



Software Product Description

PRODUCT NAME: VSI FORTRAN for OpenVMS

SPD DO-DFRSPD-01A

DESCRIPTION

This document addresses VSI FORTRAN Version 8.3-3 for OpenVMS for VSI Alpha and VSI Integrity.

VSI Fortran on OpenVMS Alpha contains the VSI Fortran 95/90 software and the VSI Fortran 77 software as well as the VSI Extended Math Library (CXML). VSI Fortran on OpenVMS Itanium contains the VSI Fortran 95/90 software.

VSI Fortran for OpenVMS is an implementation of the Fortran programming language that supports the FORTRAN 66, FORTRAN 77, Fortran 90, and Fortran 95 standards. VSI Fortran 95/90 and VSI Fortran 77 fully support the following standards:

- ANSI X3.9-1966 (FORTRAN 66)
- ANSI X3.9-1978 (FORTRAN 77)
- ISO 1539-1980(E) (FORTRAN 77)
- MIL-STD-1753
- FIPS-69-1 (VSI Fortran meets the requirements of this standard by conforming to the ANSI Standard and by including a flagger. The flagger optionally produces diagnostic messages for compile-time elements that do not conform to the Full-Level ANSI Fortran Standard.)

VSI Fortran 95/90 supports all of the standards that VSI Fortran 77 supports plus the following standards:

- ANSI X3.198-1992 (Fortran 90)
- ISO/IEC 1539-1:1997(E) (Fortran 95)

VSI Fortran for OpenVMS documentation provides comprehensive reference and usage information for all product components.

FEATURES

VSI Fortran supports extensions to the ISO and ANSI standards, including a number of extensions defined by VSI Fortran for the various VSI Fortran platforms (operating system/architecture pairs).

Major additions to the FORTRAN 77 standard introduced by the Fortran 90 standard include:

- Array operations
- Improved facilities for numeric computation
- Parameterized intrinsic data types
- User-defined data types
- Facilities for modular data and procedure definitions
- Pointers
- The concept of language evolution
- Support for DATE_AND_TIME intrinsic for obtaining dates using a four-digit year format

VSI FORTRAN for OpenVMS

VSI Fortran contains full support for the Fortran 95 standard, including the following features:

- FORALL statement and construct
- Automatic deallocation of ALLOCATABLE arrays
- DIM argument to MAXLOC and MINLOC
- PURE user-defined subprograms
- ELEMENTAL user-defined subprograms (a restricted form of a pure procedure)
- Pointer initialization (initial value)
- The NULL intrinsic to nullify a pointer
- Derived-type structure initialization
- CPU_TIME intrinsic subroutine
- KIND argument to CEILING and FLOOR intrinsics
- Nested WHERE constructs, masked ELSEWHERE statement, and named WHERE constructs
- Comments allowed in namelist input
- Generic identifier in END INTERFACE statements
- Minimal width field editing using a numeric edit descriptor with 0 width
- Detection of Obsolescent and/or Deleted features listed in the Fortran 95 standard. VSI Fortran flags these obsolescent and deleted features, but fully supports them.

VSI Fortran includes the following features and enhancements:

- CDD (Common Data Dictionary) support and DML (Data Manipulation Language) support, formerly only available in the old Fortran 77 (*old_f77*) compiler, are now available in the Fortran 90/95 compiler
- Full support for 64-bit address space, including 64-bit static space
- Support for providing cross-reference information to the DEC Source Code Analyzer component of DECset for OpenVMS
- Support for linking against static and shared libraries
- Support for creating shareable code to be put into a shared library
- Support for stack-based storage
- Support for dynamic memory allocation
- Support for reading and writing binary data files in nonnative formats, including IEEE® (little-endian and big-endian), VAX, IBM® System/360, and CRAY® integer and floating point formats
- User control over IEEE floating point exception handling, reporting, and resulting values
- Control for memory boundary alignment of items in COMMON and fields in structures and warnings for unaligned data
- Directives to control listing page titles and subtitles, object file identification field, COMMON and record field alignment, and some attributes of COMMON blocks
- Ability to CALL an external function subprogram
- 7200 Character Statement Length
- Free form unlimited line length
- Mixing Subroutines/Functions in Generic Interfaces
- Composite data declarations using STRUCTURE, END STRUCTURE, and RECORD statements, and access to record components through field references
- Explicit specification of storage allocation units for data types such as:
 - INTEGER*4
 - LOGICAL*4
 - REAL*4
 - REAL*8
 - COMPLEX*8
- Support for 64-bit signed integers using INTEGER*8 and LOGICAL*8
- Support for 128-bit floating-point real numbers (reals) using REAL*16 and COMPLEX*32
- A set of data types:
 - — BYTE
 - — LOGICAL*1, LOGICAL*2, LOGICAL*4, LOGICAL*8
 - — INTEGER*1, INTEGER*2, INTEGER*4, INTEGER*8
 - — REAL*4, REAL*8, REAL*16
 - — COMPLEX*8, COMPLEX*16, DOUBLE COMPLEX, COMPLEX*32
 - — POINTER (CRAY style)
- Data statement style initialization in type declaration statements
- AUTOMATIC and STATIC statements
- Bit constants to initialize LOGICAL, REAL, and INTEGER values and participate in arithmetic and logical expressions

