NISTIR 5287

1978 Fortran Compiler Validation System User's Guide Version 2.1

Software Standards Validation Group

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Software Standards Validation Group
Building 225, Room A266
Gaithersburg, MD 20899

NIST

NISTIR 5287

1978 Fortran Compiler Validation System User's Guide Version 2.1

Software Standards Validation Group

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Software Standards Validation Group
Building 225, Room A266
Gaithersburg, MD 20899

August 1993

U.S. DEPARTMENT OF COMMERCE
Ronald H. Brown, Secretary

TECHNOLOGY ADMINISTRATION
Mary L. Good, Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
Arati Prabhakar, Director

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
Software Standards Validation Group
Building 225, Room A266
Gaithersburg, MD 20899

1978 FORTRAN COMPILER VALIDATION SYSTEM USER'S GUIDE

VERSION 2.1

August 1, 1993

```
package RELATION_TYPES_AND_DATA is
  MAX_PERSONS : constant integer := 300;
  NAME_LENGTH : constant integer := 20;
  subtype ...
```

```
procedure division.
main-paragraph.
   open input RELATIONS.
```

```
#define NULL_REL   0200
typedef struct stname
   {
    FLOAT   ELEMENT1;
    INT     ELEMENT2;
```

```
function Connected (Identa, Startpoint, Endpoint) : boolean
var
  Index  :  1..Identifier Length;
```

```
10780   let  NAME$(CURRENT)  =  rtrim$(NAME$(CURRENT))
10790   if  GENDER$(CURRENT) = MALE$ then
10800     print ...
```

```
0 51:FN U 51 F I=1:1 R X Q:$ZC S ^LIST(I)=X
```

```
      character*(BUFLEN)
      do 100 i=1,50
        call comprtn(i)
100   continue
```

1978 FORTRAN COMPILER VALIDATION SYSTEM USER'S GUIDE

VERSION 2.1

August 1, 1993

Prepared by:

U. S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
Computer Systems Laboratory
Software Standards Validation Group
Building 225, Room A266
Gaithersburg, MD 20899
(301) 975-3274

TABLE OF CONTENTS

LIST OF EXHIBITS

1. INTRODUCTION

The 1978 Fortran Compiler Validation System (FCVS78) is based on the technical specifications contained in the American National Standard Programming Language Fortran, X3.9-1978, as adopted in the Federal Information Processing Standard Publication (FIPS PUB 69-1). It is made up of audit routines, their related data and an executive routine (FEXEC) which manages the audit routines and prepares them for compilation. Each audit routine is a Fortran source program which includes many tests and supporting procedures indicating the results of each of the tests. The audit routines making up Version 2.1 of FCVS78 collectively contain features of the Subset and Full Language levels of ANSI X3.9-1978 (FIPS PUB 69-1).

1.1 Background

The Software Standards Validation Group (SSVG) is in the Information Systems Engineering Division of the Computer Systems Laboratory (CSL), National Institute of Standards and Technology (NIST).

The validation of Fortran compilers by NIST is done in support of FIPS PUB 69-1, Federal Standard Fortran. The test results for a Fortran compiler can be used by a Federal Agency to confirm that a compiler meets the specifications of FIPS PUB 69-1, insofar as the FCVS78 tests the language elements included in ANSI X3.9-1978.

1.2 Purpose and Nature of Fortran Compiler Validation

The validation of a compiler, or any piece of software, determines the degree to which that product conforms to the technical specifications on which it was based. The use of compilers that have attained a high degree of conformance with their respective language standards (technical specifications) enhances source program interchangeability within all ADP installations which use that particular programming language.

The results of running the Fortran Compiler Validation System (FCVS) does not suggest the degree to which the compiler is usable (i.e., capable of data processing applications), but the degree to which individual language elements are usable. This gives some indication of possible conversion areas which must be considered in order to implement a source program from another computer system which was written according to FIPS PUB 69-1 specifications.

Thus, the FCVS can be used to test a Fortran compiler's adherence to the standard language syntax, and, where unambiguous, language semantics of the technical specifications upon which the compiler is based. The latter, of course, is a more difficult area because of the lack of appropriate mechanisms for precise semantic specifications. The Validation System does not evaluate the implementation of a compiler nor its quantitative performance characteristics.

1.3 Validation Services Availability

Validation Services are available to the following:

. vendors wishing to have a compiler validated for their own purposes;

- vendors wishing to have a compiler validated in response to a Government request for proposal;

- Government agencies involved in a procurement;

- Government agencies wishing to validate a compiler already in use; or

- Other organizations, where the validation of a compiler benefits the Federal Government.

The results produced during an official validation are reviewed by NIST, which prepares a Validation Summary Report (VSR). The initial dissemination is to the requestor and the vendor who supports the compiler.

The VSR classifies a compiler according to each level of the Federal Standard Fortran for which support is claimed.

To request validation services contact:

Manager
Software Standards Validation Group
National Institute of Standards and Technology
Building 225, Room A266
Gaithersburg, MD 20899
(301) 975-3274 Telephone
(301) 948-6213 FAX

2. DESCRIPTION OF SYSTEM

The FCVS consists of Fortran audit routines, their related test data, and an executive routine (FEXEC) which prepares the audit routines for compilation and execution. Each audit routine consists of series of tests of Fortran language elements, and supporting procedures which indicate the result of executing these tests. Because the routines were designed to run on any computer system purporting to support Fortran, the assumptions used to write the audit tests are very restrictive. Only the simplest forms of GO TO, Arithmetic IF, WRITE, and assignment statements are used to write the support code required for each test. A description of each of the Fortran audit routines is contained in Appendix D.

A Source Programs file of audit routines with appropriate implementor-defined parameters inserted into the source code is produced by FEXEC. FEXEC and its subroutine FEXSB are Fortran programs included in source form in the FCVS Library. Once installed, FEXEC is used each time that an audit routine or series of audit routines is selected from the FCVS Library. Basic inputs to this process are the FCVS Library (a file of all of the audit routines, the FEXEC, and related test data) and a series of control inputs to select and/or update the audit routine source code.

A Fortran compiler, in a particular computer configuration/operating system environment, is tested by the compilation and execution of each audit routine. If a compiler rejects some language element by giving fatal diagnostic messages or terminating the compilation, then the FEXEC is used to eliminate the source code containing that language element. The audit routine is then recompiled and executed. Output reports (Test Results) produced by the execution of each routine indicate whether the code generated by the compiler passed or failed each test of the routine. The Test Results together with the compilation listings constitute the raw data from which NIST produces a Validation Summary Report (VSR). The VSR itemizes the areas where the Fortran compiler being tested does not conform with the 1978 Fortran Standard specifications.

2.1 Documentation

The User's Guide presents the procedures that are required in order to use the Fortran Compiler Validation System.

2.2 FCVS Library

The FCVS Library is a file on magnetic tape containing the FEXEC routine, the Fortran audit routines and the associated test data files.

2.3 Required Resources

The following must exist in the computer system in order to validate the Fortran compiler with the FCVS:

- sufficient storage to compile and execute programs of approximately 1300 source lines of Fortran code
- an input device for FEXEC control input records
- at least one magnetic tape drive for the FCVS Library

- some form of line printer (at least 120 characters for the FEXEC report and 80 characters for the audit routine reports)

- at least two magnetic tape drives or a mass storage device must be available for processing work files in the I/O statement tests.

3. FCVS LIBRARY DESCRIPTION

The FCVS Library is a file on magnetic tape containing the FEXEC, the Fortran audit routines and, the associated test data files.

3.1 Tape Format

The physical characteristics of the magnetic tape containing the FCVS Library as distributed by NIST are:

. unlabeled
. card image format - 80 characters per record
. blocking factor - 30 records per block
. code set - ASCII
. 9-track format
. 1600 BPI
. entire tape is one file; an end-of-file follows the last record on the tape

The FCVS Library may need to be unblocked or converted to another character code before it is used. Refer to section 4.1 for some things to consider when installing the FCVS Library.

3.2 FCVS Library Categories

The FCVS Library is divided into four categories and they are:

. control records
. environment files
. Fortran audit routines
. data files

3.2.1 Control Records

Control records are used for giving information to FEXEC and to separate the files on the FCVS library. The structure or the FCVS library is shown in Exhibit 1. A *HEADER control record signals the beginning of a file and a *END-OF control record signals the end of a file. Multiple *HEADER control records may precede a *END-OF control record. A *END-OF,POPFILE control record is used to indicate the end of the FCVS library. The different types of control records are:

° `*DATE***YYMMDD`	(creation date of FCVS)
° `*OWNER xxxxxx`	(id of the user of the FCVS)
° `*AUDIT vvv`	(version of the FCVS)
° `*HEADER,ENVIR,nnnnn`	(environment name)
° `*HEADER,FORTR,FMnnn`	(main audit routine name)
° `*FILES1,FORTR,FMnnn X`	(files info)
° `*HEADER,FORTR,FMnnn,SUBRTN,FMnnn`	(subroutine)
° `*HEADER,DATA*,FMnnn`	(data file indicator)
° `*END-OF,xxxxx`	(end of file marker)

3.2.2 Environment Files

The beginning portion of the FCVS library contains several environment files. These files are used to provide information to FEXEC. It uses the information for controlling, selecting, and updating the audit routines. The environment file *HEADER,ENVIR,CHAR contains the symbolic representation of the characters on the FCVS library. It may be printed using the *LIST monitor control card with the 'C' option to verify if character conversion is required.

3.2.3 Fortran Audit Routines

The Fortran audit routines are Fortran programs which are designed to test a Fortran compiler's compliance with ANS X3.9-1978. Each source image is an 80 character record. The audit routines vary in size from 200 source lines to approximately 1300 source lines.

3.2.4 Data files

Several audit routines require an external input data file which is located immediately after the program on the FCVS library. Data files vary in size from 5 to approximately 30 records.

```
*DATE***YYMMDD
*OWNER            (identity of validation to be performed with this Library)
*AUDIT vvv              (version number of audit routines)

*HEADER,TPF


*END-OF,TPF

*HEADER,ENVIR,nnnnn


*END-OF,nnnnn

*HEADER,FORTR,FMnnn
*FILES1,FORTR,FMnnn X
```

Source images of Fortran audit routine main program

```
*END-OF,FMnnn

*HEADER,FORTR,FMnnn
*FILES1,FORTR,FMnnn X
```

Source images of Fortran audit routine main program

```
*HEADER, FORTR FMnnn, SUBRTN FMnnn
```

Source images for subroutine FMnnn

```
*HEADER,DATA*,FMnnn
```

Data records for main routine FMnnn

```
*END-OF,FMnnn

*END-OF,POPFILE        "last record on the FCVS Library tape"
```

End-of-File Mark.

EXHIBIT 1: File Structure of the FCVS Library

4. VALIDATION SYSTEM IMPLEMENTATION PROCEDURES

Preparation for a Fortran validation requires consideration in three subject areas. First of all, the character set of the FCVS Library must be compatible with the computer system on which the FCVS is to be implemented. Conversion from ASCII to other character sets may be done if necessary by a system utility, a text editor, or by a program in a language available on the system. Secondly, one must determine which input/output devices are available on the system and how these devices are assigned and related to a Fortran program. Appendix B, FILE REQUIREMENTS FOR FCVS, lists default file assignments and I/O device types. The use of X-cards (see Section 5) can be used to override these defaults. The third major area of consideration during preparation is to satisfy any unique requirements of the computer system to be validated; e.g., blocking factors, how to store the executable form of the FEXEC routine, how "job control language streams" can be used to automate the selection, compilation, and execution of the audit routines, and how to handle the work files used by the audit routines for the I/O tests.

4.1 Install the FCVS Library

The FCVS Library as distributed by NIST is on magnetic tape. The tape contains 30 records per block with 80 character records. The Library can be copied onto disk or some other rapid access storage medium in order to expedite the validation process. Also the FCVS Library may need to be deblocked or converted to a character code other than ASCII. Please note that the FEXEC as included on the FCVS Library is set up to read the FCVS library as an unblocked file, thus the FCVS Library must either be deblocked or the FEXEC modified accordingly. Using an unblocked FCVS Library is the recommended approach.

4.2 Bootstrap the FEXEC Routine

The source code for the FEXEC routine must be extracted from the FCVS Library. This can be done in a number of ways; e.g., text editor, a tailored program written to extract the source code or an old version of the FEXEC from a prior Fortran validation. The FEXEC routine is identified with a *HEADER,FORTR,FEXEC record and terminated with a *END-OF,FEXEC record.

4.3 Selection of the Audit Routines

Audit routines may be selected individually or in consecutive series from the FCVS Library by the FEXEC. Section 5 describes the control inputs for selecting the audit routines. Appendix B provides a list of audit routines and their associated file unit numbers for those audit routines using input/output data files. Appendix C provides a complete list of the audit routines on the FCVS Library.

4.4 Compile and Execute the Audit Routines

The method for compiling and executing the audit routines is completely dependent on the computer system on which the FCVS is being implemented. The FEXEC routine can provide an automatic generation of Job Control Language (JCL) for most systems. A skeleton of the JCL to compile and execute the audit routines is fed as input to the FEXEC via the Monitor Section

of the FEXEC control cards file. The FEXEC in turn takes the selected audit programs and builds a complete JCL stream which may then be submitted to the system to compile and execute the selected audit routines.

4.5 Updating the Audit Routines

When necessary audit routines can be modified using FEXEC. The FEXEC controls for updating the audit routines is described in Section 5, Routine Update Control Cards. It is recommended that all changes to the audit routines be made using the FEXEC routine instead of other methods such as a system's text editor. The advantage to using FEXEC is that it provides a report and audit trail of changes made to the FCVS. Such a record of changes is required in order to evaluate the results from running the FCVS and for producing the Validation Summary Report for a compiler validation.

4.6 Compiler Evaluation

A compiler evaluation is done by analyzing the compilation listing and report results. The compiler errors are divided into two groups, syntax errors and semantic errors.

4.6.1 Syntax Errors

Syntax errors are discrepancies found in the form or format of the source code, i.e. the syntactical and lexical forms which are allowed in the Fortran language. If a compiler requires that additional code be added in order for it to accept the language elements being tested, then this is considered an error. The source code is added with the update capabilities of the FEXEC routine so that a record of all such required updates is maintained. By the same token, should a compiler reject any of the source code in a Fortran audit routine, this is also considered a syntax error and the test containing the language element which caused the error is deleted from the audit routine by using the update capabilities of the FEXEC.

4.6.2 Semantic Errors

Semantic errors are discrepancies found in a compiler implementation during execution of the audit routines. Errors of this type are shown as failures in the Test Results which are produced from execution of each of the audit routines. For this type of error the "computed" and "correct" test results are shown on the Test Results report.

5. FEXEC ROUTINE FUNCTIONS

5.1 File Requirements

The following list shows what files are required for the FEXEC routine to select and update the audit routines (see Appendix B.1 for file assignments):

- . FCVS Library - one magnetic tape drive unless the entire file is copied onto a mass storage device.
- . control card input file - often the card reader or an on-line terminal device.
- . printer file - usually the line printer.
- . source programs file - the file of selected audit routines and job control language which may be on magnetic tape, or mass storage.
- . data file - a "scratch" magnetic tape or mass storage file which is used to store and retrieve data that may be required by an audit routine. (This file may not be needed if a E-card is used with DATA**** specified.)
- . TPF merge file - a "scratch" magnetic tape or mass storage file which is used to merge Temporary Program Fixes (TPF's) with the user's updates. (TPF's are NIST approved corrections to the FCVS78 which have not been incorporated in the audit routines of the FCVS Library). This file is not needed if TPF's do not exist or the *TPF control card is used with the N option.

5.2 Control Inputs to the FEXEC Routine

FEXEC Routine functions are controlled by submitting control cards to the FEXEC. These control cards consists of two major groups. They are Monitor Control cards which are used to direct the FEXEC in its operation and Update Control cards which are used to modify the source programs contained on the FCVS Library.

The FEXEC Control Cards are used for the selection of audit routines, updating of audit routines and merging of JCL to produce an output Job Stream. This Job Stream is called the Source Program File. Monitor control cards precede the Update control cards. The order of the Monitor control cards is completely up to the user, however, the Update control cards must be in sequential order, by program number and by line sequence number within program.

A typical control stream to select one program, update the program, insert appropriate JCL for a Univac 1108, and have all other options "defaulted" is shown in Exhibit 2.

5.2.1 Monitor Control Section

Monitor Control Cards consists of the following types of card images:

- Identification Control Cards (*DATE, *COMPILER, *PROJECT)
- List Control Card (*LIST)
- Option Card (*OPT1)
- Temporary Program Fixes Control Card (*TPF)
- Implementor-Unique Alphabet-Cards (X-Cards)
- JCL Generation Alphabet-Cards (I-Cards, B-Cards, E-Cards, T-Cards, D-Cards)
- Program Selection Cards (Plus-Cards, Minus Cards)
- Environment Control Card (*ENVIR)
- END-MONITOR Control Card

FEXEC Control Card	Action Desired (not part of the card)
`X-020  I02=6`	The number 6 is "assigned" to I02 so all WRITE (I02, nnn)... statements will address logical unit number 6

`I-01  @RUN 82UCLG,99S0050D,99S0030`
`B-01  @FOR,IS  TPF$.A,TPF$.B`
`E-01  @MAP,I  TPF$.C,TPF$.D`
`E-02  IN TPF$.`
`E-03  @XQT  TPF$.D`
`E-04  @ERS  TPF$.`
`T-01  @FIN`
	Job Control Language to be included in the SOURCE PROGRAMS file

`PFM001`	Select program FM001
`*END-MONITOR`	End of the MONITOR Control Cards
`*BEGIN-UPDATE`	Beginning of the UPDATE Control Cards
`*START,FM001`	Updates to program FM001
`=00155`	Add the following source lines after line 00155
`C  INSERTED LINES CAN BE FIRST, LAST, OR` `C  CAN GO BETWEEN EXISTING LINES OF A PROGRAM...`	
`=00500C00900`	Change lines 500 thru 900 into Comments
`=01700,01800`	Delete lines 01700 thru 01800
`IVON01 = IVON01  - 1`	Replaces deleted lines 01700 and 01800
`=02000,02000`	Delete line 02000
`*END-UPDATE`	End of UPDATE Control Cards
`*END-INPUT`	End input control stream

EXHIBIT 2: FEXEC Control Card Examples

5.2.1.1 Identification Control Cards (*DATE, *COMPILER, and *PROJECT)

The identification control cards (*DATE, *COMPILER, *PROJECT) can be used to override or supply information about the environment to FEXEC. This information is listed in the audit routine report heading. The format of the identification control cards are:

a. *DATE (date of validation - 1 to 17 characters beginning in column 8)

b. *COMPILER (compiler ID - 1 to 20 characters beginning in column 12)

c. *PROJECT (project code - 1 to 13 characters beginning in column 11)

5.2.1.2 List Control Card (*LIST)

The LIST control card (*LIST) determines the amount and what is printed on the execution report of FEXEC. The format is:

. *LIST option-1,option-2, ...

