



CONVEX FORTRAN
Quick Reference

Document No. 720-002430-000

First Edition, Rev. 1
March 1990

CONVEX Computer Corporation
Richardson, Texas USA

CONVEX FORTRAN Quick Reference
Order No. DSW-039
First Edition, Rev. 1

© 1990 CONVEX Computer Corporation
All rights reserved.

This document is copyrighted. This document may not, in whole or part, be copied, duplicated, reproduced, translated, stored electronically, or reduced to machine-readable form without prior written consent from CONVEX Computer Corporation.

Although the material contained herein has been carefully reviewed, CONVEX Computer Corporation (CONVEX) does not warrant it to be free of errors or omissions. CONVEX reserves the right to make corrections, updates, revisions or changes to the information contained herein. CONVEX does not warrant the material described herein to be free of patent infringement.

CONVEX and the CONVEX logo ("C") are registered trademarks of CONVEX Computer Corporation.

UNIX is a trademark of AT&T Bell Laboratories.

Cray is a registered trademark of Cray Research, Inc.

Sun is a trademark of Sun Microsystems, Inc.

VAX is a trademark of Digital Equipment Corp.

Printed in the United States of America

BLOCK DATA

Define a subprogram that provides initial values for variables and array elements in common blocks.

```
BLOCK DATA [name]
```

where

name is a symbolic name for the block data subprogram.

A block data subprogram begins with a BLOCK DATA statement and ends with an END statement.

CALL

Invoke a subroutine.

```
CALL subroutine ( ( [a [.a]... ] ) )
```

where

subroutine is the name of the subroutine.
a is an actual argument.

CLOSE

Disconnect a file from a unit.

```
CLOSE ( [UNIT=]u [.STATUS=p] [.ERR=label] [.IOSTAT=ios] )
```

or

```
CLOSE ( [UNIT=]u [.DISP[OSE]=p] [.ERR=label] [.IOSTAT=ios] )
```

where

u is a logical unit number.
p is a character expression that determines the disposition of the file. Its values are 'KEEP', 'SAVE', or 'DELETE'.
label is the label of an executable statement.
ios is an integer variable or integer array element.

COMMON

Allow variables or arrays in a main program or subprogram to share the same storage as variables and arrays in other subprograms.

```
COMMON [ /[cbn]/ ] nlist [ [.] /[cbn]/ nlist]...
```

where

cbn is a common block name.
nlist is a list of variable names, array names, and array declarators.

CONTINUE

This executable statement has no effect.

CONTINUE

DATA

Establish initial values for arrays, array elements, substrings, and variables.

DATA nlist/clist/ [[.] nlist/clist/]...

where

nlist	is a list of arrays, array elements, character substrings, implied DO lists, or variable names.
clist	is a list of constants or symbolic names of constants (defined with a PARAMETER statement).

DECODE

Transfer data between arrays or variables in internal storage, translating from character to internal form.

DECODE (c, f, b [.IOSTAT=ios] [.ERR=label]) [iolist]

where

c	is an integer expression.
f	is a format identifier.
b	is an array, array element, variable, or character substring reference that contains the characters to be translated.
ios	is either an integer array element or integer variable that is defined as a positive integer if an error occurs or as zero if no error occurs.
label	is the label of an executable statement.
iolist	is an I/O list that receives the data.

Compilation

The command line that invokes the CONVEX FORTRAN compiler is:

`fc [options] files [loader-options]`

where

<code>options</code>	is one or more compiler options (see below).
<code>files</code>	is one or more FORTRAN source files, object files, assembly files, or libraries.
<code>loader-options</code>	is one or more loader options.

Language-Compatibility Options

<code>-cfc</code>	Select Cray FORTRAN language definitions and library.
<code>-dfc</code>	Select DEC FORTRAN language definitions.
<code>-F66</code>	Select FORTRAN 66 language definitions.
<code>-sa</code>	Prevent compiler from generating precompiled argument packets.
<code>-sfc</code>	Select Sun FORTRAN language definitions.
<code>-vfc</code>	Select VAX FORTRAN language definitions.

Optimization Options

<code>-ds</code>	Perform dynamic loop selection.
<code>-ep n</code>	Specify estimated number of processors.
<code>-il</code>	Prepare intermediate-language file for inline substitution.
<code>-is <i>directory</i></code>	Perform inline substitution.
<code>-no</code>	Perform no optimization.
<code>-O0</code>	Perform local scalar optimization.
<code>-O1</code>	Perform global scalar optimization.
<code>-O2</code>	Perform vectorization.
<code>-O3</code>	Perform automatic parallelization.
<code>-rl</code>	Perform loop replication.
<code>-uo</code>	Perform potentially unsafe optimizations.
<code>-ur</code>	Perform loop unroll.

