Application Note 49

Linker Output Formats

Document number: ARM DAI 0049A

Issued: January 1998

ENGLAND

Advanced RISC Machines Limited
Fulbourn Road
Cherry Hinton
Cambridge CB1 4JN
UK
Telephone: +44 1223 400400
Facsimile: +44 1223 400410
Email: info@arm.com

GERMANY

Advanced RISC Machines Limited
Otto-Hahn Str. 13b
85521 Ottobrunn-Riemerling
Munich
Germany
Telephone: +49 89 608 75545
Facsimile: +49 89 608 75599
Email: info@arm.com

JAPAN

Advanced RISC Machines K.K.
KSP West Bldg, 3F 300D, 3-2-1 Sakado
Takatsu-ku, Kawasaki-shi
Kanagawa
213 Japan
Telephone: +81 44 850 1301
Facsimile: +81 44 850 1308
Email: info@arm.com

USA

ARM USA Incorporated
Suite 5
985 University Avenue
Los Gatos
CA 95030 USA
Telephone: +1 408 399 5199
Facsimile: +1 408 399 8854
Email: info@arm.com

World Wide Web address: http://www.arm.com

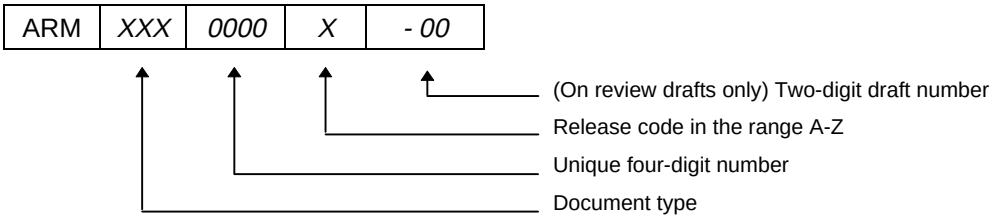
ARM
Advanced RISC Machines

Proprietary Notice

Key

Document Number

This document has a number which identifies it uniquely. The number is displayed on the front page and at the foot of each subsequent page.



Document Status

The document's status is displayed in a banner at the bottom of each page. This describes the document's confidentiality and its information status.

Confidentiality status is one of:

ARM Confidential	Distributable to ARM staff and NDA signatories only
Named Partner Confidential	Distributable to the above and to the staff of named partner companies only
Partner Confidential	Distributable within ARM and to staff of all partner companies
Open Access	No restriction on distribution

Information status is one of:

Advance	Information on a potential product
Preliminary	Current information on a product under development
Final	Complete information on a developed product

Change Log

Issue	Date	By	Change
A	January 1998	SKW	Released

Table of Contents

1 ARM Image Format (AIF)

ARM Image Format (AIF) is a simple format for ARM executable images, consisting of:

- a 128-byte header
- the image's code
- the image's initialized static data.

1.1 Requirements

In order to produce an AIF output there must be:

- no unresolved symbolic references between the input objects (each reference must resolve directly or via an input library)
- exactly one input object containing a program entry point (or no input area containing an entry point, and the entry point given using an `-Entry` option)
- either an absolute base address or the relocatable option given to armlink (the self-location option is system-dependent).

1.2 AIF flavors

Two flavors of AIF are supported.

1.2.1 Non-executable AIF

The header is not part of the image, but merely describes it. This variant is intended to be loaded by a program which interprets the header, and prepares the image following it for execution.

The fourth word of a non-executable AIF image is the offset of its entry point from its base address. The most-significant nibble of this word (in the target byte order) is 0x0.

The base address of a non-executable AIF image is the address at which its code should be loaded.

Non-executable AIF must be processed by an image loader, which loads the image at its load address and creates the zero-initialized data. The header is then discarded.

A special type of non-executable AIF is the Extended AIF, which contains a scatter loaded image. The image loader should place the load regions at the correct place in memory. For details of scatter loading, please refer to *Application Note 48: Scatter Loading* (ARM DAI 0048).

1.2.2 Executable AIF

An executable AIF file can be loaded at its load address (which may be arbitrary if the image is relocatable) and entered at the same point (at the first word of the AIF header). It prepares itself for execution by relocating itself and setting to zero its own zero-initialized data. After relocation and creation of the zero-initialized data, control passes to a branch to the image's entry point.