The allowable options are:

. U - Print all updates, replacements, and statistics.

. P - Print the selected source programs after updates and replacements have been made and print the statistics.

. S - Print statistics (number of inputs, outputs, insertions, and deletions) for each selected program (Default).

. T - Print Temporary Program Fixes (TPF) that are on the FCVS library, if any.

. E - Print each selection record retrieved from an environmental entry as it is processed (see *ENVIR).

. C - Print environmental entry *HEADER,ENVIR,CHAR to verify that the symbolic character representations on the FCVS library matches that of the system it's being implemented on.

The P, U, and S operands are mutually exclusive. There can be more than one *LIST card present in the Monitor Control Inputs section and each *LIST card can contain multiple entries.

5.2.1.3 Option Card (*OPT1)

For those audit routines which use data files, source code is included for printing (dumping) the contents of the data files. This source code is coded as comments in the audit routines but can be changed to executable code. The Option card (*OPT1) specifies whether or not the optional file dump code in the audit routines should be executed or left as comments. The format of the *OPT1 control card is:

. *OPT1 D

To leave the file dump code as comments, do NOT specify an *OPT1 card. To execute the dump code, use the *OPT1 card.

5.2.1.4 TPF Control Card (*TPF)

The TPF control card (*TPF) specifies whether or not the Temporary Program Fixes (TPF's), if any, on the FCVS Library are to be incorporated into the programs (audit routines or FEXEC itself) on the FCVS Library. The default is that they will be incorporated unless specified otherwise by the TPF control card. When the TPF's are selected they are merged with any user's updates supplied through the Update Control section of the FEXEC control card file.

TPF's are updates for the FCVS Library which have been issued by NIST after release of the current FCVS78 version. These TPF's are stored as Environment entries (following the *HEADER,ENVIR,TPF record) on the front of the FCVS Library and are applied to the FCVS programs (if selected) by FEXEC as the programs are read from the FCVS Library. The *TPF options are:

. *TPF Y (include all TPFs)

. *TPF N (exclude TPFs) - default if no *TPF control card is specified)

5.2.1.5 Program Selection Cards (P and M Cards)

a. Plus-Cards

Plus-Cards are used to select one or a sequential series of programs, subprograms, and data files from the FCVS Library and output them to the Source Programs file. The two forms of the Plus-Card are:

. PFMnnn (Select program FMnnn)
. PFMnnn,PFMnnn (Select a series)

Some examples are:

. PFM001 (Select only FM001)
. PFM100,PFM108 (Select FM100 thru FM108)

Selection of any main program will automatically include any and all subroutines, subprograms and data files associated with it.

b. Minus-Card

A Minus-card causes a program set which was previously selected by an earlier control card to be deleted from the selection list. The two forms of the Minus-Card are:

. MFMnnn (exclude program FMnnn)
. MFMnnn,MFMnnn (exclude a series)

This command logically removes the program name from the list of programs to be selected. If the program name being removed is a main program, any associated subroutines or program data will also not be selected. A Minus-card will not suppress the selection of a subroutine or subprogram if the main program is requested to be selected.

One use of the Minus-card is if a user wanted to select the full level Fortran test set (FM001 thru FM923), but only had storage space for programs FM500 through FM923. The control card sequence would be:

. *ENVIR F (select full level)
. MFM001,MFM499 (exclude FM001 - FM499)

After programs FM500 through FM923 were compiled and executed, the user could then extract programs FM001 through FM499 using the "*ENVIR S" control card.

5.2.1.6 Environment Control Card (*ENVIR)

The Environment control card (*ENVIR) is used to simulate the inclusion of a group of Plus-cards in the Monitor Control section. The *ENVIR options are:

. *ENVIR F (select both the full and subset level Fortran audit routines)

. *ENVIR S (select only the subset level Fortran audit routines)

The FCVS Library contains 272 audit routines which are divided into two levels. Appendix C explains the differences between the two levels and lists by name the audit routines included in each level.

Multiple *ENVIR control cards are not permitted.

5.2.1.7 Implementor-Unique Alphabet-Cards (X-Cards)

Implementor-unique alphabet-cards (X-cards) allow replacement of the default logical unit numbers assigned to the input/output files used by the FCVS programs. The format of the X-Card is:

. X-kk0 (Source code column 8 thru 80)
. X-kk1 (Additional source code if required)

where:

. kk = a unique two digit number which corresponds to the "CXkk0" or "CXkk1" card images coded in the FCVS programs

The source statement specified on the X-Card must be a syntactically correct Fortran statement. The X-kk0 X-Card is for assigning a logical unit number to an integer variable. (The variable name used in the assignment statement must be the letter "I" followed by the number kk.) The X-kk1 X-Card is an additional (optional) card, having the same kk number as X-kk0, which may be used to include additional source statements (such as an OPEN statement) associated with the X-kk0 card. There are provisions for up to twenty, two-card sets of X-cards that can be replaced

in the source program. The X-cards may not be needed if the operating system can use the default logical unit numbers coded in the audit routines. Appendix B.2 provides a complete list of the default unit numbers used.

In Exhibit 4, which shows how the X-card replacement is used, the following action takes place:

- When the output Source Programs File is written by the FEXEC, columns 8-73 of the X-kk0 and X-kk1 cards in the Monitor Control Inputs become columns 7-72 in the Cxkk0 and Cxkk1 card images in the source program respectively.

- The first six columns are "blanked" in the output source line when an X-Card replacement takes place.

X-cards X-190, X-191, X-200 and X-201 are special function replacement cards and are used for changing the size of the file names used by the FILE specifier. X-Cards X-190 and X-200 are used to replace the assignment statement which assigns the file name to a variable; X-Cards X-191 and X-201 are used to replace the CHARACTER statement which sets the size of the variable. The variable CDIR is a fixed variable name for Direct access files and is used to store file name "CDIR" which is also a fixed name; the variable CSEQ is a fixed variable name for Sequential access files and is used to store the file-name "CSEQ" which is also a fixed name. These variables are used in the FILE specifiers for the OPEN and INQUIRE statements.

The two X-cards, X-190 and X-191, are used to replace the following two statements respectively:

```
. CHARACTER*15 CSEQ
. CSEQ =  '           CSEQ'
```

The two X-cards, X-200 and X-201, are used to replace the following two statements respectively:

```
. CHARACTER*15 CDIR
. CDIR =  '           CDIR'
```

FILE specifier names, CSEQ and CDIR, must not be changed. Only the size of the character string for the name should be changed.

Program before X-card replacement

```
      I01 = 5
CX010 THIS CARD IS REPLACED IF X-010 CARD IS USED
CX011 THIS CARD IS REPLACED IF X-011 CARD IS USED

      READ (I01,77501) ILIST
```

X-card replacements placed in Monitor Control section

```
X-010 I01 10
X-011 CALL OPEN (I01)
*END-MONITOR
*END-INPUT
```

Program after X-card replacement by FEXEC

```
I01 = 5
I01 = 10            (replaces CX010)
CALL OPEN (I01)     (replaces CX011)

READ  (I01,77501) ILIST
```

EXHIBIT 3: X-Card Replacement

5.2.1.8 Job Control Language Generation Cards
(I, B, E, T, and D Cards)

The Job Control Language Generation Cards (alphabet-cards) are used to describe a skeleton of the Job Control Language (JCL) necessary to compile and execute the audit routines. An example of generated JCL from the alphabet-cards (I-Cards, B-Cards, E-Cards, T-Cards) to compile and execute programs FM700 through FM703 is shown in Exhibit 4.

a. Initial and Terminal Alphabet-Cards (I and T Cards)

Initial alphabet-cards (I-cards) are generated and placed in the Source Program File prior to any programs being selected. After all the requested programs have been selected the terminal alphabet-card (T-cards) are generated and placed in the Source Program file. The format of the I and T alphabet-cards are:

. I-kk (Job Control Language statement columns 8 thru 80)

. T-kk (Job Control Language statement columns 8 thru 80)

where:

. kk = a unique two-digit number for the respective I or T card. T-cards may range from 01 thru 05. I-cards have no limit.

b. Beginning and Ending Alphabet-Cards (B & E Cards)

Beginning alphabet-cards (B-cards) are generated and placed before each selected program which is written to the Source Program File. After each selected program Ending alphabet-cards (E-cards) are generated and placed in the Source Program File. A program name substitution option is available with the B and E cards for placing a unique name on the generated JCL. This option can be used for making each job unique, or for uniquely naming files associated with the source and object programs. In addition there is an indicator column which is used to control when a particular B or E card is to generate a JCL record. The format of the B and E alphabet cards are:

. B-kkmmy (Job Control Language statement columns 8 thru 80)

. E-kkmmy (Job Control Language statement columns 8 thru 80)

. E-kk DATA****

where:

. kk = a unique two-digit number for the respective B or E card. A maximum of 10 B and E cards is permitted.

. mm = optional two-digit number ranging between 01 and 69, which is the starting positioning of the 5-character program name to be substituted into the generated JCL card.

. y = optional control character (J or T that causes certain JCL to be generated at specific times. If the value for y is blank then the control card is generated before

and after each main source program selected but not for subroutines. If the value for y is the letter "J" then the control card is generated before the first selected program or after the last selected program.

If the value for y is the letter "T" then the control card is generated only before/ after a subroutine program.

. DATA**** = Special E-card which triggers the generation of the D-cards in the respective location among the E-cards.

When the JCL is generated as shown in Exhibit 4, the character in column 8 of the input monitor control card to FEXEC becomes the character in column 1 of the generated JCL instruction. Column 80 of the input monitor control card becomes column 73 of the generated card which is written to the Source Program File. When name substitution is elected for a B-card or E-card, remember that the number in columns 5-6 represents the first column in the generated instruction where substitution is to take place. Hence, if a B-card contains a JCL image (beginning in card column 8) and the field where name substitution is to take place shows on the card as column 23, the substitution position is actually column 16 (or 23 minus 7) and so "16" must be placed in columns 5-6 of the B-card.

c. Data Alphabet-Card (D-card)

The Data alphabet-cards (D-cards) are used to place JCL before and after a external data file. Audit routines FM110, FM111, FM403, FM404, FM900, FM901, FM903, FM906, AND FM923 all require external input data. The external input data file needed is located after the audit routine on the FCVS library. It's identified with a *HEADER,DATA*,FMnnn record, where FMnnn is the audit routine name, and precedes the *END-OF,FMnnn record for the audit routine.

During FEXEC execution, selection of an audit routine causes any associated data file to be extracted along with the audit routine. A D-01 monitor control card if specified will generate a JCL record which will be placed before each external input data file. A D-02 monitor control card if specified will generate a JCL record which will be placed after each external input file. The external input data files may be written to a scratch work file and each file will be separated with an end of file record. They may also be written to the Source Program file, in-line with the selected audit routines and the other generated alphabet-cards. To specify the in-line location for the placement of the data file an E-card is used with the special indicator *DATA*****. So during E-card generation if the special indicator is found the E-card is suppressed and the data file is put in it's place. A program name substitution option is also available for D-cards to place a unique name on the generated JCL. This option can be used for making each data file unique, or for any purpose a user may require. The format of the D-card is:

. D-kkmm (Job Control language statement column 8 thru 80)

where:

. kk = 01 or 02

. mm optional two-digit number ranging between 01 and 69, which is the starting positioning of the 5-character program name to be substituted into the generated JCL card.

An example of using D-cards is :

```
.   D-01  //SYSIN  DD * (optional)
.   E-nn  DATA****     (nn = two-digit no.)
.   D-02             (optional)
```

The D-01 card, if present, is released in front of the data, and the D-02 card, if present, is released behind the data.

If an E-card with the special indicator "DATA***" is not used, FEXEC will write the data to the scratch file assigned to I11 (see Appendix B.1).

5.2.1.9 End-Monitor Control Card (*END-MONITOR)

The End-Monitor control card terminates the processing of the Monitor Control Inputs section. The format is:

```
.   *END-MONITOR
```

Programs to be selected by FEXEC

FM700 - a stand-alone program
FM701 - the first program in a sequence set of three
FM702 - a "main" program which receives a data file form FM701
FM703 - a subroutine called from FM702

The FCVS Library *HEADER images

```
 *HEADER,FORTR,FM700
   *HEADER,FORTR,FM701
   *HEADER,FORTR,FM701,SUBPRG,FM702
TPF$.A,TPF$.XXXXX
   *HEADER,FORTR,FM701,SUBRTN,FM703
TPF$.B,TPF$.XXXXX
```

Alphabet-Cards

```
I-01   @RUN 82UCLG,99S0030D,99S0050
B-0120J@MSG,N THIS BEGINS XXXXX
     B - 0 2 2 1    @ F O R , I S
     B - 0 3 2 1 T @ F O R , I S
E-01   @MAP,I  TPF$.C,TPF$.ABS
E-02      IN  TPF$.
E-03   @XQT   TPF$.ABS
E-04   @ERS   TPF$.
E-0519J@MSG,N  THIS ENDS XXXXX
T-01   @FIN
```

The resultant generated Source Programs File

(continuation of column on the left)

```
@RUN  82UCLG,99S0030D,99S0050
@MSG,N  THIS BEGINS FM700
@FOR,IS  TPF$.A,TPF$.FM700
  source images for FM700
@MAP,I  TPF$.C,TPF$.ABS
 IN  TPF$.
@XQT   TPF$.ABS
@ERS   TPF$.
@MSG,N  THIS ENDS FM700
@MSG,N  THIS BEGINS FM701
@FOR,IS  TPF$.A,TPF$.FM701
 source images for FM701
```

```
@FOR,IS  TPF$.B,TPF$.FM702
   source images for FM702
@FOR,IS  TPF$.B,TPF$.FM703
   source images for FM703
@MAP,I   TPF$.C,TPF$.ABS
  IN  TPF$.
@XQT  TPF$.ABS
@ERS  TPF$.
@MSG,N  THIS ENDS FM701
@FIN
```

EXHIBIT 4: Example of JCL Generated From Alphabet-Cards

5.2.2 Update Control Section

The Update Control Section consists of the following types of card images:

. *BEGIN-UPDATE Control Card
. *START Control Card
. Update Control Cards

    . Replacement
    . Addition
    . Delete a card image(s)
    . Change to a Comment Line
    . Delete a test(s)

. *END-UPDATE Control Card
. *END-INPUT Control Card

5.2.2.1 Begin-Update Control Card (*BEGIN-UPDATE)

The Begin-Update Control card signals the beginning of the update section and follows the *END-MONITOR control card. This control card is not required if there are no programs to be modified. The format is:

```
. *BEGIN-UPDATE
```

5.2.2.2 Start Control Card (*START)

Any audit routine that is to be updated by the FEXEC routine must be selected and have a *START control card followed by the appropriate update cards. If more than one routine is to be updated, then the updates must be submitted in the same order as the programs appear in the FCVS Library. The reason for this is that the routines are processed sequentially. The format is:

. *START,FMnnn (FMnnn audit routine name)

5.2.2.3 Routine Update Control Cards

Update control cards follow the *START card for the routine that is to be updated. Source images can be added, replaced, deleted, or changed to comment lines. Also, whole tests may be deleted with one control card.

All update control cards begin with an equal sign "=" in column 1 followed by a five-digit number in columns 2 thru 6, an update type designator in column 7 and an optional five-digit number in columns 8 thru 12. The five-digit number may be either a line number or a test number depending on the update type designator specified.

The allowable update type designators are 'C', ',' or 'T'. A 'C' in column 7 changes the specified source line(s) to comments; a ',' deletes the specified source line(s); and a 'T' causes the specified test number(s) to be deleted (i.e., code for the test changed to comments and source code inserted which causes the test to be noted as 'DELETED' on the test results report).

Program source code which follows an update control card must be in the standard source statement format i.e., label beginning in column 1, continuation in column 6 and Fortran statement in columns 7 through 72. When using the update control cards, refer to the sequence numbers in columns 73-77 of the source line for specifying the source lines which are to be updated.

When deleting a test ('T' in column 7) or a series of tests, it is suggested that the test deletion update cards be entered as the first '=' update control card(s) in the set for the audit routine to be updated. They may also be placed in-line with the other update control cards, but caution must be used. When test number update cards are intermixed with line number update cards the source line numbers of these two update types must not overlap otherwise the updates may not be properly applied.

The delete test option should be used to delete test code that is not supported and will cause the compiler to produce fatal compile errors or that will not permit the program to execute to a normal end. Currently the delete test option is only supported for those programs that contain a CTnnn* label at the beginning of each test. The following paragraphs describe the different types of updating permitted with FEXEC.

a. Addition of Source Images

To add card images before the first source line of the program, put the card images immediately after the *START card and before any update control cards.

b. Insert a card image

To insert card images the following sequence would be used:

```
. =nnnnn
  (card images to be inserted)
```

The card images would be inserted after the text for source line nnnnn.

c. Replace a card image

To replace a card image the following sequence would be used:

```
. =nnnnn,nnnnn
  (new card image to replaces old card)
```

If the sequence numbers match, the card image is replaced. (Should there not be a card numbered nnnnn on the FCVS Library for that source program selected, the card will be inserted).

d. Replace a series of card images

To replace a series of card images the following sequence would be used:

```
. =nnnnn,mmmmm
  (replacement card images)
```

Card images nnnnn through mmmmm will be replaced with the new card images specified after the update control card.

e. Delete a card image

To delete a card image the following update control card would be used:

. =nnnnn,nnnnn

Source line nnnnn is deleted. If there is no corresponding card, an error message is issued.

f. Delete a series of card images

To delete a series of card images the following update control card would be used:

. =nnnnn,mmmmm

The card images from nnnnn through and including mmmmm are deleted.

g. Changing a single line to a comment

To change a source line to comments the following control card would be used:

. =nnnnnC

Source line number nnnnn is changed to a comment line (C is placed in column 1).

h. Changing a series of lines to comments

To change a series of source lines to comments the following update control card would be used:

. =nnnnnCmmmmm

The card images nnnnn though and including mmmmm will have a C inserted in column 1.

i. Delete a Test

To delete a test the following control card would be used:

. =iiiiiT

Test number iiiii is deleted. This will cause all source lines between Test 1 and Test 2 to be changed to comments and test delete code to be inserted at the beginning of the test.

j. Delete a Test series

To delete a series of consecutive tests the following update control card would be used:

. =iiiiiTjjjjj

Test numbers iiiii through and including test number jjjjj are deleted.

