in

COLLABORATORS

	TITLE : in	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 9, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

in

1.1 ConvFP Manual Title Page

```
         ConvFP: The Function Description/Pragma Conversion Utility

              Copyright © 1993 by Enchanted Blade Associates


                                USER MANUAL



                                 Continue
```

1.2 Table of Contents

1.3 Legal Distribution of ConvFP

Please select area of interest:

Distribution Information

Obtaining Updates

Bug Reports

1.4 Product Synopsis

ConvFP is a command line driven program that simplifies an assembly language programmer’s access to the Amiga’s library functions. ConvFP is capable of converting both function description (.fd) and pragmas (from the C language) into two types of assembly language include files: macro files that make it possible to call each function with a single line, and definition files that create LVO (library vector offset) equates to reduce or eliminate the time spent linking your program. It is a useful tool for both assembly programmers and anyone wishing to write custom libraries.

1.5 Product Installation Guide

To install ConvFP as a command in your C: directory, double-click on the install icon. ConvFP will then be available at any command line prompt by simply typing ConvFP and your desired arguments, and pressing RETURN.

1.6 Using ConvFP To Create Include Files

ConvFP is called from the shell and accepts a command line according to the following syntax:

```
ConvFP sourcefile[.fd|_pragmas.h] [destfile[.i]] [i|m|d|a]
```

The "sourcefile" parameter determines the function description or pragma file that will be converted. If the ".fd" or "_pragmas.h" file extension is left off, ConvFP first tries to read the file as given. If it is unable to do this, it will then attempt to open the same file with a ".fd" extension. If this fails, it will try a "_pragmas.h" file extension. If all of these attempts fail, it will give up and return a "Can’t open source file" error (Error 8). When the file type is not explicitly given (by adding the appropriate extension to the source file), ConvFP will examine the source file to try to determine the file type. If it is neither a function description or pragma file, you will get a "Can’t determine file type" (Error 4) error.

The "destfile" argument allows you to specify a destination file.

If the ".i" extension is left off it will be added automatically. If the argument is left out completely, it will be created from the source file's name by removing the existing .fd or pragma extension (if present) and appending ".i". If you choose the "a" mode parameter (see below), the destination files will be created by adding "_mac.i" to the macro file, and "_def.i" to the definition file.

The mode parameter is one of four letters which indicates how you would like the file processed:

```
   "i" activates information mode, which displays all functions
       described in the file and a few other pieces of information.
       This is the default option, and it is useful if you wish to
       syntax check a .fd or pragma file.
   "m" specifies macro file mode.  The destination file will contain
       a series of function macros, one for each library call.
   "d" specifies definition file mode.  In this mode, a LVO equate is
       generated for each function.  These can then be included to
       reduce or eliminate the linking stage of assembly.
   "a" activates all modes simultaneously.
```

Using a question mark ("?") as the only argument will give a summary of the above information. Note that pattern matching is not supported in this version of ConvFP, nor is the use of files with spaces enclosed in quotes. Information on obtaining the latest version is in the distribution section.

Pressing Ctrl-C at any time during ConvFP's operation will abort the current file processing operation.

1.7 Using ConvFP Include Files In Your Programs

ConvFP has four main uses, all of which will be described in this section. In summary, these are:

- A .fd/pragma file decoder and analyzer
- A syntax checker capable of finding most .fd/pragma file errors
- An assembly language library function macro generator, and
- An assembly language Library Vector Offset EQUate generator.

The exact structure of the macro and equate files can be found in Appendices B and C. For best results, always use .fd files instead of pragma files as sources whenever possible.

File Decoding: To decode a .fd or pragma file, simply use the source file as an argument (thus using the default information mode). The source will be read and parsed, and you will get information about each function (including register use) as it is decoded. Private functions will be marked with "(PRIVATE)" after the function name. ConvFP also determines and displays the bias (LVO of the first function), proper library base according to RKRM conventions, and the total number of lines, comments, and functions in the file.

Syntax Checking: If you are trying to create your own .fd or pragma file (if you are writing your own library, for example), ConvFP can be used as a fairly reliable syntax checker. To use ConvFP in this manner, execute ConvFP with the file extension (i.e. ".fd" or "_pragmas.h") explicitly stated as part of the filename. In this way, ConvFP will not attempt to determine the file type, and thus will complain if you accidently use a pragma instruction in a .fd file, or vice-versa.

