HP Fortran

Release Notes for OpenVMS Alpha Systems

May 2003

This document contains information about new and changed features in this version of HP Fortran for OpenVMS Alpha Systems.

Software Version: HP Fortran V7.6

Hewlett-Packard Company
Palo Alto, California

May 2003

This document was prepared using DECdocument, Version 3.3-1n.

Contents

4 General information about Compaq Fortran

5 Documentation Overview

Tables

1
HP Fortran Version V7.6 Overview

HP Fortran Version V7.6 provides new versions of the HP Fortran (formerly Compaq Fortran 90) and HP Fortran 77 (formerly Compaq Fortran 77) compilers which include new features as well as corrections to problems discovered in earlier versions.

With the Version 7.6 release, the name of the product officially changed to HP Fortran from Compaq Fortran (and DIGITAL Fortran before that). However, some documentation may not reflect the new name until it is republished. HP Fortran Version V7.6 requires OpenVMS Alpha Version V7.1 or higher. The HP Fortran product kit may be installed whether or not any earlier version of HP Fortran, Compaq Fortran, DIGITAL Fortran or DEC Fortran was previously installed. If DEC Fortran 90 V2.x was previously installed, it must be deinstalled (`PRODUCT REMOVE FORTRAN90` before installing HP Fortran. For complete installation details, see the *Compaq Fortran Installation Guide for OpenVMS Systems*.

This chapter provides general information about the HP Fortran product. The remaining chapters provide the following information:

- Chapter 2 highlights new and changed features in this version of HP Fortran.
- Chapter 3 highlights new and changed features in previous versions of HP Fortran.
- Chapter 4 provides detailed information about HP Fortran features not described in other documentation as well as general information on usage.
- Chapter 5 lists the organization of the HP Fortran documentation set and describes changes and corrections to the documentation.

1.1 Operating system compatibility

HP Fortran is supported on OpenVMS Alpha versions V7.1 through V7.3-1, the latest which has been released at the time of printing. It is likely that HP Fortran will work correctly on all later versions of OpenVMS Alpha. If you have questions about version compatibility, please contact us as described in Section 1.3.

Applications compiled with HP Fortran version V7.6 require a compatible Fortran Run-Time Library, such as the one provided with the HP Fortran kit. The updated Run-Time Library will be included in a future version of OpenVMS Alpha after V7.3-1.

To install the Run-Time Library on other systems, use the POLYCENTER Software Installation (PCSI) FORRTL kit provided with HP Fortran. This kit may be freely redistributed provided the terms specified in the Software Product Description (SPD) are honored. A copy of the HP Fortran SPD can be found in

the [.DOCUMENTATION] subdirectory of the HP Fortran product directory on the Software Product Library CD-ROM.

1.2 The HP Fortran Home Page

If you have Internet access and a web browser, you are welcome to view the following:

- The HP Fortran home page, located at the following URL:

  http://www.hp.com/software/fortran/

- The Hewlett-Packard Company home page, located at the following URL:

  http://www.hp.com/

1.3 Getting Help and Reporting Problems

If you encounter a problem while using HP Fortran, report it to HP.

If an error occurs while HP Fortran is in use and you believe the error is caused by a problem with HP Fortran, take one of the following actions:

- If you have a Software Product Services Support Agreement, consider contacting your Customer Support Center (CSC) by telephone (in the United States, 1-800-354-9000) or by using the electronic means provided with your support agreement. You can use DSNlink or other electronic means to report the problem to the CSC.
- Customers without a service contract can arrange for per-call CSC support.

When you initially contact the CSC, please indicate the following:

- The name and version number of the operating system (OpenVMS Alpha) you are using
- The name (HP Fortran or HP Fortran 77) and complete version number of the compiler you are using (The version number is displayed at the end of a compiler listing file)
- The hardware system you are using (such as a model number)
- How critical the problem is
- A very brief description of the problem (one sentence if possible)

When you submit information electronically or are speaking on the phone to the appropriate support specialist, you can provide more detailed information. This includes the specific commands used to compile and link the program, the error messages displayed, and relevant detailed information (possibly including source program listings). Please try to narrow the cause of the problem to a specific source module or lines of code.

CSC personnel may ask for additional information, such as listings of any command files, INCLUDE files, relevant data files, and so forth. If the program is longer than 50 lines, submit a copy of it electronically or provide machine-readable media (floppy diskette or magnetic tape).

Experience shows that problem reports sometimes do not contain enough information to duplicate or identify the problem. Concise, complete information helps HP give accurate and timely service to software problems.

To obtain information about purchasing HP support services, please contact your local HP sales representative.

You may also send comments, questions and suggestions about the HP Fortran product to the following Internet mail address: fortran@hp.com. Please note that this address is for informational inquiries and is not a formal support channel.

2
New and Changed Features

2.1 New and Changed Features in V7.6

This section provides highlights of problems corrected in HP Fortran V7.6

- The image identification string for the Fortran 95 compiler is V7.6-3276-48D52.
- The image identification string for the Fortran 77 compiler is V7.6-198-48D52.

2.1.1 Changes specific to HP Fortran

- 2632 - Generate correct code when character function length expression references a module procedure later in the module.
- 2637 - Diagnose passing an assumed-size array to an ELEMENTAL subroutine.
- 2663 - Eliminate spurious error for integer POINTER pointee being a procedure whose interface is visible through a USE in a host scope when IMPLICIT NONE is used.
- 2668 - Properly handle the SIZE intrinsic when the first argument is an array expression and the optional DIM argument is used.
- 2674 - Allow NO_ARG_CHECK attribute in contained procedures. Allow it to disable the check for passing a scalar to an array argument.
- 2688 - Correct parsing problem for C-string literals containing a "!" character.
- 2691 - Allow the KIND argument to be any INTEGER kind for the following intrinsics: AINT, ANINT, CEILING, CHAR, CMPLX, FLOOR, INT, LOGICAL, NINT and REAL.
- 2703 - Correct parsing error for certain expressions containing dot field component separators.
- 2762 - Eliminate internal compiler error for confusing generic routine references.
- 2770 - Suppress unused variable warning for RESULT variable in INTERFACE.
- 2792 - Allow optional KIND for ICHAR.
- 2824 - Allow specific module procedure to have the same name as the generic.
- 2876 - Eliminate internal compiler error for certain uses of character substring expression with no upper bound (eg. C(i:)), when variable is a passed-length variable in contexts where the upper bound is not needed.

- 2880 - Correct various problems involving array constructors of entities with allocatable components.
- 2883 - Allow .XOR. to be user (re)defined for two logical operands.
- 2887 - Do not sign-extend addresses when building 64-bit descriptors.
- 2906 - Allow A(N+1:) in DATA.
- 2907 - Make EXTERNAL take precedence over a module procedure of the same name.
- 2908 - Issue standards warnings for too many continuation lines.
- 2909 - Issue standards warning for array constructor character elements of differing lengths (extension to F95, but to be standard in F200x!)
- 2945 - Allow array constant elements in array constructors.
- 2946 - In free-form source, properly parse continued character literal that ends with "!" characters.
- 2947 - Correct parsing of field references to variables named PUBLIC or PRIVATE when used with dot separators.
- 2956 - Allow subobjects of constants in DATA.
- 2958 - Don't require delimiter following repeat group in FORMAT.
- 2967 - Allow w and d specifications for FORMAT items to be omitted in contexts where delimiters are omitted.
- 2968 - Allow initialized RECORD arrays of more than 32766 elements.
- 2970 - Prevent EQUIVALENCE from causing symbol "leakage" from modules.
- 2993 - Disallow calls to non-pure intrinsics from pure procedures.
- 3002 - Correct problem when only use of use-associated symbol is as a KIND specifier in the FUNCTION statement.
- 3010 - Diagnose INTENT(OUT) or (INOUT) arguments to PURE functions.
- 3012 - Allow CHARACTER(0) FUNCTION.
- 3036 - Issue standards warning for recursive use of function without RESULT.
- 3042 - Do not require slashes around common block names in ATTRIBUTES C, STDCALL, DEFAULT or ALIAS.
- 3063 - Eliminate internal compiler error when the pointer in an integer POINTER declaration is made PRIVATE.
- 3091 - Eliminate internal compiler error for TRANSFER in an initialization expression.
- 3158 - Correctly handle operator precedence in expressions involving dot field separators.
- 3172 - Eliminate internal compiler error processing !DEC$ conditional compilation directives when an END SUBROUTINE is missing.
- 3173 - Disallow forward references to type declarations.
- 3187 - Suppress message abouy undefined function value if value defined by storage association among ENTRYs.
- 3193 - Correct bad code for blank common in MODULE that is not USEd.

- 3194 - Disallow statement function in specification expression.
- 3210 - Correct generated code for bounds expressions computed for ENTRY points where arguments are in different order.
- 3223 - Significantly speed up compilation for initialization of very large arrays to a single value.
- 3224 - Eliminate spurious error for user definition of OPERATOR(-) when ONLY is used to pull in the definition.
- 3239 - Diagnose as non-standard missing PRIVATE attribute on derived type which contains PRIVATE user-type component.
- 3248 - Disallow array values for UNIT and FILE= in OPEN (except that, as an extension, FILE may be a numeric array.)
- 3264 - Improve consistency of checks where an implicitly typed named constant is named in a later type declaration.

3

New and Changed Features in Previous Releases

3.1 New and Changed Features in V7.5 ECO 01

This section provides highlights of new and changed features and problems corrected in Compaq FortranV7.5 ECO 01

- The image identification string for the Fortran 95 compiler is V7.5-2630-48C8L.
- The image identification string for the Fortran 77 compiler is X7.5-197-48C8L.

3.1.1 Changes specific to Compaq Fortran

- 1965 - Disallow situation where, through renaming, the same entity could be known by more than one local name in a scoping unit.
- 1967 - Allow REAL(16), COMPLEX(8, 16), derived types to be passed by value on all platforms.
- 1968 - Eliminate internal compiler error for initialization expression involving array constructor PARAMETER.
- 1985 - Correct a problem where some old-format module files could not be used by a newer compiler.
- 1992 - Allow the full appropriate range of signed integer constants which contain an explicit kind specifier.
- 2002 - Promote mixed-kind arguments to shift intrinsics to INTEGER(8) as needed.
- 2005 - Eliminate internal compiler error for INTERFACE block defining function with ALLOCATABLE result.
- 2007 - Disallow an array from having the same name as a derived type or STRUCTURE. This is not legal in Fortran 90, as syntactically it looks like a structure constructor, but it was allowed in Compaq Fortran 77 (which didn't have structure constructors.) In some cases, the compiler issued inappropriate and different error messages and/or an internal compiler error.
- 2008 - In structure constructor, allow typeless PARAMETER constant to take type from context.
- 2022 - Eliminate internal compiler error for TRANSFER of long character literal to numeric array.
- 2024 - Generate correct result of TRANSFER of array constructor in initialization expression.

- 2026 - Correct problem with visibility of sibling contained procedure as specification function.
- 2027 - Improve error message for I/O list with extraneous parenthesis.
- 2036 - Allow LOG and LOG10 to be passed as actual arguments. Eliminate spurious diagnostic for use of non-default kind in CASE that matches SELECT CASE. Eliminate internal compiler error for EOSHIFT of INTEGER(2) array. Give an error for an empty array constructor.
- 2037 - Allow combination of null ONLY list (ONLY:) and non-null ONLY list in a later USE.
- 2038 - Disallow COMPLEX*16 second argument to DCMPLX, COMPLEX*32 second argument to QCMPLX.
- 2039 - Give error for null CASE range.
- 2063 - Correct import of COMMON variable in USE, ONLY in contained procedure.
- 2064 - Don't allow parenthesization of expression hide compile-time overflow detection.
- 2070 - Improve detection of duplicate signature detection for generics.
- 2075 - Make ANINT generic.
- 2108 - Correct problem involving passing a discontiguous pointer array slice as an actual argument where a temporary is needed but wasn't being created.
- 2115 - Give standards warning for use of character literal constant in Hollerith context in WHERE.
- 2126 - Eliminate possible internal compiler error caused by attempt to put out ARG_TEMP_CREATED diagnostic message.
- 2137 - Eliminate internal compiler error for USE of blank COMMON in a module.
- 2157 - Don't give standards warning for use of host associated PARAMETER constant as a constant kind specifier in a statement function.
- 2197 - Give standards warning for consecutive commas in an argument list.
- 2206 - Eliminate internal compiler error for omitted optional argument to PURE function in specification expression.
- 2214 - Give standards warning for comma between control list and I/O list.
- 2216 - Make PACK directive take precedence over /ALIGN compiler switch. Add 8 as a PACK option.
- 2221 - Correct problem involving import of COMMON from a module where different imports make different sets of symbols visible.
- 2223 - When calling an intrinsic that needs to know the size of default integer (eg. DATE_AND_TIME), make sure that we look at the effects of !DEC$ INTEGER and OPTIONS /[NO]I4.
- 2245 - Disallow character literal scalar to match character*1 array of same length in argument lists.
- 2246 - Disallow use of an array as a substring bound.
- 2248 - Correct problem involving nested renames.

- 2249 - Allow ALLOCATED to be used on an ALLOCATABLE derived type component.
- 2260 - Do not automatically deallocate on exit allocatable components of INTENT(OUT) dummy arguments that have initialization.
- 2264 - Allow logical operators to be extended with non-numeric argument types.
- 2267 - Improve way that source filespecs are handled so that the compiler lets the OS determine what the syntax of a valid filespec is, rather than looking for what it thinks is a delimiter. In particular, embedded quotes are now properly handled.
- 2270 - Avoid corrupting descriptor for allocatable array dummy argument when debugging information was requested. The usual symptom of this was an inappropriate array bounds violation error.
- 2271 - Correct problem where ALLOCATED on an allocatable component passed as an actual argument might not return the correct result.
- 2275 - For INTENT(OUT) allocatable component dummy arguments, deallocate them in the prologue of only those entry points where the component is in the argument list.
- 2293 - If a compiler-generated external name would exceed the maximum length supported by the platform's linker, shorten it and give a warning.
- 2329 - Disallow passing an element of a POINTER array to a POINTER argument.
- 2342 - Allow an pointer to derived type with allocatable component to be initialized to NULL( ).
- 2344 - Eliminate spurious shape conformance error message for nested MATMUL.
- 2345 - Disallow "PRINT (101)".
- 2361 - Correct miscellaneous ASSOCIATED problems.
- 2363 - Eliminate I/O list-format synchronization problems caused by inappropriate implied-DO loop collapsing when format contains VFEs.
- 2374 - Disallow any statements from appearing between a SELECT CASE and the first CASE. We were allowing FORMAT and DATA here.
- 2376 - F95 Interp 80 - Do not allow a datatype to be implicitly assigned to what looks like a function reference to a host-associated EXTERNAL.
- 2377 - Do not give standards warning for use of BOZ constant as repeat count in DATA initializer when a regular error would have been given anyway.
- 2378 - Improve and correct diagnostics reflecting F95 interp 70.
- 2389 - Correct behavior to reflect F95 interp 72 relating to generics and elemental references.
- 2395 - Correct behavior to reflect F95 interp 83 regarding scope of implied-DO loop variable in array constructors.
- 2398 - F95 interp 70; allow length specification of char-len-param-value or array spec to be a specification expression if object is automatic.
- 2399 - Allow KIQINT, KIQNNT to be passed as actual arguments.

- 2400 - Add F95 standards checking for F2K ehancements.
- 2401 - Correct problem where if a module defines a derived type and operators on that type, but a USE imports ONLY the type and not the operator, the compiler still makes the operator visible.
- 2411 - Correct name of external math library routine used when KIXINT is passed as an actual argument.
- 2415 - Diagnose as non-standard the use of a non-intrinsic elemental as an actual argument.
- 2435 - Allow passing a PURE procedure as an actual argument when the corresponding dummy is not declared PURE.
- 2437 - If a routine name is explicitly declared INTRINSIC, do not assume EXTERNAL if argument types don't match.
- 2455 - Change compiler to assume that dummy arguments with TARGET attribute are aliased in all cases, not just for pointer assignments.
- 2457 - Disallow OPTIONAL dummy argument for DIM argument to DIM, etc.
- 2458 - Properly propagate TARGET attribute to ALLOCATABLE subcomponents of POINTER object.
- 2476 - Correct visibility problems for COMMON variables in a module in the presence of only and renames.
- 2477 - Detect invalid case of global symbol referred to by two names through nested rename.
- 2482 - More changes to accomodate F95 interpretation 70 dealing with what is allowed (or not) in an automatic bounds or character length expression.
- 2486 - Properly report error (instead of internal compiler error) when ALLOCATED is used on non-ALLOCATABLE derived type that contains an ALLOCATABLE component.
- 2559 - More work for COMMON variable with only and rename.
- 2560 - In FORALL, don't confuse subscript name with a derived type component of the same name.
- 2576 - Support %FILL in STRUCTUREs in MODULEs.
- 2577 - Correct problem with defined operator visibility.
- 2578 - Eliminate internal compiler error for reference to PARAMETER structure array in structure constructor.
- 2580 - Properly support use of pointer-returning functions in ASSOCIATED.
- 2582 - Eliminate spurious standards warning for dataype of PURE function used in specification expression.
- 2591 - Disallow passing array section to ALLOCATABLE argument. Disallow passing assumed-size array to elemental procedure where the shape is required.
- 2594 - Eliminate internal compiler error for nested TRANSFER in initialization expression.
- 2595 - Disallow use of scalar in structure constructor for allocatable component.