Exhibit 5 shows some examples of the different types of update control cards and their functions.

5.2.2.4 End-Update Control Card (*END-UPDATE)

The End-Update control card is required if any updates were supplied and signals the end of the update section. The format is:

```
.       *END-UPDATE
```

5.2.2.5 End-Input Control Card (*END-INPUT)

The End-Input control card indicates the end of all FEXEC Control Card inputs. It is the last card in the input deck and is required even if no update cards are supplied. The format is:

```
.       *END-INPUT
```

Failure to include the *END-INPUT card will cause FEXEC to terminate with an I/O error.

5.3 Outputs from the FEXEC Routine

5.3.1 Source Programs File

The source programs that are output by the FEXEC routine are sequentially written onto the logical unit specified as I04 in the source code of the FEXEC. The programs will be separated by whatever job control language was specified for generation through use of I-cards, B-cards, E-cards, L-cards and T-cards. A file mark terminates the file. The FCVS user must decide which of the devices available on his system would best be used as the Source Programs file. This file must then be input to the operating system, usually via a batch input procedure.

5.3.2 Data Files

Data files are read from the FCVS Library for routines which have card input and are written in a separate file (not the Source Programs file) when the special indicator "DATA****" is not specified. If the special indicator is used, the data follows in-line with the source program and is written to the Source Programs file.

The file used for data files when the special indicator is not specified is logical unit I11 used by the FEXEC.

5.3.3 Printer File

Depending upon the *LIST option chosen in the Monitor Control Inputs to the FEXEC routine, the printer listing will vary in size and content.

```
*BEGIN-UPDATE   Begin Update section input
*START,FMnnn    Begin updates for program FMnnn
C       FSTC    Add new text at beginning of program
=01000          Insert text after line 01000
        I01 =  5        Text to be inserted
        I02 =  6
=01100,01100    Delete a card image
=01150,01150    Replace a card image
        GO TO 25        New card image to replace line 01150
=00002T         Test 2 is deleted (source code changed to comments) and delete code
                inserted
=00004T00006    Tests 4 through 6 are deleted (source code changed to comments) and
                delete code inserted for test 4 through 6
=01180,01190    Replace a series of card images
    CONTINUE    New card images to replace lines 01180 and 01190
        IVPASS=IVPASS+1  and 01190
=01200C         Change line to comment
=01400C01600    Change series of lines to comments
=19000,19200    Delete a series of lines
*START,FMnnn
 .
 .
*END-UPDATE
*END-INPUT
```

EXHIBIT 5: Update Control Card Examples

APPENDIX A
ERROR MESSAGES FROM THE EXECUTIVE ROUTINE

A. ERROR MESSAGES FROM THE EXECUTIVE ROUTINE

The following description of messages and actions taken documents possible error situations with regard to the FEXEC Routine. In every case, the message is shown as an error number below the control record or input card which caused the error situation.

A.1 Error Messages and Action Taken

No.	Explanation	Action Taken
1	UNRECOGNIZABLE MONITOR CONTROL CARD	Card is skipped
2	ERROR IN OPT1 CARD	Card is skipped
3	ERROR IN PFMnnn CONTROL CARD	Card is skipped
4	ERROR IN X-nn CONTROL CARD	Card is skipped
5	UNRECOGNIZABLE * RECORD ON SOURCE FILE	Skip until next valid * record is read
6	ERROR IN *HEADER RECORD - SOURCE FILE	Skip until next valid * record is read
7	ERROR IN *START CONTROL CARD	Update cards skipped until next * card read
8	ERROR IN UPDATE CARD - CAN'T RECOGNIZE	Card is skipped
9	UNRECOGNIZABLE INPUT CARD	Card is skipped
10	ERROR IN PROGRAM NAME IN *HEADER ON SOURCE FILE	Program is skipped
11	ERROR IN *LIST CARD	Card is skipped
12	ERROR IN CX RECORD ON SOURCE FILE	Record is skipped
13	ERROR IN *END-OF,FMnnn RECORD ON SOURCE FILE	Record is skipped
14	ERROR ON UPDATE CONTROL CARD	Skip to next * card
15	ERROR IN UPDATE SEQUENCE	Skip to next correct sequence card
16	ERROR IN B-CARD NUMBER OR NUMBER EXCEEDS TABLE SIZE	Card is skipped

17	ERROR IN E-CARD NUMBER OR NUMBER EXCEEDS TABLE SIZE	Card is skipped
18	ERROR IN I-CARD NUMBER OR NUMBER EXCEEDS TABLE SIZE	Card is skipped
19	ERROR IN T-CARD NUMBER OR NUMBER EXCEEDS TABLE SIZE	Card is skipped
20	ERROR IN D-CARD NUMBER OR NUMBER EXCEEDS TABLE SIZE	Card is skipped
21	ERROR IN NAME SUBSTITUTION POSITION DESIGNATOR (COLUMNS 5-6)	Name substitution is skipped
22	ERROR IN NAME SUBSTITUTION POSITION DESIGNATOR (COLUMNS 5-6) OF E-CARD	Name substitution is skipped
23	ONLY ONE CARD OF THIS TYPE MAY BE SPECIFIED	Card is skipped
24	MONITOR CARD SPECIFIED WITHOUT OPTION	Card is skipped
25	INVALID OPTION SPECIFIED	Card is skipped
26	INVALID PROGRAM NUMBER	Card is not processed

APPENDIX B
FILE REQUIREMENTS FOR FCVS

B. FILE REQUIREMENTS FOR FCVS

Appendix B lists the X-cards, default logical unit numbers (LUN), associated routines and the use of the files in the Fortran Compiler Validation System.

B.1 Logical Unit Chart for FEXEC

X-card	Default LUN	Routine	Use
X-010	I01 = 5	FEXEC	Input control file
X-020	I02 = 6		Output print file
X-040	I04 = 7		Input FCVS Library file
X-100	I10 = 8		Output Source program file
X-110	I11 = 9		Output optional data file
X-120	I12 = 10		I/O work file for TPF's

B.2 Logical Unit Chart for Audit Routines

X-card	Default LUN	Routine	Use
X-0l0	I01 = 5	FM110	Input data file (access sequential)
		FM111	
		FM403	
		FM404	
		FM900	
		FM901	
		FM906	
		FM903	
		FM923	
X-020	I02 = 6	*****	Output print file for all audit routines
X-040	I04 = 8	FM411	I/O work file (sequential, unformatted)
X-050	I05 = 14	FM915	I/O work file (sequential, unformatted)
		FM920	
X-060	I06 = 7	FM100	I/O work file (sequential, formatted)
		FM104	
		FM107	
X-070	I07 = 7	FM101	I/O work file (sequential, formatted)
X-080	I08 = 7	FM102	I/O work file (sequential, formatted)
		FM105	
		FM108	
		FM401	
	I08 = 14	FM914	
X-090	I09 = 7	FM103	I/O work file (sequential, formatted)
		FM106	
		FM402	
	I09 = 14	FM919	
X-100	I10 = 24	FM407	I/O work file (direct, unformatted)

	I10 = 9	FM413	
	I10 = 24	FM910	
		FM921	
X-110	I11 = 25	FM910	I/O work file (direct, unformatted)
X-120	I12 = 14	FM917	I/O work file (direct, unformatted)
X-130	I13 = 24	FM912	I/O work file (direct, formatted, record length 80, record count 142)
X-140	I14 = 14	FM916	I/O work file (direct, formatted)
X-150	I15 = 14	FM922	I/O work file (sequential, formatted)
X-190		FM919	Replacement card for statement
		FM920	CHARACTER*15 CSEQ
		FM922	
X-191		FM919	Replacement card for statement
		FM920	CSEQ = ' CSEQ'
		FM922	
X-200		FM910	Replacement card for statement
		FM912	CHARACTER*15 CDIR
		FM921	
X-201		FM910	Replacement card for statement
		FM912	CDIR = ' CDIR'
		FM921	

APPENDIX C
LIST OF FORTRAN AUDIT ROUTINES

C. LIST OF FORTRAN AUDIT ROUTINES

The Fortran Compiler Validation System (FCVS) consists of 272 audit routines used to test a compiler with regard to ANS X3.9-1978. The audit routines are divided into two levels. Subset Level Fortran audit routines is used to test the subset language as specified in ANS X3.9-1978 and described on the lefthand pages of the standard. Full Level Fortran audit routines, which includes the Subset Level, is used to test the full language as specified in ANS X3.9-1978 and described on the righthand pages of the standard.

C.1 Subset Level Fortran

The Subset Level Fortran test suite consists of 123 main routines and 44 related subroutines or function subprograms.

Main Program	Related Subprograms or Subroutines
FM001 thru FM014	
FM016 thru FM025	
FM026	FM027
FM028	FM029
FM030 thru FM045	
FM050	FM051, FM052, FM053, FM054, FM055
FM056	FM057, FM058, FM059
FM060 thru FM062	
FM080	FM081, FM082, FM083
FM097 thru FM111	
FM200 thru FM205	
FM251 thru FM260	
FM261	FM262, FM263, FM264
FM300 thru FM301	
FM302	FM303, FM304, FM305
FM306 thru FM307	

FM308	FM309, FM310
FM311	FM312, FM313, FM314, FM315, FM316
FM317	FM318, FM319, FM320, FM321, FM322, FM323, FM324, FM325, FM326, FM327
FM328	FM329, FM330, FM331, FM332, FM333, FM334, FM335
FM35l thru FM357	
FM359 thru FM364	
FM368 thru FM379	
FM401 thru FM406	
FM407	FM408
FM411	
FM413	

C.2 Full Level Fortran

The Full Level Fortran test suite includes the Subset Level Fortran test suite and the following 70 main routines and 35 related subroutines or function subprograms.

Main Program	Related Subprograms or Subroutines
FM500	FM501, FM502
FM503	FM504, FM505
FM506	FM507, FM508
FM509	FM510, FM511, FM512, FM513
FM514	FM515, FM516
FM517	FM518, FM519
FM520	
FM700	
FM701	FM702, FM703, FM704, FM705, FM706, FM707, FM708, FM709
FM710	

FM711	FM712, FM713, FM714
FM715	FM716, FM717
FM718	
FM719	FM720, FM721
FM722	FM723, FM724, FM725
FM800 thru FM834	
FM900 thru FM901	
FM903	FM904
FM905 thru FM909	
FM910	FM911
FM912	FM913
FM914 thru FM923	

APPENDIX D
FCVS PROGRAM INFORMATION

D. FCVS PROGRAM INFORMATION

D.1 Subset Language Programs Part 1

D.1.1 FM001 (Subset)

a. Features Tested

This routine contains the procedures which are used throughout the Fortran Compiler Validation System. The output report headings for the elementary routines are printed, followed by three tests which contain the source lines for the pass, fail and delete procedures for testing language features. The run summary lines are printed at the end of the routine.

If this routine does not compile and execute correctly, then no other routines are run. There is no use in trying to validate a Fortran compiler which cannot handle such basic statements.

ANS X3.9-1978 References: 3.2.1, 3.4, 3.6, 5.1.1.1, 10.1, 11.1, 11.4, 11.11, 12.9.5.2, 12.9.5.2.3, 13.5.3.2, 13.5.9.1, 13.5.11

b. Special Considerations

There are 3 'PASS/FAIL' tests. The report should show that Test 1 PASSed, Test 2 FAILed and Test 3 DELETED.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.2 FM002 (Subset)

a. Features Tested

This routine tests comment lines which contain valid Fortran statements. Comment lines should not affect the execution of the program in any way.

ANS X3.9-1978 References: 3.2.1

b. Special Considerations

There are 9 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.3 FM003 (Subset)

a. Features Tested

This routine contains the basic CONTINUE tests. These tests insure that execution of a CONTINUE statement causes continuation of the normal program execution sequence. Only the statements in the basic assumptions are included in these tests. CONTINUE tests are contained in later routines as part of the tests for other language features such as the DO statement tests.

ANS X3.9-1978 Reference: 3.6, 11.11

b. Special Considerations

There are 8 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.4 FM004 (Subset)

a. Features Tested

This program tests the basic arithmetic IF statement and the basic unconditional GO TO statement.

ANS X3.9-1978 Reference: 3.6, 11.1, 11.4

b. Special Considerations

There are 12 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.5 FM005 (Subset)

a. Features Tested

This routine tests the basic assumptions regarding the simple formatted WRITE statement of the form:

WRITE (U,F) or WRITE (U,F) L

where U is a logical unit number, F is a format statement label, and L is a list of integer variables.

The format statement contains nH Hollerith field descriptors, nX blank field descriptors, and Iw numeric field descriptors.

This routine also tests whether the first character of a format record for printer output determines vertical spacing as follows:

blank - one line, and 1 - advance to first line of next page.

ANS X3.9-1978 Reference: 12.8.2, 12.9.5.2, 12.9.5.2.3, 13.5.2, 13.5.9.1

b. Special Considerations

There are 17 visual tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.6 FM006 (Subset)

a. Features Tested

This routine tests arithmetic assignment statements of the form:

integer variable = integer constant
integer variable = integer variable

An integer constant is written as a nonempty string of digits. The constant is the digit string interpreted as a decimal number. The integer constant may be unsigned, positive or negative.

An integer datum is always an exact representation of an integer value. It may assume positive, negative and zero values. It may only assume integral values.

This routine also contains tests which check on the use of at least 16 bits for representing integer data values. The constant values 32767 and -32766 are used in these tests.

ANS X3.9-1978 Reference: 4.3, 4.3.1, 10.1

b. Special Considerations

There are 30 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.7 FM007 (Subset)

a. Features Tested

This routine tests the use of DATA initialization statements. DATA initialization statements are used to define initial values for integer variables. The DATA statements contain unsigned, positive signed, and negative signed integer constants. The last DATA statement in the routine contains the form

J * integer constant

which indicates the constant is to be specified J times.

ANS X3.9-1978 Reference: 4.3, 4.3.1, 9

b. Special Considerations

There are 20 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.8 FM008 (Subset)

a. Features Tested

This routine tests arithmetic assignment statements of the form:

integer variable = arithmetic expression

where arithmetic expression is formed with the arithmetic operator +, integer constants and positive integer variables. Some of the tests use parentheses to group elements in an arithmetic expression.

ANS X3.9-1978 Reference: 4.3, 4.3.1, 6.1, 10.1

b. Special Considerations

There are 35 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.9 FM009 (Subset)

a. Features Tested

This routine tests arithmetic assignment statements of the form:

integer variable = arithmetic expression

where the arithmetic expression is formed with the arithmetic operator +, integer constants and positive integer variables. Some of the tests use parentheses to group elements in the arithmetic expression

ANS X3.9-1978 REFERENCE: 4.3, 4.3.1, 6.1, 10.1

b. Special Considerations

There are 30 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.10 FM010 (Subset)

a. Features Tested

This routine tests reference format of Fortran statements and statement numbers. The use of the BLANK character is tested both within the statement number field and within the Fortran statements themselves. Leading zero is tested for statements and integer constants. Variable names which look very much like Fortran reserved words are tested in arithmetic assignment statements. Naming conventions used throughout the FCVS are tested also in arithmetic assignment statements.

ANS X3.9-1978 Reference: 2.5, 3.1.6, 3.2.2, 3.4

b. Special Considerations

There are 3 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.11 FM011 (Subset)

a. Features Tested

This routine is a test of BLANK characters (section 3.1.6) which should have no meaning when embedded in Fortran reserved words.

ANS X3.9-1978 Reference: 3.1.6

b. Special Considerations

There are 7 'PASS/FAIL tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.12 FM012 (Subset)

a. Features Tested

This routine tests the Fortran DO statement from its simplest form to the more abbreviated forms. Various increments are used and branching by various methods is tested for passing control out of the DO range and returning (extended range). Nested DO statements using various terminating statements are also tested by this routine.

ANS X3.9-1978 Reference: 11.10, 11.10.3, 11.11

b. Special Considerations

There are 15 'PASS/FAIL' tests. The report should show that Test 123 is omitted.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.13 FM013 (Subset)

a. Features Tested

This routine tests the Fortran assigned GO TO statement as described in Section 10.3 and Section 11.3. First a statement label is assigned to an integer variable in the ASSIGN statement. Second, a branch is made in an assigned GO TO statement using the integer variable as the branch controller in a list of possible statement numbers to be branched to.

ANS X3.9-1978 Reference: 10.3, 11.3

b. Special Considerations

There are 5 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.14 FM014 (Subset)

a. Features Tested

This routine tests the Fortran computed GO TO statement. Because the form of the computed GO TO is so straightforward, the tests mainly relate to the range of possible statement numbers which are used.

ANS X3.9-1978 Reference: 11.2

b. Special Considerations

There are 4 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.15 FM016 (Subset)

a. Features Tested

This routine begins a series of tests of the Fortran logical IF statement in all of the various forms. The following logical operands are used for this routine - logical constants, logical variables, logical array elements, and arithmetic expressions with various relational operators. Both the true and false branches are tested in the series of tests.

ANS X3.9-1978 Reference: 4.7.1, 6, 6.1, 6.3, 6.4, 6.6, 10, 10.2, 11.5

b. Special Considerations

There are 31 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.16 FM017 (Subset)

a. Features Tested

This routine continues tests of the Fortran logical IF statement in all of the various forms. The following logical operands are used for this routine - logical constants, logical variables, logical array elements, and arithmetic expressions with various relational operators. Both the TRUE and FALSE branches are tested in the series of tests.

ANS X3.9-1978 Reference: 4.7.1, 6, 6.1, 6.3, 6.4, 6.6, 10, 10.2, 11.5

b. Special Considerations

There are 30 'PASS/FAIL' tests

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.17 FM018 (Subset)

a. Features Tested

This routine continues tests of the Fortran logical IF statement in all of the various forms. The following logical operands are used for this routine - logical constants, logical variables, logical array elements, and arithmetic expressions with various relational operators. Both the TRUE and FALSE branches are tested in the series of tests.