Code-Generation Options

<code>-c</code>	Suppress loading.
<code>-fi</code>	Use IEEE floating-point format.
<code>-fn</code>	Use native floating-point format.
<code>-in</code>	Specify default length (n) for integer and logical variables; n can be 2, 4, or 8. Obsolete; Use <code>-p8</code> or <code>-pd8</code> instead.
<code>-p8</code>	Specify default length of 8 for integer, logical, and real variables; specify default length of 16 for double precision and complex.
<code>-pd8</code>	Specify default length of 8 for integer, logical, real, and double precision variables; specify default length of 16 for complex.

- rn Specify default length (n) for real variables; n can be 4 or 8. Obsolete; use -p8 or -pd8.
- re Generate reentrant code.
- S Generate symbolic assembly code.
- tm *target* Generate code for target machine (c1 or c2).

Message and Listing Options

- LST Produce listing of compiled source program.
- LSTI Produce -LST listing with INCLUDE files.
- na Suppress advisory diagnostics.
- nw Suppress warning diagnostics.
- or *table* Produce specified table (loop, array, all, or none).
- xr Call the fxref cross-reference generator.
- xr1 Call the fxref cross-reference generator; put all symbols into one table.

Debugging and Profiling Options

- a1 Treat noncharacter arrays with last dimension of 1 as assumed-size (last dimension of *).
- cs Check that subscripts are within array bounds.
- db Produce debugging information.
- dc Compile lines with D in column 1.
- p Produce code for prof profiler.
- pb Produce code for bprof profiler.
- pg Produce code for gprof profiler.
- pa Produce code for CXpa profiler routine and loop levels.
- pab Produce code for CXpa profiler block level.
- sc Check syntax only.

Miscellaneous Options

- B*directory* Find the substitute compiler in the named directory.
- fpp Use preprocessor as first step of compilation.
- link *arg* Passes *arg* to the loader.
- o *name* Specify name for executable module.
- tl *n* Set maximum compile time to n minutes.
- vn Identify compiler version.
- 72 Process only first 72 characters of each program line.

Error Utility

fc [options] source_file.f |& error

inserts error messages as comments into source_file.f.

Directives

The general format of a compiler directive is as follows:

`C$DIR directive [.directive]`

A directive pertaining to a loop must immediately precede that loop.

ASSIGN LOCK

Designate a variable to be used as a lock for a critical section of code.

`ASSIGN LOCK (lock-name [.lock-name]...)`

...

`FREE LOCK (lock-name [.lock-name]...)`

where

`lock-name` is the name of an integer scalar variable or array element to be used as a lock. Must be `INTEGER*4` or `INTEGER*8`.

A lock is set with the `ASSIGN LOCK` directive and released with the `FREE LOCK` directive.

BEGIN ORDER

Identify an ordered critical region in which loop iterations must be performed in the written sequence.

`BEGIN ORDER (lock-name)`

...

`END ORDER (lock-name)`

where

`lock-name` is the name of the lock to be used.

An ordered critical region begins with a `BEGIN ORDER` directive and ends with an `END ORDER` directive.

BEGIN SECTION

Identify an unordered critical region, in which loop iterations need not be performed in the correct sequence.

`BEGIN SECTION (lock-name)`

...

`END SECTION (lock-name)`

where

`lock-name` is the name of the lock.

An unordered critical region begins with a `BEGIN SECTION` directive and ends with an `END SECTION` directive.

BEGIN_TASKS

Identify a sequence of tasks for parallel execution.

BEGIN_TASKS

...

NEXT_TASK

...

NEXT_TASK

...

END_TASKS

A sequence of tasks begins with a BEGIN_TASKS directive and ends with an END_TASKS directive. A NEXT_TASK directive precedes each individual task.

FORCE_PARALLEL

Parallelize the loop that follows regardless of any apparent loop dependencies. Do not allow interchange of outer loops for vectorization.

FORCE_PARALLEL

FORCE_PARALLEL_EXT

Parallelize the loop that follows regardless of any apparent loop dependencies.

FORCE_PARALLEL_EXT

FORCE_VECTOR

Vectorize the loop that follows regardless of any apparent recurrences.

FORCE_VECTOR

MAX_TRIPS

Specify the maximum number of times the loop that follows is executed.

MAX_TRIPS (n)

where

n is the maximum iteration (trip) count.