- 2607 - Do not issue "unbalanced parentheses" diagnostic for code that is not being compiled due to !DEC$ IF.
- 2608 - Correct problems with LBOUND, UBOUND and SHAPE of array components of nested derived type PARAMETERs.
- 2612 - Eliminate internal compiler error when the compiled module referenced in a nested USE can't be opened.

3.2 New and Changed Features in V7.5

This section provides highlights of new and changed features and problems corrected in Compaq Fortran V7.5

The image identification string for the Fortran 95 compiler is V7.5-1961-48BCD.
The image identification string for the Fortran 77 compiler is V7.5-197-48BCD.

3.2.1 Changes specific to Compaq Fortran

3.2.1.1 New /ASSUME=F77RTL compile qualifier

Version 7.5 adds support for the /ASSUME=[NO]F77RTL qualifier, for compatibility with the -f77rtl switch of Compaq Fortran on other platforms. If /ASSUME=F77RTL is in effect, I/O defaults change to match that of Compaq Fortran 77 compilers:

- In OPEN, the default is PAD='NO'.
- In NAMELIST output, the Compaq Fortran $ delimiters are output.

The default is /ASSUME=NOF77RTL.

3.2.1.2 Additional changes and corrected problems in V7.5

- 1687 - Improve diagnostics for ONLY symbols not found.
- 1707 - Correct a problem when, in a given program unit a reference is made to a field whose type is private when imported from one module, but public when imported from another, the visible field is not recognized.
- 1708 - When a typeless constant is passed as an actual argument, always treat it as INTEGER(8), not "default integer".
- 1712 - Eliminate internal compiler error for nested initialized pointer type.
- 1713 - Eliminate invalid warnings when :: used with initialization in STRUCTURE field declarations.
- 1714 - Eliminate internal compiler error for invalid use of array member in scalar structure constructor.
- 1720 - Correct problems when omitted OPTIONAL arguments passed to MAX and MIN.
- 1723 - ASSOCIATED of POINTER and TARGET components sometimes returns wrong result.
- 1724 - Significantly improve compilation time for certain large (and usually machine-generated) programs when unitialized variable detection is not selected and optimization level is 0.
- 1739 - Disallow OPTIONAL argument in character length specification expression.

- 1743 - Correctly recognize accessibility of defined assignment procedures imported from multiple modules.
- 1754 - Eliminate internal compiler error when alternate return specifier used with /ANALYSIS_DATA.
- 1757 - Correct parsing error when field name matches operator name and dot separators used.
- 1758 - Diagnose ambiguous generic involving OPTIONAL arguments.
- 1774 - Give diagnostic rather than internal compiler error for use of list-directed ENCODE/DECODE (not supported).
- 1778 - Eliminate internal compiler error when /CHECK=BOUNDS used and a non-contiguous array copy is passed.
- 1779 - Eliminate internal compiler error for nested WHERE and outer WHERE array is INTEGER(2).
- 1785 - Correct offsets of UNIONs when compiled /NOALIGN.
- 1786 - Warning about unused statement functions is now informational.
- 1789 - Correct generated code for complex case of user specification function used to declare size of character function result.
- 1794 - Eliminate internal compiler error for non-standard program that uses intrinsics in array constructors for initialization of character parameters.
- 1796 - When dummy argument of procedure in an interface block is an undetermined procedure (subroutine or function) - issue diagnostic if the procedure is referenced with an actual array argument.
- 1797 - Diagnose conflict between POINTER and COMMON.
- 1802 - Diagnose invalid reference to component of another function's result variable.
- 1806 - Recognize use of renamed intrinsic in initialization expression.
- 1812 - Eliminate internal compiler error for CHARACTER function whose result length is based on an argument subcomponent.
- 1813 - Diagnose a duplicate name for interfaces from different modules if at least one, but not all, are generic.
- 1823 - Eliminate incorrect alignment warnings for case involving COMMON and EQUIVALENCE.
- 1827 - Properly ignore END MODULE if it occurs within a conditional compilation section that is not being compiled.
- 1828 - Properly handle a structure constructor with variables in an argument list.
- 1836 - Eliminate excessive virtual memory consumption when compiling sources with very deeply nested USE references.
- 1837 - Allow INQUIRE(IOLENGTH=) for POINTER and ALLOCATABLE array references.
- 1841 - Eliminate internal compiler error for use of certain expressions in array constructor in initialization expression.

- 1843 - Establish that STRUCTUREs with field initialization imply SAVE semantics (compatible with past Compaq/DEC implementations) but Fortran 90 derived types with component initialization do NOT imply SAVE (per F95 standard).
- 1846 - Properly assign default integer type to an array constructor whose elements are all typeless constants.
- 1847 - Properly handle renaming of user-extended intrinsics.
- 1848 - Properly establish type of #nnnn constants in CASE expressions.
- 1858 - Eliminate erroneous overflow warning (and bad generated code) for use of 64-bit integer constant as argument to character function which uses the argument to determine the result length.
- 1865 - Disallow subcomponents of pointers as the pointer argument to ASSOCIATED.
- 1872 - Correct parsing error for expression involving record fields with dot separator.
- 1883 - Add /ASSUME=[NO]F77RTL.
- 1888 - Correct parsing error for expression involving record fields with dot separator.
- 1896 - If IMPLICIT NONE is in effect, allow a specification function to follow its use.
- 1909 - Correct problem where call of LIB$ESTABLISH in a function overwrites the function return value
- 1911 - Allow IMPLICIT type ($)
- 1914 - Fix problem with wrong value for certain expressions in implied-DO-loop array constructors.
- 1916 - Restore ability to USE certain MODULE files generated by older compiler versions.
- 1918 - Fix field alignment of substructure in UNION.
- 1928 - Don't give standards warning for "*-" combination in a DATA initializer when the * is a repeat count.
- 1929 - Allow LEN of an INTENT(OUT) argument in a specification expression.
- 1930 - When /NAMES=AS_IS specified, keep track of separate IMPLICIT entries for uppercase and lowercase letters.
- 1937 - Eliminate incorrect diagnostic for use of bin/oct/hex constant in an initialization expression in type statements, but give standards warning if requested.
- 1953 - Disallow passing a scalar actual argument to an array dummy argument.
- 1956 - Fix problem where allocatable components of a module derived type were being incorrectly deallocated on routine exit.
- 1961 - Allow ASSIGN 10 TO FNAME where FNAME is the current function.

3.2.2 Changes specific to Compaq Fortran 77

No functional changes - rebuilt using new optimizer/code-generator version.

3.3 New and Changed Features in V7.4

This section provides highlights of new and changed features and problems corrected in Compaq Fortran V7.4B

The image identification string for the Fortran 95 compiler is V7.4B-1684-46B5P.
The image identification string for the Fortran 77 compiler is V7.4B-197-46B5P.

3.3.1 Changes specific to Compaq Fortran

3.3.1.1 Change in behavior for initialized structures/derived types

The Fortran 95 specifies that a derived type object with component initialization does not acquire an implicit SAVE attribute, unlike objects of non-derived types. In addition, the standard specifies that derived type objects that are local (declared in a routine) and which do not have SAVE explicitly specified, have their component initializations reapplied on each entry to the routine.
Prior to version 7.4B, the compiler treated all objects with initialization as implicitly SAVE, but 7.4B now correctly implements the standard's rules for derived type objects.

This change has no impact on programs which use the DEC Fortran STRUCTURE extension with initialization specified for fields, for example:

```
      STRUCTURE /STRUCT/
        INTEGER F1 /3/
 END STRUCTURE
```

As in earlier versions, variables of STRUCTURE type with initialized fields imply the SAVE attribute. This is the only significant difference in semantics between STRUCTURE/RECORD and Fortran 90 derived types. If you wish variables of derived type to be SAVEd, then explicitly name them in a SAVE declaration.

3.3.1.2 New /ASSUME=[NO]PROTECT_CONSTANTS qualifier

By default, when a constant is passed as an actual argument, attempting to modify the associated dummy argument results in an access violation, as the constant is allocated in read-only memory. Such attempts to modify a constant argument are disallowed by the Fortran standard, but some customers have requested an option to allow such modifications to succeed (without changing the original constant value) in much the same way that Compaq Fortran already allows modification of arguments associated with an expression.

The new [NO]PROTECT_CONSTANTS keyword for the /ASSUME qualifier can be used to specify the behavior when constants are passed. The default is PROTECT_CONSTANTS, which disallows modification of or assignment to dummy arguments associated with constant actual arguments. If NOPROTECT_CONSTANTS is specified, the compiler passes a temporary copy of the constant instead. This copy can be modified and assigned to - the results will be discarded when the routine returns.

3.3.1.3 New /ANNOTATIONS qualifier

The new /ANNOTATIONS qualifier requests that annotations be added to the source listing indicating which of a set of optimizations were applied to particular parts of the source. Available keywords include CODE, DETAIL, FEEDBACK, INLINING, LOOP_TRANSFORMS, LOOP_UNROLLING, PREFETCHING, SHRINKWRAPPING, SOFTWARE_PIPELINING, TAIL_CALLS and TAIL_ RECURSION. Keywords ALL or NONE may be used to specify all or no annotations. Annotation keywords may exist for optimizations not supported on this platform. Annotations can be helpful in understanding why the compiler was unable to optimize a particular code sequence or to see what optimizations were performed. For more details, please see HELP FORTRAN /ANNOTATIONS or the User Manual.

3.3.1.4 New /CHECK=[NO]ARG_TEMP_CREATED qualifier

The new [NO]ARG_TENP_CREATED keyword for the /CHECK qualifier specifies whether or not the compiler generates code to provide run-time warnings when a temporary copy of an actual argument is made prior to a call. Such copies can often be time-consuming and are usually due to passing a deferred-shape array or array section to a routine for which there is no explicit interface or the interface specifies an explicit shape array. The default is to not issue the warning.

3.3.1.5 Optional KIND= specifier on selected intrinsics

An optional KIND argument is now allowed on the intrinsics LEN, SHAPE, SIZE, UBOUND, LBOUND, MAXLOC, MINLOC, INDEX, LEN_TRIM, SCAN, and VERIFY. This allows these intrinsics to return a result that is other than default integer kind. Without the KIND= specifier, these intrinsics always return a result of default integer kind.

3.3.1.6 New INT_PTR_KIND intrinsic

The new INT_PTR_KIND( ) intrinsic returns the KIND number of an integer pointer for the platform being compiled on. This allows you to declare INTEGER variables which are big enough to hold an address without platform-specific code. The result of INT_PTR_KIND is 8 on OpenVMS Alpha, Tru64 UNIX and Linux Alpha, 4 on Windows. INT_PTR_KIND may be used in the same contexts as SELECTED_INT_KIND - it does not take any arguments.

3.3.1.7 64-bit support for MALLOC/FREE

When the arguments to the MALLOC and FREE intrinsics are of type INTEGER*8, the corresponding 64-bit Run-Time Library routine will be called. This allows you to use MALLOC to allocate data in 64-bit (P3) space by specifying an INTEGER*8 size argument. Note that the 64-bit FREE routine can free storage allocated in either 64 or 32-bit space.

3.3.1.8 Change in behavior for ERR=

In order to comply with clarified wording in the Fortran standard, the compiler and run-time library have changed the behavior when an end-of-file condition occurs during a READ statement and there is an ERR= specifier but neither a END= nor IOSTAT= specifier. In previous versions, this condition would cause the ERR= branch to be taken for an end-of-file condition. However, the standard has new wording which makes it clear that the correct behavior is to terminate the program (with an error message). Similarly, if an end-of-record condition occurs and there is not an EOR= nor IOSTAT= specifier, an error will be signalled rather than take the ERR= branch.

This change will be applied to programs which are recompiled with the new compiler - already-linked programs will continue the old, incorrect behavior. To ensure correct and portable behavior, be sure to use END=, EOR= and/or IOSTAT= where appropriate and do not assume that end-of-file or end-of-record conditions will trigger an ERR= branch.

3.3.1.9 Additional changes and corrected problems in V7.4

- Eliminate many unnecessary copies of assumed-shape arrays when passed as arguments to explicit shape arrays by keeping track of whether or not the array is known to be contiguous.
- When /SEPARATE_COMPILATION is used, and there are blank lines following the END of the final program unit, do not cause the object file to be deleted.
- Automatically force the alignment of a COMMON block to be at least as large as that required by the widest variable in the COMMON block.
- Fix a number of problems with WHERE inside of FORALL.
- Make defined assignment in FORALL work properly.
- Generate correct code for CSHIFT of a non-contiguous array slice.
- Allow user-defined types to be named BYTE and DOUBLECOMPLEX.
- Improve generated code for mixed COMPLEX-REAL multiplication and division.
- In array constructors with only one implied-DO, and nothing else, avoid creating an unnecessary temp.
- Allow SIZEOF(allocatable-array)
- Improve generated code for SIZEOF(array)
- Prevent incorrect collapsing of implied-DO loop in an I/O statement where there are nested loops and the implied-DO variable of one loop is used as a bound of an inner loop.
- When the error limit has been exceeded (default 30), simply suppress further messages rather than exiting the compiler. This means that the object file will get deleted appropriately, the listing and diagnostic (VMS) files created, and subsequent source files on the command line will be processed.
- Re-allow complex constants to be passed by value. The real and imaginary parts are passed as separate arguments.
- Allow character array valued function as a format specifier.
- Allow debugging of a character function whose length is computed based on the length of a passed-length character argument.
- Eliminate internal compiler error for character function whose length expression includes LEN_TRIM of a dummy argument.
- Eliminate internal compiler error for SIZE of a derived type component pointer array.
- Elmininate more unnecessary copies of contiguous array arguments.
- Speed up processing of EQUIVALENCE groups in COMMON blocks.
- Properly handle SIZE(A(:)(I:J))

- Implement INT8 intrinsic (already documented)
- Allow omitted OPTIONAL arguments to be passed as optional arguments to intrinsics.
- If "too many errors", suppress informational/warnin messages as well.
- Allow keyword specification of arguments A10, A20, etc. to MAX/MIN.
- Eliminate internal compiler error for case of ADJUSTL with array argument.
- Where DATA initializations initialize the same array element (not allowed according to the standard, but supported by Compaq Fortran), preserve the order in which initializations were specified, so that the last one takes precedence.
- Support the full range of format width specifiers as documented.
- Correct problem where bounds for an adjustable array were not being computed properly at an ENTRY.
- Improve detection of contiguous array slices.
- Allow EOSHIFT to work on REAL*16 and COMPLEX*32 arrays
- Correct support for SIZE with argument being array with vector subscripts.
- Add support for /ASSUME=[NO]PROTECT_CONSTANTS.
- Eliminate inappropriate error messages for certain case involving multiple ENTRYs for a FUNCTION where two ENTRYs have an array function result.
- Don't give standards warning for MAX(0,2-1).
- Make sure that variables of derived type where the derived type has initialized components are SAVEd.
- Diagnose conflicting use of /IEEE_MODE and /ROUNDING_MODE when VAX float is in use.
- Correct problem where IARGCOUNT could return wrong result when /NOOPT.
- If argument to MALLOC/FREE is INTEGER*8, call the corresponding 64-bit entry point.
- Eliminate GEM assertion failure when DATA implied-DO loop has negative stride.
- Correct problem with FORALL when overlap is detected and datatype is CHARACTER.
- Improve performance by suppressing the copy-back of arguments with INTENT(IN).
- Add support for -mixed_str_len_arg switch on UNIX, which puts character argument lengths next to their addresses in the argument list (this is the default Windows method). Add support for [NO]MIXED_STR_LEN_ARG attribute in !DEC$ ATTRIBUTES.
- Eliminate spurious error for declaration of RECORD array in a COMMON block in a MODULE.
- Correct handling of generics where several routines have an optional argument at the same position.

