Topics

Figures

Notes

7 Archiver Description

The MSP430 family archiver lets you combine several individual files into a single file called an **archive** or a **library**. Each file within the archive is called a **member.** Once you have created an archive, you can use the archiver to add more files to the library, delete or replace existing members, or extract members.

You can build libraries out of any type of files. Both the assembler and the linker accept archive libraries as input; the assembler can use libraries that contain individual source files, and the linker can use libraries that contain individual object files.

One of the most useful applications of the archiver is to build a library of object modules. For example, you could write several arithmetic routines, assemble them, and then use the archiver to collect the object files into a single, logical group. You can then specify an object library as linker input. The linker will search through the library and include any members that resolve external references.

You can also use the archiver to build macro libraries. You can create several separate source files, each of which contains a single macro, and then use the archiver to collect these macros into a single, functional group. The .mlib assembler directive lets you specify the name of a macro library to the assembler; during the assembly process, the assembler will search the specified library for the macros that you call.

7.1 Archiver Development Flow

The figure shows the archiver's role in the assembly language development process. Both the assembler and the linker accept libraries as input.

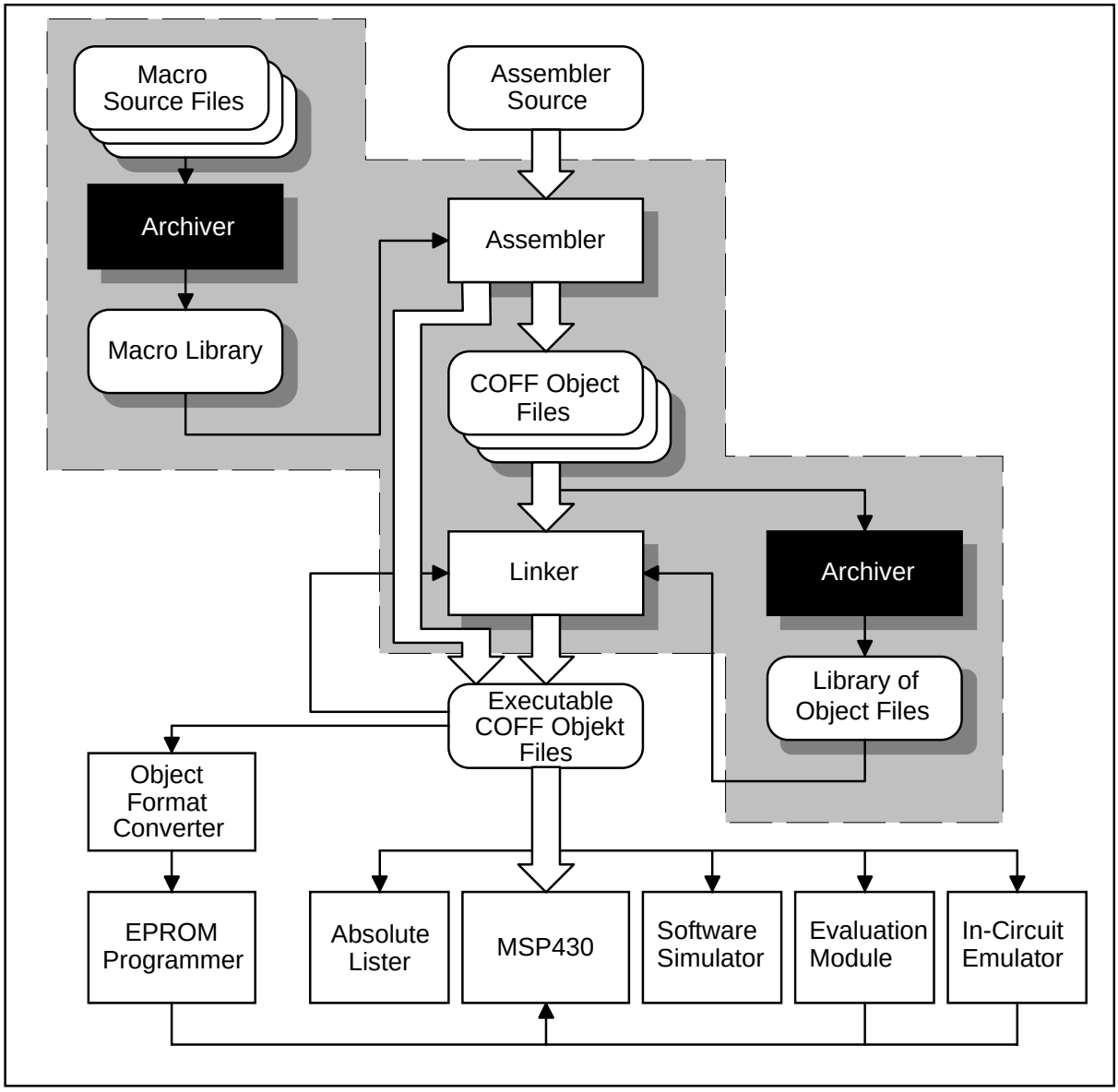

Figure 7.1: Archiver Development Flow

7.2 Invoking the Archiver

To invoke the archiver, enter:

ar430 *[-]command[option] libname [filename$_1$ ... filename$_n$]*

ar430 is the command that invokes the archiver.