VSI FORTRAN for OpenVMS

- Built-in functions %LOC, %REF, %VAL, and %DESCR
- VOLATILE statement
- Bit manipulation functions
- Binary, hexadecimal, and octal constants and Z and O format edit descriptors applicable to all data types
- I/O unit numbers that can be any nonnegative INTEGER*4 value
- Variable amounts of data can be read from and written to ^STREAM~ files, which contain no record delimiters
- ENCODE and DECODE statements
- ACCEPT, TYPE, and REWRITE input/output statements
- DEFINE FILE, UNLOCK, and DELETE statements
- USEROPEN subroutine invocation at file OPEN time
- Support for reading nondelimited character strings as input for character NAMELIST items
- Debug statements in source
- Generation of a source listing file with optional machine code representation of the executable source
- Variable format expressions in a FORMAT statement
- Optional run-time bounds checking of array subscripts and character substrings
- 31-character identifiers that can include dollar sign (\$) and underscore (_)
- Support for executing in-line assembler code using the ASM intrinsics (Alpha only)
- Support for the supercomputer intrinsics POPCNT, POPPAR, LEADZ, TRAILZ, and MULT_HIGH
- Language elements that support the various extended range and extended precision floating point architectural features:
 - 32-bit VAX F_floating data type, with an 8-bit exponent and 24-bit mantissa, which provides a range of 0.293873588E-38 to 1.7014117E38 and a precision of typically 7 decimal digits. Calculations with F_floating data on I64 are performed using the S_floating data type.
 - 64-bit VAX D_floating data type, with an 8-bit exponent and 56-bit mantissa, which provides a range of 0.2938735877055719D-38 to 1.70141183460469229D38 and a precision of typically 16 decimal digits. Calculations with D_floating data on Alpha systems use G_floating precision (53-bit instead of 56-bit mantissa). Calculations with D_floating data on I64 are performed using the T_floating data type.
 - 64-bit VAX G_floating data type, with an 11-bit exponent and 53-bit mantissa, which provides a range of 0.5562684646268004D-308 to 0.89884656743115785407D308 and a precision of typically 15 decimal digits. Calculations with G_floating data on I64 are performed using the T_floating data type.
 - 32-bit IEEE S_floating data type, with an 8-bit exponent and 24-bit mantissa, which provides a range of 1.17549435E-38 (normalized) to 3.40282347E38 (the IEEE denormalized limit is 1.40129846E-45) and a precision of typically 7 decimal digits
 - 64-bit IEEE T_floating data type, with an 11-bit exponent and 53-bit mantissa, which provides a range of 2.2250738585072013D-308 (normalized) to 1.7976931348623158D308 (the IEEE denormalized limit is 4.94065645841246544D-324) and a precision of typically 15 decimal digits
 - 128-bit IEEE extended Alpha X_floating data type, with a 15-bit exponent and a 113-bit mantissa, which provides a range of approximately 6.48Q-4966 to 1.18Q4932 and a precision of typically 33 decimal digits
 - The following combinations of floating types may be specified:
 - F, G and X (the default on Alpha)
 - S, T and X (IEEE) (the default on I64)
- Command line control for:
 - The size of default INTEGER, REAL, and DOUBLE PRECISION data items
 - The levels and types of optimization to be applied to the program
 - The directories to search for INCLUDE files
 - Inclusion or suppression of various compile-time warnings
- Inclusion or suppression of run-time checking for various I/O and computational errors
- Control over whether compilation terminates after a specific number of errors has been found
- Choosing whether executing code will be thread-reentrant
- Internal procedures can be passed as actual arguments to procedures