- In C strings, allow octals of the form \ooo to have 1, 2 or 3 octal digits and hex objects of the form \Xxx to have 1 or 2 hexadecimal chars.
- Generate debug symbol table information for labels, allowing SET BREAK %LABEL nnn.
- Add support for /check=arg_temp_created
- Annotations now are displayed on a per-routine basis.
- Enhance debug information for module functions.
- Eliminate internal error for SIZEOF of POINTER array component of derived type.
- Check for overflow in computation of size for ALLOCATE
- Correct problem in certain cases of an omitted OPTIONAL argument passed as the MASK argument to MAXLOC, etc.
- Allow character substring assignment in FORALL.
- Prevent blank COMMON definition from "leaking" from a module when USE, ONLY does not name a COMMON variable.
- Correct problem involving PRIVATE component of non-PRIVATE derived type.
- Fix internal compiler error when COMMON in a MODULE has an ALIAS.
- Improve generated code for ISHC.
- When passing NINT as an actual argument, pass address of JNINT, not ININT.
- When passing IDINT as an actual argument, pass address of JIDINT, not IIDINT.
- Disallow CHARACTER value for RECL= in OPEN.
- Fix internal compiler error involving substring assignment in FORALL.
- /STANDARD now means /STANDARD=F95.
- Eliminate spurious "shapes do not conform" error when RESHAPE is used in a PARAMETER declaration.
- Eliminate spurious error when generic interface and specific module procedure have the same name.
- Correctly diagnose error when procedure passed as argument where interface calls for numeric.
- Diagnose prefetch arguments which are not variables or array element references or structure component references.
- Diagnose missing INTENT on PURE subroutine arguments which are not procedures nor pointers.
- Diagnose mask expressions with incompatible shapes in where constructs and contained where/elsewhere.
- Enforce standard's requirement that user-defined operator names must consist of letters only.
- Don't give errors for "invalid continuation" in !DEC$ IF block that is not to be compiled.

- Correct the way that COMPLEX arguments are passed by value, and eliminate internal compiler error for some cases of passing COMPLEX by value. The correct way is to treat a COMPLEX as a pair of REAL arguments.
- Do not issue "source line truncated" warnings when F95 standards checking is enabled unless /WARN=TRUNCATED_SOURCE is specified.
- Give proper error, instead of internal compiler error, when a type is incorrectly superseded.
- Correct parsing problems in certain cases where dot field separator is used and the name to the left of the dot ends in digits.
- Correct problem with INQUIRE(IOLENGTH=FIELD%NAME).
- Detect more cases when copy-in/copy-out is not needed.
- Do not diagnose fields of private types as being inaccessible within the defining module.
- Disallow ALLOW_NULL/REFERENCE attributes for assumed-shape array arguments.
- Allow dummy arguments to be used as arguments to inquiry functions in specification expressions without requiring INTENT(IN).
- Fix problem with FORALL masks when /INTEGER_SIZE=64.
- Diagnose invalid value in bin/hex/octal constant when used as operand to unary operator.
- Accept $ as the end of the range in IMPLICIT.
- Allow a run-time format to be a character array element.
- Add EV68 as an option to /ARCH and /OPT=TUNE=.
- Correct problem where %LOC(funcname) inside the named function sometimes returns the address of the function rather than its result variable.
- Diagnose ambiguous generic if optional and non-optional specifications match.
- Eliminate internal compiler error for untyped implied-DO-in-DATA variable with IMPLICIT NONE.
- Correct problem with PACK when the MASK is .FALSE. or is an expression whose value is not known at compile-time.
- Diagnose attempt to assign to a field of an undeclared derived type variable when there exists a function of that name elsewhere in the compilation.
- Allow "self-referential" inquiry intrinsics in specification expressions.
- Add support for KISHFTC.
- For typeless PARAMETER constant used in contexts where a particular type is assumed, make sure that type is used.
- Fix problem which occurs when a symbol which exists in a public list, is visible in one 'use' chain, and is mistakenly treated as 'public' when imported from a second use chain where it is private and invisible.
- Issue diagnostic for an actual argument which has a vector subscript if dummy has intent(out) or intent(inout). Do this even if arg of defined_ assignment subroutine.

- Eliminate internal compile error if /DEFINE with no list is seen.
- Allow CHARACTER*(n) %FILL without ::.
- Generate better debug information for procedure dummy arguments.
- Allow and honor ADDRESS64 on COMMON blocks.
- Fix SIZE(A,DIM) where DIM is an optional argument.
- Allow MAX and MIN to have omitted, optional arguments.
- Prevent omitted, optional CHARACTER argument from being passed anyway.
- Improve compile-time performance for USE inside of INTERFACE.
- Make older !DEC$ ALIAS work in INTERFACE blocks.
- Fix IARGCOUNT when in a routine with no arguments.
- Don't give error for dummy argument used in specification expression whose implicit datatype is confirmed by a later type declaration.
- Correct debug information for CHARACTER*(*) array dummy arguments.
- Improve generated code for ILEN.
- Treat use of undefined preprocessor symbol in expression as zero.
- Put out appropriate debug symbol table information for all ENTRY points.
- Eliminate spurious syntax error for particular use of period field separator in expression.
- Fix problem with WHERE in FORALL where WHERE mask contains a reference to an elemental function.
- Objects of derived type whose type contains initializations are no longer implicitly SAVEd, as required by the Fortran 95 standard. If such objects are local to a procedure, the initialization is repeated on each entry to the procedure. See Section 3.3.1.1 for more details.
- Correct handling of reference to SELECTED_REAL_KIND with an omitted, OPTIONAL argument passed to it.
- Allow dot field separator in DATA implied DO expression.
- Eliminate spurious error involving generic with procedure argument where that procedure has a pointer argument.
- Eliminate internal compiler error for module import of EQUIVALENCEd COMMON.
- Eliminate internal compiler error for reference to contained pure function in specification expression of another contained function.
- Eliminate spurious unused warning when only reference is as an argument to a statement function.
- Diagnose as an extension the use of a BOZ constant outside of DATA initialization.
- Correctly process ONLY: (with no list).
- Do not give warning for declaration of non-COMMON variable in BLOCK DATA if that variable is used in a DATA implied-DO.
- Correct parsing error for use of dot separator in ALLOCATE.

- Eliminate internal compiler error for a particularly complex and deeply nested structure reference in an IF.
- Don't give error for passing section of an assumed size array to a deferred shape array.
- Improve compilation speed for certain large DATA statements.
- Eliminate internal compiler error for a certain complicated FORALL.
- Correct problem with PUBLIC/PRIVATE attributes of a COMMON block in a MODULE.
- Allow (A .EQ. .NOT. B) (an extension).
- Correct problem with some COMPLEX*32 initialization expressions.
- Eliminate spurious unused variable diagnostic for variable used in pointer assignment.
- Correct problem where asking for standards checking disabled /define (-D)
- The use of an INTENT(OUT) argument with LOC is now considered a "definition" for the purpose of uninitialized variable checking. Also, the use of LOC(ARRAY(N)) is now considered a "use" of ARRAY for unused variable checking.
- Eliminate internal compiler error for structure with %FILL component in module procedure.
- When standards checking is requested, do not give a warning for fixed-form source line exactly 72 columns long.
- Eliminate internal compiler error for assignment statement with variable whose name starts with an underscore. (Such names are not allowed, but a reasonable error should have been given.)
- Correct the problem where if a program unit contains two internal subroutines which both use host-association to access the same variable, the second one gets an inappropriate error.
- Eliminate internal compiler error for declaration of a derived type array with an initializer that is an implied-DO array constructor.
- Eliminate inappropriate error message for initialization expression of an implied-DO array constructor of length zero.
- When standards checking is enabled, give one warning, not three, for a !DEC$ ATTRIBUTES directive.
- Generate correct code for certain cases involving overloading of the .AND. and .OR. operators.
- Allow Variable Format Expression in a character literal when the I/O list has a subscripted array element.
- Eliminate compiler abort with incorrect program that names the enclosing program unit in an ONLY clause.
- Allow the extension syntax '101'B for a binary literal.
- Fix problem where $INTEGER directive was not being properly handled.
- Add support for KIND= keyword in MINLOC/MAXLOC
- Add suppprt for KIND= keyword in various string intrinsics.

- Prevent compiler abort for incorrect attempt to pass unsupported datatypes by value.
- Properly report invalid declaration EXTERNAL,INTEGER and recover so that remainder of program is properly parsed.
- Give standards warning for non-standard placement of NAMELIST.
- Eliminate internal compiler error for particular type of concatenation of substrings when Fortran 95 standards checking is requested.
- When converting a negative REAL value to COMPLEX, use +0.0 as the imaginary part rather than -0.0.
- Allow PARAMETER constant in ALIGN= specification of !DEC$ PSECT.
- Don't give shape mismatch for correct-shape RESHAPE in initialization expression.
- Don't give inappropriate alignment warnings for certain convoluted EQUIVALENCE/ COMMON combinations.
- Eliminate internal compiler error for initialization expression which contains a constant expression in a structure constructor.
- Allow EXTERNAL :: FOO
- Correct parsing error for certain record component references using dots.
- Eliminate internal compiler error for particular use of RESHAPE.
- Do not give an unused variable warning where the only reference to a variable is as the argument to LOC.
- Eliminate internal compiler error for particular use of nested STRUCTURE.
- Disallow illegal ALLOCATED(A(:))
- Allow, as an extension for compatibility with our Fortran 77 compilers, a LOGICAL value as an argument to intrinsics which accept INTEGER, such as ABS.
- Diagnose ambiguous generic routine reference.
- When an integer constant is assigned to a variable whose KIND is smaller, don't change the KIND of the constant for future references.
- Allow IMPLICIT REAL with $.
- Eliminate internal compiler error for RESHAPE of array constructor
- Correctly diagnose use of same name for COMMON and BLOCK DATA.
- Fix parsing error for typed function declaration with MS-style argument passing specification in argument list.
- Resolve incorrect generic resolution between subroutine with no arguments (but ( ) specified as argument list) and routine with one required argument.
- Allow named constant to be used in specification expression later in same statement where it was defined.
- Give error instead of compiler abort for IF (expr) SELECT CASE
- For F90 standards checking, warn about I0, etc. in FORMAT.

- Warn about variables in BLOCK DATA subprogram that are not in a COMMON.

3.3.2 Changes specific to Compaq Fortran 77

3.3.2.1 /FAST now implies /ARCH=HOST/TUNE=HOST

The Fortran 77 compiler now implements the feature, previously introduced in the Fortran 95 compiler in V7.3, where specifying /FAST implies /ARCH=HOST/TUNE=HOST. For more information on the implications of this change, see Section 3.4.1.1 and Section 3.4.2.3.

3.3.2.2 Additional changes and corrected problems in V7.4

- Allow DFOR$PREFETCHxxx for COMPLEX data types S and F.
- Fix access to COMMON variables when /ASSUME=DUMMY_ALIASED.
- Use of the DICTIONARY statement no longer results in an inappropriate "Attribute not found" error message.

3.4 New and Changed Features in V7.3

This section provides highlights of new and changed features and problems corrected in Compaq Fortran V7.3

3.4.1 Changes common to both Compaq Fortran and Compaq Fortran 77

3.4.1.1 New EV67 keyword for /ARCHITECTURE and /OPTIMIZE=TUNE=

The EV67 keyword has been added to the list of processor types specifiable with the /ARCHITECTURE and /OPTIMIZE=TUNE= qualifiers. Specifying /ARCHITECTURE=EV67 allows the compiler to generate instructions which are supported by the EV67 revision of the Alpha 21264 processor architecture, including the CIX count instructions. If programs compiled with /ARCHITECTURE=EV67 are run on systems with earlier revisions of Alpha processors, the program may run slowly due to software emulation.

Specifying /OPTIMIZE=TUNE=EV67 directs the compiler to choose optimizations which are best for EV67 processors. These optimizations may not be optimal for earlier revisions of the Alpha architecture.

The default for both /ARCHITECTURE and /OPTIMIZE=TUNE is GENERIC, which assumes any Alpha architecture revision. However, if using the Fortan 95 (and Fortran 77 as of V7.3 ECO 01) compiler and /FAST is specified, the default changes to HOST, making the architecture revision selected the same as the compiling host system. This should be taken into account when distributing applications to run on other Alpha systems. The following changes are common to both Compaq Fortran and Compaq Fortran 77.

3.4.2 Changes specific to Compaq Fortran

3.4.2.1 New COMPLEX*32 datatype

The compiler now supports the COMPLEX*32 (COMPLEX(KIND=16)) datatype. This is a COMPLEX datatype whose real and imaginary parts are of type REAL*16 (IEEE X_float). All operations valid on COMPLEX datatypes are valid for COMPLEX*32, and the set of intrinsic functions has been expanded to support the type. See HELP FORTRAN Intrinsics for more details.
Note that the OpenVMS debugger does not currently support this datatype - you will not be able to EXAMINE or DEPOSIT variables of this type. Support for COMPLEX*32 will be added in a future release of OpenVMS.

The Fortran and Mathematics run-time libraries have been enhanced to support COMPLEX*32. This means that if your application uses this type, it must be linked and run on a system which has the corresponding FORRTL kit installed.

3.4.2.2 New /ALIGN=[NO]SEQUENCE qualifier

In previous versions of Compaq Fortran, derived types with the SEQUENCE attribute were always close-packed, with no alignment padding inserted, even if fields were unaligned. This is still the default behavior. However, a new [NO]SEQUENCE keyword has been added to the /ALIGN qualifier. If /ALIGN=SEQUENCE is specified, SEQUENCE types participate in alignment padding according to the currently in effect alignment rules.

The default is NOSEQUENCE, but if /FAST is specified, the default changes to /ALIGN=SEQUENCE. /ALIGN by itself or /ALIGN=ALL also enables /ALIGN=SEQUENCE.

3.4.2.3 Changes to /FAST qualifier

The /FAST qualifier is a quick way of selecting a number of other command-line options which usually improve run-time performance. In V7.3, /FAST additionally implies /ARCHITECTURE=HOST/OPTIMIZE=TUNE=HOST/ALIGN=SEQUENCE. The complete list of altered defaults specified by /FAST are:

```
/ALIGN=(COMMONS=NATURAL,RECORDS=NATURAL,SEQUENCE)
/ARCHITECTURE=HOST
/ASSUME=NOACCURACY_SENSITIVE
/MATH_LIBRARY=FAST
/OPTIMIZE=TUNE=HOST
```

You may specifiy individual qualifiers on the command line to override the /FAST defaults.

It is important to note that the changed default of /ARCHITECTURE=HOST may make programs run slowly on systems with Alpha processors which implement earlier revisions of the Alpha architecture than the host.

3.4.2.4 Additional changes and corrected problems in V7.3

- Eliminate internal compiler error for case involving EQUIVALENCEd POINTER variables not in COMMON.
- Fix problem which could cause incorrect code to be generated for certain uses of PACK, RESHAPE and UNPACK, primarily with CHARACTER arguments.
- Speed up compilation of initialized nested structures.
- Fix problem with incorrect code generated for use of nested SPREAD intrinsic.
- Correct problem where DO loop with loop ending value of the largest integer value would instead create an infinite loop.

- Eliminate internal compiler errors for use of defined assignment in FORALL.
- Speed up compilation where very large MODULEs are repeatedly used. Further work in this area remains.
- Correct problem with PACK intrinsic where array is array of structures with an array field with constant subscript/substring values.
- Improve generated code for SPREAD intrinsic.
- Improve generated code for array reductions.
- Improve generated code for UBOUND, LBOUND, SELECTED_INT_KIND, SELECTED_REAL_KIND when the argument(s) are constants.
- Improve generated code for SIZEOF when bounds are constants.
- When generating temporary copies of arrays, don't allocate them in 64-bit space unless we know that 64-bit addresses are acceptable.
- Eliminate internal compiler error for certain cases of integer (not F90) POINTER as a module variable.
- Reduce severity of "variable has not been used" diagnostic to "informational".
- Improve generated code for MINVAL/MAXVAL.
- Improve generated code for SIZE(A(:)).
- Improve generated code for SIZE with array argument that has vector subscripts.
- Add new INT_PTR_KIND( ) intrinsic which returns the kind of an integer pointer (either 4 or 8).
- Eliminate internal compiler error for use of allocatable array reference in a variable format expression.
- Improve generated code for ILEN.
- An end-of-file condition on READ no longer triggers an ERR= branch - this is to conform with clearer wording in the recent standard. If an EOF condition occurs and there is no END= or IOSTAT=, an error is signalled.
- Add a NUL to the end of non-C character literals. This will not be reflected in the "length" of the constant. This matches the Compaq Fortran 77 behavior, which was undocumented.
- %VAL/%REF now overrides any mechanism specified in an explicit interface.
- Generate much better code for certain array constructors (such as (/0,I=1,500000/)) and allow lowerbound, upperbound and stride to have different KINDs.
- Improve compile time performance when USEing a module which contains a large COMMON block with many (thousands) of EQUIVALENCEd variables.
- Allow general CHARACTER expressions for the MOLD argument of the TRANSFER intrinsic.
- Improve debug support of deferred-shape arrays of derived types.
- Correct problem that prevented scalar numeric variables and array elements which appeared in the iolist of an output statement (WRITE, etc.) from participating in uninitialized variable analysis.

