
FORTRAN 77

reference manual

ARM Evaluation System

Acorn OEM Products



FORTRAN 77

Part No 0448,013
Issue No 1.0
15 August 1986

© Copyright Acorn Computers Limited 1986

Neither the whole nor any part of the information contained in, or the product described in, this manual may be adapted or reproduced in any material form except with the prior written permission of the copyright holder. The only exceptions are as provided for by the Copyright (photocopying) Act, or for the purpose of review, or in order for the software herein to be entered into a computer for the sole use of the owner of this book.

Within this publication the term 'BBC' is used as an abbreviation for 'British Broadcasting Corporation'.

- The manual is provided on an 'as is' basis except for warranties described in the software licence agreement if provided.
- The software and this manual are protected by Trade secret and Copyright laws.

The product described in this manual is subject to continuous developments and improvements. All particulars of the product and its use (including the information in this manual) are given by Acorn Computers in good faith.

There are no warranties implied or expressed including but not limited to implied warranties or merchantability or fitness for purpose and all such warranties are expressly and specifically disclaimed.

In case of difficulty please contact your supplier. Every step is taken to ensure that the quality of software and documentation is as high as possible. However, it should be noted that software cannot be written to be completely free of errors. To help Acorn rectify future versions, suspected deficiencies in software and documentation, unless notified otherwise, should be notified in writing to the following address:

Customer Services Department,
Acorn Computers Limited,
645 Newmarket Road,
Cambridge
CB5 8PD

All maintenance and service on the product must be carried out by Acorn Computers. Acorn Computers can accept no liability whatsoever for any loss, indirect or consequential damages, even if Acorn has been advised of the possibility of such damage or even if caused by service or maintenance by unauthorised personnel. This manual is intended only to assist the reader in the use of the product, and therefore Acorn Computers shall not be liable for any loss or damage whatsoever arising from the use of any information or particulars in, or any error or omission in, this manual, or any incorrect use of the product.

Econet® and The Tube® are registered trademarks of Acorn Computers Limited.

ISBN 1 85250 008

Published by:

Acorn Computers Limited, Fulbourn Road, Cherry Hinton, Cambridge CB1 4JN, UK

Contents

1. Introduction	1
1.1 Installation	1
2. The compiler	2
2.1 Compilation arguments	2
2.1.1 Example compiler commands	5
2.2 Compilation options	6
2.3 Compiling in separate stages	7
2.3.1 Front end	8
2.3.2 Code generator	10
2.4 Linking and execution	12
3. Extensions to the standard	13
3.1 Hexadecimal constants	13
3.2 FORTRAN 66 option	13
3.3 Naming	14
3.4 Loops	15
3.4.1 WHILE ... ENDWHILE	15
3.4.2 DO WHILE	15
3.4.3 Block DO	15
3.5 Random number generators	16
3.6 Include statement	16
4. Input/output	17
4.1 Unit numbers and files	17
4.2 Sequential files	18
4.2.1 Opens and closes	18
4.2.2 Formatted IO	19
4.2.3 Unformatted IO	21
4.3 Direct access files	21
4.4 OPEN and CLOSE	23
4.5 INQUIRE	23
4.5.1 INQUIRE by unit	23
4.5.2 INQUIRE by file	23
4.6 BACKSPACE	23
4.7 ENDFILE	23
4.8 REWIND	24
4.9 FORMAT decoding	24
4.9.1 Lower case letters	24

4.9.2 Extraneous repeat counts	24
4.9.3 Edit descriptor separators	24
4.9.4 Numeric edit descriptors	24
4.9.5 A editing	25
4.9.6 Abbreviations and synonyms	25
4.9.7 Transfer of numeric items	26
5. Errors and debugging	27
5.1 Front end error messages	27
5.2 Warning messages	28
5.3 Code-generator error messages	28
5.4 Code generator limits	29
5.5 Run-time errors	29
5.5.1 Code 1000 errors	30
5.6 Array and substring errors	31
5.7 Input/output errors	31
5.8 Tracing	31
6. Appendix A	34
6.1 Code-generator error messages	34
7. Appendix B	37
7.1 Input/output errors	40



1. Introduction

FORTRAN has long been regarded as the programming language most suited to scientific and numeric applications. FORTRAN 77 is the latest standardised version of the language, and this has been used in the production of Acorn FORTRAN 77. This manual describes the use of Acorn FORTRAN 77 running under ARM Executive ; note that it is not a tutorial.

The Acorn FORTRAN 77 compiler has been fully validated in conformance with the American National Standard Programming Language FORTRAN X3.9 -1978 (ANS FORTRAN). Detailed language specifications are given in the publication *American National Standard Programming Language FORTRAN, X3.9-1978*, which is available from the British Standards Institute. Less technical approaches are provided in *A Structured Approach to FORTRAN 77 Programming* by T M R Ellis, published by Addison Wesley, and *A Pocket Guide to FORTRAN 77* by Clive Page, published by Pitman.

