Software Product Description

PRODUCT NAME: VAX PL/I, Version 3.4 **SPD 25.30.19**

DESCRIPTION

VAX PL/I is an extended implementation of the ANSI X3.74 1981, American National Standard PL/I General Purpose Subset. VAX PL/I extensions include compatibility features with industry standard implementations and ANSI full language features as well as VMS system-specific features. VAX PL/I consists of a shareable compiler, a HELP facility and a system interface library which includes declarations for system routines. The VAX PL/I compiler runs under the VMS Operating System and generates optimized, position-independent machine code.

VAX PL/I is a comprehensive and powerful programming language that supports scientific computation, commercial data handling and data organization, and extensive string manipulation. The block-structuring provided by the PL/I language helps to reduce the costs of program development and support.

VAX PL/I allows access to VAX CDD/Plus. A compile-time preprocessor facility allows language extension and conditional compilation. All VMS System Services, the VAX Common Runtime Library and VAX system utilities are available through the PL/I CALL statement. A library of predefined ENTRY declarations provided with VAX PL/I minimizes the coding required to utilize VMS system services, the VAX Common Runtime Library and many system utilities.

As a native-mode VAX language, VAX PL/I is integrated into the VAX common language environment. This integration provides VAX PL/I users with support for the VAX Interlanguage Calling Standard, access to the VAX Symbolic Debugger (including support for source-line debugging), and callable interfaces to VAX utilities and optional products (such as SORT, VAX DATATRIEVE, and VAX DBMS).

VAX PL/I also interfaces to the VAX Language-Sensitive Editor/Source Code Analyzer. Source programs can be written and compiled using the Language-Sensitive Editor component which has built-in intelligence about the source format of PL/I programs.

VAX PL/I also provides support for low level program design, including the processing of pseudo code and the extraction of design information from comments.

Features

- Support for the VAX CDD/Plus, allowing VAX PL/I programmers to extract a designated VAX CDD/Plus record description node and represent the record as a PL/I structure declaration. VAX CDD/Plus structure declarations may be optionally included in the compiler listing.
- A full complement of VAX datatypes:

  Binary integer (FIXED BINARY), floating point (FLOAT BINARY, FLOAT DECIMAL), decimal (FIXED DECIMAL), fixed or varying length character strings (CHARACTER [VARYING]), fixed length bit strings (BIT), edited numeric data in character format (PICTURE), address manipulation (AREA, OFFSET, and POINTER), entry point (ENTRY) and label with an optional subscript (LABEL).
- Language elements that support the VAX extended range and extended precision floating point architectural features:

  64-bit G_floating point data type, with an 11-bit exponent and 53-bit mantissa, which provides a range of 0.56*10**-308 to 0.09*10**308 and a precision of 15 decimal digits.

  128-bit H_floating data type, with a 15-bit exponent and a 113-bit mantissa, which provides a range of 0.84*10**-4932 to 0.59*10**4932 and a precision of 33 decimal digits.
- An assignment operator that operates on equivalent structures or arrays of data as well as scalar variables. A scalar can be assigned to an entire array.

- Support for all VAX RMS file organizations (sequential, relative, and indexed), and access methods (sequential, direct, and keyed). A set of ENVIRONMENT options provides access to a large subset of RMS features. The OPTIONS option on READ provides extended control of record locking.
- Five storage classes:

AUTOMATIC:	Variables are allocated upon block entry.
STATIC:	Variables are allocated at compile time. Static data can be EXTERNAL or globally shared. The GLOBALDEF attribute provides program section (PSECT) control of data.
DEFINED:	Variables are overlaid upon existing variables.
BASED:	Variable allocation is dynamically controlled by the programmer.
CONTROLLED:	Variables are allocated and freed dynamically as generations. Only the most recent generation is available to the programmer.

- INITIAL values may be specified for AUTOMATIC, BASED, CONTROLLED, and STATIC variables.
- A REFER option is available for the creation of dynamically self-defining based structures. This option may be specified for any or all bounds and extents for structure members which are arrays, bit strings, character strings or areas.
- ALLOCATE statement with the SET and IN options for explicit dynamic storage allocation.

  FREE space previously allocated by the ALLOCATE statement.

  POINTER and OFFSET datatypes for address manipulation of scalar and aggregate data.

  AREA as an address base for variables of OFFSET type with language supported allocation and deallocation.
- LIKE attribute to allow the members of a structure to be declared in terms of an already-declared structure.
- UNION attribute to declare overlaid minor structures.
- A powerful set of structured program control statements.

  DO statement with TO, BY, WHILE, UNTIL, and REPEAT options.