- Missing !DEC$ ENDIF no longer causes compiler abort.
- Correct rules for when SAVE in a PURE procedure is allowed or not.
- Correct parse of assignment to field of RECORD whose name begins with TYPE.
- Eliminate E-level error when a BLOCK DATA subprogram name is the same as that of a COMMON block. A future revision will cause an appropriate diagnostic to appear.
- Issue clearer error message when a module name is repeated in a USE statement.
- Eliminate problem where UBOUND gave wrong value in certain cases.
- Allow substrings in left hand side of FORALL.
- Give proper error when a PARAMETER constant is CALLed.
- Give proper error when a variable is CALLed.
- In assignment statements, make sure that real constants are evaluated using the precision appropriate for their syntax (single/double/quad).
- Allow module ALLOCATABLE variables to be DLLIMPORTed.
- If -warn declarations is specified, do not issue diagnostic for use of IMPLICIT NONE.
- Eliminate internal compiler error for EOSHIFT with constant array argument.
- Eliminate internal compiler error for program fragment that is "ifdef-ed out".
- Eliminate internal compiler error for DLLEXPORT of ALLOCATABLE array.
- Report error for kind mismatch where an ac-value in an array constructor is an expression of a different kind than the other literal ac-values.
- Report error for assumed-size array passed as actual to deferred-shape array.
- Eliminate compiletime error for use of AIMAG in initialization expression when /real_size=128 is specified.
- Eliminate internal compiler error for format in a PRINT statement being an expression with concatenation.
- DFOR$PREFETCH, DFOR$PREFETCH_MODIFY, DFOR$PREFETCH_ EVICT_NEXT intrinsics now supported for Alpha processors.
- Generate correct values for a PARAMETER array whose element values are computed in an implied DO loop involving indexing into another PARAMETER array.
- Correct bad parse of .NE. as record field name.
- Allow a dummy argument that has an INTENT attribute specified to be specified in a NAMELIST list.
- Give standards warning for kind mismatch between source and pad arguments in RESHAPE.
- Long source lines are now correctly compiled when standards checking is requested (warning is still given.)

- Correct problem with incorrect error given for a particular complex module renaming case.
- Allow as an extension (as does Compaq Fortran 77) a LOGICAL argument to an intrinsic which expects INTEGER.
- Correctly parse format which contains ">" characters and a variable format expression.
- Eliminate access violation on some platforms for ALLOCATE of pointer in derived type.
- Correct problem where compiler could omit putting out declaration for a routine symbol.
- Handle non-present, optional dummy arguments as third argument to INDEX, SPAN and VERIFY.
- Generate correct code when passing character array slices as arguments.
- Fix case of contiguous array slice as first argument to TRANSFER.
- Fix INQUIRE by IOLIST of ALLOCATABLE arrays
- Correct problem involving pointer assignment with sections of a derived type.
- Allocate all PARAMETER constants in a read-only PSECT.
- Ensure that locally-allocated derived-type arrays are naturally aligned.
- Do COMMON/global-name conflict checking with /SEPARATE.
- With /ANALYSIS_DATA, give SCA the correct descriptor datatype for D_float and IEEE_float.
- Generate correct code for pointer assignment of an array generated from a section of a derived type.
- Eliminate internal compiler error in certain cases with dummy argument that has OPTIONAL and INTENT(OUT) attributes.
- Flag square-bracket array constructor syntax as an extension.
- Eliminate internal compiler error for certain uses of TRANSFER.
- Properly detect ambiguous generic reference when all distinguishing arguments are OPTIONAL.
- Eliminate internal compiler error for a case involving a PRIVATE POINTER in a module.
- "Directive not supported on this platform" diagnostic is now informational, not warning
- Allow array sections in DATA statement variable list
- Eliminate GEM assertion for certain program compiled /DEBUG/OPTIMIZE
- Correct problem with I/O of a slice of an assumed-size array.
- In listing summary, list zero-length COMMON PSECTs.
- Eliminate spurious warning when passing a POINTER or assumed-shape array in COMMON to a routine with a compatible dummy argument declaration.
- Fix internal compiler error involving array-valued functions with entry points.

- Generate correct code for unusual (and non-standard) dummy aliasing case involving an EQUIVALENCEd variable passed as an argument.
- Fix problem with incorrect code for a call to ALLOCATE or DEALLOCATE where STAT= is specified using an array element.
- Fix internal compiler error for certain programs which CALL a function.
- Correct compiler abort for module procedure with ENTRY.
- Allow full set of F95-permitted intrinsic functions in specification expressions.
- Correct compiler abort with invalid VFE in FORMAT.
- Correct problem with accessibility of MODULE symbols when two modules define the symbol but one has marked it PRIVATE.
- Correct problem with missing and duplicate alignment warnings.
- Allow repeated NULL( ) in DATA initialization when variables have different types.
- Correct spurious "shapes do not conform" error.
- Correct compiler abort for invalid program using wrong component in ASSOCIATED.
- When /names=as_is specified, don't make IMPLICIT case- sensitive.
- Give standards warning for Q exponent letter in floating literals.
- Generate correct code for generic which replaces MIN or MAX.
- Give more reasonable error message when variable used as control construct name.
- Eliminate spurious message for vector-valued subscript in defined assignment.
- Give error if INTENT not properly specified for defined assignment.
- Correct internal compiler error for overloaded MAX.
- Eliminate spurious warning for FORALL.
- Give warning when INTENT(IN) argument modified in PURE FUNCTION.
- Eliminate spurious error for valid DATA with array subscript.
- Allow ORDER in RESHAPE to be non-constant.
- Fix compiler abort with RESHAPE.
- Don't give unused warning for TARGET argument used in pointer assignment.
- Properly distinguish STRUCTUREs with the same name in different CONTAINed routines.
- Allow NULL( ) to initialize a pointer to derived type.
- Don't give unused warning for array constructor implied- DO variable.
- Allow INTRINSIC :: name (new in F95)
- Eliminate spurious standards warning for certain obscure uses of UNPACK.
- Eliminate compiler abort when transformational intrinsic used (illegally) in statement function.

- Raise limit of number of items in a FORMAT from 200 to 2048.
- Disallow invalid INTENT keywords
- Allow CALL of a typed identifier (DIGITAL Fortran 77 extension)
- Correct problem where USE-associated identifiers aren't seen in certain cases involving renaming.
- Correctly evaluate CEIL intrinsic when used in a specification expression.
- Allow SIZE intrinsic to be overloaded.
- Don't issue spurious "function value has not been defined" warning for case involving ENTRY and RESULT.
- Fix internal compiler error involving defined assignment.
- Fix problem with incorrect CHARACTER initialization values and CHAR function.
- Disallow array constructor being used to initialize a scalar.
- Allow ALLOCATE/DEALLOCATE of argument to PURE SUBROUTINE.
- Fix problem for certain uses of period separators for derived type fields.
- Eliminate spurious syntax error for use-associated variable in NAMELIST.
- Eliminate spurious syntax error for certain uses of variable format expression in FMT=.
- Allow as an extension the use of a name previously seen in a CALL statement as an actual argument without an EXTERNAL statement or explicit interface.
- Eliminate spurious overflow message for MS-style base-2 constant.
- Correct problem with generic routine matching.
- Correct internal compiler error when function return value used in statement function expression.
- Fix problem with wrong generated code if an OPTIONAL and omitted descriptor-based dummy argument is passed as an actual argument to a routine which declares that argument as OPTIONAL.
- Fix problem where ASSOCIATED did not always return the correct result for a pointer component that was transferred via pointer assignment.
- Enable display of array bounds larger than 32 bits in listing summary.
- Fix internal compiler error for certain uses of defined assignment where multiple defined operators appeared in the right-hand side of the assignment.
- Don't put initialized descriptors in the BSS PSECT, as this prevents demand-zero compression and can lead to much larger executable files.
- Make the default for BLANK= in OPEN match the documentation when the /NOF77 switch is specified, which is to default to BLANK='ZERO'. Previously, BLANK='NULL' was used regardless.
- Allow array constructors to have scalar CHARACTER source elements of varying size.
- Correct problem where a call to a routine with the C and VARYING attributes generates incorrect code.

- Fix internal compiler error for statement function which uses the function return variable of the host function.
- Fix internal compiler error for incorrect program which uses an component of a derived type variable in an automatic array bounds expression, the derived type is undefined and IMPLICIT NONE is used.
- Fix internal compiler error when RESULT variable has same name as a previously seen FUNCTION.
- Fix problem with PUBLIC/PRIVATE attributes in a particular complicated module usage.
- Eliminate spurious error message for valid generic procedure reference.
- Fix problem with DATA initialization of zero-origin arrays.
- Don't give "unused" warning for EQUIVALENCEd variable.
- Properly treat INT(n,KIND=) in an array constructor.
- Don't disable type checking for %LOC.
- Properly parse generic INTERFACE whose name begins with TO.
- When /ALIGN=COMMONS=NATURAL is used, make sure that POINTER objects in COMMON are aligned on quadword boundaries.
- Correctly parse program with IF construct whose name begins with IF.
- If an attempt is made to DEALLOCATE an item which is not DEALLOCATEable, such as an array slice, a run-time error is now given. Previously, the results were unpredictable.

3.4.3 Changes specific to Compaq Fortran 77

- Avoid ACCVIO when RECORD statement conflicts with previous use of name.
- Fix assertion when dummy arg name is the SAME as a SUBROUTINE/FUNCTION/ENTRY name already seen.

3.5 New and Changed Features in V7.2

This section provides highlights of new and changed features and problems corrected in Compaq Fortran Version 7.2.

3.5.1 Changes common to both Compaq Fortran and Compaq Fortran 77

The following changes are common to both Compaq Fortran and Compaq Fortran 77.

3.5.1.1 FORTRAN command invokes Fortran 95 compiler

With this release of Compaq Fortran, the FORTRAN verb invokes the Fortran 90/95 compiler by default. The Fortran 77 compiler is invoked using FORTRAN/OLD_F77. The F90 command continues to invoke the Fortran 95 (formerly Fortran 90) compiler.

No further development or enhancement will be done to the Fortran 77 compiler, although maintenance will continue for some time. Compaq recommends use of the Fortran 95 compiler for current development work - it accepts standard Fortran 77 (and Fortran IV/66) programs as well as nearly all DIGITAL Fortran 77 extensions. The significant DIGITAL Fortran 77 features not currently supported by Compaq Fortran include:

- The DICTIONARY statement.
- Double-quote syntax for octal constants, eg, "7723.
- The /ASSUME=BACKSLASH, /CROSS_REFERENCE, /SHOW=DICTIONARY, /STANDARD=(MIA,SYNTAX,SEMANTIC,SOURCE_FORM) qualifiers or keyword values.
- Support of Language Sensitive Editor placeholders in source.

There are other minor language interpretation differences - please see the Compatibility appendix of the *Compaq Fortran User Manual for OpenVMS Alpha Systems* for more details.

System managers and/or individual users may choose to override the default selection of compiler. Two command definition files have been provided:

- SYS$LIBRARY:FORT$FORTRAN-F95.CLD - FORTRAN (and F90) invokes Fortran 95, FORTRAN/OLD_F77 invokes Fortran 77.
- SYS$LIBRARY:FORT$FORTRAN-F77.CLD - FORTRAN invokes Fortran 77, F90 invokes Fortran 95.

Individual users may change the default by using SET COMMAND to establish the desired command set. For example:

```
$ SET COMMAND SYS$LIBRARY:FORT$FORTRAN-F77
```

This will change the command defaults for the current process. Note that spawned processes will not inherit command definition changes made in this fashion.

System managers may change the system-wide default as follows:

```
$ SET COMMAND /TABLES=SYS$LIBRARY:DCLTABLES -
  /OUTPUT=SYS$COMMON:[SYSLIB]DCLTABLES -
  SYS$LIBRARY:FORT$FORTRAN-F77
$ INSTALL REPLACE SYS$LIBRARY:DCLTABLES
```

The INSTALL command must be repeated on all Alpha nodes of a VMScluster. Users must log in again to see the new definitions, or do a:

```
$ SET COMMAND /TABLES=SYS$LIBRARY:DCLTABLES
```

If you have problems compiling your Fortran 77 code with the Fortran 90/95 compiler, please let us know by sending a summary (and small code example) to fortran@compaq.com.

3.5.2 Changes specific to Compaq Fortran

3.5.2.1 Support of data in 64-bit address space

Compaq Fortran version V7.2 adds support for data in 64-bit address space. The default is that all data is allocated in 32-bit address space ("P0" or "P1") unless explicitly requested otherwise by means of the new ADDRESS64 attribute. The ADDRESS64 attribute is specified on a !DEC$ ATTRIBUTES directive (see the *Compaq Fortran Language Reference Manual*) and specifies that the associated object resides in 64-bit space. The ADDRESS64 attribute may be applied to the following items and has the effect described:

- Local variable with ALLOCATABLE or POINTER attribute - Memory obtained for the variable using ALLOCATE is in 64-bit space.
- Local variable without ALLOCATABLE or POINTER attribute - Variable is statically allocated in 64-bit space. This feature is available on OpenVMS Alpha V7.2 or later, and linking requires use of LINKER64 as supplied with the ADX072 Debugger kit. The variable may not be DATA initialized nor may it be named in an AUTOMATIC statement.
- COMMON block - COMMON block is statically allocated in 64-bit space. This feature is available on OpenVMS Alpha V7.2 or later, and linking requires use of LINKER64 as supplied with the ADX072 Debugger kit. DATA initialization is not permitted. Note also that a shareable image containing a 64-bit COMMON cannot be INSTALLed using the /SHARE qualifier, due to a restriction in OpenVMS 7.2. This may be relaxed in a future version of OpenVMS.
- Dummy argument in procedure or INTERFACE block - Specifies that the associated actual argument has the ADDRESS64 attribute. Both actual and dummy arguments must have ADDRESS64 or neither may. In a future release, this restriction may be relaxed.
- Derived type containing a POINTER component - Not yet implemented.

Examples:

```
! Declare a local variable which is allocated in
! 64-bit space
!
REAL BIG_ARRAY (1000000)
!DEC$ ATTRIBUTES ADDRESS64 :: BIG_ARRAY


! Declare an ALLOCATABLE array which is to be
! allocated in 64-bit space
!
REAL, ALLOCATABLE :: BIG_ARRAY (:)
!DEC$ ATTRIBUTES ADDRESS64 :: BIG_ARRAY
...
ALLOCATE (BIG_ARRAY(1000000)


! Declare an external routine which accepts 64-bit
! arguments
!
INTERFACE
SUBROUTINE TAKES_BIG_ARRAY (BIG_ARRAY)
REAL BIG_ARRAY (*)
!DEC$ ATTRIBUTES ADDRESS64 :: BIG_ARRAY
END SUBROUTINE
END INTERFACE
```

3.5.2.2 Support for DIGITAL Source Code Analyzer

Compaq Fortran version 7.2 now supports the /ANALYSIS_DATA qualifier, which specifies that the compiler output information for use by the optional DIGITAL Source Code Analyzer product, part of the DECset package. The /DESIGN qualifier, which adds design information from structured comments, is not supported.

The following limitations of the current implementation are known:

- Information about actual arguments in routine calls is not provided, therefore CHECK CALLS won't verify argument lists. Information about Formal (dummy) arguments is provided.
- The "machine datatype" associated with type declarations reflects the default of /FLOAT=G_FLOAT. This should have little or no effect on actual use.
- STRUCTURE and derived type declarations do not have a lexical scope, and thus you can't ask what STRUCTURE/type declaration "contains" a given field declaration.
- The information generated by certain Fortran 90/95 constructs has not been extensively tested.

We expect to correct these problems in a future update.

If you notice incorrect or unexpected Analysis Data results, please report the problem to Compaq and include the smallest possible source which reproduces the problem.

3.5.2.3 Additional changes and corrections

The following changes are new in V7.2

- The compiler and run-time library now support all Fortran 95 language features.
- Detection of uninitialized variables is done even if /NOOPTIMIZE is specified.
- The ASM intrinsic is now supported to enable assembly language code to be embedded in Fortran source. See HELP F90 Intrinsic_Procedures ASM for more details.
- Support ISHC for INTEGER*8.
- Correct problem with overlapping CHARACTER assignment in FORALL.
- Correct debug information for CHARACTER POINTERs.
- Correct problems with ISHFTC which can cause alignment errors.
- Correct problem with FORALL and WHERE with non-default integer size
- The compiler now accepts a new DEFAULT keyword on the !DEC$ ATTRIBUTES directive. This tells the compiler to ignore any compiler switches that change external routine or COMMON block naming or argument passing conventions, and uses just the other attributes specified (if any). The switches which this affects are /NAMES and /ASSUME=UNDERSCORE.
- The /NOF77 switch now establishes OPEN defaults of STATUS='NEW' and BLANK='ZERO'.
- Correct a problem where calling an ELEMENTAL routine with a pointer array may give incorrect results.