VSI FORTRAN for OpenVMS

- Kind types for all of the hardware-supported data types:
 - For 1-, 2-, 4-, and 8-byte LOGICAL data:
 - LOGICAL (KIND=1)
 - LOGICAL (KIND=2)
 - LOGICAL (KIND=4)
 - LOGICAL (KIND=8)
 - For 1-, 2-, 4-, and 8-byte INTEGER data:
 - INTEGER (KIND=1)
 - INTEGER (KIND=2)
 - INTEGER (KIND=4)
 - INTEGER (KIND=8)
 - For 4-, 8-, and 16-byte REAL data:
 - REAL (KIND=4)
 - REAL (KIND=8)
 - REAL (KIND=16)
 - For single precision, double precision, and quad-precision COMPLEX data:
 - COMPLEX (KIND=4)
 - COMPLEX (KIND=8)
 - COMPLEX (KIND=16)

VSI Fortran takes advantage of OpenVMS facilities to include the following features and enhancements in both VSI Fortran 95/90 and VSI Fortran 77:

- Language elements for keyed and sequential access to OpenVMS RMS indexed organization files
- The ability to specify an OpenVMS text library module in an INCLUDE statement
- Support for calls to OpenVMS system service and Run-Time Library procedures
- Generation of symbol tables for the OpenVMS Symbolic Debugger
- LIB\$ESTABLISH and LIB\$REVERT are provided as intrinsic functions for compatibility with VSI Fortran exception handling
- Support for providing error diagnostics to the DEC Language-Sensitive Editor component of DECset for OpenVMS
- FDML (Fortran Data Manipulation Language) support (I64 only)

VSI Fortran 77 contains the following extensions to the FORTRAN 77 standard: (Alpha only)

- Support for recursive subprograms
- IMPLICIT NONE statements
- INCLUDE statement
- NAMELIST-directed I/O
- DO WHILE and ENDDO statements
- Use of exclamation point (!) for end of line comments
- Generation of Cross Reference Listings
- Support for NTT Technical Requirement TR550001, Multivendor Integration Architecture (MIA) Version 1.1, Division 2, Part 3-2, Programming Language FORTRAN
- Support for automatic arrays
- Support for the SELECT CASE - CASE - CASE DEFAULT - END SELECT statements
- Support for the EXIT and CYCLE statements and for construct names on DO - END DO statements
- Reporting of unused and uninitialized variables
- Support for DATE_AND_TIME intrinsic for obtaining dates using a four-digit year format

VSI Fortran 77 takes advantage of OpenVMS facilities to include the following features and enhancements:

- Support for translation of CDD/Repository records into Fortran records for Fortran F77 on OpenVMS Alpha and Fortran F90 on OpenVMS I64
- Support for the extraction of program design information in comments using the DEC Source Code Analyzer component of DECset for OpenVMS

VSI Fortran provides a multiphase optimizer that is capable of performing optimizations across entire programs. Specific optimizations performed by both VSI Fortran 95/90 and Fortran 77 include:

- Constant folding
- Optimizations of arithmetic IF, logical IF, and block IF-THEN-ELSE
- Global common subexpression elimination
- Removal of invariant expressions from loops
- Global allocation of general registers across program units
- In-line expansion of statement functions and routines
- Optimization of array addressing in loops

VSI FORTRAN for OpenVMS

- Value propagation
- Deletion of redundant and unreachable code
- Loop unrolling
- Thorough dependence analysis
- Software pipelining to rearrange instructions between different unrolled loop iterations
- Optimized interface to intrinsic functions
- 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

Specific optimizations performed by VSI Fortran 95/90 include:

- Array temporary elimination

Both VSI Fortran 95/90 and VSI Fortran 77 are shareable, re-entrant compilers that operate under the OpenVMS operating system. They globally optimize source programs while taking advantage of the native instruction set and the OpenVMS virtual memory system.

VSI EXTENDED MATH LIBRARY (CXML) ALPHA ONLY

VSI Extended Math Library (CXML) for OpenVMS Alpha is a set of mathematical subprograms that are optimized for VSI architectures. Included subprograms cover the areas of:

- Basic Linear Algebra
- Linear System and Eigenproblem Solvers
- Sparse Linear System Solvers
- Sorting
- Random Number Generation
- Signal Processing

The Basic Linear Algebra library includes the industry-standard Basic Linear Algebra Subprograms (BLAS) Level 1, Level 2, and Level 3. Also included are subprograms for BLAS Level 1 Extensions, Sparse BLAS Level 1, and Array Math Functions (VLIB).

