SESSION REPORT

61	A735	New Features in VS-FORTRAN Release 3	65
SHARE NO.	SESSION NO.	SESSION TITLE	ATTENDANCE
FORTRAN		Sylvia "Sunnie" Sund	SLA
	PROJECT	SESSION CHAIRMAN	INST. CODE

Stanford Linear Accelerator Center, P.O. Box 4349, Bin 96, Stanford, CA. 94305 415-854-3300
SESSION CHAIRMAN'S COMPANY, ADDRESS, and PHONE NUMBER

A copy of the speaker's talk is attached

VS FORTRAN SHARE PRESENTATION

VS FORTRAN Release 3.0

G. R. Garabedian
IBM
Santa Teresa Laboratory
San Jose, California

1.0 GENERAL OVERVIEW - RELEASE 3.0 CONTENT

Release 3.0 of VS FORTRAN was announced April 5, 1983, and is currently available in the U.S. and World Trade countries. This release includes improved CHARACTER data handling, enhanced diagnostics and debugging aids, improved usage of the INCLUDE statement, compile- time checking of OPEN/CLOSE/INQUIRE parameters which are constants, continued execution after I/O failure, and automatic formatting of direct access files identified by the OPEN statement.

2.0 CHARACTER DATA TYPE IMPROVEMENTS

- CHARACTER Argument Passing

  The key change for CHARACTER data type handling was to make subprogram calls insensitive to this data type.

  In previous VS FORTRAN releases, parameter lists, which contained CHARACTER-type arguments, had the length of those arguments embedded within the parameter list.

  The new construction provides a means of passing arguments to functions and subroutines in such a manner that the information needed for character-type arguments is "transparent"; i.e., the parameter list can be referenced without any regard to the character-type-argument information.

  The method is to provide a double parameter list for all argument lists which contain any character-type argument or for any reference to a character-type function. The primary list consists of pointers to the actual arguments; the secondary list consists of pointers to the lengths of the actual arguments. The high-order bit in the last argument position of each part of the parameter list is set on. If there are no character-type arguments or if the function being referenced is not character-type, only a primary list is passed.

  The doubling of all parameter lists (except for intrinsic functions which do not involve character arguments and for implicitly invoked function references) not only implies that the parameter lists themselves are different but that the prologues of FORTRAN subprograms are different in order to process these changed parameter lists. Therefore, if any FORTRAN program, compiled previously to Release 3, which references subprograms with character-type arguments (or is a character-type function itself) is to be used with a FORTRAN program which is compiled with Release 3, then the old program must also be re-compiled with Release 3 of VS FORTRAN.

  Note that programs compiled on releases of VS FORTRAN before Release 3.0 with LANGLVL(66) as well as programs compiled with FORTRAN H Extended, FORTRAN G1 or FORTRAN F do not have to be recompiled to operate with VS FORTRAN Release 3.0.

  Remember also that if you have changed your assembler-language programs to receive the non-transparent parameter lists, these must be changed to operate with Release 3.0 compiled VS FORTRAN programs which contain CHARACTER-type data.

  The SC option has been removed. This Release 2.0 option had been used to specify those subroutine names for which length parameters for CHARACTER-type arguments were not to be passed. This option is no longer needed with the Release 3.0 "transparent" argument passing.

- CHARACTER Data Types in COMMON with non-CHARACTER Data Types

  CHARACTER type data may now share the same COMMON (both blank and named) with data of non-CHARACTER types. A warning message will be issued only if FIPS flagging has been requested.

- CHARACTER Data Types EQUIVALENCEd to non-CHARACTER Data Types

  CHARACTER type data may now be EQUIVALENCEd to data of non-CHARACTER types. A warning message will be issued only if FIPS flagging has been requested.

- CHARLEN Compiler Option

  A compiler option, CHARLEN, has been added to VS FORTRAN Release 3.0 which allows the user to specify the maximum length he wishes to use for CHARACTER data type.

  The compiler option is of the form:

  ```
                      CHARLEN(k)
  ```

  where k is an unsigned integer constant greater than 0 but no greater than 32767. This constant represents the maximum length permitted for any character variable, array element or function.

  The limits for the values of symbolic names of character constants(255), for character constants in FORMAT statements (255), and for character constants in the STOP/PAUSE statements (72) remain the same in Release 3.0.

  If k > 32767, then an error message is given and the value 32767 is used. If the option is not used, a value of 500 will be assumed. The value 500 is the default length.

  The advantage of the compiler option is that those users who do not require large character variables will not have the large character "temporaries" (which are generated as necessary for inherited length character entities) in their object programs.

3.0 ENHANCED DIAGNOSTIC AND DEBUGGING AIDS

- Symbolic Dump Capability

To improve debugging, VS FORTRAN now provides a symbolic dump.

The compiler option, SDUMP, causes the generation of a dictionary in the object text file. SDUMP is the default setting of the option and can be set off by specifying NOSDUMP when the dictionary is not desired, with the understanding that a symbolic dump of variables in that routine is not then possible.

A symbolic dump can be produced anywhere via a call to a new library routine SDUMP. Also, it is automatically called when certain abnormal termination conditions are detected.