- Fix transfer intrinsic where the MOLD is a character substring with non-zero base, e.g., TRANSFER(X, CH(I1:I2))
- Fix problem where CSHIFT of an array of derived type generated bad code.
- Correct problem with pointer assignment when the right-hand-side is an array of derived types.
- Correct problem involving RANDOM_NUMBER of a VAX D_float array.
- Correct problems involving function return value whose size depends on properties of input arguments.
- Fix problem that caused internal compiler error with RESHAPE.
- Fix problem where IBCLR of mixed-kind arguments got wrong answer.
- Correct a problem with PACK when the first argument is a two-dimensional slice of a three-dimensional array.
- Correct problem with ADJUSTL, ADJUSTR and COTAN with array element arguments.
- Correct compiler abort with ASSOCIATED (X,(Y))
- Don't give standards warning for ELEMENTAL PURE.
- Consider FORALL index variables "used" for /warn=unused purposes.
- Disallow leading underscore in identifiers, as documented.
- Correct problem with implied DO loop in non-INTEGER array constructors in initialization expressions.
- Allow expression involving array constructors in an initialization expression.
- %LOC is treated the same as LOC for type checking purposes.
- Correct problem involving generic routine resolution.
- SEQUENCE now byte-packs fields, as the documentation says.
- Correct compiler abort with RESHAPE in initialization expression.
- Correct compiler abort for case with defined operators.
- Correct compiler abort for syntax error X(;,:)
- Give appropriate error if DO loop variable is too small for range.
- Correct compiler abort for LEN_TRIM(array) in initialization expression.
- Correct compiler abort for SIZE(non-array).
- Correct problems with ISHFT(array) in initialization expression.
- Allow SHAPE in initialization expression.
- Don't give standards warning for use of INDEX in initialization expression.
- Consider statement function dummy argument "used" for /warn=unused.
- Correct compiler abort for invalid syntax in a Variable Format Expression
- Fix internal compiler error for certain uses of LL* intrinsics.
- Prevent internal compiler error in certain cases when the size of a return value is based on a call to a pure function with the argument to this function.
- Correct problems with nested uses of SPREAD intrinsic.

- Make ASSOCIATED return the correct result when the target is an element of a deferred-shape array.
- Correct a problem with a USE..ONLY of some symbols from an EQUIVALENCE group in a module. Previously, the compiler might generate an external reference to the wrong symbol.
- Correct a problem with EOSHIFT of a structure array with a multidimensional structure component.
- Eliminate the unnecessary use of temporary array copies in many cases.
- Add support for specific names IMVBITS, JMVBITS and KMVBITS (already documented).
- Correct a problem where calling an ELEMENTAL routine with a pointer array may give incorrect results.
- Don't issue spurious UNUSED warning for argument whose interface comes from a MODULE.
- Fix internal compiler error for invalid IMPLICIT syntax.
- Eliminate inappropriate type mismatch error for certain cases of references to a generic procedure with a procedure argument
- Allow use of . field separator in addition to % in ALLOCATE/DEALLOCATE.
- Give warning of unused variable in module procedure when appropriate.
- Do not allow a non-integer/logical expression in a logical IF.
- Fix another case of recognizing a RECORD field that has the same name as a relational operator.
- Correct compiler failure for CMPLX(R8,R8) when real_size=64 is in effect
- Allow gaps in keyword names in MAX/MIN, for example MAX(A1=x,A4=y)
- Correct compiler failure when a COMPLEX array is initialized with a REAL array constructor
- Correct compiler failure when the CHAR intrinsic is used in an initialization expression
- Correct compiler failure ("possible out of order or missing USE") in certain uses of nested MODULEs and ONLY.
- Show correct source pointer for syntax error in declaration.
- Correct compiler failure with RESHAPE and SHAPE used in an initialization expression.
- Eliminate spurious error when a defined operator is used in a specification expression
- Correct compiler failure when undefined user-defined operator is seen.
- Eliminate spurious error when component of derived type named constant is used in a context where a constant is required.
- Correct problem with host association and contained procedure. Correct compiler failure with WHERE when non-default integer_size
- is in effect.

The following changes appeared in V7.1 ECO 2.

- The new /ASSUME=[NO]ALTPARAM compile command qualifier selects how the compiler treats the non-standard, non-parenthesized PARAMETER syntax, for example `PARAMETER N = 1`. If /ASSUME=ALTPARAM is in effect, the alternate PARAMETER syntax is accepted. If /ASSUME=NOALTPARAM is in effect, the alternate PARAMETER syntax is not accepted and such statements will typically be treated as assignment statements. The default is /ASSUME=ALTPARAM.
- The new /ASSUME=[NO]MINUS0 compile command qualifier selects how the SIGN intrinsic treats the case of SIGN(X,-0). If /ASSUME=MINUS0 is in effect, the result of SIGN(X,-0) is -X, in accordance with the Fortran 95 standard. If /ASSUME=NOMINUS0 is in effect, the result is X, which is consistent with what Fortran 77 implementations typically produce. The default is /ASSUME=NOMINUS0, which is a change from earlier versions of Compaq Fortran 90. We made this change because many of our customers' applications were getting unexpected results with the MINUS0 semantics. (Note that this setting is applicable only if /FLOAT=IEEE is in effect.)
- Using ASSOCIATED with F90 pointer now gives correct answer.
- Using vector subscripts in MATMUL now gives correct answer.
- Passing %REF argument to a routine with explicit INTERFACE no longer gets an internal error.
- CSHIFT of an array pointer contained within a derived type no longer gets an internal error.
- Using assignments in a defined generic assignment subroutine when the subroutine is not RECURSIVE now gets an error.
- Parameter constant is allowed as argument of a LOC intrinsic.
- Using UNION within derived type now gets correct result.
- Having EQUIVALENCEd character array elements within a MODULE no longer gets internal error.
- Duplicate SAVE statement no longer gets an error.
- Parameter constant can be used as case-value in a SELECT CASE statement.
- ALIAS attribute can now be specified in a cDEC$ ATTRIBUTES directive for an identifier that has not been declared EXTERNAL (EXTERNAL is assumed).
- Interface with optional function argument is now resolved properly.
- Extra trailing blanks are now allowed and ignored when used in specifier of OPEN statement, e.g., FORM='formatted '.
- INTEGER*2 array now gets correct result when compiled with /INTEGER_ SIZE:16.
- Fix a bug related to module importing with modules that contain PRIVATE statement.
- Parameter constant defined in a MODULE is now imported when its use is only in a variable format expression.
- Parameter constants are allowed in a structure constructor.

- A derived type component having the same name as a common block no longer gets an internal error.
- A non-standard warning is issued if the first argument of a GENERIC assignment procedure is not INTENT(OUT) or INTENT(INOUT).
- Fix a bug related to module importing with modules that contain a PRIVATE statement.
- Using a structure constructor to initialize a multi-dimensional array component of a derived-type no longer causes an internal error.
- The /ASSUME=FPCONSTANT option now works correctly for assignment to double complex variables.
- Having a D line as the first non-comment line after a conditional $ENDIF directive no longer gets an error.
- Using substring of a character array as argument of ICHAR intrinsic no longer gets internal error
- Pointer assignment of an array of character substrings (e.g. p=>a(:)(2:4)) now gets correct result
- Using array transformation intrinsics such as PACK, SPREAD, and RESHAPE with array of derived-type as argument in a PRINT statement now gets correct results
- An incorrect statement: "IF (ABS(I).GT 1) I=0" now gets an error message
- An incorrect statement: "CHARACTER(LEN=1), PARAMETER :: CPSCLR = ” ”" now gets an error message
- Having a PRIVATE, EQUIVALENCE variable in a module no longer causes compile time segmentation violation
- Specifying ONLY on one variable in a COMMON block now only declares the one variable not the entire variables in the COMMON block
- Allow user defined operator to be used as the format character expression in a PRINT or READ statement
- Using modules and the AUTOMATIC statement in the same routine no longer gets internal error
- Module variables with EXTERN attributes now work properly
- Increase an internal limit so that large programs no longer get the "text handle table overflow" message
- Flag incorrect usage of an entry dummy argument in an executable statement before its declaration in the entry statement
- Disallow optional return dummy argument following other OPTIONAL dummy arguments
- An invalid WRITE statement no longer gets an internal error
- Allow passing NULL intrinsic function as argument to other routines
- Allow AUTOMATIC variables to be used in an EQUIVALENCE statement
- Using a structure constructor with scalar value to assign an array element now produces correct result

- Using an array constructor with integer value to initialize a real or complex array now produces correct result
- Flag common block name and routine name conflict
- Fix elemental character function with varying length
- CPU_TIME now returns CPU, not elapsed time. (Requires FORRTL V7.1 ECO 1 library or later.)
- The debugger no longer reports a DST corruption error for certain separately compiled programs which use a module that defines an allocatable array.
- A compile-time evaluation of 0.0**0.0 (when /CHECK=POWER is in effect) now always results in a compile-time error instead of quietly giving zero.
- DPROD of two REAL(8) arguments no longer results in a compiler bugcheck.
- The compiler no longer bugchecks if a user-defined type and a structure have the same name.
- Derived type initialization of an array component now works.
- The compiler now correctly diagnoses incorrectly spelled OPEN keyword values.
- SAVE in main program no longer results in warning.
- The compiler no longer bugchecks for syntax of the form "character,allocatable :: c*7(:)".
- The compiler no longer incorrectly parses certain expressions involving record fields and binary operators.
- %FILL is now allowed as a field name when :: syntax is used in a STRUCTURE field.
- NEAREST with arguments of different KINDs no longer results in a standards warning.
- All internal compiler limits on the length of a NAMELIST have been removed.
- The compiler now allows SIZEOF of a field of an assumed-size record array.
- The compiler now correctly parses an array named SELECT.
- A standards warning for concatenation of passed-length character arguments is no longer given (was non-standard in F77, but not in F90).
- Pointer assignment now works properly when the target is a component of an allocatable array with a lower bound other than 1.
- The compiler no longer bugchecks if a Hollerith literal is passed to a CHARACTER(*) argument in the same file.
- A labelled DO with optional comma is now properly parsed.
- Derived type members whose name start with "fill" are now properly parsed.
- ONLY is no longer ignored for module variables.
- In a character literal VFE, variable names may now have underscores.
- %ref is no longer ignored when passing a character literal.
- RECORDs are now allowed in NAMELIST.
- TYPE definition with type name in parentheses now gives standards warning.

- Consecutive operators now give standards warning where appropriate.
- A CHARACTER, POINTER array whose character length is given as LEN of a passed-length CHARACTER dummy argument is now allowed.
- Unintended recursion in INTERFACE assignment is now diagnosed.
- An ELEMENTAL function is no longer allowed to have an adjustable length value.
- ALLOCATE of a REAL*16 array no longer causes a compiler bugcheck.
- Operators such as ==, etc. no longer cause standards warnings.
- ASSOCIATED( ) of a NULLIFIYed pointer now returns .FALSE..
- An array argument to a statement function is now properly diagnosed as erroneous.
- The ES format edit descriptor is no longer flagged as non-standard.
- The compiler no longer bugchecks in certain situations when the SIGN intrinsic is used with a VAX floating array argument.
- The compiler now correctly parses programs with an array named TYPE.
- The compiler now allows an unlimited number of actual and formal routine arguments. Note that the IARGCOUNT intrinsic will return the low 8 bits only of the argument count.
- A user-defined operator is now allowed as the format specifier in an I/O statement.

The following changes were included in V7.1 ECO 1.

- Significantly improve compilation speed of sources which contain many USEs of the same module.
- Fix SIGN intrinsic to handle -0. (Fortran 95 feature)
- Fix LOC intrinsic and %LOC of a derived type field.
- Fixed debug information for dynamic character variable (such as character*(i) c).
- Add debugging support for integer (Cray) pointers and allocatable arrays.
- Fix storing to the return value of a function returning character in a containing internal routine.
- Fix Nullification of a character*n pointer argument.
- Fix using passed length argument in a containing internal routine.
- Fix compiler abort when a source line is longer than 1024 characters in freeform source file.
- Fix using IOLENGTH in a INQUIRE statement.
- Fix FORALL problem of the form "X(X(I)) =."
- Fix contained functions returning a implicitly initialized derived-type.
- Better diagnostics for invalid programs.
- Fix compiler abort when using Nullification of a pointer in a MODULE.
- Fix a certain types of USE of a MODULE with rename list.

- Fix compiler abort when using do-loop style implicitly initialized derived-types in a MODULE.
- Sign-extending INTEGER*2 parameter constants.
- Flag invalid nested internal procedures.
- Fix compiler abort of USE of a MODULE with namelist variables in rename list.
- Issue a warning message for a intrinsic with wrong argument type and treat it as an external.
- Issue a warning message for having a SAVE common block data object.
- Fix compiler abort of USE of a MODULE with namelists.
- Fix using SIZEOF(common_block_array) in a PARAMETER statement.
- Fix using READONLY keyword as first keyword in an OPEN statement.
- Allow record name to be the same as a structure name.
- Fix parameter character constant with embedded NULL character.
- Fix compiler abort when same name used as a structure and derived type.
- Allow BLOCKSIZE keyword in an INQUIRE statement.
- Allow a record in a SAVE statement.
- Allow a module to have the name "procedures".
- Do not flag IABS intrinsic function as nonstandard.
- Do not flag DOUBLE COMPLEX as nonstandard.
- Put out an error message for ptr => pack(...).
- Issue an error message for invalid derived type parameter constant.
- Fix compiler abort when passing an array constructor as an actual argument.
- Fix using derived-type components that are same as intrinsic names.
- Fix an incorrect warning about "explicit-shaped array is being passed to a routine that expects a pointer or assumed-shape array".
- Fix incorrect results when compiled a program with - assume:dummy_ aliasing.
- Do not flag BOZ constant as nonstandard.
- Do not flag Z format as nonstandard.
- Put out a standard warning for using character constant in DATA statement.
- Fix using TRANSFER in initialization.
- Fix a problem with user defined assignment statement.
- Issue an error message when passing or receiving an optional argument by value.
- Fix an invalid message about return value of a function is not defined when the function returns an initialized derived type.
- Fix a compiler abort with "text handle table overflow" message.
- Fix a compiler abort using a SAVE statement.

- Fix a problem when an existing operator is overloaded.
- Fix argument checking of intrinsic subroutines.
- Fix generic interface of elemental functions.
- Issue an argument mismatch warning message for using an integer with a statement function that takes real argument.
- Fix compiler directives processing.
- Fix a compiler abort using an invalid PARAMETER array.
- Issue an error message for SAVE of an ENTRY result variable.
- Fix using UNION within derive type.
- Fix a compiler internal error when using C and REFERENCE attributes on a function name.
- Fix a compiler internal error when using ASSOCIATED of a function returning a pointer.
- Add support for passing complex by value.
- Fix pointer assignment with a character substring.
- Allow using ICHAR in an array constructor within the initialization part of an array declaration.
- Fix a problem with using UNION declaration within the derived type.
- Allow exporting of a module procedure which has a name that is the same as a generic name.
- Fix a problem with using user defined assignment operation.
- Allow specifying NaNs in the PARAMETER statement.
- Allow D source line to continue a non-D source line.
- Fix a problem in array initialization processing.

3.6 New and Changed Features in V7.1

This section provides highlights of new and changed features and problems corrected in Compaq Fortran V7.1.

3.6.1 Changes common to both Compaq Fortran and Compaq Fortran 77

The following changes are common to both Compaq Fortran and Compaq Fortran 77.

3.6.1.1 FORRTL V7.1 or OpenVMS Alpha V7.1 Required

This version of Compaq Fortran requires installation of appropriate Fortran Run-Time Library support. This is provided in OpenVMS Alpha V7.1 (released version only, not "field test" versions) or by installing the accompanying FORRTL kit as described in the *Compaq Fortran Installation Guide for OpenVMS Systems*.

The FORRTL V7.1 kit provides corrections to problems discovered since the release of OpenVMS Alpha V7.0 and adds support for the Fortran 90 NEAREST intrinsic. Compaq recommends that the FORRTL kit be installed on any system running a version of OpenVMS Alpha earlier than V7.1 on which Compaq Fortran programs will be run. The FORRTL kit may be redistributed according to the terms of the Compaq Fortran Software Product Description and is licensed under the OpenVMS software license.

3.6.1.2 /ARCHITECTURE Compile Command Qualifier

The new /ARCHITECTURE compile command qualifier determines the type of Alpha processor code that will be generated for this program. The /ARCHITECTURE option uses the same keywords as the /OPTIMIZE=TUNE option.

Whereas the /TUNE option is primarily used by certain higher-level optimizations for instruction scheduling purposes, the /ARCHITECTURE option determines the type of code instructions generated for the program unit being compiled.

Compaq OpenVMS Version 7.1 and subsequent releases will provide an operating system kernel that includes an instruction emulator. This emulator allows new instructions, not implemented on the host processor chip, to execute and produce correct results. Applications using emulated instructions will run correctly, but may incur significant software emulation overhead at runtime.

