CONVEX FORTRAN V7.0 Advance Notice

Document No. 720-005230-000

First Edition
October 1991

CONVEX Press
Richardson, Texas
United States of America

CONVEX FORTRAN
Advance Notice

Although the material contained herein has been carefully reviewed, 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.

UNLESS PROVIDED OTHERWISE IN WRITING WITH CONVEX COMPUTER CORPORATION (CONVEX), THE PROGRAM DESCRIBED HEREIN IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. SOME STATES DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES. THE ABOVE EXCLUSION MAY NOT BE APPLICABLE TO ALL PURCHASERS BECAUSE WARRANTY RIGHTS CAN VARY FROM STATE TO STATE. IN NO EVENT WILL CONVEX BE LIABLE TO ANYONE FOR SPECIAL, COLLATERAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING ANY LOST PROFITS OR LOST SAVINGS, ARISING OUT OF THE USE OR INABILITY TO USE THIS PROGRAM. CONVEX WILL NOT BE LIABLE EVEN IF IT HAS BEEN NOTIFIED OF THE POSSIBILITY OF SUCH DAMAGE BY THE PURCHASER OR ANY THIRD PARTY.

Printed in the United States of America

Introduction 1

This document is intended to provide users of the CONVEX FORTRAN Version 7.0 compiler with notice of improvements, new features and other pertinent changes to the compiler. This document precedes the Release Notice, which will contain more up-to-date information regarding bug fixes, open bugs, workarounds, and last-minute documentation corrections. This document is *not* to be considered a release notice.

The remaining sections in this document describe the following aspects of the CONVEX FORTRAN Version 7.0 compiler:

- Section 2 contains notes and cautions about the use of this software.
- Section 3 describes the new features provided in this release.
- Section 4 describes changes to the documentation.

The CONVEX FORTRAN compiler is a scalar optimizing compiler (automatically performing local and global scalar optimizations), a vectorizing compiler (automatically performing vectorization of loops), and a parallelizing compiler (automatically performing parallelization of loops). CONVEX FORTRAN Version 7.0 will consist of the compiler driver (`fc`), the compiler (`fskel`), the error message file (`errmsgf`), the runtime system libraries, the man pages, other FORTRAN-specific utilities, and documentation.

Notes and cautions 2

This section contains general information and words of caution about CONVEX FORTRAN Version 7.0.

Serial numbers

The compiler checks the serial number of the machine on which it is running. If the serial number does not match the expected one, a message is printed and execution is aborted.

Prerequisites

CONVEX FORTRAN Version 7.0 requires the installation of ConvexOS Version 9.1 or Version 10.0. A later release of ConvexOS is acceptable.

CONVEX FORTRAN Version 7.0 requires the installation of the CONVEX C compiler Version 4.1. A later release of CONVEX C is acceptable.

CXpa sites should ensure that CONVEX CXpa Version 1.2 is installed. A later release of CXpa is acceptable. CXpa Version 1.1 will not function correctly with CONVEX FORTRAN V7.0.

CONVEX FORTRAN Version 7.0 is compatible with the CXdb Debugger Version 1.1 or higher.

The following board revision levels are required for CONVEX FORTRAN V7.0:

Model	Board revision levels
C1 XL	
C1	
C210	
C220 C230 C240	VPC rev. G

A hardware upgrade is available for machines that did not include these features as original equipment.

The following revisions of the Diagnostic Database are recommended for CONVEX FORTRAN V7.0:

Model	Diagnostic database revision levels
C1	V1.15 or higher
C210, C220, C230, C240	V3.5 or higher

These versions of the diagnostic databases ensure that the `ATAN` instruction will produce the same answers as the `ATAN` library routine (more accurate than older versions).

New features 3

Faster I/O

Certain I/O statements with implied DO loops will execute much faster.

Binary I/O enhancements

Support for Cray unformatted sequential access files has been added, as described in the following subsections.