The Linear System and Eigenproblem Solver library provides the complete LAPACK v2 package developed by a consortium of university and government laboratories. LAPACK is an industry-standard subprogram package offering an extensive set of linear system and eigenproblem solvers. LAPACK uses blocked algorithms that are better suited to most modern architectures, particularly ones with memory hierarchies. LAPACK will supersede LINPACK and EISPACK for most users.

The Sparse Linear System library provides both direct and iterative sparse linear system solvers. The direct solver package supports both symmetric and nonsymmetric sparse matrices stored using the skyline storage scheme. The iterative solver package contains a basic set of storage schemes, preconditioners, and iterative solvers. The design of this package is modular and matrix-free, allowing future expansion and easy modification by users.

The Signal Processing library provides a basic set of signal processing functions. Included are one-, two-, and three-dimensional Fast Fourier Transforms (FFT), group FFTs, Cosine/Sine Transforms (FCT/FST), Convolution, Correlation, and Digital Filters.

Many CXML subprograms are optimized for the supported hardware platforms. Optimization techniques include traditional optimizations such as loop unrolling and loop reordering. CXML subprograms also provide efficient management of the hierarchical memory system, using techniques such as the following:

- Reuse of data within registers to minimize memory accesses
- Efficient cache management
- Use of blocked algorithms that minimize translation buffer misses and unnecessary paging

Since CXML routines can be called from all languages that support the OpenVMS calling standard, the library provides optimized computation for applications written in these languages. Where appropriate, most subprograms are available in both real and complex versions, as well as in both single and double precision. CXML for OpenVMS Alpha supports both IEEE and VAX floating-point formats.

VSI FORTRAN for OpenVMS

Basic Linear Algebra Subprograms

Linear algebra operations are fundamental to many mathematical applications, and several libraries of linear algebra subprograms exist throughout the computer industry. The CXML BLAS library contains the most commonly used linear algebra subprograms.

The CXML linear algebra library contains five groups of subprograms at three levels:

- Basic Linear Algebra Subprograms (BLAS) Level 1
- BLAS Level 1 Extensions
- BLAS Level 1 Sparse Extensions
- BLAS Level 2
- BLAS Level 3

BLAS Level 1 (Scalar/Vector and Vector/Vector Operations)

BLAS Level 1 provides a set of elementary vector functions, operating on one or two vectors. These are typically very small routines, and they make less efficient use of the computing resources of modern computer architectures than the Level 2 and 3 operations.

CXML provides the 15 standard BLAS Level 1 operations:

- The index of the element of a vector having maximum absolute value
- The sum of the absolute values of the elements of a vector
- Inner product of two real vectors
- Scalar plus the extended precision inner product of two real vectors
- Conjugated inner product of two complex vectors
- Unconjugated inner product of two complex vectors
- Square root of the sum of squares (norm) of the element of a vector
- Scalar times a vector plus a vector
- Copy one vector to another
- Apply a Givens rotation
- Apply a modified Givens plane rotation
- Generate elements for a Givens plane rotation
- Generate elements for a modified Givens plane rotation
- Product of a vector times a scalar
- Swap the elements of two vectors

BLAS Level 1 Extensions (Vector/Vector Operations)

When developing mathematical algorithms using the BLAS Level 1, scientists and engineers found that several additional constructs were used on a regular basis. These constructs are well known throughout the computer industry as BLAS Level 1 Extensions.

CXML contains 13 BLAS Level 1 Extension operations:

- Index of element having the minimum absolute value
- Index of element having the maximum value
- Index of element having the minimum value
- Largest value of the elements of a vector
- Smallest value of the elements of a vector
- Largest absolute value of the elements of a vector
- Smallest absolute value of the elements of a vector
- Sum of the values of the elements of a vector
- Set all elements of a vector equal to a scalar
- Constant times a vector set to another vector ($y = a_x$)
- Euclidean norm with no intermediate scaling
- Sum of the squares of the elements of a vector
- Constant times a vector plus a vector set to another vector ($z = a_x + y$)

BLAS Level 1 Sparse Extensions (Vector/Vector Operations)

This group of operations is similar to the BLAS Level 1 routines, but is designed to work on sparse vectors (vectors in which most of the elements are zero). Six of the routines are from industry standard Sparse BLAS 1, and the remaining three are enhancements.