The header is part of the image. This variant can be executed by entering the header at its first word. The fourth word of an executable AIF header is:

```
BL entrypoint
```

The most-significant byte of this word (in the target byte order) is 0xEB.

The base address of an executable AIF image is the address at which its header should be loaded; its code starts at base + 0x80 (128).

At its most sophisticated, executable AIF can be considered to be a collection of envelopes which enwrap a plain binary image, as follows:

- The outer layer of wrapping deals with relocating the image to its load address. Two options are supported:
  - link-time relocation
  - load-time self-relocation to the address where the image has been loaded.

  In particular, an executable AIF image is capable of self-relocation.
- Once an AIF image has relocated itself, it can create its own zero-initialized area.
- Finally, the image is entered at the (unique) entry point found by armlink in the set of input object modules.

1.3 Linker command line options

`-aif`

Generates Executable AIF. This is the default format if no format is specified. The default load address is 0x8000 (32KB). Any other address (greater than 0x80) can be specified by using the `-base` option.

`-aif -relocatable`

Generates a relocatable Executable AIF image which self-relocates to the load address when entered. The default load address is 0x8000 (32KB). Any other address (greater than 0x80) can be specified by using the `-base` option.

`-bin-aif`

Generates a plain binary image preceded by an AIF header which describes it (Non-Executable AIF). This format is intended for use by simple loaders. Such an image cannot be executed by loading it at its load address and entering the first word: the AIF header must be discarded first, and the image entered at its entry point. Like plain binary images, the default load address is 0, but can be changed using the `-base` option. Note that the load address here is not the load address of the AIF header, but of the binary image contained within. A separate base address can be given for the image's data segment by using the `-RW` option, otherwise, by default, the data is linked immediately following the code. This option directly supports the images with code in ROM and data in RAM.

Note *For a detailed description of AIF, refer to the Software Development Toolkit Reference Guide (ARM DUI 0041),* ***Chapter 12 ARM Image Format****.*

2 Plain Binary Format

An image in plain binary format is a sequence of bytes to be loaded in memory at a known address. Armlink is not concerned with how the entry point address is communicated to the loader.

2.1 Requirements

In order to produce such an output, there must be

- no unresolved symbolic references between the input objects. Each reference must resolve directly or via an input library.
- an absolute base address.

2.2 Linker command line options

You can select plain binary output by using the `-bin` option.

2.3 Execution address

The default execution address for the image is 0. You can specify a non-zero base address using the `-base` option.

2.4 Notes

Scatter loading

If the binary file is part of a scatter loaded application, the zero-initialized areas are not present. The information generated by armlink enables the application to create its own zero-initialized data.

-split option

Using armlink's `-split` option tells armlink to output the read-only and read-write image sections to separate output files. It should be used only in conjunction with the `-bin` image type, and is meaningful only if separate read-only and read-write base addresses have been specified.

3 ARM Executable ELF (ELF)

Armlink can write image files in ARM Executable ELF format. ELF is an open standard for image files and object files supported by many third party tools vendors.

3.1 Linker command line options

You can select Executable ELF output by using the `-elf` option.

Note *For a detailed description of ARM Executable ELF, refer to the Software Development Toolkit Reference Guide (ARM DUI 0041):* ***Chapter 16 ELF File Format****.*

4 ARM Object Format (AOF)

AOF is a simple object format, similar in complexity and expressive power to the UNIX `a.out` format. It provides a generalized superset of `a.out`'s descriptive facilities.

Armlink supports AOF files both as input and output. When AOF is used as an output format, armlink:

- merges similarly named and attributed areas from its input object files
- performs PC-relative relocation between merged areas.

Unresolved references remain unresolved, and the output AOF file may be used as the input to a further link step.

4.1 Linker command line options

You can select AOF output by using the `-aof` option.

4.2 Usage

Armlink produces AOF output in partially linked form, which is suitable for inclusion in a subsequent link step.

Note *For a detailed description of AOF, refer to the Software Development Toolkit Reference Guide (ARM DUI 0041),* ***Chapter 14 ARM Object Format****.*