From now on, unless otherwise stated, or made obvious from the context, 'FORTRAN 77' is taken to mean the implementation of FORTRAN 77 on Acorn ARM computers.

If you want to use the floating point emulator, refer to the *ARM UTILITIES reference manual*.

1.1 Installation

FORTRAN 77 for the ARM is distributed on ADFS format floppy discs.

2. The compiler

The FORTRAN 77 compiler is made up of two parts: a front end which checks that the source code conforms to the standard, and a code generator which creates the equivalent machine code program. This program is in Acorn object format (AOF) and is linked into an executable form using a linker program. A single command file is normally used to run both parts of the compiler and the linker. There are a number of arguments which can be issued to the compiler to give extra control over the compilation, and allow the compilation options to be used.

The command `f77` executes a command file which runs the two parts of the compiler and the linker in sequence, and so compiles the program without the need for the user to give three separate commands. It also informs the user how to find out help information from the front end and code generator, and accepts the compiler arguments.

A help file called `README` on the Fortran distribution floppy disc, contains details about the file structure needed for the `f77` command to work as described in this manual.

You can write your own command files (see *ARM UTILITIES reference guide*) for running the Fortran compiler in a different way, using other file structures. If you do this, it is advisable to use a different command name and leave `f77` consistent with the examples in this manual.

There is also a command `f77q`, which compiles more quickly because it does not produce program listings and maps. The map gives the name, type and location of local and common variables in each program unit.

To run the two parts of the compiler and the linker separately, the command `f77fe` is used for the front end, `f77cg` is used to generate the machine code, and `link` is used to link AOF files into relocatable image files (`rif`).

2.1 Compilation arguments

The behaviour of the compiler can be modified through the use of compilation arguments. These can be used to specify input and output files, listings, identification, and also the compilation options, which are a type of

argument specifying lower-level compiler activity.

The form of the command line is as follows:

```
f77 (-source) name { (-to name) (-list name)
      (-error name) (-opt options) (-map name)
      (-help) (-m) (-link name) (-library name)
      (-image name) }
```

where name stands for a user-supplied filename, and options represents a list of one or more compiler options. Braces enclose optional items. The arguments can be given in any order. Explanations of each follow:

-source name

The source file is the only argument which is not optional (although the keyword `-source` is). It specifies the name of the file which contains the code to be compiled. The file must be in directory `$.f77_files.source`.

-list {name}

Unless the quick version of the compiler command file, `f77q`, is used, a listing of the compiled program along with line numbers of the source is generated. This listing can be sent to a file or device specified in the argument (for example, `printer`). If no filename or device is specified, then the compiler sends the listing to a file whose name is based on the source filename, for example, a file called `Fprog` in the source directory will have a listing file called `$.f77_files.list.Fprog`.

-opt options

Several options are accepted by the compiler. These are given in the `opt` argument. The options available are listed below under the heading `Compilation options`.

-error name

Compiler error messages are sent to the vdu by default, but may be re-directed using the `error` argument to a specified file or device.

-to name

The Acorn Object Format output file of the compiler is given a name based upon the source filename by default that is, a file called `fort1` in the source directory will be given the AOF name `$.f77_files.AOF.fort1`. The `-to` argument can be used to specify an alternative name.

-map name

Unless the quick version of the compiler command file, `f77q`, is used, a storage map of the compilation is produced by the code generator and sent to the filename specified. This will be put into the directory `$.f77_files.map`.

-help

The help argument displays a reminder of all the other arguments you can use with the `f77` command. The reminder is a list of keywords as used at the start of command files (see *ARM UTILITIES reference guide*); `/a` indicates that an argument is compulsory, `/k` that it is optional. For explanations of the use of the arguments, refer to the manual (this section), as none is given in response to use of the help argument.

-m n

The module limit option is used to define the limit on the number of program units (main program and subprograms) in a compilation. A limit is necessary because of the chunk file structure of object files. The default limit is 10. To set the limit to 15, use the option `-m 15`.

-link name

If you use the `-to` argument to rename the AOF file, then the link argument must be used to direct the linker to take its input from the AOF file.

library name

This argument is used to change from the default run-time library, and give the compiler the name of the file to use instead (see *ARM UTILITIES reference manual*).

-image name

The image file contains the final executable output from the linker. The image argument is used to send the output to a file with a different name from the default one.

You cannot change the name of the file for the output from the front end of the compiler. It always has the name of the source file and is in directory `$.f77_files.fcode`.