Cray unblocked sequential access files

Cray unformatted unblocked sequential access files can now be accessed by supplying the FORM = UNFORMATTED/CRAYUB keyword definition in the OPEN statement.

Cray blocked sequential access files

A conversion program for Cray unformatted sequential access "blocked" binary files is included in this release. The conversion program converts Cray-style "blocked" unformatted sequential access files to a format readable by CONVEX FORTRAN Version 7.0.

Usage:

```
fcUnblock  < inputfile  > outputfile
```

You must specify FORM = UNFORMATTED/CRAY in the OPEN statement to access *outputfile,*.

Short circuiting evaluation of IF conditional expressions

CONVEX FORTRAN Version 7.0 automatically short circuits certain expressions in IF statements. When the conditional expression in the IF statement contains .AND. and .OR. operators, these are usually short circuited. Short-circuit evaluation works with all types of IF statements (arithmetic, logical, and block). Performing arithmetic (+, -, *, /) on or applying non-logical operands or functions to a logical expression disables short circuit evaluation within that expression. Logical valued expressions used as arguments to function calls within an IF statement's conditional expression are not short-circuited.

Note that short-circuiting can skip the evaluation of parts of an expression. Expected function call side effects may not occur. The -nosc flag disables this optimization.

New cross-referencer

A new cross-referencer, with a different output format, is now the default (when the -xr flag is specified). The old cross-reference package (fxref) is still available, but is no longer supported and does not accept all language features. Use the -xro flag if you wish to use the old cross-referencer. See the fxref(1F) man page for details.

The new cross-referencer allows multiple source files to be referenced together. The -xra flag is used to generate cross-reference data files but not a report. The fcxref program can then be invoked to generate a report after several source files have been processed. This utility is covered in detail in Chapter 4 of the *CONVEX FORTRAN User's Guide*.

IF-DO optimizations

IF-DO optimizations are performed at -O2 and -O3 optimization levels in CONVEX FORTRAN Version 7.0. In general, they modify loops containing tests to improve vector performance. Tests can be promoted out of their containing IF or DO loops or eliminated completely. By minimizing the number of tests within a loop, the compiler reduces the number of masked vector instructions that must be executed, thereby improving performance. The *CONVEX FORTRAN Optimization Guide*, included with the CONVEX FORTRAN Version 7.0 documentation set, contains a complete description of IF-DO optimizations in Chapter 3.

IF-DO optimizations are enumerated in the following sections.

Redundant index test elimination

The compiler recognizes tests against some index variable that always evaluate to `.TRUE.` or `.FALSE.` and eliminates the code.

Example:

Original loop:

```
DO I = 1, N
  IF (I .GT. 0) THEN
    DO J = 1, I
      A(I,J) = 0
    ENDDO
  ENDIF
ENDDO
```

Optimized loop:

```
DO I = 1, N
  DO J = 1, N
    A(I,J) = 0
  ENDDO
ENDDO
```

This optimization is especially useful when optimizing FORTRAN 66 codes that contain `DO` loops surrounded by `IF` tests.

Loop boundary value peeling

Loop boundary value peeling involves removing the first and/or last iterations of a loop when doing so removes conditional tests from the loop. This optimization is performed when the loop contains a test involving an explicit reference to the loop index variable that always evaluates to `.TRUE.` or `.FALSE.` for the first and/or last iteration of the loop.

Example:

Original loop:

```
DO I = 1, 100
  IF (I .EQ. 1) THEN
    A(I) = B(I)
  ELSE IF(I .EQ. 100) THEN
    A(I) = C(I)
  ELSE
    A(I) = -A(I)
  END IF
ENDDO
```

Peeled loop:

```
A(1) = B(1)
DO I = 2, 99
  A(I) = -A(I)
ENDDO
A(100) = C(100)
```

In some cases boundary-value peeling requires replicating large amounts of code and can greatly increase the size of the executable. By default, boundary-value peeling is performed and code replication is allowed up to a predetermined conservative limit. The following options and directives are available to control code replication:

- `-nopeel` and `NOPEEL`: disables loop peeling.
- `-peel` and `PEEL`: specifies a higher default replication limit.
- `-peelall` and `PEEL_ALL`: allows code replication without bound. In codes containing large numbers of boundary value operations, this can greatly lengthen compile time and increase the size of the code enough to exceed the limits of some of the compiler's internal tables.

