

# FORTRAN 77



REFERENCE  
MANUAL



# FORTRAN 77



PART NO 0410, 008  
ISSUE NO 1  
JULY 1985

© Copyright Acorn Computers Limited 1985

Neither the whole or any part of the information contained in, or the product described in, this manual may be reproduced in any material form except with the prior written approval of Acorn Computers Limited (Acorn Computers).

The product described in this manual and products for use with it, are subject to continuous developments and improvement. All information of a technical nature and particulars of the product and its use (including the information in this manual) are given by Acorn Computers in good faith.

In case of difficulty please contact your supplier. Deficiencies in software and documentation should be notified in writing, using the Acorn Scientific Fault Report Form to the following address:

Sales Department  
Scientific Division  
Acorn Computers Ltd  
Fulbourn Road  
Cherry Hinton  
Cambridge  
CB1 4JN

All maintenance and service on the product must be carried out by Acorn Computers' authorised agents. Acorn Computers can accept no liability whatsoever for any loss or damage 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.

Published by Acorn Computers Limited,  
Fulbourn Road, Cherry Hinton, Cambridge CB1 4JN.

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

**NOTE:** A User Registration Card is supplied with the hardware. It is in your interest to complete and return the card. Please notify Acorn Scientific at the above address if this card is missing.

ISBN 0 907876 41 2 Acorn Scientific

# Contents

1	Introduction to 32000 FORTRAN 77	1
1.1	Installation	1
2	The Compiler	3
2.1	Compilation arguments	3
2.2	Compilation options	5
3	Extensions to the standard	9
3.1	Hexadecimal constants	9
3.2	FORTRAN 66 option	9
3.3	Naming	10
4	Input/output	11
4.1	Unit numbers and files	11
4.2	Sequential files	12
4.2.1	Opens and closes	12
4.2.2	Formatted IO	13
4.2.3	Unformatted IO	15
4.3	Direct Access files	16
4.4	OPEN and CLOSE	17
4.5	INQUIRE	17
4.5.1	INQUIRE by unit	17
4.5.2	INQUIRE by file	18
4.6	BACKSPACE	18
4.7	ENDFILE	18
4.8	REWIND	18
4.9	FORMAT decoding	18
4.9.1	Lower case letters	18
4.9.2	Extraneous repeat counts	19
4.9.3	Edit descriptor separators	19
4.9.4	Numeric edit descriptors	19
4.9.5	A editing	20
4.9.6	Abbreviations and synonyms	20
4.9.7	Transfer of numeric items	20
5	Errors and debugging	21
5.1	Front end error messages	21
5.2	Warning messages	22
5.3	Code generator error messages	23
5.4	Code generator limits	24

5.5	Run-time errors	24
5.5.1	Code 1000 errors	26
5.6	Array and substring errors	26
5.7	Input/output errors	27
5.8	Tracing	27
6	FORTRAN 77 with other languages	31
6.1	Introduction	31
6.2	Parameters	32
6.3	Data types	36
6.4	Parameter passing	39
6.5	Results	41
6.6	Subroutines with alternate return parameters	41
6.7	Modules and Naming	42
7	The Interface Library	47
7.1	Introduction	47
7.2	Naming conventions	47
7.3	Calling conventions	48
	Appendix A: Front end error messages	57
	Appendix B: Code generator error messages	63
	Appendix C: Run-time error messages	67

# 1 Introduction to 32000 FORTRAN 77

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 32000 FORTRAN 77. This manual describes the use of Acorn 32000 FORTRAN 77 running under the Panos operating system; note that it is not a tutorial.

The Acorn 32000 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.

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 NS32000 based computers under the Panos operating system.

## 1.1 Installation

The language is provided on an Acorn DFS format disc, and before use it must be installed as described in the appropriate *User Guide* onto either the DFS, the ADFS (typically for use with a Winchester disc), or the NFS (in conjunction with the Econet). Note that the installation procedures must be followed even if you are installing FORTRAN 77 onto floppy disc, despite the fact that you have received it on this medium. The *User Guide* also contains some simple examples in the use of the compiler, and it is recommended that you read the relevant chapter before continuing. See also the *Panos Guide to Operations* which provides information about the Panos operating system at the level of the user interface, such as the use of utilities, including the linker.



## 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. 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' is, in fact, a command file which runs the two parts of the compiler in sequence, and so compiles the program without the need for the user to give two 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. The *User Guide* contains more details about the 'f77' command, and the *Panos Guide to Operations* provides instructions for creating or modifying command files.

### 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.

Any number of arguments can be given, and in any order. The form of the command line is as follows:

```
-> f77 [-source] name { [-list {name}] [-aof name]
      [-error name] [-opt options] [-map name]
      [-identify] [-help] }
```

where 'name' stands for a user-supplied file name, and 'options' represents a list of one or more compiler options. Braces enclose optional items.

#### -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. If the supplied name has no extension, then '-f77' will automatically be appended. See the *Panos Guide to Operations* for details about the Acorn file extension conventions.