2.1.1 Example compiler commands

The minimal command

```
f77 Fprog
```

This command compiles the source program Fprog, with default directories. The compiler expects the source code to be in file \$.f77_files.source.Fprog'. The output will be directed to the following files:

```
the fcode is in $.f77_files.fcode.Fprog
the AOF code is in $.f77_files.AOF.Fprog
the program listing is in $.f77_files.list.Fprog
the map is in $.f77_files.map.Fprog
the image code is in $.f77_files.image.Fprog
```

Error messages are directed to the VDU, and default compilation options are used (see section 2.2).

If the quick compiling command:

```
f77q Fprog
```

is used, the result is the same except that no list or map file is produced.

Specifying the input and output files

```
f77 -source Fprog -to $.Afiles.Fprog -list $.f77_files.list.proglist
-link $.Afiles.Fprog -error printer
```

The compiler front end expects the source code to be in file\$.f77_files.source.Fprog. The AOF code is in \$.Afiles.Fprog, so the linker has to be redirected to find its input in \$.Afiles.Fprog. The rest of the output is directed to the default files as in the minimal command, except for the program listing, which is in \$.f77_files.list.proglist.

C. Using some options and specifying a map file:

```
f77 Fprog -opt +tW0 -map $.map.FFprog
```

The program Fprog' is compiled, with tracing specified in the code, warning messages inhibited, and a storage map sent to the file \$.f77_files.map.Fprog'.

2.2 Compilation options

The `-opt` argument is followed by a list of compilation options (in upper or lower case).

The options B, H, T, 6 and 7 are enabled or disabled by preceding them with + or -. The options X, L, M and W must be followed by a number. The default for the full set of options is:

```
L1W2X0 -BHT6
```

This means that code generator line numbering is set to level 1; level 2 warning messages are given; there is no cross-referencing output, no bound checking, and Hollerith constants are not allowed; tracing and FORTRAN 66 are disabled.

B

Causes the compiler to generate bounds checking code. Array or substrings subscripts out of range will cause run-time errors to be reported in programs compiled with this option.

H

When enabled, this option allows Hollerith constants to be used in DATA statements to initialise non-character variables (for example, INTEGER).

L n

This option is followed by a number which indicates the level of line numbering included in the code for backtrace purposes (see chapter 5). The levels available are:

- 0 no line numbering
- 1 numbers lines containing subprogram CALLS
- 2 statements which can cause a run-time exception
- >2 numbers every line.

Higher levels cause more code to be generated. If a hardware exception occurs in a module compiled with level 1, the backtrace system will not be able to determine the exact line number; instead, a range of numbers will be given (for example, 100/106). The error will lie in this range.

M n

This sets the module limit. (10)

T

This causes the compiler to plant tracing code in the output file (see chapter 5).

W n

This sets the warning message level. A following digit of 0-4 is interpreted from the zero level, as suppress all warnings' to print all warnings' (level 4). See chapter 5 for more details.

X n

This is followed by a cross-reference listing width (of 18 or more for maximum legibility). A value of zero suppresses cross-referencing. The upper limit depends upon the device to which the listing is being sent (for example, printer). Cross-reference information is given immediately after the END statement of a program unit. For each name, the type is given, together with the lines on which it is referenced. For each statement label, the type (executable or non-executable) and line number of the statement is given, as well as the lines on which the label is referenced.

6

This option allows FORTRAN 66 features to be used; if enabled, it implies the H option.

7

This option is used to control warnings about the use of FORTRAN 77 language extensions. If unset, warnings are not produced; otherwise messages are produced if the warning level (Wn) is 2 (the default) or greater. The option is unset by default, so that the extensions may be used without messages, whatever the warning level.

2.3 Compiling in separate stages

As an alternative to using the *f77* command to execute a command file that runs both parts of the compiler, the two parts can be run separately. This section gives details of how to run the front end and the code generator separately.

2.3.1 Front end

The FORTRAN 77 front end accepts the full language as defined in the ANSI standard, together with a number of extensions. Upper- and lower-case letters are equivalent in all contexts, and are always converted to upper case in the FCODE output (for example, `::x=abc(y)::` and `::X=ABC(y)::` would both refer to the function `::ABC::`). Identifiers may contain up to 255 characters.

Command format

The front end is a BCPL command with the following argument string:

`"FROM/A, TO=FCODE/K, LIST/K, OPT/K, VER/K"`

- FROM** FORTRAN 77 source program.
- TO** FCODE output file. If this argument is not quoted, no FCODE is produced.
- LIST** Listing file. If LIST is quoted, a listing of the source program with line numbers is sent to the file, together with any error messages. Otherwise error messages are sent to the initial output stream, and no listing is produced.
- OPT** Front end option string. The options available are described below.
- VER** Output file for compiler messages and errors; if omitted output is to the terminal.