ANS X3.9-1976 Reference: 4.7.1, 6, 6.1, 6.3, 6.4, 6.6, 10, 10.2, 11.5

b. Special Considerations

There are 30 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.18 FM019 (Subset)

a. Features Tested

This routine continues tests of the Fortran logical IF statement by testing various forms of relational expressions with arithmetic expressions. Positive and negative signs are used in conjunction with parentheses. Combinations of logical .AND. .OR. .NOT. are used to test the more complex expressions.

ANS X3.9-1978 Reference: 4.7.1, 6, 11.5

b. Special Considerations

There are 23 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.19 FM020 (Subset)

a. Features Tested

This routine tests the Fortran in-line statement function of type logical and integer. Integer constants, logical constants, integer variables, logical variables, integer arithmetic expressions are all used to test the statement function definition and the value returned for the statement function when it is used in the main body of the program.

ANS X3.9-1978 Reference: 8.4.1, 15.3.2, 15.4, 15.4.1, 15.4.2, 15.5.2

b. Special Considerations

This program assumes that the compiler supports the Intrinsic Functions SQRT and FLOAT.

There are 12 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.20 FM021 (Subset)

a. Features Tested

This routine tests the Fortran data initialization statement. Integer, real, and logical data types are tested using unsigned constants, signed constants, and logical constants. Integer, real logical, and mixed type arrays are also tested.

ANS X3.9-1978 Reference: 4.1.3, 4.4.3, 9

b. Special Considerations

There are 39 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.21 FM022 (Subset)

a. Features Tested

This routine tests arrays with fixed dimension and size limits set either in a blank common or dimension statement. The values of the array elements are set in various ways such as simple assignment statements, set to the values of other array elements (either positive or negative), set by integer to real or real to integer conversion, set by arithmetic expressions, or set by use of the equivalence statement.

ANS X3.9-1978 Reference: 8, 8.1, 8.2, 8.3, 8.4, 9

b. Special Considerations

There are 28 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.22 FM023 (Subset)

a. Features Tested

Two DIMENSIONed arrays are used in this routine. This routine tests arrays with fixed dimension and size limits of the array elements set either in a blank COMMON or DIMENSION statement. The values of the elements are set in various ways such as simple assignment statements, set to the values of other array elements (either positive or negative), set by integer to real or real to integer conversion, set by arithmetic expressions, or set by use of the EQUIVALENCE statement.

ANS X3.9-1978 Reference: 8, 8.1, 8.2, 8.3, 8.4, 9

b. Special Considerations

There are 13 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.23 FM024 (Subset)

a. Features Tested

Three dimensioned arrays are used in this routine. This routine tests arrays with fixed dimension and size limits set either in a blank common or dimension statement. The values of the array elements are set in various ways such as simple assignment statements, set to the values of other array elements (either positive or negative), set by integer to real or real to integer conversion, set by arithmetic expressions, or set by use of the equivalence statement.

ANS X3.9-1978 Reference: 8, 8.1, 8.2, 8.3, 8.4, 9

b. Special Considerations

There are 8 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.24 FM025 (Subset)

a. Features Tested

This routine tests IF statements, DO loops, assigned and computed GO TO statements in conjunction with array elements specified in COMMON or DIMENSION statements. One, two, and three dimensioned arrays are used. The subscripts are integer constants or sometimes integer variables when the elements are in loops and all arrays have fixed size limits. Integer, real, and logical arrays are used with the type sometimes specified with the explicit type statement.

ANS X3.9-1978 Reference: 8, 8.1, 8.3, 8.4, 9, 11.2, 11.3, 11.10

b. Special Considerations

There are 11 'PASS/FAIL' tests. The report should show that tests 663 and 664 have been omitted.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.25 FM026 (Subset)

a. Features Tested

This routine contains the basic subroutine reference tests. The subroutine FS027 is called by this program. The subroutine FS027 increments the calling argument by 1 and returns to the calling program.

Execution of a subroutine reference results in an association of actual arguments with all appearances of dummy arguments in the defining sub program. Following these associations execution of the first executable statement of the defining subprogram is undertaken.

ANS X3.9-1978 Reference: 15.6.2

b. Special Considerations

There are 4 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: FM027

X-Numbers: 02

D.1.26 FM028 (Subset)

a. Features Tested

This routine contains external function reference tests. The function subprogram FF029 is called by this program. The function subprogram FF029 increments the calling argument by 1 and returns the incremented value to the calling program as the function value.

The external function FF029 is referenced by using its reference as a primary in an arithmetic expression. Execution of the function reference results in an association of actual arguments with all appearances of dummy arguments in the defining subprogram. Following these associations, execution of the first executable statement of the defining subprogram is undertaken.

ANS X3.9-1978 Reference: 15.5.2

b. Special Considerations

There are 4 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: FM029

X-Numbers: 02

D.1.27 FM030 (Subset)

a. Features Tested

This routine tests arithmetic assignment statements of the form:

integer variable = arithmetic expression

where the arithmetic expression is formed with the arithmetic operator (-) integer constants and integer variables.

ANS X3.9-1978 Reference: 4.3, 4.3.1, 6.1, 10.1

b. Special Considerations

There are 35 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.28 FM031 (Subset)

a. Features Tested

This routine tests arithmetic assignment statements of the form:

integer variable = arithmetic expression

where the arithmetic expression is formed with the arithmetic operator integer constants and integer variables. Some of the tests use parentheses to group elements in an arithmetic expression. The integer variables contain positive and negative values.

ANS X3.9-1978 Reference: 43, 4.3.1, 6.1, 10.1

b. Special Considerations

There are 30 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.29 FM032 (Subset)

a. Features Tested

This routine tests arithmetic assignment statements of the form:

integer variable = arithmetic expression

where the arithmetic expression is formed with the arithmetic operator (-) integer constants, and integer variables. Some of the tests use parentheses to group elements in an arithmetic expression.

ANS X3.9-1978 Reference: 4.3, 4.3.1, 6.1, 10.1

b. Special Considerations

There are 30 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.30 FM033 (Subset)

a. Features Tested

This routine tests arithmetic assignment statements of the form:

integer variable = arithmetic expression

where the arithmetic expression is formed with the arithmetic operator and integer constants. Some of the tests use parentheses to group elements in the expression and to allow the use of negative constants following the operator.

ANS X3.9-1978 Reference: 43, 4.3.1, 6.1, 10.1

b. Special Considerations

There are 35 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.31 FM034 (Subset)

a. Features Tested

This routine tests arithmetic assignment statements of the form:

integer variable = arithmetic expression

where the arithmetic expression is formed with the arithmetic operator (*), integer variables and integer constants. Some of the tests use parentheses to group elements in the arithmetic expression and to permit the use of negative constants following the (*) operator. The integer variables contain positive and negative values.

ANS X3.9-1978 Reference: 4.3, 4.3.1, 6.1, 10.1

b. Special Considerations

There are 35 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.32 FM035 (Subset)

a. Features Tested

This routine tests arithmetic assignment statements of the form:

integer variable = arithmetic expression

where the arithmetic expression is formed with the arithmetic operator (*), integer variables, and an integer constant. The integer variables contain positive and negative values. Some of the tests use parentheses to group elements in the arithmetic expressions. Parentheses are also used to enclose negative constants.

ANS X3-1978 Reference: 4.3, 4.3.1, 6.1, 10.1

b. Special Considerations

There are 32 'PASS/FAIL' tests:

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.33 FM036 (Subset)

a. Features Tested

This routine tests arithmetic assignment statements of the form:

integer variable = arithmetic expression

where the arithmetic expression formed with the arithmetic operator and integer constants. Both positive and negative integer constants are used in the arithmetic expression.

Some of the tests require no truncation of the result while other tests require truncation before the result is stored in the integer variable. The standard states 'The value of an integer factor or term is the nearest integer whose magnitude does not exceed the magnitude of the mathematical value represented by that factor or term.'

ANS X3.9-1978 Reference: 4.3, 4.3.1, 6.1, 6.6, 10.1

b. Special Considerations

There are 29 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.34 FM037 (Subset)

a. Features Tested

This routine tests arithmetic assignment statements of the form:

integer variable = arithmetic expression

where the arithmetic expression is formed with the arithmetic operator / and three integer constants. Both positive and negative integer constants are used in the arithmetic expression.

Some of the tests require no truncation of the result while other tests require truncation before the result is stored in the integer variable. The Standard states, 'The value of an integer factor or term is the nearest integer whose magnitude does not exceed the magnitude of the mathematical value represented by that factor or term. The associative and commutative laws do not apply in the evaluation of integer terms containing division, hence evaluation of such terms must effectively proceed from left to right.'

ANS X3.9-1978 Reference: 4.3, 4.3.1, 6.1, 6.6, 10.1

b. Special Considerations

There are 29 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.35 FM038 (Subset)

a. Features Tested

This routine tests arithmetic assignment statements of the form:

integer variable = arithmetic expression

where the arithmetic expression is formed with the arithmetic operator /, integer constants and an integer variable. Both positive and negative values are used for the integer constants and the integer variable.

Some of the tests require no truncation of the result while other tests require truncation before the result is stored in the integer variable. Parentheses are used to group elements in the arithmetic expressions with three operands. The use of parentheses to group the last two operands overrides the evaluation from left to right of integer terms which contain division. The second division must be performed first.

ANS X3.9-1978 Reference: 4.3, 4.3.1, 6.1, 6.6, 10.1

b. Special Considerations

There are 32 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.36 FM039 (Subset)

a. Features Tested

This routine tests arithmetic assignment statements of the form:

integer variable = arithmetic expression

where the arithmetic expression is formed with the arithmetic operator /, integer constants and an integer variable. Both positive and negative values are used for the integer constants and the integer variable.

ANS X3.9-1978 Reference: 4.3, 4.3.1, 6.1, 6.6, 10.1

b. Special Considerations

There are 30 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.37 FM040 (Subset)

a. Features Tested

This routine tests arithmetic assignment statements of the form:

integer variable = arithmetic expression

where the arithmetic expression is formed with the arithmetic operator /, integer variables and an integer constant. Both positive and negative values are used for the integer variables and the integer constant.

ANS X3.9-1978 Reference: 4.3, 4.3.1, 6.1, 6.6, 10.1

b. Special Considerations

There are 33 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.38 FM041 (Subset)

a. Features Tested

This routine tests arithmetic assignment statements of the form:

integer variable = primary ** primary

where the first of two primaries is an integer variable or an integer constant and the second primary is an integer variable.

ANS X3.9-1978 Reference: 4.3, 4.3.1, 6.1, 10.1

b. Special Considerations

There are 34 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.39 FM042 (Subset)

a. Features Tested

This routine tests arithmetic assignment statements of the form:

integer variable = primary primary

where the first of two primaries is an integer variable or an integer constant and the second primary is an integer variable.

ANS X3.9-1978 Reference: 4.3, 4.3.1, 6.1, 10.1

b. Special Considerations

There are 34 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.40 FM043 (Subset)

a. Features Tested

This routine tests arithmetic assignment statements of the form:

```
INTEGER VAR. =  INTEGER VAR. <OP1>
                INTEGER VAR. <OP2> INTEGER VAR.
```

where <OP1> and <OP2> are arithmetic operators, but <OP1> is not the same as <OP2>. All combinations of parenthetical associations are also exercised.

ANS X3.9-1978 Reference: 4.3, 4.3.1, 6.1, 6.6, 10.1

b. Special Considerations

There are 36 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.41 FM044 (Subset)

a. Features Tested

This routine tests arithmetic assignment statements of the form:

```
INTEGER VAR. =  INTEGER VAR. <OP1>
                INTEGER VAR. <OP2> INTEGER VAR.
```

where <OP1> and <OP2> are arithmetic operators.

ANS X3.9-1978 Reference: 4.3, 4.3.1, 6.1, 6.6, 10.1

b. Special Considerations

There are 28 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.42 FM045 (Subset)

a. Features Tested

This routine tests arithmetic assignment statements containing integer variables connected by a series of arithmetic operators. Different combinations of parenthetical notation are exercised.

ANS X3.9-1978 Reference: 4.3, 4.3.1, 6.1, 6.6, 10.1

b. Special Considerations

There are 13 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.43 FM050 (Subset)

a. Features Tested

This routine contains basic subroutine and function reference tests. Four subroutines and one function are called or referenced. FS051 is called to test the calling and passing of arguments through unlabeled COMMON. No arguments are specified in the call line. FS052 is identical to FS051 except that several returns are used. FS053 utilizes many arguments on the CALL statement and many RETURN statements in the subroutine body. FF054 is a function subroutine in which many arguments and RETURN statements are used. And finally FS055 passes a one dimensional array back to FM050.

ANS X3.9-1978 Reference: 15.5.2, 15.6.2

b. Special Considerations

There are 30 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: FM051, FM052, FM053, FM054, FM055

X-Numbers: 02

D.1.44 FM056 (Subset)

a. Features Tested

FM056 is a main program which tests the argument passing linkage of a 2 level nested subroutine and an external function reference. The main program FM056 calls subroutine FS057 passing one argument. Subroutine FS057 calls subroutine FS058 passing two arguments. Subroutine FS058 references external function FF059 passing 3 arguments. Function FF059 adds the values of the 3 arguments together. Subroutines FS057 and FS058 then merely return the result to FM056 in the first argument.

The values of the arguments that are passed to each subprogram and function, and returned to the calling or referencing program are saved in an integer array. FM056 then uses these values to test the compiler's argument passing capabilities.

ANS X3.9-1978 Reference: 15.6.2

b. Special Considerations

There are 12 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: FM057, FM058, FM059

X-Numbers: 02

D.1.45 FM060 (Subset)

a. Features Tested

This routine contains basic arithmetic IF statement tests for the format:

IF (E) K1, K2, K3

where E is a simple real expression of the form:

Real Variable
Real Variable - Real Constant
Real Variable + Real Constant

and Kl, K2 and K3 are statement labels.

The routine FM060 also tests arithmetic assignment statements of the form.

Real Variable = Real Constant
Real Variable = Real Variable
Real Variable = -Real Variable

The real constants and real variables contain both positive and negative values.

ANS X3.9-1978 Reference: 4.4, 4.4.1, 6.1, 10.1, 11.4

b. Special Considerations

There are 31 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.46 FM061 (Subset)

a. Features Tested

This routine tests arithmetic assignment statements of the form:

Integer variable = Real constant
Integer variable = Real variable
Real variable = Integer variable
Real variable = Integer constant

The constants and variables contain both positive and negative values.

ANS X3.9-1978 Reference: 4.4, 4.4.1, 6.1, 6.6, 10.1, 11.4

b. Special Considerations

There are 30 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.47 FM062 (Subset)

a. Features Tested

This routine tests arithmetic assignment statements where an arithmetic expression formed from real variables and constants connected by arithmetic operators is assigned to a real variable. In cases involving the exponentiation operator, real values are raised to integer powers only.

ANS X3.9-1978 Reference: 4.4, 4.4.1, 6.1, 6.6, 10.1

b. Special Considerations

There are 31 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.48 FM080 (Subset)

a. Features Tested

This routine contains external function reference tests. The function subprograms called by this routine are FF081, FF082 and FF083. The function subprograms are defined as FF081 = integer, FF082 = real, FF083 = implicit real. The function sub program dummy arguments must agree in order, number and type with the corresponding actual arguments of the main program. The arguments of the function subprograms will correspond to actual argument list references of variable-name, array-name, array-element-name and expression respectively.

This routine will test the value of the function and the function arguments returned following the function reference call.

ANS X3.9-1978 Reference: 2.6, 15.5.2, 17.2

b. Special Considerations

There are 17 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: FM081, FM082, FM083

X-Numbers: 02

D.1.49 FM097 (Subset)

a. Features Tested

This routine tests intrinsic functions where the function type is real and the arguments are either integer or real. The real and integer variables and the real and integer constants contain both positive and negative values. The intrinsic functions tested by FM097 are ABS, AINT, AMOD, AMAX0, AMAX1 AMIN0, AMIN1, FLOAT, SIGN and DIM.

ANS X3.9-1978 Reference: 4.1.2, 15.3, 15.3.2

b. Special Considerations

There are 32 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.50 FM098 (Subset)

a. Features Tested

This routine tests intrinsic functions where the function type is integer and the arguments are either integer or real. The real and integer variables and the real and integer constants contain both positive and negative values. The intrinsic functions tested by FM098 are IABS, INT, MOD, MAX0, MAX1, MIN0, MIN1, IFIX, ISIGN and IDIM.

ANS X3.9-1978 Reference: 4.1.2, 15.3, 15.3.2

b. Special Considerations

There are 32 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.51 FM099 (Subset)

a. Features Tested

This routine tests various mathematical functions where both the function type and arguments are real. The real variables and constants contain both positive and negative values. The functions tested in FM099 are EXP, ALOG, ALOG10, SIN, COS, TANH, SQRT, ATAN and ATAN2.

ANS X3.9-1978 Reference: 8.7, 15.5.2

b. Special Considerations

There are 26 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.52 FM100 (Subset)

a. Features Tested

This routine is a test of the I format and is tape and printer oriented. The routine can also be used for disk. Both the READ and WRITE statements are tested. Variables in the input and output lists are integer variable and integer array element or array name references. ALL READ and WRITE statements are done with format statements. The routine has an optional section of code to dump the file after it has been written. DO loops and DO-implied lists are used in conjunction with a one dimensional integer array for the dump section.

This routine writes a single sequential file which is rewound and read sequentially forward. Every fourth record is checked during the Read Test Section plug the last two records and the end of file on the last record.

The line continuation in column 6 is used in READ, WRITE, and format statements. For both syntax and semantic tests, all statements should be checked visually for the proper functioning of the continuation line.

ANS X3.9-1978 Reference: 8, 9, 11.10, l2, 12.8.2, 12.9.5.2, 13, 13.2.1, 13.5.9.1

b. Special Considerations

This test routine should be executed twice. The first time, assign LUN I06 to tape, then reassign LUN I06 to disk.

There are 11 'PASS/FAIL' tests. The report should show the message 'FILE I06 CREATED WITH 31 SEQUENTIAL RECORDS'.