**-list name**

A listing of the compiled program along with line numbers of the source is generated when this argument is given. This listing can be sent to a file or device specified in the argument (e.g. printer:). If no file name or device is specified, then the compiler sends the listing to 'name-lis'; a file whose name is based on the source file name, e.g. a file called Fprog-f77 will have a listing file called Fprog-lis. See figure 1 for a demonstration of this command.

**-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.

**-aof name**

The Acorn Object Format output file of the compiler is given a name based upon the source file name by default i.e. a file called 'fort1-f77' will be given the aof name 'fort1-aof'. The -aof argument can be used to specify an alternative name.

**-map name**

A storage map of the compilation is produced by the code generator and sent to the device or filename specified. This is given the file extension '-map' by default.

**-identify**

The identify argument requests compiler identification.

**-help**

This argument, if given to the command file 'f77', tells you how to obtain help information from the front end and code generator. The actual commands necessary are f77fe -help and f77cg -help.

## Example compiler commands

### A. The minimal command

```
f77 Fprog
```

This command will compile the source program 'Fprog-f77'; default settings are used throughout, i.e. the intermediate file is called 'fcode-tmp', the aof file is called 'Fprog-aof', no map or listing is produced, error messages are directed to the vdu, and default compilation options are assumed (see section below).

### B. Specifying the input and output files

```
f77 -source Fprog -fcode $.junk.code
-aof $.Afiles.Fprog -list Proglis -error printer:
```

The optional '-source' argument is supplied, and the source program 'Fprog-f77' is compiled with the intermediate file being called '\$.junk.code', the aof file named as '\$.Afiles.Fprog-aof', a listing is sent to the file 'Proglis-lis', and all error messages resulting from the compilation are sent to the printer.

### C. Using some options and requesting a map:

```
f77 Fprog -opt +tW0 -map dfs::0.FProg
```

The program 'Fprog-f77' is compiled, with tracing specified in the code, warning messages inhibited, and a storage map sent to the file 'dfs::0.Fprog-map'.

## 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', and '6', are enabled or disabled by preceding them with + or -. The options 'X', 'L', 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 substring 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 (e.g. 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 (e.g. divide by zero);
- >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 (e.g. 100/106). The error will lie in this range.

**T**

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

**6**

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

**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 (e.g. 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.



## 3 Extensions to the standard

32000 FORTRAN 77 offers several enhancements to the standard which are described in this chapter. Further extensions concerning input/output are described in chapter 4.

### 3.1 Hexadecimal constants

32000 FORTRAN 77 allows hexadecimal constants to be used wherever an ordinary constant is allowed. A hexadecimal constant is of the form:

?<type> <digits>

<type> is a letter, specifying the type of the constant. It must be one of I, R, D, C, L or H (for INTEGER, REAL, DOUBLE PRECISION, COMPLEX, LOGICAL and CHARACTER respectively).

The <type> letter is followed by hexadecimal <digits> (0-9, A-F). There must always be an even number of digits (i.e. an exact number of bytes).

The bytes in a CHARACTER hexadecimal constant are given in the order in which they are to appear in store; with other constants, the most significant byte is given first. If the type of the constant is REAL, DOUBLE PRECISION or COMPLEX, the number of bytes must match the size of the item in store (4 or 8); for INTEGER and LOGICAL constants, there may be fewer bytes. For example:

```
CHARACTER WINDOW*(*)  
PARAMETER (WINDOW = ?H1C05141E0C)  
J = ?I1234
```

Here, 'window' consists of the bytes 1C 05 14 1E 0C, and 'j' is set to the decimal value 4660.

### 3.2 FORTRAN 66 option

Quoting the +6 option in the command line (see chapter 2) specifies that the compiler will be in FORTRAN 66 mode. When this option is enabled, the action of FORTRAN changes as follows:

- DO loops will always execute at least once.
- Hollerith constants (nH) are allowed in DATA and CALL statements, and quoted constants are not CHARACTER type.

When the FORTRAN 66 switch is used, both Hollerith and quoted constants in CALLS and DATA are treated in the same way - 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') AND
CALL JIM(4HABCD)
```

Run-time FORMATS may be non-CHARACTER array names if the +6 option is quoted. For example:

```
DOUBLE PRECISION D(3),NUM
DATA D(1),D(3)/8H(1X,D20.,5H,I5)/
..
DATA NUM /2H10/
..
WRITE (6,D) 2.3D0,10
..
```

This facility was introduced for the compilation of FORTRAN 66 programs. It is strongly recommended that new programs use CHARACTER formats.

### 3.3 Naming

In 32000 FORTRAN 77, all lower case letters (except in FORMAT s and character constants) are converted into upper case upon reading the source. Thus all statements, identifiers etc. may be in lower case. Names may be up to 255 characters long, though a warning message is given if names longer than the 'standard' limit of 6 characters are used. It is worth noting, to save confusion, that there is no limit on the length of CHARACTER values.