If the user program does terminate abnormally, the dictionary of the data within the program is interrogated and the value of each variable in storage is printed according to the defined data type. The user is able to determine both where the program terminated with the traceback information as well as the status of the data at termination in terms of the FORTRAN data types.

A description of the SDUMP routine and examples of its output are described in Appendix E of the VS FORTRAN Language Reference manual.

135

Improved Compile Time Messages

Two options, TRMFLG and SRCFLG, have been added to control the display of the error messages on the terminal and listing data sets.

- Terminal display

  The option, TRMFLG, can be used to display the source statement in error on the SYSTERM data set along with the diagnostic message. (The error message is formatted for the terminal being used.) This facilitates finding the statement in error in the original source file.

  Certain error messages occurring after the scan of the source statement are displayed at the terminal alone. These are more global type of messages not linked to a particular source statement - like messages about ill-nested DO-loops.

  If you are using an operating system which does not allow interactive transactions (such as OS/VS batch or DOS/VSE), then specify NOTRMFLG and NOTERMINAL when you install VS FORTRAN in order to avoid messages about "no terminal on line" when this option is used.

- Batch compiler listing

  The option, SRCFLG, may be specified to insert diagnostic messages in the printed source listing. Error messages will be displayed immediately following the source statement in error during the scan of the source statement.

  Error messages occurring after the scan of the source statement will be displayed at the end of the listing.

  SRCFLG cannot be specified if NOSOURCE is also used.

- SYM Card Support

A new compiler option, SYM, is used to invoke the production of SYM cards in the object text file. These SYM cards can be used by TSO TEST for interactive debugging or can be obtained by a user program and used for any function desired.

This option produces SYM cards containing location information for variables within a FORTRAN program. Information for all referenced variables in each program unit is recorded by name and location in the appropriate CSECT. All entities in COMMON blocks are recorded by name and location.

4.0 IMPROVEMENTS IN INCLUDE FACILITY

The INCLUDE statement has been improved in two ways:

- Blocked Files for INCLUDE statements

  Statements to be INCLUDEd can now be placed in files which have a record format of fix-blocked when processed in the MVS/370, MVS/XA and VS1 environments. DOS/VSE does not provide a library structure which supports fix-blocked files.

  The block size of the data sets containing the members to be INCLUDEd may now be a multiple of 80. Moreover, when concatenating partitioned data sets, the block sizes may be different. However, the first data set in a concatenated sequence must have the largest block size. Therefore, when concatenating data sets with different block sizes, one may need to use a dummy data set first to establish the maximum block size.

- Conditional INCLUDE

  A new compiler option is provided of the form: CI(n1,n2,...).

  By setting the CI compiler option, one can select which INCLUDE statements in the source file will be activated during compilation.

  The syntax of the INCLUDE statement is now of the form:

  ```
                      INCLUDE(member) [n]
  ```

  where, if n is a number corresponding to one in the CI option list, then that INCLUDE will be activated.

5.0 COMPILE-TIME DIAGNOSTICS FOR OPEN/CLOSE/INQUIRE

The OPEN, CLOSE, and INQUIRE statement processing has been enhanced to include compile-time syntax checking for all parameters which are expressed as CHARACTER constants in the statement and appropriate error diagnostics generated if required. Previously, all parameters, constant and variable, were checked at execution-time only.

6.0 CONTINUED EXECUTION AFTER I/O ERRORS

A change has been made in the handling of transmission errors which result in the issuing of message IFY218I. The first time such an error occurs, the message is issued and the ERR= exit is taken (if specified) or an Extended error user exit taken if one has been established. If neither of these exits have been given, the failing record will be bypassed with continued execution of the program. The IOSTAT variable (if specified) is set to the value of 218.

However, if a second error of this type occurs in the processing of the next I/O on the same unit, then if neither an ERR= exit nor a user exit has been established, the program terminates execution.

This allows users to continue processing the data in spite of intermittent I/O errors. We specifically allow one error but chose to interpret the second consecutive error to mean that the data is permanently unavailable. Note that the system has already performed its normal error recovery processing before signalling the error to the FORTRAN library.

7.0 FORMATTING DIRECT ACCESS DATASET FOR OPEN

The FORTRAN statement OPEN does not specify a parameter for the maximum number of records which may be contained in a direct access dataset (as the LANGLVL(66) statement, DEFINE FILE, does). Therefore, the preformatting for new direct access datasets had not been done for the datasets defined in the OPEN statement. Migration of users from LANGLVL(66) programs to LANGLVL(77) has been difficult because the direct access dataset must exist and be formatted before the data management READ/WRITE services can be used.

To aid this migration, the first extent of a DISP=NEW dataset will be formatted in the OS/VS and VM/CMS environments. Since the DOS/VSE user has always had the CLEAR DISK system utility to preformat the dataset, no support is required for this environment.

Only the first extent of a multi-extent dataset will be formatted. An informational message is produced at execution time indicating the size of the file that has been created.

An additional facility to aid this migration is to allow the mode of a file to be switched from SEQUENTIAL to DIRECT and vice versa when the file is not connected to a unit. This way, the program can open a file for SEQUENTIAL access, write fixed length records, close the file and reopen for DIRECT access.