VSI FORTRAN for OpenVMS

The nine sparse BLAS Level 1 operations are:

- Scalar times a sparse vector plus a vector
- Sum of a sparse vector and a full vector
- Inner product of a sparse vector and a full vector
- Gather a sparse vector from a full vector
- Gather a sparse vector from the scaled elements of a full vector
- Gather a sparse vector from a full vector and zero corresponding elements of full vector
- Apply Givens rotation to a sparse vector and a full vector
- Scatter a sparse vector into a full vector
- Scale and scatter a sparse vector into a full vector

BLAS Level 2 (Matrix/Vector Operations)

The BLAS Level 2 codes make more effective use of the data in the registers, reducing the number of register loads and stores required. In addition, loop un-rolling techniques are used to minimize cache misses and page faults. The BLAS Level 2 subprograms use the following types of operations:

- Matrix/vector products
- Rank-1 and rank-2 matrix updates
- Solutions of triangular systems of equations

Six types of matrices are supported by these BLAS Level 2 routines:

- General
- General band
- Symmetric/Hermitian
- Symmetric/Hermitian band
- Triangular
- Triangular band

BLAS Level 3 (Matrix/Matrix Operations)

The BLAS Level 3 routines operate at a level that makes the most efficient use of machine resources. CXML optimizes these routines by partitioning matrices into blocks and computing matrix/matrix operations on each block. This approach avoids excessive memory accesses by providing full reuse of data while each block is in the cache or the registers. BLAS Level 3 routines provide this kind of blocking for three basic types of operations:

- Matrix/matrix products
- Rank-k and rank-2k updates of a symmetric matrix
- Solving triangular systems of equations with multiple right-hand sides

Three types of matrices are supported by these BLAS Level 3 routines:

- General
- Symmetric/Hermitian
- Triangular

A set of additional matrix-matrix routines is provided:

- Add two matrices
- Subtract one matrix from another
- Transpose a matrix, in-place or out-of-place

Array Math Functions

The Array Math Functions provide a set of basic math functions that operate on arrays of numbers rather than on scalars. On vector and superscalar architectures, such functions have a performance advantage over a loop of scalar operations. The library includes the following array functions for double precision numbers:

- Sine of array
- Cosine of array
- Cosine and sine of array
- Exponent of array
- Logarithm of array
- Square root of array
- Reciprocal of array

VSI FORTRAN for OpenVMS

LAPACK Library Contents

LAPACK is a library of linear algebra subprograms intended to solve a wide range of problems in linear algebra. LAPACK can be used to solve dense systems of linear equations, linear least squares problems, eigenvalue problems, and singular value problems. It is also useful in doing other computations such as matrix factorizations and estimations of condition numbers.

The CXML LAPACK library provides the complete LAPACK v2 package. CXML's version of LAPACK is provided as a packaged library, compiled, tested, and ready to use. Combined with the optimized BLAS Level 3 routines, the CXML LAPACK will provide optimal performance on all supported platforms. LAPACK should be used in place of LINPACK and EISPACK, because it is more efficient, accurate, and robust.

LAPACK supports both real and complex, single and double precision data. It operates on the following types of matrices:

- Bidiagonal
- General band
- General unsymmetric
- General tridiagonal
- Hermitian
- Hermitian, packed storage
- Upper Hessenberg, generalized problem
- Upper Hessenberg
- Orthogonal
- Orthogonal, packed storage
- Symmetric/Hermitian positive definite band
- Symmetric/Hermitian positive definite
- Symmetric/Hermitian positive definite, packed storage
- Symmetric/Hermitian positive definite tridiagonal
- Symmetric band
- Symmetric, packed storage
- Symmetric tridiagonal
- Symmetric
- Triangular band
- Triangular, generalized problem
- Triangular, packed storage
- Triangular
- Trapezoidal
- Unitary
- Unitary, packed storage

LAPACK provides the following operations:

- Triangular factorization
- Unblocked triangular factorization
- Solve a system of linear equations (based on triangular factorization)
- Compute the inverse (based on triangular factorization)
- Compute a split Cholesky factorization of a symmetric/Hermitian positive definite band matrix
- Unblocked computation of inverse
- Estimate condition number
- Refine initial solution returned by solver
- Perform QR factorization without pivoting
- Unblocked QR factorization
- Solve linear least squares problem (based on QR factorization)
- Solve the linear equality constrained least squares (LSE) problem
- Solve the Gauss-Markov linear model problem
- Perform LQ factorization without pivoting
- Unblocked LQ factorization
- Solve underdetermined linear system (based on LQ factorization)
- Generate a real orthogonal or complex unitary matrix as a product of Householder matrices
- Unblocked generation of real orthogonal or unitary matrix
- Multiply a matrix by a real orthogonal or complex unitary matrix by applying a product of Householder matrices
- Unblocked version of multiplication of a matrix by a real orthogonal or complex unitary matrix by applying a product of Householder matrices

VSI FORTRAN for OpenVMS

- Reduce a square matrix to upper Hessenberg form
- Unblocked version of square matrix reduction
- Reduce a symmetric matrix to real symmetric tridiagonal form
- Reduce a band matrix to bidiagonal form
- Unblocked version of symmetric matrix reduction
- Reduce a rectangular matrix to bidiagonal form
- Reduce a band symmetric/Hermitian matrix to tridiagonal form
- Reduce a symmetric/Hermitian-definite banded generalized eigenproblem to standard form
- Compute various norms of a complex Hermitian tridiagonal matrix
- Compute eigenvalues and optional Schur factorization or eigenvectors using QR algorithm
- Compute selected eigenvectors by inverse iteration
- Compute eigenvectors from Schur factorization
- Compute eigenvectors using the Pal-Walker-Kahan variant of the QL or QR algorithm
- For a pair of N-by-N real nonsymmetric matrices, compute the generalized eigenvalues, the real Schur form, and the left and/or right Schur vectors
- For a pair of N-by-N real nonsymmetric matrices, compute the generalized eigenvalues, and the left and/or right generalized eigenvectors
- Solve the generalized nonsymmetric eigenproblem $Ax = \lambda Bx$
- Solve the generalized definite banded eigenproblem $Ax = \lambda Bx$
- Solve the generalized symmetric/Hermitian-definite banded eigenproblem
- Solve the symmetric eigenproblem using divide-and-conquer algorithm
- Compute singular values and, optionally, singular vectors using the QR algorithm
- Compute the generalized (quotient) singular value decomposition
- Compute the generalized singular value decomposition (GSVD) on the M-by-N matrix A and P-by-N matrix B
- Solve a generalized linear regression model problem

Sparse System Solver Subprograms

The CXML Sparse System Solver library contains a set of subprograms that can be used to solve sparse linear systems of equations. Two packages providing direct and iterative methods are supported.

Direct Method Sparse Solver Package:

The direct solver package includes skyline (profile) solvers for symmetric and nonsymmetric matrices. Separate factorization and solver routines are provided to allow repeated use of the solver for multiple right hand sides, without repeating the factorization. To make the subprograms easier to use, both simple and expert driver routines are provided. Functions provided include:

- LDU factorization
- Solve
- Norm evaluation
- Condition number estimation
- Iterative refinement
- Simple and expert drivers

These storage schemes are supported for symmetric and nonsymmetric matrices:

- Profile-in storage
- Structurally symmetric, profile-in storage (for nonsymmetric only)
- Diagonal-out storage

Iterative Method Sparse Solver Package:

For the iterative method, the library provides a modular set of storage schemes, preconditioners, and solvers. These solvers and preconditioners are easily accessed through an integrated driver routine.

Six iterative sparse solvers for real, double precision data are supplied:

- Preconditioned conjugate gradient method
- Preconditioned least squares conjugate gradient method
- Preconditioned biconjugate method
- Preconditioned conjugate gradient squared method
- Preconditioned generalized minimum residual method
- Preconditioned transpose free QMR method

VSI FORTRAN for OpenVMS

Routines for three storage schemes are provided, or the user can develop routines to employ a custom storage scheme. The supplied storage schemes include:

- Symmetric diagonal
- Unsymmetric diagonal
- General storage by rows

Three preconditioners are supplied, which can be selectively applied to the data. Users can also supply custom preconditioners. The preconditioners supplied include:

- Diagonal
- Polynomial (Neumann)
- Incomplete LU with zero diagonals added

Sorting Subprograms

Two sort subprograms using the Quicksort algorithm and two general purpose radix sort subprograms are provided, as follows:

- Sort elements of a vector using the Quicksort algorithm
- Sort an indexed vector of data using the Quicksort algorithm
- Sort data using a radix sort algorithm
- Sort an indexed vector of data using a radix sort algorithm
- All of the above sorts operate on data stored in memory.