libname names an archive library. If you don't specify an extension for *libname,* the archiver uses the default extension **.lib**.

filename names individual member files that are associated with the library. If you don't specify an extension for a *filename,* the archiver uses the default extension **.obj**.

command tells the archiver how to manipulate the members in the library. A command can be preceded by an optional hyphen. You **must** use one of the following commands when you invoke the archiver, but you can use only **one** command per invocation. These are valid archiver commands:

- **-a** adds the specified files to the library. Note that this command **does not replace** an existing member that has the same name as an added file; it simply *appends* new members to the end of the archive. It is possible to have several members with the same name in an archive. If you want to *replace* existing members, use the **r** command.
- **-d** deletes the specified members from the library.
- **-r** replaces the specified members in the library. If you don't specify any filenames, the archiver replaces the library members with files of the same name in the current directory. If the specified file is not found in the library, the archiver adds it instead of replacing it.
- **-t** prints a table of contents of the library. If you specify filenames, only those files are listed. If you don't specify any filenames, the archiver lists all the members in the specified library.
- **-x** extracts the specified files. If you don't specify any member names, the archiver extracts all the members in the library. When the archiver extracts a member, it simply copies the member into the current directory; it *doesn't* remove it from the library.

In addition to one of the *commands,* you can specify the following *options*:

- **-e** tells the archiver not to use the default extension .obj for member names. This allows the use of filenames without extensions.
- **-q** (quiet) suppresses the banner and status messages.
- **-s** prints a list of the global symbols that are defined in the library. (This option is valid only with the -a, -r, and -d commands.)
- **-v** (verbose) provides a file–by–file description of the creation of a new library from an old library and its constituent members.

Note: Naming Library Members

It is possible (but not desirable) for a library to contain several members with the same name. If you attempt to delete, replace, or extract a member, and the library contains more than one member with the specified name, the archiver deletes, replaces, or extracts the first member with that name.

7.3 Archiver Examples

Here are some examples of using the archiver.

- **Example 1**

  This example creates a library called function.lib that contains the files sine.obj, cos.obj, and flt.obj.

```
ar430 -a function sine cos flt
MSP430 Archiver                          Version 1.00
Copyright (c) 1994 Texas Instruments Incorporated
    ==>    new archive 'function.lib'
    ==>    building archive 'function.lib'
```

  Because these examples use the default extensions (.lib for the library and .obj for the members), it is not necessary to specify them.

- **Example 2**

  You can print a table of contents of function.lib with the -t option:

```
ar430 -t function
MSP430 Archiver                          Version 1.00
Copyright (c) 1994 Texas Instruments Incorporated
      FILE NAME    SIZE   DATE
-----------------  -----  ----------------------
      sine.obj     248    Mon Feb 14 01:25:44 1994
      cos.obj      248    Mon Feb 14 01:25:44 1994
      flt.obj      248    Mon Feb 14 01:25:44 1994
```

- **Example 3**

  You can explicitly specify extensions if you don't want the archiver to use the default extensions; for example:

```
ar430 -av function.fn sine.asm cos.asm flt.asm
MSP430 Archiver                          Version 1.00
Copyright (c) 1994 Texas Instruments Incorporated
    ==>     add 'sine.asm'
    ==>     add 'cos.asm'
    ==>     add 'flt.asm'
    ==>     building archive 'function.fn'
```

  This creates a library called function.fn that contains the files sine.asm, cos.asm, and flt.asm. (-v is the verbose option.)

- **Example 4**

  If you want to add new members to the library, specify

```
ar430 -as function tan.obj arctan.obj area.obj
MSP430 Archiver                          Version 1.00
Copyright (c) 1994 Texas Instruments Incorporated
    ==>     symbol defined: 'R2D2'
    ==>     symbol defined: 'Christmas'
    ==>     building archive 'function.lib'
```

  Because this example doesn't specify an extension for the libname, the archiver adds the files to the library called function.lib. If function.lib didn't exist, the archiver would create it. (The -s option tells the archiver to list the global symbols that are defined in the library.)

- **Example 5**

  If you want to modify a member in a library, you can extract it, edit it, and replace it. In this example, assume there's a library named macros.lib that contains the members push.asm, pop.asm, and swap.asm.

  ```
  ar430 -x macros push.asm
  ```

  The archiver makes a copy of push.asm and places it in the current directory; it doesn't remove push.asm from the library, though. Now you can edit the extracted file. To replace the copy of push.asm that's in the library with the edited copy, enter:

  ```
  ar430 -r macros push.asm
  ```