Related Functions, Subroutines or Programs: None

X-Numbers: 02, 06

D.1.53 FM101 (Subset)

a. Features Tested

This routine is a test of the F format and is tape and printer oriented. The routine can also be used for disk. Both the READ and WRITE statements are tested. Variables in the input and output lists are real variables and real array elements or array name references. All READ and WRITE statements are done with format statements. The routine has an optional section of code to dump the file after it has been written. DO loops and DO-implied lists are used in conjunction with a one dimensional integer array for the dump section.

This routine writes a single sequential file which is rewound and read sequentially forward. Every fourth record is checked during the READ Test Section plus the last two records and the end of file on the last record.

The line continuation in column 6 is used in READ, WRITE, and FORMAT statements. For both syntax and semantic tests, all statements should be checked visually for the proper functioning of the continuation line.

ANS X3.9-1978 Reference: 8, 9, 11.10, 12, 12.8.2, 12.9.5.2, 13, 13.2.1

b. Special Considerations

This test routine should be executed twice. The first time, assign LUN I07 to tape, then reassign LUN I07 to disk.

There are 11 'PASS/FAIL' tests. The report should show the message 'FILE I07 CREATED WITH 31 SEQUENTIAL RECORDS'.

Related Functions, Subroutines or Programs: None

X-Numbers: 02, 07

D.1.54 FM102 (Subset)

a. Features Tested

This routine is a test of the A format and is tape and printer oriented. The routine can also be used for disk. Both the READ and WRITE statements are tested. Variables in the input and output lists are alphanumeric integers and array elements or array name references. All READ and WRITE statements are done with FORMAT statements. The routine has an optional section of code to dump the file after it has been written. DO loops and DO-implied lists are used in conjunction with a one dimensional integer array for the dump section.

This routine writes a single sequential file which is rewound and read sequentially forward. Every record is read and checked for accuracy and the end of file on record 31 is also checked. During the read and check process the file is rewound twice. The first pass checks the odd numbered records and the second pass checks the even numbered records.

The line continuation in column 6 is used in READ, WRITE and FORMAT statements. For both syntax and semantic tests, all statements should be checked visually for the proper functioning of the continuation line.

ANS X3.9-1978 Reference: 8, 9, 11.10, 12, 12.8.2, 12.9.5.2, 13, 13.2.1

b. Special Considerations

This test routine should be executed twice. The first time, assign LUN I08 to tape, then reassign LUN I08 to disk.

There are 32 'PASS/FAIL' tests. The report should show the message FILE I08 CREATED WITH 31 SEQUENTIAL RECORDS'.

Related Functions, Subroutines or Programs: None

X-Numbers: 02, 08

D.1.55 FM103 (Subset)

a. Features Tested

This routine is a test of the X format and is tape and printer oriented. The routine can also be used for disk. Both the READ and WRITE statements are tested. Variables in the input and output lists are integer or real variables, integer array elements or array name references. READ and WRITE statements are done with FORMAT statements. The routine has an optional section of code to dump the file after it has been written. DO loops and DO-implied lists are used in conjunction with a one-dimensional integer array for the dump section.

This routine writes a single sequential file which is rewound and read sequentially forward. Every record is read and checked for accuracy and the end of file on record 31 is also checked. During the read and check process the file is rewound twice. The first pass checks the odd numbered records and the second pass checks the even numbered records.

The line continuation in column 6 is used in READ, WRITE, and FORMAT statements. For both syntax and semantic tests, all statements should be checked visually for the proper functioning of the continuation line.

ANS X3.9-1978 Reference: 8, 9, 11.10, 12, 12.8.2, 12.9.5.2, 13, 13.2.1

b. Special Considerations

This test routine should be executed twice. The first time, assign LUN I09 to tape, then reassign LUN I09 to disk.

There are 32 'PASS/FAIL' tests. The report should show the message 'FILE I09 CREATED WITH 31 SEQUENTIAL RECORDS'.

Related Functions, Subroutines or Programs: None

X-Numbers: 02, 09

D.1.56 FM104 (Subset)

a. Features Tested

This routine is a test of the / format and is tape and printer oriented. The routine can also be used for disk. Both the READ and WRITE statements are tested. Variables in the input and output lists are integer variable and integer array element or array name references. All READ and WRITE statements are done with FORMAT statements. The routine has an optional section of code to dump the file after it has been written. DO loops and DO-implied lists are used in conjunction with a one dimensional integer array for the dump section.

This routine writes a single sequential file which is rewound and read sequentially forward. Every record is read and checked during the Read Test Section for values of data items and the end of file on the last record is also checked.

The line continuation in column 6 is used in READ, WRITE and FORMAT statements. For both syntax and semantic tests, all statements should be checked visually for the proper functioning of the continuation line.

ANS X3.9-1978 Reference: 8, 9, 11.10, 12, 12.8.2, 12.9.5.2, 13, 13.2.1, 13.5.9.1

b. Special Considerations

This test routine should be executed twice. The first time, assign LUN I06 to tape, then reassign LUN I06 to disk.

There are 8 'PASS/FAIL' tests. The report should show the message 'FILE I06 CREATED WITH 28 SEQUENTIAL RECORDS'

Related Functions, Subroutines or Programs: None

X-Numbers: 02, 06

D.1.57 FM105 (Subset)

a. Features Tested

FM105 tests repeated ( ) format fields and is tape and printer oriented. The routine can also be used for disk. Both the READ and WRITE statements are tested. Variables in the input and output lists are integer variable and integer array element or array name references. All READ and WRITE statements are done with FORMAT statements. The routine has an optional section of code to dump the file after it has been written. DO loops and DO-implied lists are used in conjunction with a one dimensional integer array for the dump section.

Routine FM105 is exactly like routine FM104 except that format numbers 77751 and 77752 have been changed to use three (3) repeated fields, i.e.,... 3(/ ...) this should still make the routine write and then read four (4) 80 character records for each single WRITE or READ statement. Other format conversions used are the X and I format fields. Because of the number of characters to be written or read in each set of four records, the entire repeated field is used.

This routine writes a single sequential file which is rewound and read sequentially forward. Every record is read and checked during the Read Test Section for values of data items and the end of file on the last record is also checked.

The line continuation in column 6 is used in READ, WRITE, and FORMAT statements. For both syntax and semantic tests, all statements should be checked visually for the proper functioning of the continuation line.

ANS X3.9-1978 Reference: 8, 9, 11.10, 12, 12.8.2, 12.9.5.2, 13, 13.2.1, 13.5.9.1

b. Special Considerations

This test routine should be executed twice. The first time, assign LUN I08 to tape, then reassign LUN I08 to disk.

There are 8 'PASS/FAIL' tests. The report should show the message FILE I08 CREATED WITH 28 SEQUENTIAL RECORDS'.

Related Functions, Subroutines or Programs: None

X-Numbers: 02, 08

D.1.58 FM106 (Subset)

a. Features Tested

This routine is a test of the E format and is tape and printer oriented. The routine can also be used for disk. Both the READ and WRITE statements are tested. Variables in the input and output lists are real variables and real array elements or array name references. All READ and WRITE statements are done with FORMAT statements. The routine has an optional section of code to dump the file after it has been written. DO loops and DO-implied lists are used in conjunction with a one dimensional integer array for the dump section.

This routine writes a single sequential file which is rewound and read sequentially forward. Every fourth record is checked during the Read Test Section plus the last two records and the end of file on the last record.

The line continuation in column 6 is used in READ, WRITE and FORMAT statements For both syntax and semantic tests, all statements should be checked visually for the proper functioning of the continuation line.

ANS X3.9-1978 Reference: 8, 9, 11.10, 12, 12.8.2, 12.9.5.2, 13, 13.2.1

b. Special Considerations

This test routine should be executed twice. The first time, assign LUN I09 to tape, then reassign LUN I09 to disk.

There are 11 'PASS/FAIL' tests. The report should show the message 'FILE I09 CREATED WITH 124 SEQUENTIAL RECORDS'.

Related Functions, Subroutines or Programs: None

X-Numbers: 02, 09

D.1.59 FM107 (Subset)

a. Features Tested

This routine is a test of the I format and is tape and printer oriented. The routine can also be used for disk. Both the READ and WRITE statements are tested. Variables in the input and output lists are integer variable and integer array element or array name references. All READ and WRITE statements are done with FORMAT statements. The routine has an optional section of code to dump the file after it has been written. DO loops and DO-implied lists are used in conjunction with a one dimensional integer array for the dump section.

The major purpose of this routine is to test whether the last set of parentheses will be repeated in a FORMAT statement if the number of data items in the input/output list is greater than the number of field specifications within the FORMAT statement. In addition the use of two and three dimensioned arrays is tested in the implied-DO lists in both the WRITE and READ sections.

This routine writes a single sequential file which is rewound and read sequentially forward. Every fourth record is checked during the Read Test Section plus the last two records and the end of file on the last record.

The line continuation in column 6 is used in READ, WRITE, and FORMAT statements. For both syntax and semantic tests, all statements should be checked visually for the proper functioning of the continuation line.

ANS X3.9-1978 Reference: 8, 9, 11.10, 12, 12.8.2, 12.9.5.2, 13, 13.2.1

b. Special Considerations

This test routine should be executed twice. The first time, assign LUN I06 to tape, then reassign LUN I06 to disk.

There are 11 'PASS/FAIL' tests. The report should show the message 'FILE I06 CREATED WITH 137 SEQUENTIAL RECORDS'.

Related Functions, Subroutines or Programs: None

X-Numbers: 02, 06

D.1.60 FM108 (Subset)

a. Features Tested

This routine is a test of the X format and is tape and printer oriented. The routine can not be used for disk. Both the READ and WRITE statements are tested. Variables in the input and output lists are integer or real variables, integer array elements, or array name references. READ and WRITE statements are done with FORMAT statements. The routine has an optional section of code to dump the file after it has been written. DO loops and DO-implied lists are used in conjunction with a one dimensional integer array for the dump section.

With the exception of the record preambles on each record, all of the I,F, and A-fields have a minus sign (-) in the leftmost character position of each field.

This routine writes a single sequential file which is rewound and read sequentially forward and then read sequentially backward by using the BACKSPACE command. The forward read is used to check all of the odd records and the read "reverse" is used to check all of the even numbered records. The ENDFILE command is also used after the write section, but because the result of attempting to read or read beyond the end file mark is not possible to predict for all machines, the endfile mark is never actually read.

The line continuation in column 6 is used in READ, WRITE, and FORMAT statements. For both syntax and semantic tests, all statements should be checked visually for the proper functioning of the continuation line.

ANS X3.9-1978 Reference: 8, 9, 11.10, 12, 12.8.2, 12.9.5.2, 13, 13.2.1

b. Special Considerations

This test routine should be executed using LUN I08 assigned to tape.

There are 31 'PASS/FAIL' tests. The report should show the message 'FILE I08 CREATED WITH 31 SEQUENTIAL RECORDS'.

Related Functions, Subroutines or Programs: None

X-Numbers: 02, 08

D.1.61 FM109 (Subset)

a. Features Tested

This routine tests the basic options regarding the simple formatted WRITE statement of the form:

WRITE (U, F) or WRITE (U, F) L

where U is a logical unit number, F is a FORMAT statement number, and L is an output list. The FORMAT specifications contain nH - Hollerith descriptions, nX blank field descriptions, and Iw numeric integer descriptions.

This routine tests whether the first character of a FORMAT record for printer output determines vertical spacing as follows:

1 - advance to the first line at the top of the next page
0 - advance two lines before printing
+ - do not advance before printing the next line
blank - advance one line before printing

ANS X3.9-1978 Reference: 8, 9, 11.10, 12, 12.8.2, 12.9.5.2, 13, 13.2.1

b. Special Considerations

There are 22 visual tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.1.62 FM110 (Subset)

a. Features Tested

This program tests additional features of READ and WRITE statements, formatted records, and format statements for integer and real data types.

ANS X3.9-1978 Reference: 12.8, 13

b. Special Considerations

Correctness of the statements is determined by a comparison of pairs of lines in the output file.

There are 40 input records. The data can be selected off the population file or generated by the user (the data records are listed in the source program as comments).

A visual inspection of the test results is required.

There are 11 visual tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 01, 02.

D.2 Subset Language Programs Part 2

D.2.1 FM111 (Subset)

a. Features Tested

This program tests additional features of READ and WRITE statements, formatted records, format statements for integer and real data types and character constants as format specifiers.

ANS X3.9-1978 Reference: 12.1.1, 12.8, 13.1.1, 13.2.1, 13.3, 13.5.2, 13.5.9, 13.5.9.1

b. Special Considerations

Correctness of the statements is determined by a comparison of lines in the output file.

There are 8 input records. The data can be selected off the population file or generated by the user (the data records are listed in the source program as comments).

The standard does not specify the form of zero on output, so that the exponent may or may not be present; if present it need not be E+00, however it may not be E-00.

A visual inspection of the test results is required.

There are 4 visual tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 01, 02

D.2.2 FM200 (Subset)

a. Features Tested

The routine FM200 is the first audit routine to contain a PROGRAM statement. The following features from Section 3., "Characters, Lines and Execution Sequence" are tested in this routine:

(1) Asterisk in column 1 to designate a comment line.

(2) Use of non-Fortran characters within a comment line.

(3) Statement labels on nonexecutable statements.

(4) Digit 0 in column 6 of an initial line.

(5) Continuation Lines - Maximum nine continuation lines (660 characters).

(6) Blank characters within statements.

(7) Blank comment line, blank characters in columns 1-72.

ANS X3.9-1978 Reference: 3.1.6, 3.2.1, 3.2.2, 3.2.3, 3.3, 3.4, 14.1

b. Special Considerations

There are 13 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.2.3 FM201 (Subset)

a. Features Tested

The routine FM201 verifies that:

(1) The value of a signed zero is the same as the value of an unsigned zero for integer and real variables. The integer constants 0, +0, and -0 are tested in the routines FM018 and FM019.

(2) A basic real constant may be written with more digits than a processor will use to approximate the value of the constant.

(3) An IMPLICIT statement can be used to change the default implicit integer and real typing.

(4) The implicit integer and real typing of an IMPLICIT statement may be overridden by the appearance of a variable name in a type-statement.

ANS X3.9-1978 Reference: 4.1.3, 4.4.1, 6.1.5, 8.4, 8.5

b. Special Considerations

There are 22 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.2.4 FM202 (Subset)

a. Features Tested

The routine FM202 is the first routine to test character data types. CHARACTER type-statements specify character variables of length one and length two. The tests in this routine determine that the following language features function correctly.

(1) Character assignment statements.

(2) The representation of an apostrophe in a character constant is two consecutive apostrophes with no intervening blanks.

(3) Character relational expressions.

ANS X3.9-1978 Reference: 4.8, 4.8.1, 6.2, 6.3.4, 6.3.5, 8.4.2, 10.4

b. Special Considerations

There are 30 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.2.5 FM203 (Subset)

a. Features Tested

The routine FM203 continues the testing of character data types which was started in FM202. The CHARACTER type-statements specify character variables and one-dimensional character arrays of length one and length two.

ANS X3.9-1978 Reference: 4.8, 4.8.1, 6.2, 6.3.4, 6.3.5, 8.4.2, 10.4

b. Special Considerations

There are 30 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.2.6 FM204 (Subset)

a. Features Tested

The routine FM204 continues the testing of character variables and character arrays of length one. The character features tested in FM202 and FM203 are used in the tests in this routine. The following character features are tested:

(1) Initial definition of character entities of length one by specifying them in a DATA statement.

(2) The subset Fortran language collating sequence.

(3) The intrinsic function ICHAR.

ANS X3.9-1978 Reference: 3.1.5, 4.8, 6.2, 6.3.4, 6.3.5, 8.4.2, 9.4, 10.4, 15.3, 15.10

b. Special Considerations

In addition to the 26 'PASS/FAIL' tests, the report should show 2 visual tests. The visual tests list the Fortran character set in ascending sequence and the ICHAR intrinsic function values for the Fortran character set.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.2.7 FM205 (Subset)

a. Features Tested

The routine FM205 tests character constants, character variables, and character array elements with a maximum length of 57 characters.

ANS X3.9-1978 Reference: 4.8, 4.8.1, 6.2, 6.3.4, 6.3.5, 8.4.2, 10.4

b. Special Considerations

There are 30 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.2.8 FM251 (Subset)

a. Features Tested

This program tests the IMPLICIT statement for declaring variables as type LOGICAL. The type of a variable (logical, integer, or real) is set by both IMPLICIT statements and also by explicit TYPE statements. Tests are made to check that explicit TYPE statements override the type set by an IMPLICIT statement for the variables listed.

ANS X3.9-1978 Reference: 4.7, 8.4.1, 8.5

b. Special Considerations

There are 13 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.2.9 FM252 (Subset)

a. Features Tested

This program tests redefinition of Statement Labels with the ASSIGN statement in conjunction with the assigned GO TO statement. The optional comma in the syntax of the assigned GO TO is tested. The range of statement labels (from 00001 to 99999) is tested using the ASSIGN statement and the assigned GO TO statement. It also tests the optional comma in the syntax of the computed GO TO statement and has tests on the range of the index in the computed GO TO.

ANS X3.9-1978 Reference: 10.3, 11.2, 11.3

b. Special Considerations

There are 11 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.2.10 FM253 (Subset)

a. Features Tested

This routine is a test of the IF-block. Tests within this routine are for the syntax of the basic IF ( ) THEN through END IF block structure.

There is also a series of tests to check the hierarchy and order of evaluation in expressions that contain a combination of arithmetic, relational, and logical operators.

ANS X3.9-1978 Reference: 11.6, 11.6.1, 11.6.2, 11.6.3

b. Special Considerations

There are 28 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.2.11 FM254 (Subset)

a. Features Tested

This routine is a test of the ELSE IF-block. Tests within this routine are for the syntax of the basic ELSE IF statement and ELSE IF-block structure.

ANS X3.9-1978 Reference: 11.7, 11.7.1, 11.7.2

b. Special Considerations

There are 12 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.2.12 FM255 (Subset)

a. Features Tested

This routine is a test of the ELSE statement. Tests within this routine are for the syntax of the basic ELSE statement and ELSE block structures. The END IF statement is used in all block IF structures for the routines FM253, FM254, and FM255. For each block IF statement, there must be a corresponding END IF statement in the same program unit.

ANS X3.9-1978 Reference: 11.8, 11.8.1, 11.8.2, 11.9

b. Special Considerations

There are 16 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.2.13 FM256 (Subset)

a. Features Tested

This routine is a test of the DO statement. The DO is tested both outside and inside the block-IF structure. Tests are made of the DO-variable when the DO becomes inactive. Other tests check loop and incrementation processing. The DO-loop execution is tested for those conditions which make the DO-loop inactive.