  LEAVE statement to transfer control out of one or more levels of containing DO-groups.

  SELECT-WHEN-OTHERWISE group allowing CASE-like selection of a statement or statement group.

  IF...THEN...ELSE conditional statement.

  CALL statement and function reference (RETURN).

  GOTO statement for transfer of control.
- Condition Handling.

  ON statement to establish ON-units (for AREA, CONDITION, CONVERSION, ENDFILE, ENDPAGE, KEY, UNDEFINEDFILE, FIXEDOVERFLOW, OVERFLOW, UNDERFLOW, ZERODIVIDE, STORAGE, STRINGRANGE, SUBSCRIPTRANGE, ERROR, FINISH, ANYCONDITION, and VAXCONDITION conditions).

  REVERT statement to cancel ON-units.

  RESIGNAL statement that allows conditions to be passed to other ON-units.
- Input/Output Control.

  OPEN and CLOSE file control statements.

  READ, WRITE, DELETE, and REWRITE record-oriented I/O statements.

  GET and PUT stream-oriented I/O statements (with FILE, STRING, EDIT, LIST, PAGE, and SKIP options).

  ENVIRONMENT and OPTIONS clauses provide access to RMS features including a USEROPEN feature and extended record locking control.
- Program Structuring Statements.

  PROCEDURE blocks: Internal (nested) and EXTERNAL.

  BEGIN...END blocks, allowing local variable declaration.

  DO groups that provide compound statement capabilities.

  ENTRY statement that allows a routine to have multiple entry points.
- Preprocessor statements.

  %REPLACE statement for compile-time replacement of arithmetic, bit-, or character-string constants.

  %INCLUDE statement for compile-time source copying, with full library support for INCLUDE modules and default and user-specified system libraries.

  %DICTIONARY for CDD record extraction.

%DECLARE, %ACTIVATE, %DEACTIVATE statements for declaration and control of compile-time variables.

%DO...%END, %IF...%THEN...%ELSE, and %GOTO statements for compilation control.

%PROCEDURE to define compile-time procedures

%INFORM, %WARN, %ERROR, %FATAL statements for user-generated diagnostics.

%[NO]LIST_ALL, %[NO]LIST_DICTIONARY, %[NO]LIST_INCLUDE, %[NO]LIST_MACHINE, and %[NO]LIST_SOURCE statements for selective listing control.

%PAGE, %TITLE, %SBTTL statements for listing format control.

- Preprocessor expressions and built-in functions.
- Built-In Functions.

  A full set of arithmetic functions: ABS, ADD, CEIL, DIVIDE, FLOOR, MAX, MIN, MOD, MULTIPLY, ROUND, SIGN, SUBTRACT, TRUNC.

  A full set of mathematical (transcendental) functions: ACOS, ASIN, ATAN, ATAND, ATANH, COS, COSD, COSH, EXP, LOG, LOG10, LOG2, SIN, SIND, SINH, SQRT, TAN, TAND, TANH.

  String functions: BOOL, COLLATE, COPY, EVERY, HIGH, INDEX, LENGTH, LOW, MAXLENGTH, REVERSE, SEARCH, SOME, STRING, SUBSTR, TRANSLATE, TRIM, VERIFY.

  Conversion functions: BINARY, BIT, BYTE, CHARACTER, DECIMAL, DECODE, ENCODE, FIXED, FLOAT, RANK, UNSPEC, INT, POSINT.

  Condition-handling functions: ONARGSLIST, ONCHAR, ONCODE, ONFILE, ONKEY, ONSOURCE.

  Array-handling functions: DIMENSION, HBOUND, LBOUND, PROD, SUM.

  Storage functions: ADDR, ALLOCATION, EMPTY, NULL, OFFSET, POINTER, SIZE.

  Timekeeping functions: DATE, DATETIME, TIME.

  File Control functions: LINENO, PAGENO.

  Pseudovariables (functions allowed on left-hand side of an assignment): INT, ONCHAR, ONSOURCE, PAGENO, POSINT, STRING, SUBSTR, UNSPEC.

  Calling mechanism support functions: DESCRIPTOR, REFERENCE, VALUE, ACTUALCOUNT, PRESENT

  Picture Variable Validation function: VALID

- Built-in subroutines are provided for:

  File-handling: DISPLAY, EXTEND, FLUSH, FREE, NEXT_VOLUME, RELEASE, REWIND, SPACEBLOCK.

  Condition-handling: RESIGNAL.

- Other VAX PL/I language capabilities include:

  Expressions in format lists.

  Replication factors for string constants.

  Preprocessor statements in any context.