Function Macros: When called with the "m" or "a" modes, ConvFP will create an assembly language include file that contains macros to call any of the functions described by the source file. The macro names are created by adding a single underscore ("_") to the function name, to avoid conflicts. The parameters supplied to the macro are loaded into the proper registers, using move.l or movea.l instructions as appropriate, and the function is called. All macros assume that the library base is located at a label called "_[basename]", where [basename] is the conventional basename variable for that library as described in the RKRM Libraries tome (for example, _GfxBase for graphics.library). As an example, the dos.library function CreateProc looks like this: CreateProc(name,pri,segList,stackSize). To use a macro created with ConvFP, you would first open dos.library and put its address at _DOSBase. You would then include the macro file, and call the macro when needed using "CreateProc_ name,pri,segList, stackSize", replacing the variables with whatever values you wanted. After the macro's code was finished, d0 would contain the result. If the macros are generated from .fd files, the macro include will put a comment with a description of the parameters used on each function. If a function uses more than nine paramters, it must be split into two macros, since most assemblers only support parameters from \1 to \9. To use such macros, call the macro normally (use [function name]_) with the first nine parameters, and then call the second part with the remaining parameters using [function name]_2.

LVO Equates: It can often be a pain to make the numerous XREF's to various function calls to obtain their LVOs, and then have to wait each time you assemble the program while the linker fills in the values from amiga.lib. Instead, using mode d or a, you create a file containing EQUate instructions for each function in a given library, and then include the file. The equates are generated in the standard LVO_[function name] form to maintain compatibility with existing programs and routines. Note that the macros created with the macro function of ConvFP do not need the LVO's because they use the values directly. Should the LVO's of a particular library change (not likely to happen), simply use ConvFP on the new .fd or _pragma.h file and the macros will again work perfectly.

Notes: Neither the macro files nor the LVO definition files will include lines for functions marked private by an .fd file. Most assemblers let you create "preassembled" versions of macro and equate files, thus saving space and time. Check the manual of your assembler.

1.8 Appendix A: Error Explanations

There are two types of errors you may encounter while using ConvFP: Errors and Warnings. Errors are fatal; processing of the file ends when the error occurs. On the other hand, warnings do not inhibit file conversion, they merely indicate unusual situations you should

```
be aware of.  If the error occurs during the processing of the file,
the line number of the error and the line itself will be displayed
along with the error message.  The following index describes all the
errors and warnings by number.

ERROR(1)  No arguments found: ConvFP requires at least one argument to
          be present: the name of the source file.

ERROR(2)  Extraneous arguments: ConvFP has found additional arguments
          on the command line.  This can occur if one of the filenames
          contains a space as well.

ERROR(3)  Unrecognized mode parameter: A mode was specified on the
          command line that is not one of the four legal codes: i,
          m, d, and a.  The mode must always be given last.

ERROR(4)  Can’t determine file type: The file type was not explicitly
          stated on the command line, and ConvFP has examined the
          file in order to determine the type from its structure.
          When this error occurs, ConvFP has examined the entire file
          and could not find commands of either type.  This means that
          either the file is "empty" and contains no actual functions,
          or that it is neither a .fd or pragma file.

ERROR(5)  Can’t open destination macro file: Either mode "m" or mode
          "a" has been activated, but ConvFP was unable to open the
          destination file.

ERROR(6)  Can’t open destination def’n file: Either mode "d" or mode
          "i" has been activated, but ConvFP was unable to open the
          destination file.

ERROR(7)  Syntax error in source file: Some error has been detected
          in the structure of the source file and it must be corrected
          before ConvFP can convert it.

ERROR(8)  Can’t open source file: For some reason, ConvFP was unable
          to open the source file.  Check the spelling and the path.
          Note that ConvFP can detect and open a file with a ".fd"
          or "_pragmas.h" extension automatically without it being
          given on the command line.  For example, given the source
          specification "sc:include/pragmas/asl", ConvFP will
          correctly open "sc:include/pragmas/asl_pragmas.h".

ERROR(9)  Function defined before bias determined: This indicates an
          error within a function description file.  A proper .fd
          file contains a ##bias instruction that determines what
          the library vector offset of the first function is
          (normally 30).  When a function description occurs before
          this initial bias is defined, ConvFP cannot correctly
          calculate the offset.

ERROR(10) Function defined before library base determined: This is
          similar to error 10, except that it revolves around the
          ##base instruction, which determines which library the
          functions are in.  When a function is found before the
          base is known, ConvFP cannot use the correct library.
```

```
ERROR(11) Function missing paramter description: This is a specific
          type of syntax error that indicates that the paramter list
          for a function is missing.

ERROR(12) 100 sequential null lines without end of file: Some filing
          systems do not correctly indicate the end of a file. If 100
          empty lines are read in a row, ConvFP will assume that this
          is why and stop reading the file.  The destination file(s),
          if any, should still work normally.


WARNING(1) Unrecognized instruction ignored: In .fd files, a "##"
           sequence indicates a command.  ConvFP understands all
           currently used commands, so this may in fact indicate
           an error in the file.  If new commands are added in
           the future, they are ignored.  Pragma files can contain
           'C' instructions; these will also give this warning.

WARNING(2) A6/A7 declared as parameter: Normally, this shouldn't
           happen, but if it does for whatever reason, ConvFP
           will still continue with the conversion.

WARNING(3) Unexpected end of source file: A proper .fd file is
           terminated with a "##end" instruction.  ConvFP can
           still correctly handle a file that does not end
           correctly, but most other .fd related utilities cannot.

WARNING(4) More than nine paramters; made two macros: Most assemblers
           with macro capabilities only allow nine parameters (\1 to
           \9).  When a function call with more than nine paramters
           is converted into a macro, it is divided into two. The
           first macro (named [function name]_) should be called with
           the first nine paramters, followed immediately by the
           next macro (named [function name]_2) which is given the
           remaining parameters and actually calls the function.
```