ANS X3.9-1978 Reference: 11.10, 11.10.1, 11.10.2, 11.10.3, 11.10.4, 11.10.5, 11.10.6, 11.10.7

b. Special Considerations

There are 24 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.2.14 FM257 (Subset)

a. Features Tested

This routine is a test of the PAUSE and STOP statements. These statements can now be followed by a string of not more than five digits, or a character constant.

ANS X3.9-1978 Reference: 11.12, 11.13

b. Special Considerations

The following series of tests check the various forms of the PAUSE statement. In each case the word PAUSE (followed by a string of characters as noted in each test description), should be displayed on the operators console. For each test the operator need only do whatever is necessary to tell the system to continue the execution of the routine. The string forms are as described in Section 11.13.

(1) Test 001 checks the PAUSE statement that is not followed by a string of anything except blanks. Only the word PAUSE should be displayed.

(2) Test 002 should display the word PAUSE followed by a single character zero (0).

(3) Test 003 should display the word PAUSE followed by a string of five zeros (00000).

(4) Test 004 should display the word PAUSE followed by the string of five characters (19283).

(5) Test 005 should display the word PAUSE followed by the string of four nines (9999).

(6) Test 006 is for the STOP statement.

Since the STOP statement can only be executed once in a program unit, various formats of the STOP statement will be checked for syntax only by the use of a computed GO TO statement. Once the STOP statement has been executed, then the routine FM257 should no longer execute. Any continuation is considered as a failure of this test.

The report should show 5 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.2.15 FM258 (Subset)

a. Features Tested

Tests Block IF statements; specifically checking proper action with IF (e) THEN, ELSE, ELSEIF, and ENDIF. Also tested, for correct compilation and execution, is the case of the empty block.

ANS X3.9-1978 Reference: 11.6 - 11.9

b. Special Considerations

At the end of the control structure, (following ENDIF), the expected value of the test variable is subtracted from the actual value. A zero indicates a successful test. A visual inspection of the test results is required. There are 8 visual tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.2.16 FM259 (Subset)

a. Features Tested

Tests Block IF statements; specifically checking proper action of GOTO, COMPUTED GOTO, ASSIGNed GOTO, and DO statements.

ANS X3.9-1978 Reference: 11.1 - 11.3, 11.6 - 11.10

b. Special Considerations

The proper branching of the Block IF is tested for all cases. At the end of the control structure, (following ENDIF), the expected value of the test variable is subtracted from the actual value. Each test generates one or several results. For a successful test, all results should be zero.

There are 3 visual tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.2.17 FM260 (Subset)

a. Features Tested

Tests Block IF statements; specifically checking proper action with IF (e) THEN, ELSE, ELSEIF, and ENDIF.

ANS X3.9-1978 Reference: 11.1 - 11.3; 11.6 - 11.10

b. Special Considerations

The proper branching of the Block IF is tested for all cases. At the end of the control structure, (following ENDIF), the expected value of the test variable is subtracted from the actual value. Each test generates one or several results. For a successful test, all results should be zero.

There are 2 visual tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.2.18 FM261 (Subset)

a. Features Tested

Tests Block IF with CALL statement; specifically with CALL both external to and internal to BLOCK IF. The RETURN statement is also tested with BLOCK IF.

ANS X3.9-1978 Reference: 11.6 - 11.9; 15.6

b. Special Considerations

FM261 is the main program which calls subroutines SN262 and SN263 and references Integer Function IF264.

There are 2 visual tests.

Related Functions, Subroutines or Programs: FM262, FM263, FM264

X-Numbers used: 02

D.2.19 FM300 (Subset)

a. Features Tested

FM300 tests the use of the EQUIVALENCE statement to equate storage units of variables, arrays and array elements. Only integer, real, logical and character data types are tested. No attempt is made to test data of different types that are equated with the EQUIVALENCE statement. The subset level features of the EQUIVALENCE statement are also tested in routines FM022, FM023 and FM024.

ANS X3.9-1978 Reference: 8.1, 8.2, 9

b. Special Considerations

There are 19 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.2.20 FM301 (Subset)

a. Features Tested

This routine tests the use of the type-statement to explicitly define the data type for variables, arrays and statement functions. Only integer, real, logical and character data types are tested in this routine. Integer and real variables and arrays are tested in a manner which both confirms and overrides the implicit typing of the data entities.

ANS X3.9-1978 Reference: 4.1, 8.4, 8.5, 15.4

b. Special Considerations

There are 19 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.2.21 FM302 (Subset)

a. Features Tested

This routine tests the subset level features of the COMMON specification statement. Integer, real and logical variables and arrays are passed back-and- forth between the main program, external functions and subroutines. Both named and unnamed (blank) common are tested. Specific tests are included for renaming entities in common between program units, the passing of data through common by equivalence association, and the specifying of unnamed common of different lengths in different program units. The subset level features of the COMMON statement are also tested in FM022 through FM025, FM050 and FM056.

ANS X3.9-1978 Reference: 8.2, 8.3, 15.5, 15.6 15.9.4

b. Special Considerations

There are 16 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: FM303, FM304, FM305

X-Numbers: 02

D.2.22 FM306 (Subset)

a. Features Tested

This routine tests the use of the subset level features of the IMPLICIT specification statement. The default implied integer and real typing is either confirmed or overridden to specify integer, real and logical typing. All 26 alphabetic letters are used to indicate the implicit typing. Variable and array entities are used to test the actual typing. The subset level features of the IMPLICIT statement are also tested in routine FM201 and FM251.

ANS X3.9-1978 Reference: 4.1.2, 8.5

b. Special Considerations

There are 12 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.2.23 FM307 (Subset)

a. Features Tested

This routine tests intrinsic functions where the function type is real and the arguments are either integer or real. The function NINT is an exception and has an integer function type. The real or integer arguments consist of positive, negative and unsigned constants, variables and array element values. Each intrinsic function is tested with three or four different combinations of actual arguments. The arguments were chosen to test not only the various combinations of data usages, but also to test the range of argument and function values where appropriate. The intrinsic functions tested in this routine are REAL, ANINT, NINT, TAN, ASIN, ACOS, SINH, COSH.

ANS X3.9-1978 Reference: 15.3, 15.9.2, 15.9.3, 15.10.1

b. Special Considerations

There are 31 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.2.24 FM308 (Subset)

a. Features Tested

This routine tests intrinsic functions where the actual arguments consist of intrinsic function references, external function references, statement function references, and expressions involving operators. The argument and function types of all intrinsic functions tested are either integer or real. The INTRINSIC and EXTERNAL specification statements are specified in order to allow intrinsic and external functions to be used as actual arguments. The IMPLICIT statement and type-statement are tested to ensure that they do not change the type of an intrinsic function. The COMMON statement is used to pass data entities to an external function. The DATA statement is used to ensure that initially defined entities can be used as actual arguments. The EQUIVALENCE statement is used to equate a variable used as an actual argument. The intrinsic functions tested in this routine are INT, IFIX, FLOAT, REAL, AINT, ANINT, NINT, IABS, ABS, MOD, AMOD, ISIGN, SIGN, IDIM, DIM, MAXO, AMAXO, MAX1, AMIN1, MIN1, SQRT, EXP, ALOG, SIN, COS, TAN, ASIN, ACOS, ATAN, SINH, COSH AND TANH.

ANS X3.9-1978 Reference: 8.2, 8.3, 8.4, 8.5, 8.7, 8.8, 9, 15.3, 15.4, 15.5, 15.5.2, 15.9.2, 15.9.3, 15.10.1

b. Special Considerations

There are 32 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: FM309, FM310

X-Numbers: 02

D.2.25 FM311 (Subset)

a. Features Tested

This routine tests the use of the Fortran in-line statement function of types integer, real and logical. Specific features tested include:

(1) Real statement functions using real constants and variables in the expression and as actual arguments.

(2) Statement functions which require conversion of the expression to real and integer typing.

(3) The use of variables, array elements, external references, and initially defined entities in the expression.

(4) Various definitions and uses of the dummy arguments.

(5) Actual arguments consisting of expressions, intrinsic function references, and external function references.

(6) Confirming and overriding the typing of statement functions and dummy arguments.

(7) Use of statement functions and dummy arguments in the main program and in external function and subroutine subprograms.

ANS X3.9-1978 8.3, 8.4, 8.5, 8.7, 8.8, 9, 15.3, 15.4, 15.5, 15.6, 15.9.1, 15.9.2, 15.9.3

b. Special Considerations

There are 37 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: FM312, FM313, FM314, FM315, FM316

X-Numbers: 02

D.2.26 FM317 (Subset)

a. Features Tested

This routine tests subset level features of external function subprograms. Tests are designed to check the association of all permissible forms of actual arguments with variable, array and procedure name dummy arguments.

ANS X3.9-1978 Reference: 2.8, 5.1.2.2, 5.5, 8.1, 8.3, 8.4, 8.7, 8.8, 15.2, 15.3, 15.5, 15.6, 15.9

b. Special Considerations

There are 32 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: FM318, FM319, FM320, FM321, FM322, FM323, FM324, FM325, FM326, FM327

X-Numbers: 02

D.2.27 FM328 (Subset)

a. Features Tested

This routine tests subset level features of subroutine subprograms. Tests are designed to check the association of all permissible forms of actual arguments with variable, array and procedure name dummy arguments.

All data passed to the referenced subprograms are passed via arguments values, while all results returned to FM328 are returned via variables in named common.

ANS X3.9-1978 Reference: 2.8, 5.1.2.2, 5.5, 8.1, 8.3, 8.4, 8.7, 8.8, 15.2, 15.3, 15.5, 15.6, 15.9

b. Special Considerations

There are 22 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: FM329, FM330, FM331, FM332, FM333, FM334, FM335

X-Numbers: 02

D.2.28 FM351 (Subset)

a. Features Tested

FM351 contains tests for compound arithmetic expressions which necessitate the application of the rules for arithmetic operator precedence. These tests include ones which exercise the:

(1) Use of all arithmetic operator types in the same statement.

(2) Use of parentheses to override default precedence.

(3) Use of all classes of primary operands.

(4) Use of nested function references.

(5) Use of mixed data types.

ANS X3.9-1978 Reference: 6.1, 6.5, 6.6

b. Special Considerations

There are 25 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.2.29 FM352 (Subset)

a. Features Tested

FM352 contains tests for basic relational expressions involving operands of real data type. In each test, not only the relational expression is tested, but the trichotomy law of mathematical relationships is also tested (e.g., if A .LT. B, then A can not be .GT. B, and A can not be .EQ. B).

ANS X3.9-1978 Reference: 4.4, 6.3, 6.5

b. Special Considerations

There are 28 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.2.30 FM353 (Subset)

a. Features Tested

This segment tests the intrinsic functions INT and IFIX, which convert real arguments into integer values.

ANS X3.9-1978 Reference: 15.3, 15.10

b. Special Considerations

There are 14 visual tests. All results should be zero for the test segment to be successful.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.2.31 FM354 (Subset)

a. Features Tested

This program tests the intrinsic functions FLOAT and REAL, which convert integer expressions into real values.

ANS X3.9-1978 Reference: 15.3, 15.10

b. Special Considerations

There are 14 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.2.32 FM355 (Subset)

a. Features Tested

This program tests the intrinsic functions AINT, ANINT, and NINT. AINT truncates a real expression to a real number. ANINT rounds a real expression to the nearest whole number. NINT rounds a real expression to the nearest integer.

ANS X3.9-1978 Reference: 15.3, 15.10

b. Special Considerations

This program assumes the FLOAT function works properly.

There are 48 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.2.33 FM356 (Subset)

a. Features Tested

This program tests the intrinsic function ABS which returns the absolute value of any real expression, and the intrinsic function IABS which returns the absolute value of any integer expression. ABS returns a real number and IABS returns an integer.

ANS X3.9-1978 Reference: 15.3, 15.10

b. Special Considerations

There are 10 'PASS/FAIL' tests

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.2.34 FM357 (Subset)

a. Features Tested

This program tests the remaindering functions AMOD and MOD. AMOD accepts any two real expressions and returns a real value. MOD accepts any two integer expressions and returns an integer.

ANS X3.9-1978 Reference: 15.3, 15.10

b. Special Considerations

There are 22 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.2.35 FM359 (Subset)

a. Features Tested

This program tests the intrinsic functions SIGN, and ISIGN. SIGN accepts two real expressions and returns a real value. ISIGN accepts two integer expressions and returns an integer value.

ANS X3.9-1978 Reference: 15.3, 15.10

b. Special Considerations

There are 22 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.2.36 FM360 (Subset)

a. Features Tested

This program tests the intrinsic functions DIM and IDIM. DIM accepts two real expressions and returns the real value that is the positive difference of the two arguments. IDIM is similarly defined for integers.

ANS X3.9-1978 Reference: 15.3, 15.10

b. Special Considerations

There are 14 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02.

D.2.37 FM361 (Subset)

a. Features Tested

This program tests the intrinsic functions AMAX0 AMAX1, MAX0, AND MAX1. AMAX0 accepts a sequence of at least two integer arguments. AMAX1 accepts a sequence of at least two real arguments. MAX0 accepts a sequence of at least two integer arguments. MAX1 accepts a sequence of at least two real arguments. Each function returns the largest value from the respective sequence.

ANS X3.9-1978 Reference: 15.3, 15.10

b. Special Considerations

There are 48 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.2.38 FM362 (Subset)

a. Features Tested

This program tests the intrinsic functions AMIN0, AMIN1, MIN0, and MIN1. AMIN0 accepts a sequence of at least two integer arguments. AMIN1 accepts a sequence of at least two real arguments. MIN0 accepts a sequence of at least two integer arguments. MIN1 accepts a sequence of at least two real arguments. Each function returns the smallest value from the respective sequence.

ANS X3.9-1978 Reference: 15.3, 15.10

b. Special Considerations

There are 47 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.2.39 FM363 (Subset)

a. Features Tested

This program tests 23 intrinsic functions. The intrinsic functions should be able to accept as arguments any expression of the type specified in the intrinsic functions table (ANS REF - 15.10).

ANS X3.9-1978 Reference: 15.3, 15.10

b. Special Considerations

This program assumes the following segments are working: XINT, XREAL, XAINT, XABS, XAMOD, XSIGN, XDIM, XMAX, XMIN.

There are 14 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.2.40 FM364 (Subset)

a. Features Tested

This program tests intrinsic functions INT, IFIX, FLOAT, REAL, AINT, NINT, ANINT, ABS, IABS, AMOD, MOD, SIGN, ISIGN, IDIM, DIM, AMAX0, AMAX1, MAX0, MAX1, AMIN0, AMIN1, MIN0, MIN1, in mixed mode arithmetic expressions.

ANS X3.9-1978 Reference: 15.3, 15.10

b. Special Considerations

There are 14 'PASS/FAIL' tests.

Related Functions, Subroutines, or Programs: None

X-Numbers used: 02

D.2.41 FM368 (Subset)

a. Features Tested

This program tests the intrinsic function SQRT, which is the square root function applied to real arguments.

The arguments are non-negative real constants, variables and expressions. Two special cases are 0 and 1, since they are fixed points of the function.

ANS X3.9-1978 Reference: 15.3, Table 5.

b. Special Considerations

There are 13 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.2.42 FM369 (Subset)

a. Features Tested

This program tests the intrinsic function EXP which is the exponential function applied to real arguments.

ANS X3.9-1978 Reference: 15.3, Table 5.

b. Special Considerations

The computed values are compared with the correct values. The arguments used are real constants, variables, and expressions. The real constants used are zero, one, values close to one, and values close to 1/e. The results contain real values, representing the result of each test of EXP. The values returned are expected to have a maximum relative error no greater than 0.00005. The expected results are the number 'e' (approximately = 2.72) raised to the value of the argument.

There are 19 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.2.43 FM370 (Subset)

a. Features Tested

This segment tests the function ALOG, which is the natural logarithm applied to REAL arguments.

The arguments used are positive real constants, variables, and expressions. Special values to be tested are one, values close to e, values close to one and values close to zero.

ANS X3.9-1978 Reference: 15.3, Table 5

b. Special Considerations

There are 16 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.2.44 FM371 (Subset)

a. Features Tested

This segment tests the function ALOG10, which is the colon logarithm function applied to REAL arguments.

The arguments used are positive real constants, variables, and expressions. Special values to be tested are one, values close to 10, values close to one and values close to zero.

ANS X3.9-1978 Reference: 15.3, Table 5

b. Special Considerations

There are 15 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.2.45 FM372 (Subset)

a. Features Tested

This segment tests the function SIN, which is the sine function applied to real arguments.

ANS X3.9-1978 Reference: 15.3, Table 5

b. Special Considerations

The computed values are compared with the correct values. The arguments used are real constants, variables, and expressions. Special values to be tested are zero, values close to pi, values close to 2*pi, values close to pi/2, and close to 3*pi/2, and values of large magnitude. The expected results are the real sine of the argument. The results contain real values, representing the result of each test of SIN.

There are 17 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.2.46 FM373 (Subset)

a. Features Tested

This program tests the intrinsic function COS, which is the cosine function applied to real arguments.

ANS X3.9-1978 Reference: 15.3., Table 5.

b. Special Considerations

The arguments used are real constants, variables and expressions. Special constants used are zero, values near pi and near 2*pi, values near pi/2 and near 3*pi/2, and values of large magnitude. The results contain real values, representing the result of each test of COS.

There are 18 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.2.47 FM374 (Subset)

a. Features Tested

This program tests the intrinsic function TAN, which is the tangent function applied to real arguments.

ANS X3.9-1978 Reference: 15.3., Table 5.

b. Special Considerations.

Special constants used are zero, values near odd multiples of pi, values near multiples of pi/2, and values of large magnitude. The results contain real values, representing the result of each test of TAN.

There are 12 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.2.48 FM375 (Subset)

a. Features Tested

This segment tests the functions ASIN and ACOS, which are the arcsine function and the arccosine function applied to REAL arguments.

ANS X3.9-1978 Reference: 15.3, Table 5

b. Special Considerations

Special values to be tested are -1 and 1 to check principal values at endpoints, and comparisons of ASIN and ACOS to test their relationship. The expected results are the principal values of the arcsine and arccosine of the argument.

The first set of results contains real values, representing the results of each test of ASIN. The second set of results contains similar results for each test of ACOS.