All Alpha processors implement a core set of instructions. Certain Alpha processor versions include additional instruction extensions.

Supported /ARCHITECTURE keywords are as follows:

- GENERIC generates code that is appropriate for all Alpha processor generations. This is the default.

  Running programs compiled with the GENERIC keyword will run all implementations of the Alpha architecture without any instruction emulation overhead.

- HOST generates code for the processor generation in use on the system being used for compilation.

  Running programs compiled with this keyword on other implementations of the Alpha architecture might encounter instruction emulation overhead. If HOST is specified, the actual processor model code used is displayed in the listing qualifier summary.

- EV4 generates code for the 21064, 21064A, 21066, and 21068 implementations of the Alpha architecture.

  Running programs compiled with the EV4 keyword will run without instruction emulation overhead on all Alpha processors.

- EV5 generates code for some 21164 processor implementations of the Alpha architecture that use only the base set of Alpha instructions (no extensions).

  Running programs compiled with the EV5 keyword will run without instruction emulation overhead on all Alpha processors. For the purposes of /ARCHITECTURE, the EV5 keyword is equivalent to specifying EV4.

- EV56 generates code for some 21164 processor implementations that use the byte and word manipulation instruction extensions of the Alpha architecture.

Running programs compiled with the EV56 keyword might incur emulation overhead on EV4 and EV5 processors, but will still run correctly on Compaq OpenVMS Version 7.1 (or later) systems.

- PCA56 generates code for the 21164PC processor implementation that uses the byte and word manipulation instruction extensions and multimedia instruction extensions of the Alpha architecture.

  Running programs compiled with the PCA56 keyword might incur emulation overhead on EV4 and EV5 and EV56 processors, but will still run correctly on Compaq OpenVMS Version 7.1 (or later) systems.

3.6.1.3 New /OPTIMIZE=TUNE Keywords

The /OPTIMIZE=TUNE qualifier now accepts the EV56 and PCA56 keywords. In addition, use of HOST will cause the processor model used to be displayed in the listing qualifier summary.

3.6.1.4 Problem with NFS-mounted sources fixed

A problem was corrected in both compilers where the compiler would fail to read a source file from an NFS-mounted disk - the error message given was "RMS-F-COD, invalid or unsupported type field in XAB ".

3.6.2 Changes specific to Compaq Fortran

3.6.2.1 Partial Fortran 95 support

This release includes a partial implementation of the proposed Fortran 95 standard. The following features of the proposed Fortran 95 standard have already been implemented by previous versions of Compaq Fortran 90:

- FORALL statement and construct
- Automatic deallocation of ALLOCATABLE arrays
- Dim argument to MAXLOC and MINLOC

The following features of the proposed Fortran 95 standard have been implemented by this version of Compaq Fortran 90. See HELP F90 for information on most of these features.

- Initial association status for pointers
- Detection of Obsolescent and/or Deleted features listed in the proposed Fortran 95 standard
- Masked ELSEWHERE statement
- Nested WHERE constructs
- NULL Intrinsic
- CPU_TIME intrinsic
- Generic identifier in END Interface Statements
- Kind argument to CEILING and FLOOR intriniscs
- Pure user-defined subprograms
- Automatic deallocation of ALLOCATABLE arrays
- Implicit initialization of derived-type objects (static case only)

The compiler is capable of checking for syntax conformance to either the Fortran 90 standard or the draft Fortran 95 standard. The /STANDARD compile command qualifier has been enhanced to optionally allow one of the following keywords:

F90 - Check for conformance to Fortran 90
F95 - Check for conformance to Fortran 95
NONE - Do not check for standards conformance.

The default is /NOSTANDARD which is the same as /STANDARD=NONE. If /STANDARD is specified without a keyword, the default is /STANDARD=F90. This default may change to F95 in a future release.

3.6.2.2 /PAD_SOURCE Qualifier Now Supported

Compaq Fortran now supports the /PAD_SOURCE qualifier in the same manner as Compaq Fortran 77. If /PAD_SOURCE is specified, the compiler will implicitly pad with blanks any fixed-form source records which are shorter than the source width in effect (72 or 132 columns, dependent on /EXTEND_SOURCE.) The default is /NOPAD_SOURCE, for compatibility with other Compaq Fortran compilers. /PAD_SOURCE can be used for compatibility with some non-Compaq Fortran compilers. This affects character constants which are continued across multiple source records.

3.6.2.3 Additional changes and corrections

The following additionalchanges and corrections are in this version of Compaq Fortran:

- If /REAL_SIZE=64 or =128 is specified, and a real constant includes an explicit kind specifier but no exponent letter, the compiler no longer gives an error.
- The compiler no longer fails if a derived type field is named "fill".
- The compiler now allows a source line to begin with a semicolon in free-form. This is an extension to the standard and is flagged as such.
- The compiler now allows D-lines (/D_LINES) to be continued.
- The unformatted indexed REWRITE statement no longer causes OUTSTAOVE errors when transmitting an aggregate.
- BYTE or INTEGER*2 PARAMETER constants which are defined using a constant value in the "unsigned" range are now properly sign-extended when required.
- The compiler no longer fails if two or more program units in the same compiler invocation use the IARGCOUNT intrinsic.
- The compiler now correctly treats a valueless OPEN keyword (such as READONLY) as a keyword when it appears at the start of the OPEN keyword list.
- The compiler no longer gives incorrect errors when a structure and record have the same name.
- Compile-time evaluation of character intrinsics whose arguments include a NUL character no longer produce inconsistent results.

- The NEAREST intrinsic function now returns correct values for VAX floating types. This requires new math library support which is provided in the accompanying FORRTL kit or in OpenVMS Alpha V7.1. On OpenVMS versions earlier than V7.1, calling the NEAREST intrinsic to obtain the next higher value from HUGE( ) or the next lower value from TINY( ) will result in an ACCVIO error rather than the correct math library error reported. This will be fixed in OpenVMS Alpha V7.1.
- LOC(f90_pointer) no longer causes the compiler to fail.
- The compiler no longer generates code in certain cases involving a string array element argument to an intrinsic routine, such as ADJUSTL.
- The compiler no longer generates incorrect code for an unformatted REWRITE of an array or record structure.
- Compile-time integer overflows are now warnings rather than errors.
- The compiler no longer incorrectly merges rename lists from a CONTAINEd subprogram into those from the host program unit.
- Compiler had limited INCLUDE files to 20 per compilation unit. There is no longer any fixed limit for the number of INCLUDE files. INCLUDE nesting depth limit is 20.
- Compiler limit of 99 continuation lines was raised to 511.
- Compiler did not completely honor OPTIONS /[NO]G_FLOAT.
- Compiler abort with certain types of pointer assignment.
- Incorrect error message for nested STRUCTUREs.
- Inconsistent severity for undeclared variable message with IMPLICIT NONE or command line switch.
- Incorrect error about passing LOGICAL*4 to a LOGICAL*1 argument.
- Lack of standards warning for non-integer expressions in computed GOTO.
- Incorrect flag NAME= as nonstandard in INQUIRE.
- Lack of standards warning for AND, OR, XOR intrinsics.
- VOLATILE attribute not honored for COMMON variables.
- COMPLEX expression not allowed in variable format expression.
- Adjustable array not allowed to be declared AUTOMATIC (AUTOMATIC declaration is ignored.)
- /RECURSIVE not honored in main program.
- Parsing error with DO-loop has bounds of -32768,32767.
- Compiler abort when extending generic intrinsic.
- SAVEd variable in inlined routine didn't always get SAVEd.
- Compiler abort with initialization of CHARACTER(LEN=0) variable
- Incorrect values of PRECISION, DIGITS, etc. for floating types.
- Incorrect value of INDEX with zero-length strings.
- Incorrect value for SELECTED_REAL_KIND in PARAMETER statement.
- Incorrect compile-time result of VERIFY.

- Routine using IARGPTR or IARGCOUNT corrupts address of passed CHARACTER argument.
- Lack of standards warning for CMPLX( ) in initialization expression.
- Compiler abort when %LOC(charvar) used in statement function.
- Incorrect initialization of STRUCTURE array.
- Compiler abort with large statement function.
- RESHAPE of array with a zero bound aborts at runtime.
- /INTEGER_SIZE not honored.
- SIZEOF(SIZEOF( )) should be 8.
- Error parsing a derived type definition with a field name starting with "FILL_ ".
- With OPTIONS /NOI4, compiler complained of IAND with arguments of an INTEGER*4 variable and a typeless PARAMETER constant.
- Incorrect standards warning for DABS.
- Lack of Error message for ambiguous generic.
- FOR$RAB was not always considered as returning an INTEGER*4.
- Error parsing field array reference in IF.
- Bit constants in argument lists should be typed based on value, not "default integer".
- Compiler did not allow module to use itself.
- Lack of standards warning for Hollerith constant.
- Wrong values for TINY, HUGE for VAX floating.
- EXPONENT( ) with /FLOAT=D_FLOAT references non-existent math routine.

3.6.3 Changes specific to Compaq Fortran 77

3.6.3.1 G Format Edit Descriptor for Integer Variables

If an integer variable is transmitted in formatted I/O using a G format edit descriptor, Compaq Fortran 77 now adopts the Fortran 90 semantics of allowing this combination and using an equivalent I format. In some earlier releases, and on OpenVMS VAX, this combination was disallowed and would result in a FORVARMIS error. Some applications explicitly disabled this error (through ERRSET on VAX and /CHECK=NOFORMAT on Alpha), where the behavior would then be that the data was assumed to be REAL*4.) This change in behavior may affect some applications which use an integer variable in I/O to transmit data that may be real.

3.6.3.2 DATE_AND_TIME Intrinsic

The DATE_AND_TIME intrinsic from the Fortran 90 standard has been added to Compaq Fortran 77 to provide a standard interface for obtaining the date with a four-digit year format. Use of this routine, in place of the non-standard DATE or IDATE intrinsics, will allow an application to function correctly in the year 2000 and beyond.

The interface to DATE_AND_TIME is as follows:

CALL DATE_AND_TIME ([date] ,[time] ,[zone] ,[values]) date - CHARACTER*8 containing CCYYMMDD {CC is century} time - CHARACTER*10 containing hhmmss.sss zone - CHARACTER*5 containing ±hhmm {difference from UTC} values - INTEGER*4 (8) array containing 4-digit year, month, day, zone, hour, minutes, seconds, milliseconds

Any of the arguments may be omitted, but any preceding commas for omitted arguments must be specified.

3.6.3.3 Use of POINTER Variables in DEBUG

Compaq Fortran now correctly describes pointees, declared in a POINTER statement, in the Debug Symbol Table information for use with the OpenVMS Debugger. However, the debugger does not contain full support for pointer-type variables in Fortran. Scalar numeric pointees can be examined by prefacing the pointee variable name with the "@" character. For example, given the following declarations:

```
REAL X
POINTER (P,X)
```

you could use the DEBUG command `E @X` to examine the contents of X (pointed to by P). If the pointee has a character, array or record type, however, the debugger will not properly access the contents. Until the debugger is enhanced in this area, a suitable workaround is to tell the debugger that the language is Pascal, by means of a `SET LANGUAGE PASCAL` DEBUG command, and use Pascal syntax, as follows:

- Place a caret "^" after the pointee variable name.
- Use square brackets "[]" instead of parentheses in array and character references.

As in Compaq Fortran, a period is used as a record field separator. Also, the debugger will properly use Fortran-style column-major order array indexing even though the language is set to Pascal. Note that the debugger does not support COMPLEX constants in expressions, but can examine COMPLEX variables.

The following table shows equivalent Pascal syntax for accessing various Fortran variable types.

Table 3–1 Pascal DEBUG Syntax for Fortran Pointee Expressions

Fortran Type	Pascal DEBUG Syntax
Whole variable X	X^
Character substring X(2:3)	X^[2:3]
Array reference X(2,3)	X^[2,3]
Field reference X.F	X^.F
Field array reference X(3).F	X^[3].F

OpenVMS Alpha V7.1 will contain Fortran DEBUG support for pointer variables.

3.6.3.4 Improved DEBUG support for PARAMETER constants

Compaq Fortran 77 now supports /DEBUG=PARAMETER=NONE | USED | ALL in the following manner:

/DEBUG not present

/DEBUG=(PARAMETERS=NONE,NOSYMBOLS,TRACEBACK)

/DEBUG alone

/DEBUG=(PARAMETERS=USED,SYMBOLS,TRACEBACK)

/DEBUG=ALL

/DEBUG=(PARAMETERS=ALL,SYMBOLS,TRACEBACK)

/DEBUG=NONE

/DEBUG=(PARAMETERS=NONE,NOSYMBOLS,NOTRACEBACK)

Note that the behavior for Compaq Fortran 77 for OpenVMS Alpha differs in two ways from that of Compaq Fortran 77 for OpenVMS VAX:

/DEBUG=PARAMETER

Alpha: /DEBUG=(PARAMETERS=ALL,NOSYMBOLS,TRACEBACK)
VAX: /DEBUG=(PARAMETERS=ALL,SYMBOLS,TRACEBACK)

/DEBUG=PARAM=USED

Alpha: /DEBUG=(PARAMETERS=USED,NOSYMBOLS,TRACEBACK)
VAX: /DEBUG=(PARAMETERS=USED,SYMBOLS,TRACEBACK)

In the OpenVMS VAX compiler, /DEBUG=PARAM is looked at only if /DEBUG=SYMBOLS is set whereas the Alpha compiler processes /DEBUG=PARAM independent of /DEBUG=SYMBOLS.

3.6.3.5 Additional changes and corrections

The following changes and corrections are in this version of Compaq Fortran 77:

Note

/ASSUME=UNDERSCORE is available but as yet undocumented in Compaq Fortran 77. When set, a trailing underscore {"_"} is appended to external names (which is the default convention for Compaq Fortran 77 on Compaq UNIX.) The default is /ASSUME=NOUNDERSCORE. Users should use /ASSUME=(ALL,NOUNDERSCORE) instead of /ASSUME=ALL if they do not want this UNIX behavior.

- Deferred function argument expression can cause problems, eg, "IF(F(I,(.NOT.X).AND.Y))THEN".
- Be more tolerant of large typeless things like SIZEOF(X'FFFFFFFFFFFFFFFFFFFF').
- Do not delete adjustable array bounds that are equivalenced into COMMON.
- Construct names can be a problem for .ANA files, especially with no preceding real label.

- Allow CHARACTER C*10(20) syntax in addition to CHARACTER C(20)*10.
- Fix large (INTEGER*8 sized) constant subscripts.
- Fix F90 automatic and pointer arrays with variable bounds.
- Allow debugging with a pointer to an array in COMMON.
- Allow $FORT T1.F/LIS,T2.F/NOLIS or $FORT T1.F/OBJ,T2.F/NOOBJ.
- Allow SIZEOF(CHARACTER_ARRAY(1)) - do runtime evaluation if necessary.
- Do NOT do .ANA file processing {eg, for USEROPEN} without /ANA present.
- Fix IBITS to not overflow on small integer types.
- Fix (another) NAMELIST/SAVE/EQUIVALENCE problem.
- Use less memory at compile-time for routine prolog code for adjustably dimensioned arrays (a(n,n),b(n,n)).
- Fix a.b.c offset computation.
- "Small integer*1 op small integer*2 constant" should give integer*1 result (ala VAX).
- Change error message for YEAR2000 (suggest using DATE_AND_TIME, not DATE).
- Data initialization incorrect for: "RECORD /S1/R1,/S2/R2" (r2 used s1 initialization).
- Do not allow substrings to be merged in I/O lists (N,STR(1:N) for example).
- Fix a bug in nested uninitialized STRUCTUREs.
- Allow CHARACTER *(*) C; PARAMETER (C='abcdefghijk') even with /DEBUG=PARAM=ALL.
- Better "function type" message string when mismatch between caller and called function.
- Give warning instead of assertion for REAL*4 X(10); REAL*8 X.
- Fix IBITS on small integers so it doesn't overflow.
- Do NOT allow VOLATILE to apply to a function name.
- Fix illegal memory reference in compiler when the I/O format is a CHARACTER function.
- "CALL FOO(LEN(C))" with argument checking now gives "Integer/Logical*8" for type instead of "Unknown".
- Fix USEROPEN routine when using /ANA so a bad .ANA file is not produced.
- Implement multiple entry point functions with mixed COMPLEX entry names.
- Add /STAND=SEMANTICS warning of entry (of function) name used before ENTRY statement.
- Optimize the representation of field references.
- Merge I/O lists with "simple" elements into a single I/O call.
- Implement /ARCHITECTURE = EV4 | EV5 | EV56 | PCA56 | GENERIC | HOST.