Test promotion

Test promotion involves promoting a test out of the loop that encloses it by replicating the containing loop(s) for each branch of the test. The replicated loops contain fewer tests than the originals (or no tests at all), so they execute much faster.

Multiple tests can be promoted, with loop replications made for each. In nested loops, tests are promoted to the level of the outermost nest, and all subordinate loops are replicated.

Example:

Original loop:

```
DO I = 1, 100
  IF (G) THEN
    A(I) = B(I)
  ELSE
    A(I) = C(I)
  END IF
END DO
```

Optimized loop:

```
IF (G) THEN
  DO I = 1, 100
    A(I) = B(I)
  END DO
ELSE
  DO I = 1, 100
    A(I) = C(I)
  END DO
END IF
```

By default, the compiler promotes tests and replicates code up to a predetermined conservative limit. The following options are available to control this limit:

- `-noptst` and `NO_PROMOTE_TEST`: disables test promotion.
- `-ptst` and `PROMOTE_TEST`: specifies a higher replication limit. Can cause a noticeable increase in compile time.
- `-ptstall` and `PROMOTE_TEST_ALL`: allows code replication without bound. This can cause a large increase in compile time and can increase the size of the code enough to exceed the limits of some of the compiler's internal tables.

Test promotion and loop peeling together may substantially increase the size of programs with many large complicated loop nests. These optimizations can interact with each other to exponentially increase the size of a program (and the time to compile it) under certain circumstances. Indiscriminate use of the `-peelall` and `ptstall` flags can result in excessive

compilation times as well as cause the compiler to exceed certain internal limits. The default level of loop peeling and test promotion was chosen to minimize the added overhead in compilation time and program size while still providing significant benefit from these optimizations.

Fortran 90 support

This release of CONVEX FORTRAN supports the `-f90` flag, which enables support for the Fortran 90 features enumerated in the following sections.

Automatic arrays

Automatic arrays are implicitly defined by the occurrence of a local array (not in `COMMON` and not a dummy argument) which is dimensioned with a non-constant expression. These arrays are allocated on the stack at runtime when the procedure is called. Also supported under the `-cfc` flag.

Allocatable arrays

Allocatable arrays are declared with the `ALLOCATABLE` statement. Storage for these arrays must be declared with the `ALLOCATE` statement, and storage should be freed with the `DEALLOCATE` statement.

Example:

```
        .
        .
        .
INTEGER B(:)
ALLOCATABLE (Z,A(:),B,C(:,:,:))
ALLOCATE (A(6),B(-3:2),C(100,100,100))
DO I=1,6
  A(I) = I
  B(I-4) = I-4
END DO
DEALLOCATE (A,B,C)
        .
        .
        .
```

Limited Fortran 90 array notation

The `-f90` flag enables support for Fortran 90 array objects, array sections, and array expressions in assignment statements.

Example:

```
          .
          .
          .
      INTEGER T(300,300)
      REAL Q(300,300)
C     TRIANGULAR MATRIX INITIALIZATION
      DO I=1,300
        T(I:300,I) = I
      END DO
      Q = 7
      T = T * 2 + Q
      END
          .
          .
          .
```

Supported Fortran 90 array intrinsics

CONVEX FORTRAN Version 7.0 supports the following Fortran 90 array manipulation intrinsics:

MERGE	SPREAD	PACK	UNPACK
MATMUL	DOT_PRODUCT	TRANSPOSE	SUM
PRODUCT	MAXVAL	MINVAL	COUNT
ANY	ALL	MAXLOC	MINLOC

Cray TASK COMMON support

Cray TASK COMMON blocks provide thread-local storage for the variables they contain. TASK COMMON blocks must reside in functions, subroutines and BLOCK DATA subprograms. A block cannot be declared both COMMON and TASK COMMON. Unnamed TASK COMMON blocks are not allowed.