There are 12 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.2.49 FM376 (Subset)

a. Features Tested

This program tests the intrinsic functions ATAN and ATAN2, which are, respectively, the arctangent and two-argument arctangent function applied to real arguments.

ANS X3.9-1978 Reference: 15.3., Table 5.

b. Special Considerations

Special constants tested are large argument values for ATAN, values of the form (0, positive) and (0, negative) for ATAN2, values near but not equal to zero for ATAN2, and comparison of ATAN and ATAN2. The first set of results contain real values, representing the result of each test of ATAN. The second set of results contains real values representing the result of each test of ATAN2. The program assumes the intrinsic function SQRT is working.

There are 13 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.2.50 FM377 (Subset)

a. Features Tested

This program tests the intrinsic functions SINH and COSH, which are, respectively, the hyperbolic sine and hyperbolic cosine functions applied to real arguments.

ANS X3.9-1978 Reference: 15.3, Table 5

b. Special Considerations

There are 15 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.2.51 FM378 (Subset)

a. Features Tested

This program tests the intrinsic function TANH, which is the hyperbolic tangent function applied to real arguments.

ANS X3.9-1978 Reference: 15.3., Table 5.

b. Special Considerations

The arguments used are real constants, variables, and expressions. Special constants tested are zero and values of large magnitude. The results contain real values, representing the result of each test of TANH. The expected results are the hyperbolic tangents of the arguments.

There are 9 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.2.52 FM379 (Subset)

a. Features Tested

This program tests intrinsic functions used in important trigonometric identities. The intrinsic functions tested are ALOG, EXP, SIN, COS, ASIN, ACOS, TAN, ATAN, ATAN2, SQRT, ALOG10, ALOG, SINK, TANH, and COSH. The tests are applicable for real arguments.

ANS X3.9-1978 Reference: 15.3., Table 5.

b. Special Considerations

The results contain real values, representing the result of evaluating each side of the trigonometric identity and then forming the difference between the two sides.

There are 10 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D..53 **FM401 (Subset)**

a. Features Tested

This routine tests for proper editing of logical data by the L edit descriptor of the Format specification. The L edit descriptor is first tested for proper editing on output by directing the edited result to a print file.

Next a file which is connected for sequential access is created with logical data fields and then repositioned to the first record in the file. The file is then read using the same edit descriptors as were used to create the file, and the internal data representation as a result of reading the logical data is checked.

The following L editing tests are made to see that:

(1) the value T and F is produced on output when the internal datum is true and false respectively,

(2) the value of the input list item is true and false when the input field is T and F respectively,

(3) the values .T, .F, T, F, .TRUE., .FALSE., .T, and .F are acceptable logical data forms for input fields,

(4) the input values T and F may be followed by additional characters in the field,

(5) the repeatable edit descriptor for L editing functions correctly,

(6) the fields containing logical data can be written using one L edit descriptor and read using a different form of the L edit descriptor.

ANS X3.9-1978 4.7, 13.1.1, 13.5.10

b. Special Considerations

This test routine should be executed using LUN I08 assigned to disk.

In addition to the 22 'PASS/FAIL' tests, the report should show 7 tests (tests 1 - 7) that must be visually verified.

Related Functions, Subroutines or Programs: None

X-Numbers: 02, 08

D.2.54 FM402 (Subset)

a. Features Tested

This routine tests the A(w) (w is the size of field in characters) edit descriptor of the format specification both with and without the optional w. The A edit descriptor is used with an input/output list item of type character. IF a field width w is not specified with the A edit descriptor, the number of characters in the field is the length of the character input/output list item. This routine first tests for proper editing of character data on output by directing the edited result to a print file.

Next an external file connected for sequential access is created with character data. Finally, the file is rewound and read with the A(w) edit descriptor and checked for proper editing on input.

ANS X3.9-1978 Reference: 3.1, 4.8, 8.4.2, 10.4, 13.5.11

b. Special Considerations

This test routine should be executed using LUN I09 assigned to disk.

In addition to the 20 'PASS/FAIL' tests, the report should show 14 tests (tests 1 - 14) that must be visually verified.

Related Functions, Subroutines or Programs: None

X-Numbers: 02, 09

D.2.55 FM403 (Subset)

a. Features Tested

This program tests simple formats and formatted data transfer statements in external sequential I/O. The tests in this program are performed on integer, real, and logical data types.

ANS X3.9-1978 Reference: 12.9.5.2, 13.3, 13.5.9

b. Special Considerations

There are 27 input records. The data can be selected off the population file or generated by the user (the data records are listed in the source program as comments).

There are 59 visual tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 01, 02

D.2.56 FM404 (Subset)

a. Features Tested

This segment tests simple formats and formatted data transfer statements in external sequential I/O. The tests in this segment are performed on character data types.

ANS X3.9-1978 Reference: 12.9.5.2, 13.3, 13.5.11

b. Special Considerations

There are 6 input records. The data can be selected off the population file or generated by the user (the data records are listed in the source program as comments).

There are 5 visual tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 01, 02

D.2.57 FM405 (Subset)

a. Features Tested

This program tests Internal file Input with data types Integer, Real, Logical, and Character, and with all legal Edit descriptors.

ANS X3.9-1978 Reference: 12.2.5

b. Special Considerations

There are 15 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.2.58 FM406 (Subset)

a. Features Tested

This program tests internal file output. Data types tested are integer, real, logical, and character. All the legal edit descriptors are used. Internal file output is produced by the WRITE statement. Variables are initialized with Predetermined values and then, via the WRITE statement, the values are moved from the data variable list to the internal file.

ANS X3.9-1978 Reference: 12.2.5

b. Special Considerations

Computed values with absolute value less than one are permitted to contain either a leading zero or a leading blank. Computed values requiring "E" format may be expressed as aE+nn or a+Onn, where a represents the mantissa, and nn represents the exponential power. A leading plus sign is also permitted for computed values which are positive.

There are 12 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.2.59 FM407 (Subset)

a. Features Tested

This segment tests unformatted READs and WRITEs to direct access files. Only INTEGER, REAL, LOGICAL, and CHARACTER data types are tested in this segment.

The primary purpose of this segment is to test that the direct access write and read operations positions the file to the proper record number. Subroutine SN408 is called to initialize four arrays of ten elements each. The arrays are of type INTEGER, REAL, LOGICAL, and CHARACTER. Ten unformatted records are written, in sequential order, to a direct access file. Each record contains an element from each of the arrays. Next these records are read, first in sequential order, and then in non-sequential order, and are compared to the values that were written. Finally, ten records will be written in non-sequential order, read back in both sequential and non-sequential order, and compared to the values that were written.

ANS X3.9-1978 Reference: 12.10.1

b. Special Considerations

A default number (number 24) is used, and must be changed using the X-Card (X-100 I10 = nn) if the unit is not capable of being opened as a direct, unformatted file.

There are 4 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: FM408

X-Numbers used: 02, 10

D.2.60 FM411 (Subset)

a. Features Tested

This routine tests for proper processing of unformatted records with a file connected for sequential access. Unformatted records may be read or written only by unformatted input/output statements. This routine tests several syntactical variations of the unformatted read and write statements as well as the file positioning statements BACKSPACE, ENDFILE and REWIND. In addition unformatted records may have both character and noncharacter data. This data is transferred without editing between the current record and entities specified by the input/output list items. This routine both reads and writes records containing data of logical, real and integer type with I/O list items represented as variable names, array element names and array names.

ANS X3.9-1978 Reference: 4.1, 12.1.2, 12.2.1, 12.2.4, 12.2.4.1, 12.3.3, 12.7.2, 12.8, 12.8.1, 12.8.2, 12.8.2.1, 12.8.2.2, 12.8.2.3, 12.9.5.1, 12.10.4, 12.10.4.1, 12.10.4.2, 12.10.4.3

b. Special Considerations

This test routine should be executed using LUN I04 assigned to disk.

There are 35 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02, 04

D.2.61 FM413 (Subset)

a. Features Tested

This routine tests for proper processing of unformatted records in files connected for direct access. For the subset language a file connected for direct access must have unformatted records. This routine first tests several syntactical variations of the read and write statements used in creating and accessing records of the file. The open statement is used to connect the file to a unit and establish its connection for direct access. The first series of tests create and access the records of the file in record number sequence and the last series of tests create and access records of the file in random order. Unformatted records may have both character and noncharacter data and this data is transferred without editing between the current record and the entities specified by the input/output list. This routine both reads and writes records containing the data types of integer, real and logical with I/O list items represented as variable names, array element names and array names.

ANS X3.9-1978 Reference: 4.1, 12.1.2, 12.2.4, 12.2.4.2, 12.3.3, 12.7.2, 12.8, 12.8.1, 12.8.2, 12.8.2.1, 12.8.2.2, 12.8.2.3, 12.9.5.1, 12.10.1

b. Special Considerations

This test routine should be executed using LUN I10 assigned to disk.

There are 35 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02, 10

D.3 Full Language Programs

D.3.1 FM500 (Full)

a. Features Tested

Test BLOCK DATA SUBPROGRAM features; including: IMPLICIT, PARAMETER, and SAVE.

FM500 is a main program that CALLs a subroutine, SN251. SN251 determines if the BLOCK DATA program, AN251 correctly initialized all of the variables.

ANS X3.9-1978 Reference: 5.1.1, 8.6, 8.7, 8.9, 9.1, 16.1, 16.2

b. Special Considerations

There are 28 'PASS/FAIL' tests and 9 visual tests.

Related Functions, Subroutines or Programs: FM501, FM502

X-Numbers: 02

D.3.2 FM503 (Full)

a. Features Tested

This program tests internal data forms of BLOCK DATA subprograms. A subroutine SN505 is called to check that the unnamed BLOCK DATA subprogram correctly initialized the variables.

ANS X3.9-1978 Reference: 16.1, 16.2

b. Special Considerations

There are 8 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: FM504, FM505

X-Numbers used: 02

D.3.3 FM506 (Full)

a. Features Tested

This program tests the ability of BLOCK DATA subprograms to correctly handle odd and even lengths of character variables juxtaposed within a single COMMON block without alignment problems. A subroutine (SN508) is called to check that the BLOCK DATA subprogram (AN507) correctly initialized the variables.

ANS X3.9-1978 Reference: 16.1, 16.2

b. Special Considerations

There are 4 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: FM5O7, FM508

X-Numbers used: 92

D.3.4 FM509 (Full)

a. Features Tested

This program tests subroutine subprograms and function subprograms with multiple entries, the ENTRY statement, substring names as arguments, and array element substrings as arguments.

ANS X3.9-1978 Reference: 15.6.1, 15.7, 15.9.2, 15.9.3.2, 15.9.3.3

b. Special Considerations

There are 16 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: FM510, FM511, FM512, FM513.

X-Numbers used: 0[illegible]

D.3.5 FM514 (Full)

a. Features Tested

This program tests the subroutine statement with asterisks as dummy arguments, and tests the use of alternate return specifiers as actual arguments for a subroutine.

ANS X3.9-1978 Reference: 15.6.1, 15.9.3.5, 15.6.2.3

b. Special Considerations

There ARE 2 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: FM515, FM516

X-Numbers used: 02

D.3.6 FM517 (Full)

a. Features Tested

This program tests the RETURN (e) statement, where "e" is an integer expression.

ANS X3.9-1978 Reference: 15.8.1, 15.8.3

b. Special Considerations

The first set of results contain integer values representing the result of each test of the RETURN (e) statement where "e" is an integer expression.

The second set of results contain integer values representing the result of each test of the RETURN (e) statement where "e" has a value which is out range.

There are 5 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: FM518, FM519

X-Numbers used: 02

D.3.7 FM520 (Full)

a. Features Tested

This program tests integer and real arithmetic expressions using only symbolic names of arithmetic constants. The PARAMETER statement is used to give the constants symbolic names.

ANS X3.9-1978 Reference: 6.1, 6.1.3, 6.6.3, 8.6

b. Special Considerations

There are 30 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 02

D.3.8 FM700 (Full)

a. Features Tested

This program tests the DATA statement with variable names, array names, array element names, substring names, and implied-DO lists.

The clist of the DATA statement may contain a symbolic name of a constant. If necessary, the clist constant is converted to the type of the nlist entity according to the rules for arithmetic conversion. Each subscript expression in an implied-DO list may contain implied-DO variables of the list that has the subscript expression within its range.

ANS X3.9-1978 Reference: 9.1, 9.2, 9.3

b. Special Considerations

There are 23 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.9 FM701 (Full)

a. Features Tested

This program tests values of dimension bounds in array declarators.

The bounds presented may be integer constants, integer variables, or integer arithmetic expressions. The value of either the lower or upper dimension bound may be positive, zero, or negative. However, the value of the upper bound must be greater than or equal to the value of the lower bound.

Each array declarator is either an actual array declarator or a dummy array declarator. A dummy array declarator may be either a constant array declarator, an adjustable array declarator, or an assumed-size array declarator. A dummy array declarator may appear only in a function or subroutine subprogram.

ANS X3.9-1978 Reference: 5.1.1.1, 5.1.1.2

b. Special Considerations

Each array is initialized in a DATA statement and may be passed as a dummy array to a subroutine, which may reassign values to certain selected array elements. For each array declarator, the value of a particular array element or a simple combination of two array elements is compared with the expected value.

Tests 1 through 16 and 26 through 35 use integer arrays and result in integer values. Tests 17 through 25 use character arrays and result in character values.

There are 35 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: FM702, FM703, FM704, FM705, FM706, FM707, FM708, FM709

X-Numbers used: 02

D.3.10 FM710 (Full)

a. Features Tested

This program tests subscript expressions, subscript values, character substring names, and substring expressions.

The subscript expressions may contain array element references and function references. Subscript values are tested to determine if array elements may be properly identified by array element name.

Either the leftmost character position or the rightmost character position of a character substring may be explicitly specified or implied. Substring expressions may contain array element references and function references.

ANS X3.9-1978 Reference: 5.4.2, 5.4.3, 5.7.1, 5.7.2

b. Special Considerations

The first set of results contains integers representing the result of each subscript expression test. Test 3 tests the ability of the processor to handle the range of an implied-DO list corresponding to a part of an array in a WRITE statement. The test is successful if the computed line agrees with the correct line displayed.

The second set of results contains character strings representing the result of each character substring test. This program assumes the intrinsic functions INT and IABS are working.

There are 18 'PASS/FAIL' tests and one visual test.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.11 FM711 (Full)

a. Features Tested

This program tests adjustable arrays and adjustable dimensions where the lower and/or upper bounds are arguments and/or in common.

Also tested is the use of array names. Specific tests involve the ability to use an array element name as a unit identifier for an internal file in an I/O statement, the ability to use an array name as a format identifier in an I/O statement, and the ability to use an array name in a SAVE statement.

ANS X3.9-1978 Reference: 5.5.1, 5.6

b. Special Considerations

The first set of results contains integer values representing the result of each test of the adjustable arrays.

Test 3 results in an integer value representing the result of a test of the ability to use an array element name as a unit identifier for an internal file in an I/O statement. Test 4 results in a character value representing the result of a test of the ability to use an array name as a format identifier in an I/O statement. Test 5 results in an integer value representing the result of a test of the ability to use an array name in a SAVE statement.

There are 5 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: FM712, FM713, FM714

X-Numbers used: 02

D.3.12 FM715 (Full)

a. Features Tested

This program tests character expressions and concatenation operations.

The simplest form of a character expression is a character constant, symbolic name of a character constant, character variable reference, character array element reference, character substring reference, or character function reference. More complicated character expressions may be formed by using one or more character operands together with concatenation operators and parentheses.

ANS X3.9-1978 Reference: 6.2, 6.2.1, 6.2.2, 6.2.2.2, 6.6.5

b. Special Considerations

The first set of results contains character or integer values representing the result of each test of character expressions. A character value is resulted if an assignment statement is used. An integer value is resulted if a relational expression in an IF statement is used.

The second set of results contains character or integer values, representing the result of each test of concatenation operations. A character value is resulted if an assignment statement is used. An integer value is resulted if a relational expression in an IF statement is used.

This program assumes the intrinsic function LEN is working.

There are 34 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: FM716, FM717

X-Numbers used: 02

D.3.13 FM718 (Full)

a. Features Tested

This program tests logical expressions and logical operators.

The simplest form of a logical expression is a logical constant, symbolic name of a logical constant, logical variable reference, logical array reference, logical function reference, or relational expression. More complicated logical expressions may be formed by using one or more logical operands together with logical operators and parentheses.

The precedence of the logical operators, from highest to lowest, is as follows: .NOT., .AND., .OR., EQV. or .NEQV..

ANS X3.9-1978 Reference: 6.4, 6.4.2, 6.4.3, 6.4.4

b. Special Considerations

The set of results contains logical or integer values, representing the result of each test of logical expressions. A logical value is resulted if an assignment statement is used. An integer value is resulted if a logical IF statement is used.

There are 29 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.14 FM719 (Full)

a. Features Tested

This program tests the DO statement using real DO variables and double precision DO variables. Also tested are active and inactive DO-loops, and DO statements with mixed integer, real, double precision, and complex variables.

ANS X3.9-1978 Reference: 11.10, 11.10.2, 11.10.3, 15.6, 15.7, 15.8

b. Special Considerations

This program assumes that the ENTRY statement for subroutines is supported.

There are 14 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: FM720 FM721

X-Numbers used: 02

D.3.15 FM722 (Full)

a. Features Tested

This routine tests the TYPE STATEMENT for declaring variables, arrays, constants, functions, and dummy procedures as type DOUBLE PRECISION and COMPLEX. Tests are made to check that explicit TYPE STATEMENTS override the implicit type integer designation by the first letter of the variable name.

ANS X3.9-1978 Reference: 4.1, 4.1.2, 8.4.1., 8.6

b. Special Considerations

This program assumes that the PARAMETER statement is supported.

There are 12 'PASS/FAIL' tests.

Related Function, Subroutines or Programs: FM723, FM724, FM725

X-Numbers used: 02

D.3.16 FM800 (Full)

a. Features Tested

This program tests the intrinsic function IDINT which converts double precision expressions into integer values.

ANS X3.9-1978 Reference: 15.3, 15.10

b. Special Considerations

There are 12 'PASS/FAIL' tests

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.17 FM801 (Full)

a. Features Tested

This program tests the intrinsic functions DINT, DNINT, and IDNINT. DINT truncates a double precision expression to an integer number. DNINT rounds a double precision expression to the nearest whole number. IDNINT rounds a double precision expression to the nearest integer.