Options

- +** turn following options on
- turn following options off
- T** include tracing
- Xn** width of cross-reference output (no output if width is 0).
- Wn** Set warning level
- 7** Strict FORTRAN 77 mode
- 6** FORTRAN 66 mode

The default settings are X0W2-T67.

The `+T` switch causes the front end to embed calls to special trace routines at various points in the program, such as program unit entry, DO statements, labelled executable statements and subprogram calls. See the later section on tracing for more details.

The +w option controls the production of warning messages: level 0 suppresses them, level 1 permits the most significant, and 2, 3 and 4 allow gradually less serious warnings to be produced.

The +x option controls the width in which cross-reference information is written, with a width of zero causing the output to be suppressed. Cross-reference information is given immediately after the END statement of a program unit. For each name, the type is given, together with the lines on which it is referred to. For each statement label, the type (executable or non-executable) and line number of the statement is given, as well as the lines on which the label is referred to.

The FORTRAN 66 switch (+6) is intended to help in the compilation of programs written in the previous version of FORTRAN. When set, most constructs which have different meanings in the two versions are interpreted according to the FORTRAN 66 definition. In particular:

- (a) DO loops will always execute at least once
- (b) Hollerith (nH) constants are allowed in DATA and CALL statements, and quoted constants in calls are not of CHARACTER type.
- (c) Non-CHARACTER array names are allowed as format specifiers.

The strict FORTRAN 77 option (+7) is used to control warnings about language extensions. If unset, warnings are not produced; otherwise messages are produced if the warning level (Wn) is 2 (the default) or greater. The +7 option is unset by default, so that the extensions may be used without messages, whatever the warning level.

When the FORTRAN 66 switch is used, Hollerith and quoted constants are treated in the same way when used as arguments in CALLs - they are not of CHARACTER type. The option is provided for use with FORTRAN 66 programs which store character information in numeric data types.

For example, the following calls will have identical effects at run time if the FORTRAN 66 switch is used:

```
call jim('abcd')  
call jim(4habcd)
```

The +6 option must also be used with the code generator if it is wished to use Hollerith constants in DATA statements.

If the FORTRAN 66 switch is set, run-time FORMATs specifiers may also be non-CHARACTER array names.

For example:

```
double precision d(3),num
data d(1),d(3)/8h(1X,D20., 5h,I5//)
data num /2h10/
...
d(2) = num
...
write(6, d) 2.3d0, 10
...
```

This facility was introduced to assist in the implementation of FORTRAN 66 programs; it is strongly recommended that new programs use CHARACTER formats.

Examples

```
f77fe f77.prog -to tmp.fcode
```

Compile source program in *f77.prog* to FCODE in *tmp.fcode*.

```
f77fe f77.prog -ver x
```

Compile *f77.prog*, producing no FCODE output, with messages sent to the file *x*.

```
f77fe f77.prog -to tmp.fcode -list list.prog
```

Compile as before, but also send source listing to the file *prog* in directory *list*.

```
f77fe f77.prog -to tmp.fcode -opt t
```

Compile with tracing calls included.

2.3.2 Code generator

The code generator takes an FCODE file and produces an object file and/or assembler output. It requires a BCPL system with at least 800 globals, and access to the floating point instruction set for certain operations involving REAL and DOUBLE PRECISION constants.

Command line

The code generator is a BCPL command with the following argument string:

```
"FCODE/A, OBJ=TO/K, ASM=A/K, VER/K, OPT/K, MAP/K"
```

- FCODE** FCODE input file.
- TO** Object output file. If this argument is not quoted, no object file is produced. The object file is in an AOF file, and may be merged with other AOF files using the linker Link to produce a large compiled program (see section 2.4).
- ASM** Assembler output file. If this argument is quoted, a disassembled version of the object code is sent to the file.
- VER** Output file for code generator messages and errors; if omitted output is to the terminal.
- OPT** Option string. The options available are described below.
- MAP** The file used for the code generator map output. The MAP gives the name, type and location of local and COMMON variables in each program unit. The location is relative to the start of the static area for a local variable and is the offset in the block for a COMMON variable. The offset of each statement number from the start of the code is also given.

Options

- + turn following options on
- turn following options off
- B insert code for array and substring bound checking
- 6 or H allow Hollerith constants in DATA statements
- Ln set line number level.
- Mn set module limit (10).

The default options are L1-B6. The front end options T, 7, W and X are ignored by the code generator, whilst the front end ignores B and L, so that the same option string may be given to both programs, if required (except for the M option – this should be specified separately).

The line number option (+L) is used to control the amount of line number information included in the code. The possible levels are: