also

Tasks

- *Replacing individual files in an object library* on page 4-2
- *Replacing all files in a library with specific files* on page 4-3.

Reference

- *armar command-line syntax* on page 2-4
- *-a pos_name* on page 6-4
- *-b pos_name* on page 6-5
- *-i pos_name* on page 6-19.

4.4 Moving files in a library

Use the `-m` option with one of the following options to move the specified files to a new location in a library:

`-a` *`pos_name`* Move the files after the file *`pos_name`*.

`-b` *`pos_name`* Move the files before the file *`pos_name`*.

`-i` *`pos_name`* Move the files before the file *`pos_name`*.

no option Move the files to the end of the library.

You can use this option in conjunction with other compatible command-line options.

4.4.1 Example

To move the file `file1.o` to a new location after `file2.o` in the `mylib.a` library, enter:

```
armar -m -a file2.o mylib.a file1.o
```

4.4.2 See also

Reference

- *armar command-line syntax* on page 2-4
- *-a pos_name* on page 6-4
- *-b pos_name* on page 6-5
- *-i pos_name* on page 6-19
- *-m pos_name* on page 6-20.

4.5 Deleting files from a library

Use the `-d` option to delete one or more files from a library. You specify the files with the *`file_list`* parameter.

You can use this option in conjunction with other compatible command-line options.

4.5.1 Example

To delete the files `file1.o` and `file2.o` from the `mylib.a` library, enter:

```
armar -d mylib.a file1.o,file2.o
```

4.5.2 See also

Reference

- *armar command-line syntax* on page 2-4
- *-d* on page 6-9.

4.6 Extracting files from a library

Use the `-x` option to extract the files in the *`file_list`* parameter from the library. The contents of the library are not changed. If no file operands are given, all files in the library are extracted. If the filename of a file extracted from the library is longer than that supported in the destination directory, the results are undefined.

The files are extracted to the current location.

You can use this option in conjunction with other compatible command-line options.

4.6.1 Example

To extract the files `file1.o` and `file2.o` from the `mylib.a` library in the directory `C:\temp` to the `C:\temp\obj`, enter:

```
cd C:\temp\obj
armar -x ..\mylib.a file1.o,file2.o
```

4.6.2 See also

Reference

- *armar command-line syntax* on page 2-4
- *-x* on page 6-35.

4.7 Creating or suppressing symbol tables

Use the `-s` option to create a symbol table in the library. This option is useful for libraries that have been created:

- using the `-n` option
- with an archiver that does not automatically create a symbol table.

Use the `-n` option to suppress the creation of a symbol table in the library.

Note

By default, `armar` always creates a symbol table when you create a library of object files.

4.7.1 Example

To create a library without a symbol table, containing all object files in the current directory, enter:

```
armar -n --create mylib.a *.obj
```

To create a symbol table in a library that was created using the `-n` option, enter:

```
armar -s mylib.a
```

4.7.2 See also

Reference

- *armar command-line syntax* on page 2-4
- *-n* on page 6-21
- *-s* on page 6-25
- *--zs* on page 6-36.

Chapter 5
Displaying information about libraries

The following topics describe how to display information about a library with the ARM® librarian, `armar`:

Tasks

5.1 Displaying object files with entry points

Use the `--entries` option to list all object files in the library that have an entry point defined using the assembler `ENTRY` directive.

The format for the listing is:

`ENTRY at offset num in section name of member`

5.1.1 Example

To list the entry point of each object file in `myasm.a`, enter:

```
> armar --entries myasm.a
ENTRY at offset 0 in section adrlabel of adrlabel.o
ENTRY at offset 0 in section ARMex of armex.o
ENTRY at offset 0 in section Block of blocks.o
ENTRY at offset 0 in section Jump of jump.o
ENTRY at offset 0 in section LDRlabel of ldrlabel.o
ENTRY at offset 0 in section Loadcon of loadcon.o
ENTRY at offset 0 in section StrCopy of strcopy.o
ENTRY at offset 0 in section subrout of subrout.o
ENTRY at offset 0 in section Tblock of tblock.o
ENTRY at offset 0 in section ThumbSub of thumbsub.o
ENTRY at offset 0 in section Word of word.o
```

5.1.2 See also

Reference

- *armar command-line syntax* on page 2-4
- *--entries* on page 6-16.

5.2 Displaying the contents of files in the library

Use the `-p` option to display the contents of files in the library to `stdout`.