Miscellaneous changes

CONVEX FORTRAN contains the following miscellaneous changes:

- The preprocessor, `fcpp1`, which shipped with CONVEX FORTRAN V6.1, has been deleted. The functionality it provided has been included in the compiler in a more general optimization.
- The `fc` driver now optionally records assorted information about each attempted compilation. Refer to the *CONVEX FORTRAN Version 7.0 Installation Instructions* for more details.
- Cray style Boolean octal constants of the form `I =` *nn...n*`B` are supported under the `-cfc` option.
- Hex constants of the form `I = X'`*nn...n*`'` are supported in all modes.
- Cray zero-filled left-justified Hollerith constants of the form `A =` *n*`Ltt...t` are supported under the `-cfc` option.
- Cray-style `FORMAT` statements of the form `FORMAT (*`*literal text*`*)` are supported under the `-cfc` option in compile time formats, and in all modes in runtime formats.

Summary of new flags

The following table contains a summary of the new flags contained in CONVEX FORTRAN Version 7.0.

Flag	Purpose
`-f90`	Allows use of array expressions, sections, and objects. Allows use of automatic arrays and allocatable arrays.
`-mi` *n*	Allows expected memory interleave to be specified for executable.
`-nopeel`	Disables the loop peeling optimization.
`-noptst`	Disables the test promotion optimization.
`-nosc`	Disables the short circuiting optimization in conditional expressions in `IF` statements.
`-peel`	Increases code replication limit for loop peeling at optimization levels `-O2` and `-O3`.

Flag	Purpose
`-peelall`	Allows code replication without bound when loop peeling is performed at optimization levels `-O2` and `-O3`.
`-ptst`	Increases code replication limit for test promotion at optimization levels `-O2` and `-O3`.
`-ptstall`	Allows code replication without bound when test promotion is performed at optimization levels `-O2` and `-O3`.
`-tm C3[248]`	Machine types for the `-tm` flag now include C3 Series arguments.
`-xr`	Invokes the new cross-reference facility. This flag and `-xra` have several subordinate cross-referencer flags which are enumerated in detail in the *CONVEX FORTRAN User's Guide*.
`-xra`	Used to cross reference several source files in one report.
`-xro`	Invokes the old cross-reference facility.

Documentation changes 4

The documentation set supplied with CONVEX FORTRAN Version 7.0 has been extensively revised and reformatted. The *CONVEX FORTRAN User's Guide, Language Reference Manual,* and *Programmer's Reference* have been published in a new small-page format as one softcover document, the *CONVEX FORTRAN Guide*. Each book is included as a part in the *Guide* in its entirety, but the books share common front matter and a master index.

Stylistic conventions have changed as explained in the preface to the *CONVEX FORTRAN Guide*.

The *CONVEX FORTRAN Quick Reference* and *CONVEX FORTRAN Optimization Guide* remain separate documents, and both have been revised for CONVEX FORTRAN Version 7.0.

The following table enumerates the new CONVEX FORTRAN Version 7.0 documentation set.

FORTRAN compiler documentation

Item	Qty	Type	Part number	Description
1	1	Manl.	720-000130-101	*CONVEX FORTRAN Guide*
2	1	Manl.	720-002430-002	*CONVEX FORTRAN Quick Reference,* Second Edition
3	1	Manl.	720-000930-202	*CONVEX FORTRAN Optimization Guide,* Third Edition
4	1	Manl.	720-001930-013	*CONVEX FORTRAN Installation Instructions*
5	1	Manl.	720-001830-013	*CONVEX FORTRAN Release Notice*

All five of these documents will be included in the documentation set for this release.