Random Number Subprograms

CXML provides four random number generator subprograms:

- Produce a vector of uniform $[0,1]$, long-period random numbers using the L'Ecuyer multiplicative method. Two auxiliary input routines are provided to allow this subprogram to be called from within a parallel section of a program.
- Produce a vector of $N(0,1)$, normally-distributed random numbers. Two auxiliary input routines are provided to allow this subprogram to be called from within a parallel section of a program.
- Produce single precision random numbers using a linear multiplicative algorithm
- Produce single precision random numbers using a Lehmer multiplicative generator

Signal Processing Subprograms

The CXML Signal Processing library contains a set of subprograms in four basic areas of signal processing:

- Fast Fourier Transforms (FFT)
- Fast Cosine and Fast Sine Transforms (FCT and FST)
- Convolution and correlation
- Digital filters

Fast Fourier Transforms and Cosine and Sine Transforms

CXML provides one-dimensional, two-dimensional, three-dimensional, and group FFT routines and one dimensional FCT/FST routines. Each routine is supplied in two forms:

- The first form computes the transform in one unit operation. This is convenient for programs requiring speed on only one or a few operations.
- The second form is provided for programs requiring speed on repeated operations. With this form, each routine is subdivided into three routines. One routine builds the rotation factors, a second routine applies them to perform the transform, and a third routine deallocates any virtual memory allocated in the first routine. Thus, for repeated operations, the rotation factors need to be built only once.

Convolution and Correlation

CXML provides routines for computing one-dimensional discrete convolutions and correlations. These routines can process both periodic and nonperiodic data.

Digital Filters

CXML provides support for one-dimensional, nonrecursive digital filtering. Based on the Kaisers Sinh-Bessel algorithm, these routines allow programming of bandpass, bandstop, low-pass, and high-pass filters.

VSI FORTRAN for OpenVMS

Cray SciLib Portability Support

SCIPOINT is a VSI implementation of v7 of the Cray Research scientific numerical library, SciLib. SCIPOINT provides 64 bit single-precision and 64-bit integer interfaces to underlying CXML routines for Cray users porting programs to Alpha systems running OpenVMS. SCIPOINT also provides equivalent versions of almost all Cray Math Library and CF77 (Cray Fortran 77) Math intrinsic routines.

In order to be completely source code compatible with SciLib, the SCIPOINT library calling sequence supports 64-bit integers passed by reference. However, internally, SCIPOINT uses 32 bit integers. Consequently, some run-time uses of SciLib are not supported by SCIPOINT.

SCIPOINT provides the following:

- 64-bit versions of all Cray SciLib single-precision BLAS Level 1, Level 2, and Level 3 routines
- All Cray SciLib LAPACK routines
- All Cray SciLib Special Linear System Solver routines
- All Cray SciLib Signal Processing routines
- All Cray SciLib Sorting and Searching routines

These routines are completely interchangeable with their Cray SciLib counterparts up to the runtime limit on integer size, and with the exception of the ORDERS routine, require no program changes to function correctly. Owing to endian differences of machine architecture, special considerations must be given when the ORDERS routine is used to sort multibyte character strings.

RUN-TIME LIBRARY REDISTRIBUTION

The Fortran kit may include updated Run-Time Library shareable images. VSI grants the user a nonexclusive royalty-free worldwide right to reproduce and distribute the executable version of the Run-Time Library (the "RTLs"), provided that the user does all of the following:

- Distributes the RTLs only in conjunction with and as a part of the user's software application product that is designed to operate in the OpenVMS environment.
- Does not use the name, logo, or trademarks of VSI to market the user's software application product.
- Includes the copyright notice of VSI Fortran on the user's product disk label and/or on the title page of the documentation for software application product.
- Agrees to indemnify, hold harmless, and defend VSI from and against any claims or lawsuits, including attorney's fees, which arise or result from the use or distribution of the software application product.

Except as expressly provided herein, VSI grants no implied or express license under any of its patents, copyrights, trade secrets, trademarks, or any license or other proprietary interests and rights.