5.2.1 Example

To display the contents of `file1.o` in `mylib.a`, enter:

```
armar -p mylib.a file1.o
```

5.2.2 See also

Reference

- *armar command-line syntax* on page 2-4
- *-p* on page 6-23.

5.3 Displaying command-line processing

Use the `--show_cmdline` option to show how armar processes the command line. It shows the commands normalized and expands the contents of any `via` files.

5.3.1 Example

To show how armar processes the command-line options for the replacement of file `obj1.o` in `mylib.a`, enter:

```
> armar --show_cmdline -r mylib.a obj1.o
[armar --show_cmdline -r mylib.a obj1.o]
```

5.3.2 See also

Reference

- *armar command-line syntax* on page 2-4
- *--show_cmdline* on page 6-26.

5.4 Displaying file sizes

Use the `--sizes` option to list the `Code`, `RO Data`, `RW Data`, `ZI Data`, and `Debug` sizes of each member in the library.

5.4.1 Example

The following example shows the sizes of the files in `mylib.a`:

```
> armar --sizes mylib.a

Code    RO Data    RW data    ZI Data    Debug   Object Name
 464          0          0          0     8612   app_1.o
3356          0          0      10244    11848   app_2.o
3820          0          0      10244    20460   TOTAL
```

5.4.2 See also

Reference

- *armar command-line syntax* on page 2-4
- *--entries* on page 6-16
- *--sizes* on page 6-27
- *--zs* on page 6-36.

5.5 Displaying a table of contents

Use the `-t` option to display a table of contents for the library. The files specified by *`file_list`* are included in the written list. If *`file_list`* is not specified, all files in the library are included in the order of the archive.

5.5.1 Example

To display the table of contents of `mylib.a`, enter:

```
> armar -t mylib.a
app_1.o
app_2.o
```

5.5.2 See also

Reference

- *armar command-line syntax* on page 2-4
- *-t* on page 6-28.

5.6 Displaying the version number

Use the `--vsn` or `--version` options to display the version number to `stderr`.

5.6.1 Example

```
> armar --vsn
ARM Librarian, N.n [Build num]
```

5.6.2 See also

Reference

- *armar command-line syntax* on page 2-4
- *--vsn* on page 6-34.

5.7 Displaying the symbol table

Use the `--zs` option to display the symbol table for all files in the library.

5.7.1 Example

To list the symbol table in the `mylib.a` library, enter:

```
> armar --zs mylib.a
__ARM_use_no_argv    from hello.o    at offset    210
main                 from hello.o    at offset    210
__ARM_use_no_argv    from fncalls.o  at offset   1870
add                  from fncalls.o  at offset   1870
main                 from fncalls.o  at offset   1870
_Z1fv                from test.o     at offset   7758
_ZN1T1fEi            from test.o     at offset   7758
__ARM_use_no_argv    from s.o        at offset   9242
main                 from s.o        at offset   9242
altstack             from s.o        at offset   9242
```

5.7.2 See also

Reference

- *armar command-line syntax* on page 2-4
- *-n* on page 6-21
- *-s* on page 6-25
- *--zs* on page 6-36.

5.8 Displaying member sizes and entry points for a library

Use the `--zt` option to list member sizes and entry points for all files in the library.

5.8.1 Example

To list member sizes and entry points for all files in the `mylib.a` library, enter:

```
> armar --zt mylib.a

      Code    RO Data    RW Data    ZI Data      Debug   Object Name
        36          0          0          0       3883   test.o
        36          0          0          0        140   t.o
        72          0          0          0       4023   TOTAL

No ENTRY points found.
```

5.8.2 See also

Reference

- *armar command-line syntax* on page 2-4
- *--entries* on page 6-16
- *--sizes* on page 6-27
- *--zt* on page 6-37.

Chapter 6
armar command reference

The following topics describes the command-line options of the ARM® librarian, `armar`, provided with the ARM Compiler toolchain:

See also *armar command-line syntax* on page 2-4.

6.1 *archive*

Specifies the location of the library to be created, modified, or read.

Note

If you include a list of files in *file_list*, they must be specified after the library file.

6.1.1 See also

Reference

- Chapter 6 *armar command reference*.

6.2 -a *pos_name*

Places new files in the library after the file *pos_name*. The effect of this option is negated if you include -b (or -i) on the same command line.

6.2.1 Example

To add or replace files obj3.o and obj4.o immediately after obj2.o in mylib.a, enter:

```
armar -r -a obj2.o mylib.a obj3.o obj4.o
```

6.2.2 See also

Tasks

- *Controlling the placement of files when adding to a library* on page 4-4.