- An unused automatic data item no longer causes the compiler to abort.
- Code generation bugs were fixed in /OPTIMIZE=LOOPS and /OPTIMIZE=LEVEL > 3.
- .AND. and .OR. expressions in logical IFs now evaluate left-to-right more often, find more short circuits, and generate better code.
- Allow dummy arguments to ENTRY statements to occur in the declaration phase, eg: PARAMETER SX = SIZEOF (X) where X only occurs as an ENTRY argument later.
- Allow %VAL(expression) as the argument to LIB$ESTABLISH.
- Make it easier for the debugger to find the start of a routine.
- Make CPPFILERR a warning {not an error} so that # 2000 "xxx.F" is ignored when xxx.F is not present.
- Put out better information for FUNCTION/ENTRY names for debugging.
- Make ALTRETOMI an informational for return with alternate label when no * in any dummy argument list.
- Give better error message for bad character in octal/hex constant.
- Allow I/O keywords to be blank padded (for compile time constants).
- /INTEGER_SIZE= should affect constants in output statements, e.g. TYPE *,1,2,3. This is an incompatible change.
- Allow continued DICTIONARY statement.
- Let DICTIONARY definitions use their own /ALIGN setting (instead of forcing /ALIGN=RECORD=PACKED).
- Give informational (BRNCHINTOBLK) for jumping into block or loop from outside: "Questionable branch into loop or block".
- Give an error (INVSUBUSE) when subroutine/entry names are used as functions: "This name cannot be used as a function reference".
- Make it possible to pass CHARACTER*(*) function names again.
- Allow POINTER to functions/subroutines.
- Fix the listing of /BY_REF_CALL to list more than the first routine name.
- Allow an optional ":" between CDEC$ TITLE | SUBTITLE and the "[sub]title".
- Add YEAR2000 warning for DATA/IDATE, suppressed by /WARN=NOUSAGE : "Two-digit year return value may cause problems with the year 2000".
- Suppress BRNCHINTOBLK under /WARN=NOUSAGE.
- Don't issue CHACONCONTD more than once per statement.
- Fix variable lower bounds for second dimension so they don't cause assertions.
- Give error message (BADRANDU) for incorrect arguments to RANDU: "Intrinsic RANDU arguments should be: (integer*2, integer*2, real*4)"
- Catch adjutable array used as namelist element (avoid internal error).
- Code generation bugs were fixed in CHARACTER argument setup, register reuse, loop reductions, and backwards loops.

3.6.3.6 Known problems in Compaq Fortran 77

- A bug in USEROPEN processing sometimes aborts the compiler. The workaround is to compile with /ANALYSIS_DATA specified.

3.7 New and Changed Features in V7.0

This section provides highlights of new and changed features in Compaq Fortran 7.0, as compared to the earlier separate products DEC Fortran 90 V2.0 and DEC Fortran V6.3.

3.7.1 Changes common to Compaq Fortran and Compaq Fortran 77

3.7.1.1 cDEC$ PSECT ALIGN= Keywords Supported

The cDEC$ PSECT ALIGN= specifier now supports the keywords BYTE, WORD, LONG, QUAD, OCTA and PAGE in addition to the integer constant expression previously allowed. These keywords are equivalent to the constants 0, 1, 2, 3, 4 and 16, respectively. This feature is also supported by version 6.4 of Compaq Fortran for OpenVMS VAX Systems, where the keyword PAGE will be treated as if it were the constant 9 (the page size on OpenVMS VAX systems is 2**9 (512) bytes.)

3.7.1.2 DPROD Intrinsic Now Generic

The DPROD intrinsic function has been extended to operate as a generic function which can accept arguments of types REAL*4 or REAL*8 (both arguments must be the same type.) If the arguments to DPROD are REAL*8, the function result is of type REAL*16. Note that if the DPROD intrinsic name is passed as an actual argument, the routine passed is the version with a REAL*8 function result.

3.7.1.3 /ASSUME=ACCURACY_SENSITIVE Changes

The compilers have been changed so that the optimization resulting in calls to a special "reciprocal square root" routine for expressions of the form 1.0/SQRT(x) or A/SQRT(B) will be enabled only if /ASSUME=NOACCURACY_SENSITIVE is in effect. The default is /ASSUME=ACCURACY_SENSITIVE. Note that /FAST enables /ASSUME=NOACCURACY_SENSITIVE.

3.7.1.4 /ASSUME=BYTERECL Command Qualifier

Compaq Fortran now supports the /ASSUME=BYTERECL qualifier on the command line. This controls the size of the unit used for the RECL= keyword in an OPEN or INQUIRE statement for unformatted files. The default, NOBYTERECL, specifies that RECL is in units of 4-byte longwords, compatible with current and past Compaq Fortran products. If BYTERECL is specified, the unit for RECL is one byte, which is used on some non-Compaq platforms.

Use of the non-default BYTERECL value requires that the application be running on a system which has the associated Fortran Run-Time Library support. This support is provided by installing one of the following:

Compaq Fortran version V7.0 or later
OpenVMS Alpha V7.0 or later

If a program which specifies /ASSUME=BYTERECL is run on a system without the proper Run-Time Library support, it will fail with an INVARGFOR, Invalid argument to Fortran Run-Time Library error.

3.7.1.5 /CHECK=NOPOWER Compile Command Keyword

By default, Compaq Fortran follows the Fortran standard and disallows (gives a compile-time or run-time error) for real exponentiation of 0.0**0.0 or for a negative real number raised to an integer-valued power. Other languages and other Fortran implementations may allow these expressions. The /CHECK=NOPOWER compile command qualifier may be used to specify that Compaq Fortran allow these expressions so that 0.0**0.0 gives a value of 1.0 and that negative real bases to raised integer-valued real powers are treated as if the exponent were integer valued (for example, -3.0**3.0 results in -27.0).

3.7.1.6 New /OPTIMIZE features

/OPTIMIZE=LEVEL=5 now enables a new set of loop transformation optimizations that apply to array references within loops, including:

- Loop blocking
- Loop distribution
- Loop fusion
- Loop interchange
- Loop scalar replacement
- Outer loop unrolling

The loop transformations can be selectively disabled (as a group) by specifying /OPTIMIZE=NOTRANSFORM_LOOPS. Also, software pipelining, a set of optimizations previously enabled at optimization level 5, can be disabled using /OPTIMIZE=NOPIPELINE.

3.7.1.7 /REENTRANCY Compile Command Qualifier

Compaq Fortran now supports writing applications which use multithread or AST reentrancy during RTL calls. The new /REENTRANCY qualifier allows you to specify keyword values to indicate what type of reentrancy to support. (Use of this qualifier requires OpenVMS Alpha V7.0 or later - it is ignored if an application is run on earlier versions of OpenVMS Alpha). See the online HELP for further details.

3.7.1.8 Improvements in IEEE constant handling

The compilers have improved handling of IEEE exceptional values (NaN, infinity) created during compile-time constant expression evaluation. If /FLOAT=IEEE is specified and a constant expression overflows or underflows or would result in a NaN, the correct exceptional value is used. This may result in run-time exceptions if the default /IEEE_MODE=FAST is in effect.

3.7.1.9 Improved optimizations

To improve run-time performance, new optimizations are now available and certain improvements have been made, including:

- Certain intrinsic procedures specific to Fortran 90 (not available in FORTRAN-77)
- Subprogram calls with array arguments
- New command-line qualifiers that activate new optimizations, including the loop transformation optimizations (/OPTIMIZE=LOOPS) or (/OPTIMIZE=LEVEL=5). The software pipelining optimization is now activated by using /OPTIMIZE=PIPELINE or /OPTIMIZE=LEVEL=5.

3.7.1.10 Enhanced DEBUG support for optimized programs

Compaq Fortran V7.0 contains enhanced support for debugging programs compiled with optimization as described in *OpenVMS Version 7.1 New Features Manual*.

3.7.1.11 USEROPEN routines must be INTEGER

If a program specifies a USEROPEN routine in an OPEN statement and the routine is declared explicitly as some type other than INTEGER*4 or INTEGER*8, the compiler will now give a "Name previously used with conflicting data type" (INVTYPUSE) error. If the routine was not explicitly given a type, the compiler now assigns it the INTEGER*4 type, even if an IMPLICIT statement would specify a different type. However, if IMPLICIT NONE is in effect, and the type was not specified explicitly, then an IMPNONE error will be generated. Previously, the type conflict was not detected nor was a warning generated if IMPLICIT NONE was in effect. USEROPEN routines must return an INTEGER*4 or INTEGER*8 function value.

3.7.2 Changes specific to Compaq Fortran

3.7.2.1 cDEC$ ALIAS directive now supported

The cDEC$ ALIAS directive is now supported in the same manner as in Compaq Fortran 77. This directive provides the ability to specify that the external name of an external subprogram is different than the name by which it is referenced in the calling subprogram. For further details, see Section 4.2.3.

3.7.2.2 cDEC$ ATTRIBUTES directive

The cDEC$ ATTRIBUTES directive lets you specify properties for data objects and procedures. These properties let you specify how data is passed and the rules for invoking procedures. This directive is intended to simplify mixed language calls with Compaq Fortran routines written in C or Assembler.

For more information on the cDEC$ ATTRIBUTES directive, see Section 4.2.4.

3.7.2.3 Variable Format Expressions in character literals now supported

Simple forms of Variable Format Expressions (VFEs) are now allowed in character literal constant formats. The VFE may contain a single constant or variable name, but not an expression. VFEs in FORMAT statements may contain expressions. In addition, invalid Format strings in character literal constants are now detected at compile-time rather than at run-time.

3.7.2.4 Abbreviations allowed in OPTIONS statement

The compiler now allows qualifier and keyword values in the OPTIONS statement to be abbreviated to the shortest unique value.

3.7.2.5 Enhanced support for the FORALL statement and construct

The FORALL construct now allows the following statements in the forall body:

- Pointer assignment statement
- FORALL statement or construct (nested FORALL)
- WHERE statement or construct

Please note that each statement in the FORALL body is executed completely before execution begins on the next FORALL body statement.

The compiler now correctly defines the scope of a FORALL subscript name to be the scope of the FORALL construct. That is, the subscript name is valid only within the scope of the FORALL. Its value is undefined on completion of the FORALL construct.

3.7.2.6 /NAMES qualifier now supported

The /NAMES qualifier, with keyword values of UPPERCASE, LOWERCASE and ASIS, is now supported in the same manner as in Compaq Fortran 77. See the online HELP for further details.

3.7.3 Changes specific to Compaq Fortran 77

The following additional changes are in this version of Compaq Fortran 77:

- CDEC$ OPTIONS /[NO]WARNINGS[=[NO]ALIGNMENT]

  The "CDEC$ OPTIONS" directive has been extended to accept

  /WARNINGS[=[NO]ALIGNMENT]
  /NOWARNINGS

  to turn off or on the local reporting of alignment warnings in STRUCTUREs or COMMONs.

- The new directive

```
CDEC$ RESTRICT pointee [, pointee]...
CDEC$ RESTRICT *
```

  advises the compiler that objects pointed to using the POINTER statement are not accessed by any other name than the name in the POINTER statement {"pointee" above}. "*" means all such pointees are exclusively accessed by their pointee name. This allows for better optimizations of objects pointed to because the compiler can assume the only way to access the object is by its pointee name and no other.

- The following program shows a problem with Compaq Fortran 77:

```
PROGRAM X
IMPLICIT NONE
INTEGER*2 Y(ICHAR('A'):ICHAR('Z'))
INTEGER*4 Z
Z = ICHAR('A')
END
```

  reports a compile-time error of "untyped name ICHAR" in the Y declaration. The work-around is to declare the intrinsic:

```
INTEGER*4 ICHAR
```

4

General information about Compaq Fortran

This chapter provides general information applicable to Compaq Fortran.

4.1 Information common to Compaq Fortran and Compaq Fortran 77

4.1.1 INQUIRE(RECL) on OpenVMS Alpha vs. OpenVMS VAX

INQUIRE(RECL) for unformatted files with the default RECL unit (longwords) gives different answers on OpenVMS VAX and OpenVMS Alpha if the existing file has a record length which is not a multiple of 4 bytes. Use /ASSUME=BYTERECL and specify the proper RECL in bytes in the OPEN statement.

4.1.2 Additional explanation of certain tuning options

Here is some additional information on the use of tuning options:

- /fast implies /arch=host
- The /arch setting establishes the floor for the /opt=tune setting, so /opt=tune must be greater than or equal to the /arch setting
- The compiler is aware of EV67, EV68, and EV7 and knows how to tune for the caches on these Alphas.

4.1.3 Fully specify datatypes in MODULEs and INCLUDE files

When creating MODULEs and INCLUDE files that contain datatype declarations, Compaq recommends that such declarations explicitly specify the "kind" of the datatype, such as INTEGER(KIND=4) or INTEGER*4. If a kind is omitted, and the application which uses the declaration is compiled with a non-default /INTEGER_SIZE, /REAL_SIZE or /DOUBLE_SIZE qualifier value, the declarations in your module or INCLUDE file can be affected. When using the Fortran 95 compiler, you can use the SELECTED_INT_KIND and SELECTED_REAL_KIND intrinsics to determine the appropriate kind values for the datatype size you desire - this is better than hard-coding the kind value. This mechanism is not available in the Fortran 77 compiler.

4.1.4 The Year 2000 Problem

What is the Year 2000 Problem? It stems from the common practice of using two digits instead of four when writing dates and having multiple internal time formats. When this practice is extended into computer hardware and software, it causes arithmetic operations, comparisons, and data sorting procedures to yield incorrect results when working with years beyond 1999.

As Compaq's customers begin to consider the readiness of their computing systems for the transition to the year 2000, many have begun asking what impact it will have on their OpenVMS systems.

We are happy to answer that, because of insightful development by the engineers who created it, OpenVMS was born ready. The OpenVMS operating system base date uses a 64-bit format that is totally unaffected by the transition to the year 2000.

Customers who have consistently used the system base date and the associated system services that provide the ability to input and retrieve four digit ASCII year strings, will make a seamless transition into the year 2000 when accessing their data.

The Compaq Fortran compilers and their Run-Time Library are similarly unaffected by the transition to the year 2000. However, applications which use the DATE or IDATE intrinsic subroutines may need to be modified, as these routines return the year in a two-digit format. To determine if your application uses DATE or IDATE, specify the `/MAP/FULL/CROSS` qualifiers to the `LINK` command when linking your application. Then use the OpenVMS SEARCH utility or a text editor to search the resulting .MAP file for references to one of the following routines:

```
DFOR$DATE_NUMERIC
DFOR$DATE_T_DS
DFOR$IDATE
DFOR$JDATE
DFOR$KDATE
```

The cross-reference section of the linker map will list the names of the application modules which reference these routines. Then examine the code which uses the routines to determine if modifications are necessary. For further information on the DATE and IDATE intrinsic subroutines, see the *Compaq Fortran Language Reference Manual* or *DEC Fortran Language Reference Manual*.

Both the Fortran 95 and Fortran 77 compilers issue an informational diagnostic if DATE or IDATE is used in a program.

4.1.5 Optimizations on References to Dummy Arguments and COMMON

The Compaq Fortran compiler can may make local copies of dummy arguments and/or variables in COMMON blocks for improved performance. The compiler is careful to always copy values back or fetch new values when the language semantics require it, but some applications may have taken advantage of other implementations' limitations in this area and may produce incorrect results.

For dummy arguments, it is important to specify /ASSUME=DUMMY_ALIASES or to declare the variable as VOLATILE if a dummy argument can be addressed through some path other than the argument name; for example, if a dummy argument is also a COMMON variable, or if the routine can be called with some arguments omitted. By default, Compaq Fortran assumes that the application conforms to the Fortran 90 and FORTRAN-77 standards in that it does not

alias dummy arguments and that all actual arguments agree in order, number and data type (or structure, for record arguments), with their corresponding dummy arguments. See the appropriate Language Reference Manual for more information on the /ASSUME=DUMMY_ALIASES option.

For variables in COMMON, it is important to name in a VOLATILE statement any variables or COMMON blocks that can be referenced (either read or written) other than by direct reference or during a routine call. For example, if a variable in COMMON is referenced in an OpenVMS AST routine, condition handler or exit handler, or is in a shared global section, you must declare as VOLATILE that variable or the COMMON block to which it belongs. See the appropriate Language Reference Manual for more information on the VOLATILE statement.

4.1.6 Form of logical constants in formatted input

Please note that in formatted, list-directed, and NAMELIST input of logical values, any string that starts with "F" is accepted as .FALSE. and any string that starts with "T" is accepted as .TRUE.

4.1.7 Control of Dynamic IEEE Rounding Modes

The description of /ROUNDING_MODE=DYNAMIC is incorrect in saying that one can use the SYS$IEEE_SET_FP_CONTROL system service to control the dynamic rounding mode. Changing the dynamic rounding mode on OpenVMS requires use of an assembler routine at the present time.