NO_RECURRENCE

Disregard any apparent recurrences in the loop that follows.

NO_RECURRENCE

NO_SIDE_EFFECTS

The specified functions do not modify the value of a parameter or common variable, read or write, or call another routine.

NO_SIDE_EFFECTS [(func [,func]...)]

where

func is a user-defined function.

If no argument is given, the directive applies to all functions that follow.

NO_PARALLEL

Do not parallelize the loop that follows.

NO_PARALLEL

NO_VECTOR

Do not vectorize the loop that follows.

NO_VECTOR

PREFER_PARALLEL

Parallelize the loop that follows, if possible. Do not interchange outer loops for vectorization.

PREFER_PARALLEL

PREFER_PARALLEL_EXT

Parallelize the loop that follows, if possible.

PREFER_PARALLEL_EXT

PREFER_VECTOR

Vectorize the loop that follows, if possible, if there is a choice of loops in a nest to vectorize.

PREFER_VECTOR

PSTRIP

Strip mine the following parallel loop using the specified length.

PSTRIP (integer_constant)

where

integer_constant is an integer that specifies the strip-mine length.

ROW_WISE

The designated array names have their dimensions reversed.

ROW_WISE (array [.array]...)

where

array is the name of the array.

SCALAR

Prevent the loop that follows from being vectorized or parallelized.

SCALAR

SELECT

Generate multiple versions of a loop and the code to select, at runtime, which version to execute.

SELECT (vtrip, ptrip, pvtrip)

where

vtrip	is the iteration (trip) count at which the compiler is to select vector execution for the loop.
ptrip	is the trip count at which the compiler is to select parallel execution for the loop.
pvtrip	is the trip count at which the compiler is to select parallel-vector execution for the loop.

SYNCH_PARALLEL

Execute the loop that follows in parallel even though it requires synchronization that may result in less than full efficiency.

SYNCH_PARALLEL

UNROLL

Perform loop unrolling on the body of the loop that follows.

UNROLL

VSTRIP

Strip mine the following vector loop using the specified length.

VSTRIP (k)

where

k is an integer constant.

Statements

ACCEPT

Sequentially read data from the implicit input unit.

ACCEPT f [.iolist]

or

ACCEPT * [.iolist]

or

ACCEPT namelist

where

f is an expression or a format label.

* specifies list-directed formatting.

iolist is an I/O list.

namelist is a namelist block name.

ASSIGN

Assign a statement label to an integer variable.

ASSIGN label TO i

where

label is the label of an executable statement.

i is an integer variable name.

BACKSPACE

Position a file connected to the specified unit to the beginning of the previous record.

BACKSPACE u

or

BACKSPACE ([UNIT=]u [.IOSTAT=ios] [.ERR=label])

where

u is a logical unit specifier.

label is the label of an executable statement.

ios is an integer variable or integer array element.

EQUIVALENCE

Cause two or more entities in a program to refer to the same storage area.

EQUIVALENCE (list) [(list)]...

where

list is a list of two or more variables, array elements, array names, or character substring names separated by commas.

EXTERNAL

Identify a name as representing an externally defined procedure or dummy procedure.

EXTERNAL n [,n]...

where

n is the symbolic name of a subprogram, block data subprogram, or dummy procedure.

FIND

Position a direct-access file to a particular record and set the associated variable to that record number.

FIND ([UNIT=]n, REC=rec, [.ERR=label] [.IOSTAT=ios])

where

n is a logical unit number that refers to a direct-access file.

rec is the direct-access record number.

label is the label of an executable statement to which control transfers if an error occurs.

ios is an integer variable or integer array element that is positive if an error occurs or zero if no error occurs.

FORMAT

Provide information to create the desired format for I/O statements.

label FORMAT (list)

where

label is a required statement label.

list is a nonempty format list.

FUNCTION

Specify the name of a function and its arguments.

[type] FUNCTION name [*m] [([d [.d]...])]

where

type is a logical, character, or numeric data type specifier.

name is the name of the function subprogram.

m is an unsigned, nonzero integer constant specifying the length of the data type.

d is a dummy argument name.

A function subprogram begins with a FUNCTION statement and ends with an END statement.

GOTO

Transfer control to a designated statement.

GOTO label

or

GOTO (list) [,] expr

or

GOTO var [[,] (list)]

where

label is the label of an executable statement.

list is a list of labels of executable statements.

expr is an arithmetic expression.

var is an integer variable name defined by an ASSIGN statement with the value of an executable statement label.

IF

Transfer control based on the value of an arithmetic or logical expression.