Reference

- *-b pos_name* on page 6-5
- *-i pos_name* on page 6-19
- *-m pos_name* on page 6-20
- *-r* on page 6-24.

6.3 -b *pos_name*

Places new files in the library before the file *pos_name*. This option takes precedence if you include -a on the same command line.

6.3.1 See also

Tasks

- *Controlling the placement of files when adding to a library* on page 4-4.

Reference

- *-a pos_name* on page 6-4
- *-i pos_name* on page 6-19
- *-m pos_name* on page 6-20
- *-r* on page 6-24.

6.4 -c

Suppresses the diagnostic message normally written to `stderr` when a library is created.

6.4.1 See also

Reference

- Chapter 6 *armar command reference*.

6.5 -C

Instructs the librarian not to replace existing files with like-named files when performing extractions. Use this option with -T to prevent truncated filenames from replacing files with the same prefix.

An error message is displayed if the file to be extracted already exists in the current location.

6.5.1 See also

Reference

- *-T* on page 6-29
- *-x* on page 6-35.

6.6 --create

Creates a new library containing only the files specified in *file_list*. If the library already exists, its previous contents are discarded.

6.6.1 Example

To create a new library by adding all object files in the current directory, enter:

```
armar --create mylib.a *.o
```

6.6.2 See also

Tasks

- *Creating a new object library* on page 3-2.

6.7 -d

Deletes one or more files specified in *file_list* from the library.

6.7.1 See also

Tasks

- *Deleting files from a library* on page 4-6.

6.8 --debug_symbols

By default, debug symbols are excluded from an archive. Use `--debug_symbols` to include debug symbols in the archive.

6.8.1 See also

Concepts

- *About the ARM librarian* on page 2-2.

6.9 --diag_error=*tag[,tag,...]*

This option sets diagnostic messages that have a specific tag to error severity.

6.9.1 Syntax

`--diag_error=`*`tag[,tag,...]`*

Where *`tag`* can be:

- a diagnostic message number to set to error severity
- `warning`, to treat all warnings as errors.

6.9.2 See also

Reference

- *--diag_remark=tag[,tag,...]* on page 6-12
- *--diag_style={arm|ide|gnu}* on page 6-13
- *--diag_suppress=tag[,tag,...]* on page 6-14
- *--diag_warning=tag[,tag,...]* on page 6-15.

6.10 --diag_remark=*tag[,tag,...]*

This option sets diagnostic messages that have a specific tag to remark severity.

6.10.1 Syntax

`--diag_remark=`*`tag[,tag,...]`*

Where *tag* is a comma-separated list of diagnostic message numbers.

6.10.2 See also

Reference

- *--diag_error=tag[,tag,...]* on page 6-11
- *--diag_style={arm|ide|gnu}* on page 6-13
- *--diag_suppress=tag[,tag,...]* on page 6-14
- *--diag_warning=tag[,tag,...]* on page 6-15.

6.11 --diag_style={arm|ide|gnu}

Specifies the style to use for diagnostic messages.

6.11.1 Syntax

`--diag_style=`*`string`*

Where *`string`* is one of:

`arm` Display messages using the ARM style.

`ide` Include the line number and character count for any line that is in error. These values are displayed in parentheses.

`gnu` Display messages in the format used by GNU.

6.11.2 Default

If you do not specify a `--diag_style` option, it assumes `--diag_style=arm`.

6.11.3 See also

Reference

- *--diag_error=tag[,tag,...]* on page 6-11
- *--diag_remark=tag[,tag,...]* on page 6-12
- *--diag_suppress=tag[,tag,...]* on page 6-14
- *--diag_warning=tag[,tag,...]* on page 6-15.

6.12 --diag_suppress=*tag*[,*tag*,...]

This option disables diagnostic messages that have the specified tags.

6.12.1 Syntax

`--diag_suppress=`*`tag`*`[,`*`tag`*`,...]`

Where *tag* can be:

- a diagnostic message number to be suppressed
- `error`, to suppress all errors
- `warning`, to suppress all warnings.

6.12.2 See also

Reference

- *--diag_error=tag[,tag,...]* on page 6-11
- *--diag_remark=tag[,tag,...]* on page 6-12
- *--diag_style={arm|ide|gnu}* on page 6-13
- *--diag_warning=tag[,tag,...]* on page 6-15.

6.13 --diag_warning=*tag[,tag,...]*

This option sets diagnostic messages that have a specific tag to warning severity.