4.2 Information specific to Compaq Fortran

4.2.1 Differences from Compaq Fortran 77

This section lists significant Compaq Fortran 77 for OpenVMS Alpha features which are not implemented by Compaq Fortran. For further details, please see Appendix A of the *Compaq Fortran User Manual for OpenVMS Alpha Systems*.

The following FORTRAN command qualifiers and qualifier keywords are not supported by the F90 verb.

- /ASSUME=BACKSLASH
- /CHECK=ASSERTIONS
- /CROSS_REFERENCE
- /DESIGN
- /DML
- /PAD_SOURCE
- /SHOW=(DICTIONARY,PREPROCESSOR,SINGLE)
- /STANDARD=(MIA,SEMANTIC,SOURCE_FORM,SYNTAX)
- /SYNTAX_ONLY
- /TERMINAL
- /WARNINGS=(UNREACHABLE)

The following Compaq Fortran 77 language features are not supported by Compaq Fortran.

- DICTIONARY statement

- Octal notation for integer constants ("0014)
- Dummy arguments with nonconstant bounds in a NAMELIST group
- Extraneous parentheses in I/O lists

The following semantic differences exist between Compaq Fortran and Compaq Fortran 77 for OpenVMS Alpha Systems.

- The Fortran-77 language specification did not define a default value for the BLANK= setting when reading from preconnected or internal files. For compatibility with VAX FORTRAN (and earlier Fortran-66 programs), Compaq Fortran 77 for OpenVMS uses BLANK='ZERO'. The Fortran 90 language specifies that the default for internal files must be BLANK='NULL', therefore Compaq Fortran uses that setting, which is different from Compaq Fortran 77 for OpenVMS. However, the Fortran 90 standard does not specify the BLANK= setting for preconnected files and Compaq Fortran for OpenVMS uses the same value, 'ZERO', as does Compaq Fortran 77. (Note that if /NOVMS is specified, the default changes to BLANK='NULL' for both internal and preconnected files in both compilers.)
- The Fortran-77 language did not allow a formatted READ to require more characters than were present in the external record. Compaq Fortran 77 implements an extension called "short field termination" where if there are insufficient characters remaining in the record to satisfy a format edit descriptor's field width, that width is reduced to the number of remaining characters.

  For example, consider the following program:

```
READ (*,'(I5)') J
WRITE (*,*) J
END
```

  If the program was compiled with Compaq Fortran 77 for OpenVMS and input was taken from an interactive terminal as follows:

  123 Return

  Compaq Fortran 77 would reduce the format field width from I5 to I3 and the program would display the value 123 for J.

  The Fortran 90 language added a new feature, the PAD= keyword to the OPEN statement, which specifies how short records are to be treated during formatted input. The default for all types of files is PAD='YES', which causes short records to be treated as if they were padded with blank characters to the required width. Because the Compaq Fortran for OpenVMS default is BLANK='ZERO' for preconnected files, the above example would display a value of 12300 for J; the blanks used for padding are treated as zeroes.

  This change in behavior may cause problems for some applications. To avoid problems in this area, Compaq recommends adding an explicit BN format item to formats used for interactive I/O on preconnected units.

  If you have installed the Compaq Fortran Run-Time Library ECO FORRTLAVE01070 (or a later version of the Run-Time Library), the Fortran 90 default for BLANK= is changed to be 'NULL' for all types of files. This change will also appear in a version of OpenVMS Alpha after V7.0. For more information, see the release notes for FORRTLAVE01070.

For further information on language constraints, see Appendix A of the *Compaq Fortran User Manual for OpenVMS Alpha Systems*.

4.2.2 Timezone support in DATE_AND_TIME intrinsic

The DATE_AND_TIME intrinsic has an optional parameter which returns the timezone differential from Coordinated Universal Time (UTC). By default, OpenVMS systems do not maintain information on timezone differential unless DECnet/OSI is installed or the system manager defines the timezone using the SYS$MANAGER:UTC$CONFIGURE_TDF.COM command procedure. If the timezone information is not available, the DATE_AND_TIME intrinsic will return a blank for the character ZONE argument and -1 for the differential element of the VALUES argument.

If timezone information is desired, the system manager should define the timezone differential using either DECnet/OSI procedures or SYS$MANAGER:UTC$CONFIGURE_TDF.COM if DECnet/OSI is not installed.

4.2.3 cDEC$ ALIAS Directive

The cDEC$ ALIAS directive is now supported in the same manner as in Compaq Fortran 77. This directive provides the ability to specify that the external name of an external subprogram is different than the name by which it is referenced in the calling subprogram.

The syntax is:

cDEC$ ALIAS internal-name, external-name

The *internal-name* is the name of the subprogram as used in the current program unit and *external-name* is either a quoted character constant or a *symbolic name*.

If external-name is a character constant, the value of that constant is used as the external name for the specified internal-name. The character constant is used as it appears, with no modifications for case or addition or removal of punctuation characters.

The Compaq Fortran compiler will force the name into uppercase unless directed otherwise.

If external-name is a symbolic name, the symbolic name (in upper case) is used as the external name for the specified internal-name. Any other declaration of the specified symbolic name is ignored for the purposes of the ALIAS directive. Note that the OpenVMS linker may have restrictions on what may appear in an external name and that upper and lower case in external names is significant.

For example, in the following program (free source form):

```
PROGRAM ALIAS_EXAMPLE
!DEC$ ALIAS ROUT1, 'ROUT1A'
!DEC$ ALIAS ROUT2, 'routine2_'
!DEC$ ALIAS ROUT3, rout3A
      CALL ROUT1
      CALL ROUT2
      CALL ROUT3
END PROGRAM ALIAS_EXAMPLE
```

The three calls would be to external routines named ROUT1A, routine2_, and ROUT3A. Because rout3A was not in quotation marks (character constant), its name was converted to uppercase.

This feature can be useful when porting code between UNIX and OpenVMS systems where different routine naming conventions are in use. On UNIX systems, calling C functions from Fortran use the convention of using a lowercase routine name with a trailing underscore. If you wrote a C routine intended to be

called from Fortran, you would have to name it in accordance to this convention. For example, ROUT2 would be coded in C as routine2_.

If this application were ported to OpenVMS where routine names are not generally modified, the result would be that the linker would not resolve the reference to ROUT2 (aliased as routine2_), unless the cDEC$ ALIAS directive were removed.

By adding or removing the cDEC$ ALIAS directive, you can specify an alternate routine name without recoding the application.

4.2.4 The cDEC$ ATTRIBUTES Directive

The cDEC$ ATTRIBUTES directive lets you specify properties for data objects and procedures. It takes the following form:

```
cDEC$ ATTRIBUTES att [,att]... :: object [,object]...
```

Where:

c

Is the letter or character (c, C, *, !) that introduces the directive (see your language reference manual).

att

Is one of the following:

C
ALIAS : external-name : internal-name
REFERENCE
VALUE

object

Is the name of a data object used as an argument or procedure.

Table 4–1 shows valid combinations of properties with the various types of objects:

Table 4–1 cDEC$ ATTRIBUTES Properties and Object Types

	Object Types		
Property	**Argument Data Declarations**	**Common Block Names[1]**	**Subprogram Specification and EXTERNAL Statements**
C	Not allowed	Allowed	Allowed
ALIAS	Not allowed	Not allowed	Allowed
REFERENCE	Allowed	Not allowed	Not allowed
VALUE	Nonarrays only	Not allowed	Not allowed

[1]Common block names are specified as [/]common-block-name[/].

For example, the C property can only be used with common block names and subprogram and EXTERNAL names, but not with variable declarations. The ALIAS property also applies to subprogram and EXTERNAL names (but not to other objects).

In contrast, the REFERENCE and VALUE properties do not apply to subprogram and EXTERNAL procedure names (or common block names). They only apply to data declarations, but only REFERENCE (not VALUE) can be used for arrays or pointers to arrays.

The cDEC$ ATTRIBUTES properties can be associated with function and subroutine definitions, in type declarations, and with the INTERFACE and ENTRY statements.

Properties applied to entities available through use or host association are in effect during the association. For example (free source form):

```
MODULE MOD1
INTERFACE
  SUBROUTINE SUB1
!DEC$ ATTRIBUTES C :: SUB1
  END SUBROUTINE
END INTERFACE
CONTAINS
  SUBROUTINE SUB2
  CALL SUB1
  END SUBROUTINE
END MODULE
```

In this case, the call to SUB1 within SUB2 uses the C property specified in the interface block.

The properties are described as follows:

- C

  This property lets you specify how data is to be passed when you use routines written in C or assembler with Fortran routines.

  When applied to a subprogram, the C property defines the subprogram as having a specific set of calling conventions. Note that Compaq Fortran on other platforms will apply the appropriate defaults and C rules for that platform.

  Table 4–2 summarizes the differences between the calling conventions:

Table 4–2 C Property and External Names

Item	Fortran Default	C Property Specified
Trailing underscore added to procedure names	No	No
Case of external subprogram names	Uppercase, unless the ALIAS property is specified	Uppercase, unless the ALIAS property is specified or /NAMES=ASIS or /NAMES=LOWERCASE is specified
Argument passing	See Table 4–3	See Table 4–3

  In addition to the case of external names and the trailing underscore, the C property affects how arguments are passed, as described in Table 4–3.

Table 4–3 C Property and Argument Passing

Argument Variable Type	Fortran Default	C Property Specified for Routine
Scalar (includes derived types)	Passed by reference	Passed by value
Scalar, with VALUE specified	Passed by value	Passed by value
Scalar, with REFERENCE specified	Passed by reference	Passed by reference
String	Passed by descriptor	String (1:1) padded to integer length
String, with VALUE specified	Error	String (1:1) padded to integer length
String, with REFERENCE specified	Passed by reference	Passed by reference
Arrays, including pointers to arrays	Always passed by reference	Always passed by reference

If C is specified for a subprogram, arguments (except for arrays and characters) are passed by value. Subprograms using standard Fortran 90 conventions pass arguments by reference.

Character arguments are passed as follows:

- By Fortran default, passed by class S or NCA (for arrays) descriptor.
- If C is specified without VALUE or REFERENCE for the arguments, the first character of the string is passed (padded with zeros out to INTEGER*4 length).
- If C is specified with REFERENCE for the argument (or if only REFERENCE is specified), the string is passed by reference.

When the C property is specified, the case of the EXTERNAL name is forced to uppercase, even if /NAMES=ASIS or /NAMES=UPPERCASE was specified during compilation. To allow mixed case or lowercase names when the C property is specified, specify the ALIAS property for the same subprogram or EXTERNAL name.

- ALIAS

  You can specify the ALIAS property as cDEC$ ALIAS or as cDEC$ ATTRIBUTES ALIAS; both are equivalent. The ALIAS property provides the ability to specify that the external name of an external subprogram is different than the name by which it is referenced in the calling subprogram (see Section 4.2.3).

  When both ALIAS and C properties are used for a subprogram or EXTERNAL name, the ALIAS property takes precedence over the C property. This allows you to specify case-sensitive names (the C attribute sets them to lowercase).

- REFERENCE and VALUE

  These properties specify how a dummy argument is to be passed.

  REFERENCE specifies a dummy argument's memory location is to be passed instead of the argument's value.

  VALUE specifies a dummy argument's value is to be passed instead of the argument's memory location.

When a dummy argument has the VALUE property, the actual argument passed to it can be of a different type. If necessary, type conversion is performed before the subprogram is called.

Character values, substrings, assumed-size arrays, and adjustable arrays cannot be passed by value. When REFERENCE is specified for a character argument, the string is passed with no length.

VALUE is the default if the C property is specified in the subprogram definition.

Consider the following free-form example which declares a routine that accepts an integer argument by value:

```
interface
  subroutine foo (a)
   !DEC$ ATTRIBUTES C :: foo
      integer a
  end subroutine foo
end interface
```

This subroutine can be invoked from Fortran using the name foo (no underscore):

```
integer i
i = 1
call foo(i)

end program
```

The actual subroutine code:

```
 subroutine foo (i)
  !DEC$ ATTRIBUTES C :: foo
     integer i
     i = i + 1
     .
     .
 end subroutine foo
```

5
Documentation Overview

The sections in this chapter:

- Describe HP Fortran documentation and online information (Section 5.1)

5.1 HP Fortran Documentation and Online Information

The HP Fortran documentation set includes the following:

- *Compaq Fortran Installation Guide for OpenVMS Alpha Systems* (AA–PU3AF–TE)

  Explains how to install HP Fortran on an OpenVMS Alpha operating system, including requirements.

  The installation guide is included with the HP Fortran document kit, QA–MV1AA–GZ.7.*n*. It is also included in ASCII and PostScript™ form on the Software Product Library CD-ROM and is on the Online Documentation Library CD-ROM in Bookreader form.

- *Compaq Fortran Language Reference Manual* (AA–Q66SD–TK)

  Describes the HP Fortran source language for reference purposes, including the format and use of statements, intrinsic procedures, and other language elements. It also provides an overview of new Fortran 95 features (not available in FORTRAN-77).

  It identifies extensions to the Fortran 95 standard by *blue color* in the printed document and by *shading* in Bookreader. When using Bookreader Version 4.0, note that the shading of extensions may be inaccurate.

  This document is included with the HP Fortran document kit, QA–MV1AA–GZ.7.*n* and is on the Online Documentation Library CD-ROM in Bookreader form.

- *Compaq Fortran User Manual for OpenVMS Alpha Systems* (AA–QJRWB–TE)

  Describes the HP Fortran program development and run-time environment on OpenVMS Alpha systems. It describes compiling, linking, running, and debugging HP Fortran programs, performance guidelines, run-time I/O and error-handling support, data types, numeric data conversion, calling other procedures, and compatibility with HP Fortran 77.

  This document is included with the HP Fortran document kit, QA–MV1AA–GZ.7.*n* and is on the Online Documentation Library CD-ROM in Bookreader form.

- *DEC Fortran Language Reference Manual* (AA–PU45B–TK)

  Describes the HP Fortran 77 source language for reference purposes, including the format and use of statements, intrinsic procedures, and other language elements.

It identifies extensions to the Fortran 77 standard by *blue color* in the printed document and by *shading* in Bookreader. When using Bookreader Version 4.0, note that the shading of extensions may be inaccurate.

This document is included with the HP Fortran 77 document kit, QA–MV1AB–GZ.7.*n* and is on the Online Documentation Library CD-ROM in Bookreader form.

- *DEC Fortran User Manual for OpenVMS AXP Systems* (AA–PU39A–TE)

  Describes the HP Fortran 77 program development and run-time environment on OpenVMS Alpha systems. It describes compiling, linking, running, and debugging HP Fortran 77 programs, performance guidelines, run-time I/O and error-handling support, data types, numeric data conversion, calling other procedures, and compatibility with Compaq Fortran on other platforms.

  This document is included with the HP Fortran 77 document kit, QA–MV1AB–GZ.7.*n* and is on the Online Documentation Library CD-ROM in Bookreader form.

- *Read Before Installing or Using Compaq Fortran Version 7.0 for OpenVMS Alpha Systems* (AV–PU3BK–TE)

  This cover letter contains information about installing HP Fortran that may not be included in the installation guide or in the release notes.

  This cover letter is included with the HP Fortran document kit, QA–MV1AA–GZ.7.*n*. It is also included on the Software Product Library CD-ROM in ASCII and PostScript form.

The HP Fortran Software Product Description (SPD) is provided on the Software Product Library CD-ROM.

The following HP Fortran online information is available (once installed on the system):

- HP Fortran online release notes

  Provide more information on this version of HP Fortran, including known problems.

  To display or print the HP Fortran release notes before installing, use the PRODUCT EXTRACT RELEASE_NOTES FORTRAN command (see the *Compaq Fortran Installation Guide for OpenVMS Alpha Systems*).

  Once installed, the ASCII version of the online release notes are located in:

  SYS$HELP:FORTRAN.RELEASE_NOTES

  Other forms of the release notes (PostScript and Bookreader) are also provided, using the file name:

  SYS$HELP:FORTRAN_RELEASE_NOTES.*

- HP Fortran online help

  The HP Fortran HELP module in SYS$HELP:HELPLIB.HLB provides online access to HP Fortran and HP Fortran 77 help, including descriptions of F90 command qualifiers, a summary of the language elements (statements, intrinsic procedures, and so on), error message descriptions, and other information.

  To view the online HP Fortran help file using the HELP command, type:

```
$ HELP FORTRAN
```

for HP Fortran, or:

```
$ HELP F77
```

for HP Fortran 77.

You can specify topics to navigate the help hierarchy. For example:

```
$ HELP FORTRAN /ALIGN
```

The *Compaq Fortran Installation Guide for OpenVMS Alpha Systems*, the "read first" cover letter, and the SPD are available on the OpenVMS Alpha Software Product Library CD-ROM in ASCII and PostScript format.

All HP Fortran documents except the cover letter, SPD, and these release notes are available on the Online Documentation Library CD-ROM in Bookreader format.