Cflink ColdFire Linker

rev. 2.26

Contents

2. General

The ColdFire Linker , cflink, is a freeware linker for ColdFire processors.

The job of the linker is to take all the object files required to make an executable program and join them together. During the linking process, it will match all the references between object files so that the final executable is complete.

2.1 Command Line Options

Usage: cflink [options] [files] -o [outputfilename]

Options:

-e<label>	Select xdef label name for embedded object mode.
-f<filename>	File containg command line information. If the file list is too long for a command line interface to handle, then place all information into a text file and use this option.
-g	Generate library file.
-i	Display build information.
-l<files>	Library files to include for routine linking.
-m	Display link map.
-r<hex address>	Generate binary ROM file. Address is a 32-bit start address for the ROM in hexadecimal.
-s<hex address>	Generate S-Records file. Address is a 32-bit start address for the ROM in hexadecimal.

2.2 Linking

The usual format for linking looks like this;

cflink startup.o myprogram.o -l nuos.lib -o myprogram.prg

where:-

cflink	- is the name of the linker.
startup.o	- is the name of the startup object code module for use with NuOS. You only need a file like this for program file generation.
myprogram.o	- is the name of your program object file. You can have more than one object file here.
nuos.lib	- is the general set of library functions. You can have more than one library file here.
myprogram.prg	- is the name of the program file.

The files you need to supply are that of the ‘myprogram.o’ which should have been produced from a Compiler or Assembler. All “.o” object files need to be in the NuOS object code format. If you used cfcc (ColdFire C Compiler) or cfasm (ColdFire Assembler - with the object output option enabled) then you would use the results from these programs.

Using the ROM option will produce only code and data binary output instead of the block file format. XREF (external references) and XDEF (external definitions) will be resolved, so that the code can be run in place.

2.3 Library File Generation

Library files are made from a set of object code files for use with the generation of program files. The linker will use these library files and extract the appropriate object code and data segments for inclusion into the program file. This is achieved by matching external definitions in the library file with the external references in the set of object files supplied.

Library files are also a simple way of reducing the number of object files in the command line.

The usual format for library generation looks like this;

```
cflink  -g file1.o  file2.o file3.o file4.o -o mylibrary.lib
```

Where;

cflink	- is the name of the linker program.
file1.o,file2.o, file3, file4.o	- are the names of the object code files to join together as a library.
mylibrary.lib	- is the name of the library file to create.

3. Errors

Error messages are meant to be self-explanatory. The general error messages that you may encounter are;

File not found — One of the required files could not be found

No command file to process
The file containing the command line options could not be found.

Unrecognised flag -<?>
The flag passed is not valid.

File <name> is not an valid object file
The file <name> is not a valid NuOS object code file.You should check the output options for the assembler or the name of the file you supplied.

Unknown data block type in <filename>
The object file contains a data block type that is not valid.

Unknown bss block type in <filename>
The object file contains a bss block type that is not valid.

Byte relative out of range for reference <name>
Word relative out of range for reference <name>
The variable <name> was used in a manner that puts it outside the range for either byte or word addressing modes. You need to change the model size for the source file.

Reference <name> not found
A reference was made to the variable <name> but it was not found in any of the object files. You need tocheck for correct spelling in your files, or that the library contains the reference.

Merged data >64k
This results because of an overflow in the small data model. All data combined from the object files must not exceed 65536 bytes in size. (Once the large data model is available, this may fix the problem). This does not refer to the size of the code, but rather the 'data' contained in each object file, such as tables or text. Code size can theoretically be as large as 2Gigabytes.

4. Limitations

1. The linker only supports the small data model. Code size has no restrictions, only the merged data size must be less than 64kbytes (i.e. All the data contained in each object file, not code, must add up to less than 64kbytes). You can circumvent this with assembler files by using the ‘section code’ option rather than ‘section data’ for large data items.

4.1 Known Bugs

There are no known program bugs for this current version.

If you find any program bugs, then please send reports to:-

bugs@austexsoftware.com

5. Changes

2.26 Fixed bug with basedata handling.
2.24 Fixed bug with s-records address handling.
2.22 Fixed bug with hexaddr conversion.
2.20 Added S-Record output support.
2.10 Program block output modified to allow allocation of memory for task spawning easier.
2.04 Bumped for new release.
2.02 Fixed handling of various relocation blocks.
2.00 Modified output format.
1.98 Library functions now working.
1.96 Started work on library functions.
1.94 Added processor block handling.
1.92 Modified block handling for new base register blocks.
1.90 Modified relocatable block handling.
1.86 Modified for multiple data blocks.
1.84 Changes to the object code format regarding relocations.
1.82 Modified data handling with rom mode.
1.80 Pre-Library support added.
1.40 Information output format modified.
1.30 Copyright name change.
1.28 Memory freeing sequence modified..
1.26 Linker now return()s error.
1.24 Fixed map output name corruption.
1.22 Enhanced XREF details on linking errors.
1.20 Finialised embedded object mode. Fixed maxfilesize calculations.
1.16 XDEF handling modified (requires at least v1.30 of cfasm). Fixed initialisation problem
1.10 Adding embedded object file option. Added map format output.
1.00 Possible problem in Link_REL16 now causes an error, rather than just show message.