6.13.1 Syntax

`--diag_warning=`*`tag[,tag,...]`*

Where *tag* can be:

- a diagnostic message number to set to warning severity
- `error`, to downgrade all errors to warnings.

6.13.2 See also

Reference

- *--diag_error=tag[,tag,...]* on page 6-11
- *--diag_remark=tag[,tag,...]* on page 6-12
- *--diag_style={arm|ide|gnu}* on page 6-13
- *--diag_suppress=tag[,tag,...]* on page 6-14.

6.14 --entries

Lists all object files in the library that have an entry point defined using the assembler ENTRY directive.

The format for the listing is:

```
ENTRY at offset num in section name of member
```

6.14.1 Example

The following example lists the entry point of each object file in myasm.a:

```
> armar --entries myasm.a
ENTRY at offset 0 in section adrlabel of adrlabel.o
ENTRY at offset 0 in section ARMex of armex.o
ENTRY at offset 0 in section Block of blocks.o
ENTRY at offset 0 in section Jump of jump.o
ENTRY at offset 0 in section LDRlabel of ldrlabel.o
ENTRY at offset 0 in section Loadcon of loadcon.o
ENTRY at offset 0 in section StrCopy of strcopy.o
ENTRY at offset 0 in section subrout of subrout.o
ENTRY at offset 0 in section Tblock of tblock.o
ENTRY at offset 0 in section ThumbSub of thumbsub.o
ENTRY at offset 0 in section Word of word.o
```

6.14.2 See also

Reference

- *--sizes* on page 6-27
- *--zt* on page 6-37.

armasm Reference Guide:

- *ENTRY* on page 10-34.

6.15 *file_list*

A space-separated list of ELF-compliant files, such as ELF objects and ELF libraries.

Each file must be fully specified by its path and name. The path can be absolute, relative to drive and root, or relative to the current directory.

Note

The list of files must be specified after the library file.

Only the filename at the end of the path is used when comparing against the names of files in the library. If two or more path operands end with the same filename, the results are unspecified. You can use the wild characters `*` and `?` to specify files.

If one of the files is a library, `armar` copies all members from the input library to the destination library. The order of members on the command line is preserved. Therefore, supplying a library file is logically equivalent to supplying all of its members in the order that they are stored in the library.

6.15.1 See also

Reference

- Chapter 6 *armar command reference*.

6.16 --help

Displays a summary of the main command-line options.

This is the default if you do not specify any options or source files.

6.16.1 See also

Reference

- *--version_number* on page 6-32
- *--vsn* on page 6-34.

6.17 -i *pos_name*

Places new files in the library before the member *pos_name* (equivalent to -b *pos_name*).

6.17.1 See also

Tasks

- *Controlling the placement of files when adding to a library* on page 4-4.

Reference

- *-a pos_name* on page 6-4
- *-b pos_name* on page 6-5
- *-m pos_name* on page 6-20
- *-r* on page 6-24.

6.18 -m *pos_name*

Moves files. If -a, -b, or -i with *pos_name* is specified, files are moved to the new position. Otherwise, move files to the end of library.

6.18.1 See also

Tasks

- *Moving files in a library* on page 4-5.

Reference

- *-a pos_name* on page 6-4
- *-b pos_name* on page 6-5
- *-i pos_name* on page 6-19.

6.19 -n

Suppresses the creation of a symbol table in the library.

Note

By default, armar always creates a symbol table when you create a library of object files.

You can recreate the symbol table in the library using the -s option.

6.19.1 Example

To create a library without a symbol table, enter:

```
armar -n --create mylib.a *.obj
```

6.19.2 See also

Reference

- *-s* on page 6-25.

6.20 --new_files_only

Updates an object file in the archive only if a new object that has a later timestamp. When used with the `-r` option, files in the library are replaced only if the corresponding file has a modification time that is newer than the modification time of the file in the library.

6.20.1 See also

Reference

- *-r* on page 6-24
- *-u* on page 6-30.

6.21 -p

Prints the contents of files in a library to `stdout`.

6.21.1 See also

Tasks

- *Displaying the contents of files in the library* on page 5-3.

6.22 -r

Replaces, or adds, files in the library. If the library does not exist, a new library file is created and a diagnostic message is written to standard error. -q is an alias for -r.

If *file_list* is not specified and the library exists, the results are undefined. Files that replace existing files do not change the order of the library.

If the -u option is used, then only those files with dates of modification later than the library files are replaced.