1.9 Appendix B: Macro File Structure

The macro files created by ConvFP follow a similar skeletal outline. This structure begins with a brief header stating the source and ConvFP version. It is a good idea to keep this header intact, since it will quickly determine when and where a source file came from. If, for example, a bug is found and corrected in a later version of ConvFP, you will quickly be able to see if a given file needs updating, and if so, what source file is needed to do it.

The header is immediately followed by a macro (or two) for each function described in the source file. Taglist and tagcall versions of a library function are ignored by ConvFP, and only the standard libcall version is converted. Each macro is followed by a single blank line, except for split macros created when a function has more than nine parameters. The format of these macros is as follows: the MACRO command, followed by the name of the function with "_" appended, possibly followed by a list of what each parameter is for if the source file was a .fd file and the function has any parameters. This is then followed by a single move.l or movea.l instruction for each parameter, in the proper calling order, that puts the parameter given when the macro is expanded into the correct register. The library base is then moved (using movea.l) into a6 from a label created by prefixing the library basename with a "_". The function is then called by JSRing to the correct offset. This offset is hardcoded into the macro. If there are more than nine parameters, a similar approach is used, but the first macro only loads the first nine parameters. The second fills in the remaining parameters and calls the function. Graphically, this looks like the following:

```
*
* Library macros for [source file]
*
* Created using ConvFP [version] by C. Jennings
*
 MACRO FirstFunction_   [;(param1,param2...)]
  move[a].l \1,[register]
  move[a].l \2,[register]
  ...
  movea.l _[lib base],a6
  jsr [offset](a6)
 ENDM

 MACRO LongFunction_   [;(p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11)]
  move[a].l \1,[register]
  ...
  move[a].l \9,[register]
 ENDM
 MACRO LongFunction_2
  move[a].l \1,[register]
  move[a].l \2,[register]
  movea.l _[lib base],a6
  jsr [offset](a6)
 ENDM

 ...
```

1.10 Appendix C: LVO Definition File Structure

The structure of the definition files is quite simple. It starts with a header (see the macro structure section for more information), and then proceeds with an equate of the form LVO_[function] EQU [offset] for each function in the source file. Graphically, this is as follows:

```
*
* LVO definition file for [source file]
*
* Created using ConvFP [version] by C. Jennings
*
LVO_[function 1]                                     EQU -[off]
LVO_[function 2]                                     EQU -[off]
...
LVO_[function n]                                     EQU -[off]
```

1.11 ConvFP Index

1.12 Distribution of ConvFP

1.13 Obtaining the Latest Version

The latest version can be obtained directly from the author for the small fee of $15 Canadian (basically enough to cover the cost of buying disks, envelopes, stamps, and so on). This "registered" version will include the latest version (whatever that may be) and a GUI frontend for the program, and enough other "goodies" to fill up a disk. This program is freely distributable; it is not shareware. There is no obligation to send money. You only need to send anything if you want the latest version, the GUI frontend, and other software as described above.

To order the next available update, send cash (a bit risky) or a money order for $15, payable to Enchanted Blade Associates, to the following address*:

Enchanted Blade Associates
R4 Langton, ON
N0E 1G0
Canada

(519) 875-2137

Enchanted Blade Associates can also be reached via E-Mail:

lynnjenn@vef.north.net

For fastest (and cheapest) response to questions and comments, send email with "CONVFP" in the subject header.

* Be sure to enclose the following along with your payment: your name and mailing address, the version number of the version of ConvFP you currently have, and any suggestions and bug reports you might wish to make.

THIS IS NOT SHAREWARE; THERE IS NO OBLIGATION TO SEND MONEY. THE MONEY IS ONLY TO COVER MY COSTS AND IS ONLY IF YOU WANT AN UPDATED VERSION OF THE SOFTWARE.

1.14 Bug Reports

If you find a bug in ConvFP, please write me a letter and tell me about it. Please give me as much information as possible, and at least a description of the problem and how to recreate it. If possible, send a disk with the files or source code that seem to be causing the problem. Tell me the version of ConvFP you are using. Please send all bug reports to:

```
                      Attn: ConvFP Bugs
                      Enchanted Blade Associates
                      R4 Langton, ON
                      N0E 1G0
                      Canada

                      (519) 875-2137

Or via E-Mail to:    lynnjenn@vef.north.net
```

For fastest (and cheapest) response to questions and comments, send email with "CONVFP" in the title.