IF (arith-expr) label-1, label-2, label-3

or

IF (logical-expr) statement

where

arith-expr is an arithmetic expression of type integer or real.

label is the label of an executable statement. For an arithmetic IF, all three labels must be given. Control is transferred to label 1 if the value of the expression is less than zero, label 2 if it equals zero, and label 3 if it is greater than zero.

logical-expr is a logical expression.

statement is an executable statement other than a DO, END DO, END, logical IF, or block IF.

IF THEN

Execute one or more statements depending upon the value of a logical expression.

IF (logical-expr) THEN

...

END IF

or

IF (logical-expr) THEN

...

ELSE

...

END IF

or

IF (logical-expr-1) THEN

...

ELSE IF (logical-expr-2) THEN

...

ELSE IF (logical-expr-3) THEN

...

ELSE

...

END IF

where

logical-expr is a logical expression.

Any number of ELSE IF...THEN statements may be specified. Use of the ELSE statement is optional.

IMPLICIT

Override the implied data typing of symbolic names.

IMPLICIT typ (a [.a]...) ...

or

IMPLICIT NONE

where

typ is an INTEGER[*len], REAL[*len], DOUBLE PRECISION, CHARACTER[*len] DOUBLE COMPLEX, COMPLEX[*len], LOGICAL[*len], or structure data type.

a is a letter or range of letters.

len is an optional length specifier.

The IMPLICIT NONE form overrides all implicit defaults.

INCLUDE

Includes source code from the specified file into the program being compiled.

INCLUDE 'filename'

where

filename is the pathname of the file from which source code is to be read.

INQUIRE

Determine specific information about a file or unit.

INQUIRE (FILE=fi, list)

or

INQUIRE ([UNIT=]u, list)

where

fi is a character expression, numeric array name, numeric variable name, or numeric array element name whose value is the name of the file being queried.

list is a list of specifiers that indicate the information to be determined (see the CONVEX FORTRAN Language Reference Manual, Chapter 7).

u is the external unit number.

INTRINSIC

Allow the use of intrinsic function names as arguments to subprograms.

```
INTRINSIC n [.n]...
```

where

n is one of the FORTRAN intrinsic functions.

NAMELIST

Associate a name with a list of variables or array names.

```
NAMELIST /name/varlist [ [.] /name/varlist]...
```

where

name is a symbolic name representing the list of entries to be read or written.

varlist is a list of variable or array names, separated by commas, to be associated with /fname/fR.

OPEN

Connect an existing external file to a unit, change the attributes of a connected file, or create a new file and connect it to a unit.

```
OPEN (keyword [.keyword] )
```

where

keyword is a keyword and, possibly, an associated value (see Chapter 7 of the /fICONVEX FORTRAN Language Reference Manual/fR).

OPTIONS

Specify or override compiler options. If used, must be the first statement in the program.

```
OPTIONS option [option]...
```

where

option is one of the compiler options (see the /fICONVEX Language Reference Manual/fR, Table 1-2).

PARAMETER

Assign a symbolic name to a constant.

PARAMETER (name=c [.name=c]...)

or

PARAMETER name=c [.name=c]...

where

name is a symbolic name.

nlist is a constant.

The second form of the statement with just one constant is only available under the -vfc compiler option.

PAUSE

Temporarily halt program execution; resume execution on operator command.

PAUSE [message]

where

message is the information to be displayed upon halting and can be a character constant or a string of up to five digits.

PRINT

Transfer formatted records to the implicit output device using sequential access.

PRINT format [.iolist]

or

PRINT * [.iolist]

or

PRINT namelist

where

format is a format specifier.

iolist is an I/O list.

* indicates list-directed formatting.

namelist is a namelist specifier.

PROGRAM

Assign a name to the main program unit. This statement is optional.

PROGRAM name

where

name is the symbolic name for the main program.

READ

Transfer data from an external file into internal storage, or transfer data between internal storage locations.

READ fmt [,iolist]

or

READ (u [,fmt] [,IOSTAT=ios,ERR=label,END=label]) [iolist]

or

or

READ (u' rec [,fmt] [,IOSTAT=ios,ERR=label,END=label]) [iolist]

or

READ (iu, fmt [,IOSTAT=ios,ERR=label,END=label]) [iolist]

where

fmt is a format specifier or an asterisk (*) indicating list-directed format.

iolist is an I/O list that identifies the data to be transferred.

u is a logical unit number.

ios is an integer variable or array element.

label is the label of an executable statement.

rec is a record number. The apostrophe before the record number is available only in -vfc mode.

iu is the name of a character variable, array, array element, or substring used as an internal unit.