If the -a, -b, or -i option is used, then *pos_name* must be present and specifies that new files are to be placed after (-a) or before (-b or -i) *pos_name*. Otherwise the new files are placed at the end.

6.22.1 Example

To add or replace obj1.o, obj2.o, and obj3.o files in a library, enter:

```
armar -r mylib.a obj1.o obj2.o obj3.o
```

To replace files in a library, and only if the file in the library is older than the specified file, enter:

```
armar -ru mylib.a k*.o
```

6.22.2 See also

Tasks

- *Replacing individual files in an object library* on page 4-2
- *Replacing all files in a library with specific files* on page 4-3
- *Controlling the placement of files when adding to a library* on page 4-4.

Reference

- *-a pos_name* on page 6-4
- *-b pos_name* on page 6-5
- *-i pos_name* on page 6-19
- *-u* on page 6-30.

6.23 -s

Creates a symbol table in the library. This option is useful for libraries that have been created:

- using the -n option
- with an archiver that does not automatically create a symbol table.

Note

By default, armar always creates a symbol table when you create a library of object files.

6.23.1 Example

To create a symbol table in a library that was created using the -n option, enter:

```
armar -s mylib.a
```

6.23.2 See also

Reference

- *-n* on page 6-21
- *--zs* on page 6-36.

6.24 --show_cmdline

This option shows how armar has processed the command line. It shows the command-line after processing by armar, and can be useful to check:

- the command-line a build system is using
- how armar is interpreting the supplied command-line, for example, the ordering of command line options.

The commands are shown in their preferred form, and the contents of any via files are expanded.

6.24.1 See also

Reference

- *--via=filename* on page 6-33
- Chapter 6 *armar command reference*.

6.25 --sizes

Lists the Code, RO Data, RW Data, ZI Data, and Debug sizes of each member in the library.

6.25.1 Example

The following example shows the sizes of app_1.o and app_2.o in mylib.a:

```
> armar --sizes mylib.a

Code    RO Data    RW data    ZI Data    Debug   Object Name
 464          0          0          0     8612   app_1.o
3356          0          0      10244    11848   app_2.o
3820          0          0      10244    20460   TOTAL
```

6.25.2 See also

Reference

- *--entries* on page 6-16
- *--zt* on page 6-37.

6.26 -t

Prints a table of contents for the library. The files specified by *file_list* are included in the written list. If *file_list* is not specified, all files in the library are included in the order of the archive.

6.26.1 Example

To list the table of contents of a library in verbose mode, enter:

```
> armar -tv mylib.a
rw-rw-rw-    0/    0   7512 Jun 22 11:19 2009 app_1.o (offset   736)rw-rw-rw-    0/
0   1452 May 19 16:25 2009 app_2.o (offset   8308)
```

6.26.2 See also

Reference

- *-v* on page 6-31.

6.27 -T

Enables truncation of filenames when extracted files have library names that are longer than the file system can support.

By default, extracting a file with a name that is too long is an error. A diagnostic message is written and the file is not extracted.

Be aware that if the multiple files in the library have the same truncated name, each subsequent file that is extracted overwrites the previously extracted file with that name. To prevent this, use the `-C` option.

6.27.1 See also

Reference

- *-C* on page 6-7
- *-x* on page 6-35.

6.28 -u

Updates older files. When used with the -r option, files in the library are replaced only if the corresponding file has a modification time that is at least as new as the modification time of the file within library.

6.28.1 See also

Reference

- *--new_files_only* on page 6-22
- *-r* on page 6-24.

6.29 -v

Gives verbose output.

The output depends on what other options are used:

`-d, -r, -x`
: Write a detailed file-by-file description of the library creation, the constituent files, and maintenance activity.

`-p`
: Writes the name of the file to the standard output before writing the file itself to the `stdout`.

`-t`
: Includes a long listing of information about the files within the library.

`-x`
: Prints the filename preceding each extraction.

6.29.1 See also

Reference

- *-d* on page 6-9
- *-p* on page 6-23
- *-r* on page 6-24
- *-t* on page 6-28
- *-x* on page 6-35.

6.30 --version_number

This option displays the version of armar you are using.

6.30.1 Syntax

```
armar --version_number
```

armar displays the version number in the format nnbbbb, where:

- nn is the version number
- bbbb is the build number.

6.30.2 Example

Version 6.0 build 0019 is displayed as 600019.

6.30.3 See also

Reference

- *--help* on page 6-18
- *--vsn* on page 6-34.