Compiler Options

Compile-time command qualifiers provide a variety of options:

- /[NO]ANALYSIS_DATA: Causes Source Code Analyzer component information to be generated.
- /[NO]CHECK: Produce extra code to check array and string references.
- /[NO]CROSS_REFERENCE: Produce an alphabetical symbol cross-reference.
- /[NO]DEBUG: Causes DEBUG information to be included with the object code.
- /[NO]DIAGNOSTICS: Causes Language-Sensitive Editor component information to be generated.
- /[NO]G_FLOAT: Specifies the default floating point representation.
- /[NO]LIST: Controls the production of a listing file.
- /SHOW: Selects specific listing options: source, CDD definitions, include files, map, statistics, trace, and expansion.
- /[NO]ERROR_LIMIT: Controls the compiler diagnostic message limit.
- /VARIANT: Permits specification of compilation variants.
- /[NO]MACHINE_CODE: Causes machine code to be listed with the source.
- /[NO]OBJECT: Controls the production of the object file.

- /[NO]OPTIMIZE: Controls optimizations performed by the compiler.
- /[NO]WARNINGS: Controls the printing of compiler warning messages.
- /LIBRARY: Indicates the associated file is a library of source text modules specified by %INCLUDE statements.
- /[NO]DESIGN: Controls the generation of design information for the Program Design Facility.

At the end of each compilation in which messages are generated, the VAX PL/I compiler will display the number of informational, warning, and error messages.

Optimizations

VAX PL/I generates efficient object code. Optimizations include:

- Value Propagation
- Subexpression elimination
- Allocation of local variables to registers
- Removal of invariant computations from loops
- Simplification of Boolean expressions
- Extensive special case code generation
- Pattern replacement in generated code
- Inline expansion of procedure calls

Industry PL/I Compatability

VAX PL/I provides many of those PL/I features often used by mainframe PL/I programmers. Conversion effort depends upon the individual program and the set of PL/I features used by the programmer. Well-structured programs that do not rely on system-specific or implementation-specific features convert with a minimum of effort (from no changes to a few percent of the lines in the program). Programs that use implementation-specific features such as ENVIRONMENT and OPTIONS can require a larger conversion effort.

Digital does not provide any special programs or other conversion aids. The user is responsible for determining the extent of any conversion effort and for providing appropriate conversion tools to convert programs and data.

Unusual conversion requirements may be necessary if programs use the machine-dependent representation of data. The VAX architecture organizes bytes within an integer differently than most other vendors' hardware. This can lead to different results when UNSPEC or DEFINED operations are used to convert between BIT and FIXED BINARY data.

HARDWARE REQUIREMENTS

VAX, MicroVAX, VAXstation or VAXserver configuration as specified in the System Support Addendum (SSA 25.30.19-x).

SOFTWARE REQUIREMENTS*

For Systems Using Terminals (No DECwindows Interface):

- VMS Operating System

For Workstations Running VWS:

- VMS Operating System
- VMS Workstation Software

For Workstations Running DECwindows:

- VMS Operating System (and necessary components of VMS DECwindows)

VAX PL/I supports a DECwindows compiler interface for compile qualifier selection on workstations. This interface is accessible from FileView.

* Refer to the System Support Addendum (SSA 25.30.19-x) for availability and required versions of prerequisite/optional software and for information regarding components of VMS DECwindows.

ORDERING INFORMATION

Software Licenses: QL-114A*-**
Software Media: QA-114A*-**
Software Documentation: QA-114AA-GZ
Software Product Services: QT-114A*-**

* Denotes variant fields. For additional information on available licenses, services and media, refer to the appropriate price book.

SOFTWARE LICENSING

This software is furnished under the licensing provisions of Digital Equipment Corporation's Standard Terms and Conditions.

For more information about Digital's licensing terms and policies, contact your local Digital office.

LICENSE MANAGEMENT FACILITY SUPPORT

This layered product supports the VMS License Management Facility. License units for this product are allocated on a CPU-capacity basis.

For more information on the License Management Facility, refer to the VMS Operating System Software Product Description (SPD 25.01.xx) or the License Management Facility manual of the VMS Operating System documentation set.

For more information about Digital's licensing terms and policies, contact your local Digital office.

SOFTWARE PRODUCT SERVICES

A variety of service options are available from Digital. For more information contact your local Digital office.

SOFTWARE WARRANTY

Warranty for this software product is provided by Digital with the purchase of a license for the product as defined in the Software Warranty Addendum of this SPD.

™ The DIGITAL Logo, VMS, DECwindows, VAX, MicroVAX, VAXstation, VAXserver, PL/I, RMS, and DATATRIEVE are trademarks of Digital Equipment Corporation.