The RTL image is designated as DEC\$FORRTL.EXE. VSI Fortran may include a separate installation kit for the purpose of installing the VSI Fortran Run-Time Library. This kit, installable with the POLYCENTER® Software Installation Utility (a component of OpenVMS), must be used to install the RTL image on other systems.

HARDWARE REQUIREMENTS

Processors Supported:

- Integrity: Any Integrity system capable of running the VSI OpenVMS Integrity Operating System Version 8.4-2 or higher.
- Alpha: Any AlphaServer system capable of running the VSI OpenVMS Alpha Operating System Version 8.4-2L1 or higher.

Refer to the latest VSI OpenVMS Integrity or Alpha Software Product Description for information about supported servers.

Disk Space Requirements on VSI OpenVMS Alpha (Block Cluster Size = 1)

Disk space required for compiler installation:	50,000 blocks (25.0 MB)
Disk space required for use (permanent):	40,200 blocks (20.1 MB)
Disk space required for CXML installation:	150,000 blocks (75 MB)
Disk space required for CXML use (permanent):	150,000 blocks (75 MB)

VSI FORTRAN for OpenVMS

Disk Space Requirements on VSI OpenVMS Integrity (Block Cluster Size = 1)

Disk space required for compiler installation:	62,000 blocks (31.0 MB)
Disk space required for use (permanent):	60,000 blocks (30.0 MB)

These counts refer to the disk space required on the system disk. The sizes are approximate. Actual sizes may vary depending on the user's system environment, configuration, and software options.

SOFTWARE REQUIREMENTS

On Integrity servers, VSI OpenVMS Integrity Version 8.4-2 or higher is the required operating system version for this product. On AlphaServer systems, VSI OpenVMS Alpha Version 8.4-2L1 or higher is the required operating system version for this product.

The OpenVMS operating system can be configured to include or omit certain components. VSI Fortran requires the following components to be included:

- Programming Support
- Utilities

The default for OpenVMS Alpha installation is to include all components.

SOFTWARE LICENSING

A software license is required in order to use the VSI Fortran software product.

- For Integrity servers, the license is a Concurrent Use license. Version update licenses are not available for the Integrity servers platform. Rights to use future revisions of VSI Fortran are available only through a Support Agreement or through a new license purchase.
- For AlphaServer systems, the license to use VSI Fortran is included in the ALPHA-LP license.

For more information about OpenVMS licensing terms and policies, contact your VSI account representative. Information is also available at the following website:

<http://vmssoftware.com/services>

LICENSE MANAGEMENT FACILITY SUPPORT

VSI Fortran for OpenVMS supports the *OpenVMS License Management Facility*.

For more information about the License Management Facility, refer to the *VSI OpenVMS License Management Utility Manual* in the OpenVMS documentation set.

CLUSTER ENVIRONMENT

This layered product is fully supported when installed on any valid and licensed OpenVMS Cluster configuration, which are fully described in the *OpenVMS Cluster Software Product Description (SPD DO-VIBHAA-032)*. See the HARDWARE REQUIREMENTS section in this document for hardware requirements.

OPTIONAL SOFTWARE

- DECset for VSI OpenVMS Alpha
- CDD/Repository for OpenVMS Alpha
- DECset for VSI OpenVMS I64

GROWTH CONSIDERATIONS

The minimum hardware and software requirements for any future version of this product may be different from the requirements for the current version.

VSI FORTRAN for OpenVMS

ORDERING INFORMATION

For VSI Fortran on OpenVMS, licenses are available as electronic licenses (E-LTU) or physical licenses (P-LTU):

For VSI Integrity:

- VSI Fortran for VMS I64 Concurrent E-LTU SL-LIFO0E-82V
- VSI Fortran for VMS I64 Concurrent P-LTU SL-LIFO0P-82V

For VSI Alpha:

- VSI Fortran for OpenVMS Included in the ALPHA-LP license bundle

SOFTWARE PRODUCT SERVICES

A variety of service options are available from VSI. For more information, contact your VSI account representative or distributor. Information is also available at the following website:

<http://vmssoftware.com/services>

SOFTWARE WARRANTY

This software product is provided by VSI with a 90-day conformance warranty in accordance with the VSI warranty terms applicable to the license purchase.

Copyright © 2017 VMS Software, Inc., Bolton Massachusetts, USA

Confidential computer software. Valid license from VSI required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

The information contained herein is subject to change without notice. The only warranties for VSI products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. VSI shall not be liable for technical or editorial errors or omissions contained herein.