RETURN

Return control from a subroutine subprogram to the calling program.

RETURN [expr]

where

expr is an optional integer expression that specifies an alternate return.

REWIND

Position a file to its initial point.

```
REWIND ( [UNIT=]u [.IOSTAT=ios] [.ERR=label] )
```

where

u	is a logical unit specifier.
label	is the label of an executable statement.
ios	is an integer variable or integer array element.

SAVE

Retain the values of designated variables and arrays in a subroutine or function when a RETURN or END statement is executed.

```
SAVE [name [,name]...]
```

where

name	is a variable name, array name, or named common block contained between slashes.
------	--

If the SAVE statement has no arguments, the values of all allowable entities are retained.

STOP

Terminate program execution.

```
STOP [string]
```

where

string	is a string of up to five digits or a character constant to be displayed.
--------	---

SUBROUTINE

Specify the name of a subroutine and its arguments.

```
SUBROUTINE name [ ( [d [.d]...] ) ]
```

where

name	is a symbolic name of the subroutine.
d	is a dummy argument list containing variable, dummy procedure, and array names, or asterisks (*).

A subroutine begins with a SUBROUTINE statement and terminates with an END statement.

DO

Begin a loop.

```
DO [ label [, ] ] var = expr1, expr2, expr3
```

where

label	is the label of an executable statement. If no label is specified, the loop must end with an END DO statement.
var	is the name of an integer, real, or double-precision variable.
expr1	is an integer, real, or double-precision expression that specifies the value of var upon initial execution of the loop.
expr2	is an integer, real, or double-precision expression that specifies the ending value of var.
expr3	is an integer, real, or double-precision expression that specifies the increment for var after each iteration of the loop.

A DO loop begins with a DO statement and ends with an END DO statement or a statement marked by *label*.

DO WHILE

Execute a loop as long as a logical expression remains true.

```
DO [label [, ] ] WHILE (expr)
```

where

label	is the label of an executable statement.
expr	is a logical expression.

A DO WHILE loop begins with a DO WHILE statement and ends with an END DO or a statement marked by *label*.

DIMENSION

Specify the bounds of arrays.

```
DIMENSION array (dim [,dim]... ) ...
```

where

array	is the name of an array.
dim	is a dimension of the array in the form <code>[[lower_bound]:] upper_bound</code> .

ENCODE

Transfer data between arrays or variables internal storage, translating from internal to character form.

```
ENCODE ( c .f. b [.IOSTAT=ios] [.ERR=label] ) [iolist]
```

where

- c is an integer expression.
- f is a format identifier.
- b is an array, array element, variable, or character substring to receive the data.
- ios is either an integer array element or integer variable that is defined as positive if an error occurs or zero if no error occurs.
- label is the label of an executable statement to which control transfers if an error occurs.
- iolist is an I/O list that contains the data to be translated.

END

Indicate the end of a program unit. It terminates a main program without a message or returns control from a function or subroutine.

```
END
```

ENDFILE

Write an endfile record on the file connected to a unit.

```
ENDFILE ( [UNIT=]u [.IOSTAT=ios] [.ERR=label] )
```

where

- u is a logical unit specifier.
- label is the label of an executable statement.
- ios is an integer variable or integer array element.

ENTRY

Specify an alternate entry point for a function or subroutine.

```
ENTRY name [ ( [d [.d]...] ) ]
```

where

- name is a symbolic name of the entry point.
- d is a dummy argument.

TYPE

Transfer formatted records to the implicit output device using sequential access.

TYPE format [.iolist]
or
TYPE * [.iolist]
or
TYPE namelist

where

format	is a format specifier.
iolist	is an I/O list.
*	indicates list-directed formatting.
namelist	is a namelist specifier.

WRITE

Transfer data from internal storage to an external unit, or transfer data between internal storage locations.

WRITE fmt [.iolist]
or
WRITE (u [.fmt] [.IOSTAT=ios,ERR=label]) [.iolist]
or
WRITE (u' rec [.fmt] [.IOSTAT=ios,ERR=label]) [.iolist]
or
WRITE (iu, fmt [.IOSTAT=ios,ERR=label]) [.iolist]

where

fmt	is a format specifier or an asterisk (*) indicating list-directed formatting.
iolist	is an I/O list that identifies the data to be transferred.
u	is a logical unit number.
ios	is an integer variable or array element.
label	is the label of an executable statement.
rec	is a record number. The apostrophe before the record number is available only in -vfc mode.
iu	is the name of a character variable, array, array element, or substring used as an internal unit.