6.31 --via=*filename*

Reads an additional list of input filenames and librarian options from *filename*.

6.31.1 Syntax

```
--via=filename
```

Where filename is the name of a via file containing options to be included on the command line.

6.31.2 Usage

You can enter multiple `--via` options on the ELF file converter command line. The `--via` options can also be included within a via file.

6.31.3 See also

Reference

- Appendix A *Via File Syntax*.

6.32 --vsn

This option displays the version number to stderr. For example:

```
> armar --vsn
ARM Librarian, N.nn [Build num]
```

6.32.1 See also

Reference

- *--help* on page 6-18
- *--version_number* on page 6-32.

6.33 -x

Extracts the files in *file_list* from the library. The contents of the library are not changed. If no file operands are given, all files in the library are extracted. If the filename of a file extracted from the library is longer than that supported in the destination directory, the results are undefined.

Be aware that if the name of a file in the library is longer than the file system can support, an error is displayed and the file is not extracted. To extract files with long filenames, use the `-T` option to truncate the names of files that are too long.

6.33.1 See also

Tasks

- *Extracting files from a library* on page 4-7.

Reference

- *-T* on page 6-29.

6.34 --zs

Displays the symbol table for all files in the library.

6.34.1 Example

To list the symbol table in a library, enter:

```
armar --zs mylib.a
```

6.34.2 See also

Tasks

- *Displaying the symbol table* on page 5-8.

Reference

- *-n* on page 6-21
- *-s* on page 6-25.

6.35 --zt

Lists both the member sizes and entry points for all files in the library. See `--sizes` and `--entries` for the output format.

6.35.1 See also

Tasks

- *Displaying member sizes and entry points for a library* on page 5-9.

Reference

- *--entries* on page 6-16
- *--sizes* on page 6-27.

Appendix A
Via File Syntax

This appendix describes the syntax of via files accepted by all the ARM development tools:

A.1 Overview of via files

Via files are plain text files that contain command-line arguments and options to ARM development tools.

Typically, you can use a via file to overcome the command-line length limitations. However, you might want to create multiple via files that:

- Group similar arguments and options together.
- Contain different sets of arguments and options to be used in different scenarios.

Note

In general, you can use a via file to specify any command-line option to a tool, including `--via`. This means that you can call multiple nested via files from within a via file.

A.1.1 Via file evaluation

When `armar` is invoked it:

1. Replaces the first specified `--via` *`via_file`* argument with the sequence of argument words extracted from the via file, including recursively processing any nested `--via` commands in the via file.
2. Processes any subsequent `--via` *`via_file`* arguments in the same way, in the order they are presented.

That is, via files are processed in the order you specify them, and each via file is processed completely including processing nested via files before processing the next via file.

A.2 Via file syntax

Via files must conform to the following syntax rules:

- A via file is a text file containing a sequence of words. Each word in the text file is converted into an argument string and passed to the tool.
- Words are separated by whitespace, or the end of a line, except in delimited strings. For example:

  `--debugonly --privacy` (two words)

  `--debugonly--privacy` (one word)
- The end of a line is treated as whitespace. For example:

  ```
  --debugonly
  --privacy
  ```

  is equivalent to:

  ```
  --debugonly --privacy
  ```
- Strings enclosed in quotation marks ("), or apostrophes (') are treated as a single word. Within a quoted word, an apostrophe is treated as an ordinary character. Within an apostrophe delimited word, a quotation mark is treated as an ordinary character.

  Use quotation marks to delimit filenames or path names that contain spaces. For example:

  `--output C:\My Project\output.txt` (three words)

  `--output "C:\My Project\output.txt"` (two words)

  Use apostrophes to delimit words that contain quotes.
- Characters enclosed in parentheses are treated as a single word. For example:

  `--option(x, y, z)` (one word)

  `--option (x, y, z)` (two words)
- Within quoted or apostrophe delimited strings, you can use a backslash (\) character to escape the quote, apostrophe, and backslash characters.
- A word that occurs immediately next to a delimited word is treated as a single word. For example:

  `--output"C:\Project\output.txt"`

  is treated as the single word:

  `--outputC:\Project\output.txt`
- Lines beginning with a semicolon (;) or a hash (#) character as the first nonwhitespace character are comment lines. If a semicolon or hash character appears anywhere else in a line, it is not treated as the start of a comment. For example:

  ```
  -o objectname.axf      ;this is not a comment
  ```

  A comment ends at the end of a line, or at the end of the file. There are no multi-line comments, and there are no part-line comments.