ANS X3.9-1978 Reference: 15.3, 15.10

b. Special Considerations

This program assumes the FLOAT function works properly.

There are 45 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.18 FM802 (Full)

a. Features Tested

This program tests the intrinsic function DABS which returns the absolute value of a double precision expression. The number returned is again a double precision number.

ANS X3.9-1978 Reference: 15.3, 15.10

b. Special Considerations

There are 6 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.19 FM803 (Full)

a. Features Tested

This program tests the intrinsic function CABS which returns the absolute value of any complex expression. CABS returns a real number.

ANS X3.9-1978 Reference: 15.3, 15.10

b. Special Considerations

There are 9 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.20 FM804 (Full)

a. Features Tested

This program tests the intrinsic function DMOD the double precision remaindering function. The number returned is again a double precision number.

ANS X3.9-1978 Reference: 15.3, 15.10

b. Special Considerations

There are 11 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.21 FM805 (Full)

a. Features Tested

This program tests the intrinsic functions DDIM and DPROD. DDIM accepts two double precision expressions and returns the double precision value which is the positive difference of the two arguments. DPROD accepts two real expressions and returns the double precision value which is the product of the two arguments.

ANS X3.9-1978 Reference: 15.3, 15.10

b. Special Considerations

Because the standard does not indicate whether the result of DPROD(R1,R2) should yield the identical result as for DBLE(R1)*DBLE(R2) or REAL(R1)*REAL(R2), the maximum relative error for real numbers is adopted for the second set of results and is no greater than .00005.

There are 18 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.22 FM806 (Full)

a. Features Tested

This program tests the intrinsic function DMAX1 which accepts a sequence of at least two double precision arguments, and returns the largest value from this sequence.

ANS X3.9-1978 Reference: 15.3, 15.10

b. Special Considerations

The maximum number of arguments presented to DMAX1 will be five.

There are 12 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.23 FM807 (Full)

a. Features Tested

This program tests the intrinsic function DMIN1 which accepts a sequence of at least two double precision arguments, and returns the smallest value from this sequence.

ANS X3.9-1978 Reference: 15.3, 15.10

b. Special Considerations

There are 12 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.24 FM808 (Full)

a. Features Tested

This program tests the intrinsic function DBLE which converts any arithmetic expression into a double precision value. DBLE should be able to accept as an argument any real expression; including real constants and variables, and should be able to return the double precision value to any environment which allows a double precision data object.

ANS X3.9-1978 Reference: 15.3, 15.10

b. Special Considerations

There are 8 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.25 FM809 (Full)

a. Features Tested

Test the intrinsic functions CMPLX, AIMAG and CONJG. CMPLX accepts two real expressions and converts them to a complex number. AIMAG accepts a complex expression and returns its imaginary part. CONJG accepts a complex expression and converts it to its complex conjugate.

ANS X3.9-1978 Reference: 15.3, 15.10

b. Special Considerations

The output consists of three sets of results representing the tests for CMPLX, AIMAG, and CONJG, respectively.

There are 25 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.26 FM810 (Full)

a. Features Tested

This program tests the intrinsic functions IDINT, SNGL, DINT, DNINT, DABS, DMOD, DSIGN, DDIM, DPROD, DMAXI, DMINI, and DBLE in expressions involving integer, real, double precision, and mixed mode arithmetic.

ANS X3.9-1978 Reference: 15.3, 15.10, 6.1.4

b. Special Considerations

This program assumes the following segments are working: XINT, XREAL, XAINT, XABS, XAMOD, XSIGN, XDIM, XMAX, XMIN, YIDINT, YSNGL, YDINT, YDABS, YCABS, YDMOD, YDSIGN, YDMAX1, YDMIN1, YDBLE, YCONJG.

There are 10 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.27 FM811 (Full)

a. Features Tested

Test occurrences in mixed mode expressions of the following intrinsic functions of the full language: IDINT, SNGL, DINT, DNINT, DABS, CABS, DMOD, DSIGN, DDIM, DPROD, DMAX1, DMIN1, DBLE, CMPLX, AIMAG, and CONJG. The intrinsic functions should be able to accept as arguments any expression of the type specified in the intrinsic function table (ANS REF - 15.10).

ANS X3.9-1978 Reference: 15.3, 15.10, 6.1.4

b. Special Considerations

This program assumes the following segments are working: XINT, XREAL, XAINT, XABS, XAMOD, XSIGN, XDIM, XMAX, XMIN, YIDINT, YSNGL, YDINT, YDABS, YCABS, YDMOD, YDSIGN, YDMAX1, YDMIN1, YDBLE, YCONJG.

There are 10 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.28 FM812 (Full)

a. Features Tested

This segment tests the function DSQRT, which is the square root function applied to DOUBLE PRECISION arguments.

The arguments used are non-negative double precision constants, variables and expressions.

ANS X3.9-1978 Reference: 15.3, Table 5

b. Special Considerations

There are 13 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.29 FM813 (Full)

a. Features Tested

This program tests the intrinsic function CSQRT, which is the square root function applied to complex arguments.

The arguments used are complex constants, variables and expressions. Special complex constants used are zero, positive real numbers, negative real numbers and purely imaginary numbers.

ANS X3.9-1978 Reference: 15.3., Table 5.

b. Special Considerations

There are 13 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.30 FM814 (Full)

a. Features Tested

This program tests the intrinsic function DEXP which is the exponential function applied to double precision arguments.

The arguments used are double precision constants, variables and expressions. The double precision constants used are zero, one, values close to one, and values close to 1/e.

ANS X3.9-1978 Reference: 15.3, Table 5.

b. Special Considerations

The expected results are the number 'e' (approximately = 2.72) raised to the value of the argument.

There are 19 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.31 FM815 (Full)

a. Features Tested

This program tests the intrinsic function CEXP, which is the exponential function applied to complex arguments.

The arguments used are complex constants, variables, and expressions. Special complex constants used are zero: purely real numbers, and purely imaginary numbers. The results contain complex values, consisting of a real and an imaginary part representing the result of each test of CEXP.

ANS X3.9-1978 Reference: 15.3., Table 5.

b. Special Considerations

This program assumes that the intrinsic functions AIMAG and CABS are working.

There are 9 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.32 FM816 (Full)

a. Features Tested

This program tests the intrinsic function DLOG which is the natural logarithm function applied to double precision arguments.

The arguments used are positive double precision constants, variables, and expressions. Special values to be tested are one, values close to e, values close to one and values close to zero.

ANS X3.9-1978 Reference: 15.3.

b. Special Considerations

There are 16 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.33 FM817 (Full)

a. Features Tested

This program tests the intrinsic function CLOG, which is the natural logarithm function applied to complex arguments.

The arguments used are complex constants, variables, and expressions. Special complex constants used are positive real numbers, and negative real numbers.

ANS X3.9-1978 Reference: 15.3., Table 5.

b. Special Considerations

There are 11 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.34 FM818 (Full)

a. Features Tested

This segment tests the function DLOG10, which is the common logarithm function applied to double precision arguments.

ANS X3.9-1978 Reference: 15.3, Table 5

b. Special Considerations

The computed values are compared with the correct values. The arguments used are positive double precision constants, variables, and expressions. Special values to be tested are one, values close to 10, values close to one and values close to zero. The expected results are the common logarithms of the arguments. The results contain double precision values, representing the result of each test of DLOG10.

There are 15 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.35 FM819 (Full)

a. Features Tested

This segment tests the function DSIN, which is the sine function applied to double precision arguments.

ANS X3.9-1978 Reference: 15.3, Table 5

b. Special Considerations

The computed values are compared with the correct values. The arguments used are double precision constants, variables, and expressions. Special values to be tested are zero, values close to pi, values close to 2*pi, values close to pi/2, and close to 3*pi/2, and values of large magnitude. The expected results are the double precision sine of the argument. The results contain double precision values, representing the result of each test of DSIN.

There are 19 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.36 FM820 (Full)

a. Features Tested

This program tests the intrinsic functions CSIN and CCOS, which are the sine function and the cosine function applied to complex arguments.

ANS X3.9-1978 Reference: 15.3., Table 5.

b. Special Considerations

The computed values are compared with the correct values. The arguments used are complex constants, variables, and expressions. Special complex constants used are zero, values on the real line, and values with a zero real part. The first set of results contain complex values, consisting of a real and an imaginary part, representing the result of each test of CSIN. The second set of results also contain complex values with a real and imaginary part, representing the result of each test of CCOS. All values returned are expected to have a maximum relative error no greater than 0.00005 for each part. The expected results are the complex sine and cosine of the arguments. It is assumed that the intrinsic function CABS is working.

There are 18 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.37 FM821 (Full)

a. Features Tested

This program tests the intrinsic function DCOS, which is the cosine function applied to double precision arguments.

ANS X3.9-1978 Reference: 15.3., Table 5.

b. Special Considerations

The computed values are compared with the correct values. The arguments used are double precision constants, variables, and expressions. Special double precision constants used are zero, values near pi and near 2*pi, values near pi/2 and near 3*pi/2, and values of large magnitude. The results contain double precision values, representing the result of each test of DCOS. The values returned are expected to have a maximum relative error no greater than 0.0000000005. The expected results are the double precision cosines of the arguments.

There are 19 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.38 FM822 (Full)

a. Features Tested

This program tests the intrinsic function DTAN, which is the tangent function applied to double precision arguments.

ANS X3.9-1978 Reference: 15.3., Table 5.

b. Special Considerations

The computed values are compared with the correct values. The arguments used are double precision constants, variables, and expressions. Special constants used are zero, values near odd multiples of pi, values near multiples of pi/2 ,and values of large magnitude. The results contain double precision values, representing the result of each test of DTAN. The expected results are the tangents of the arguments.

There are 14 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.39 FM823 (Full)

a. Features Tested

This segment tests the functions DASIN and DACOS, which are the arcsine function and the arccosine function applied to double precision arguments.

ANS X3.9-1978 Reference: 15.3, Table 5

b. Special Considerations

The tests will present the DASIN and DACOS functions with double precision constants, variables and expressions as arguments. The magnitude of these values must not exceed 1. Special values to be tested are -1 and 1 to check principal values at endpoints, and comparisons of DASIN and DACOS to test their relationship. The expected results are the principal values of the arcsine and arccosine of the argument. The computed values are compared with the expected values. The first set of results contains double precision values, representing the results of each test of DASIN. The second set of results contains similar results for each test of DACOS.

There are 12 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.40 FM824 (Full)

a. Features Tested

This program tests the intrinsic functions DATAN and DATAN2, which are, respectively, the arctangent and two-argument arctangent function applied to double precision arguments.

ANS X3.9-1978 Reference: 15.3., Table 5.

b. Special Considerations

The arguments used are double precision constants, variables, and expressions. Special double precision constants tested are large argument values for DATAN, values of the form (0, positive) and (0, negative) for DATAN2, values near but not equal to zero for DATAN2, and comparison of DATAN and DATAN2. The first set of results contain double precision values, representing the result of each test of DATAN. The second set of results also contains double precision values, representing the result of each test of DATAN2. The expected results are the arctangent of the arguments. The program assumes the intrinsic function DSQRT is working.

There are 13 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.41 FM825 (Full)

a. Features Tested

This program tests the intrinsic functions DSINH and DCOSH, which are, respectively, the hyperbolic sine and hyperbolic cosine functions applied to double precision arguments.

ANS X3.9-1978 Reference: 15.3, Table 5

b. Special Considerations

The first set of results contains double precision values, representing the results of each test of DSINH. The second set of results contains similar results for each test of DCOSH. The expected results are, respectively, the hyperbolic sines and hyperbolic cosines applied to double precision arguments.

There are 16 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.42 FM826 (Full)

a. Features Tested

This program tests the intrinsic function DTANH, which is the hyperbolic tangent function applied to double precision arguments.

ANS X3.9-1978 Reference: 15.3., Table 5.

b. Special Considerations

The arguments used are double precision constants, variables, and expressions. Special double precision constants tested are zero and values of large magnitude. The results contain double precision values, representing the result of each test of DTANH. The expected results are the hyperbolic tangents of the arguments.

There are 9 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.43 FM827 (Full)

a. Features Tested

This program tests intrinsic functions used in important trigonometric identities. The intrinsic functions tested are DLOG, DSIN, DCOS, DASIN, DACOS, DTAN, DATAN, DATAN2, DSQRT, DLOG10, DLOG, DSINH, DEXP, DTANH, and DCOSH. The tests are applicable for double precision arguments.

ANS X3.9-1978 Reference: 15.3., Table 5.

b. Special Considerations

The arguments used are double precision constants, variables, and expressions. The results contain double precision values, representing the result of evaluating each side of the trigonometric identity and then forming the difference between the two sides.

There are 10 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.44 FM828 (Full)

a. Features Tested

This program tests intrinsic functions used in important trigonometric identities. The intrinsic functions tested are CSQRT, ATMAG, CABS, CONJG, ATAN2, CEXP, CLOG, EXP, AMOD, CMPLX, COS, SIN, CCOS, and COS. The tests are applicable for complex arguments.

ANS X3.9-1978 Reference: 15.3., Table 5.

b. Special Considerations

The results contain complex values, consisting of a real and/or an imaginary part, representing the result of evaluating each side of the trigonometric identity and then forming the difference between the two sides.

There are 9 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.45 FM829 (Full)

a. Features Tested

This program tests the type conversion generic functions INT, REAL, DBLE, and CMPLX. The data types of the arguments are chosen so as not to duplicate previous testing.

ANS X3.9-1978 Reference: 15.3, Table 5

b. Special Considerations

The first and fifth sets of results contain integer values, representing the results of each test of INT. The second and sixth sets of results contain real values, representing the results of each test of REAL. The third and seventh sets of results contain double precision values, representing the results of each test of DBLE. DBLE converts its argument to double precision type. If the argument is double precision as in Test 15, the returned double precision value is the same as the argument itself and is expected to have a maximum relative error no greater than .0000000005. If the argument is an integer or real as in Tests 13, 14, 30, and 31, DBLE results in a double precision value with as much precision of the significant part of the argument as possible. The returned values are expected to have a maximum relative error no greater than .00005. If the argument is complex as in Tests 16 and 32, DBLE results in a double precision value with as much precision of the significant part of the real portion of the complex argument as possible.

The returned values are expected to have a maximum relative error no greater than .00005. The fourth and eighth sets of results contain complex values, with a real and an imaginary part, representing the results of each test of CMPLX. The complex values returned are expected to have a maximum relative error no greater than .00005 for each part. The expected results are the type conversions of the arguments.

There are 35 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.46 FM830 (Full)

a. Features Tested

This program tests arithmetic expressions containing generic functions. The generic functions tested are AINT, ANINT, NINT, SQRT, EXP, LOG, and LOG10. The data types of the arguments are chosen so as not to duplicate previous testing.

ANS X3.9-1978 Reference: 15.3, Table 5

b. Special Considerations

The computed values are compared with the expected values. The first result is an integer value, representing the result of a test of NINT. The second set of results contains double precision values, representing the results of tests with the functions AINT, ANINT, SQRT, EXP, LOG, LOG10, and NINT. The double precision values returned are expected to have a maximum relative error no greater than .0000000005. The third set of results contains complex values, with a real and an imaginary part, representing the results of tests with the functions SQRT, EXP, LOG, AINT, and NINT. The complex values returned are expected to have a maximum relative error no greater than .00005 for each part. The expected results are the arithmetic evaluations of the expressions, given the arguments supplied.

There are 9 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.47 FM831 (Full)

a. Features Tested

This program tests arithmetic expressions containing generic functions. The generic functions tested are ABS, MOD, SIGN, SIN, COS, TAN, SINH, and TANH. The data types of the arguments are chosen so as not to duplicate previous testing.

ANS X3.9-1978 Reference: 15.3, Table 5

b. Special Considerations

The first result is an integer value, representing the result of a test of ABS and SIGN. The second set of results contains double precision values, representing the results of tests with the functions ABS, MOD, SIGN, SIN, COS, TAN, SINH, COSH, and TANH. The third set of results contains complex values, with a real and an imaginary part, representing the results of tests with the functions ABS, MOD, SIN, COS, and TAN.

There are 12 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.48 FM832 (Full)

a. Features Tested

This program tests generic functions using as arguments real values and their double precision counterparts. The generic functions tested are SQRT, EXP, LOG, LOG10, COS, SINH, TANH, ASIN, ATAN and ATAN2.

ANS X3.9-1978 Reference: 15.3, Table 5

b. Special Considerations

The expected results are the functions evaluated with single precision and then double precision accuracy. Both sets of results are printed in double precision notation.

There are 20 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.49 FM833 (Full)

a. Features Tested

This program tests for equality between generic functions and their specific function counterparts. The generic functions tested are SIGN, MAX, EXP, TANH, ASIN, ANINT, MOD, ABS, SQRT, LOG, SIN.

ANS X3.9-1978 Reference: 15.3, Table 5

b. Special Considerations

The first result is an integer value, representing the result of a test with SIGN. The second result is a real value, representing the result of a test with MAX. The third set of results contain double precision values, representing the results of each test of the generic function. The fourth set of results contain complex values, with a real and an imaginary part, representing the results of each test of the generic function.

There are 11 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.50 FM834 (Full)

a. Features Tested

To test the handling of generic functions used as arguments to other generic functions. The generic functions tested are ABS, MIN, MOD, SIGN, COS, SQRT, MAX, LOG, LOG10, EXP, SINH, TAN, ATAN, CMPLX.

ANS X3.9-1978 Reference: 15.3, 15.10

b. Special Considerations

The results of Tests 1 and 2 contain integers and represent the result of each test of the generic functions used in integer expressions. The results of Tests 3 and 4 contain real numbers and represent the result of each test of the generic functions used in real expressions. The results of Tests 5 and 6 contain double precision numbers and represent the result of each test of the generic functions used in double precision expressions. The result of Test 7 contains a complex number and represents the result of a test of the generic functions used in the complex expression in Test 7.

There are 7 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.51 FM900 (Full)

a. Features Tested

This segment tests simple formats and formatted data transfer statements in external sequential I/O. The tests in this segment are performed on double precision and complex data types.

Formatted reads are used to transfer values from the systems input file to local variables. Formatted writes are used to transfer values from local variables to the systems output file.

ANS X3.9-1978 Reference: 12.9.5, 13.3, and 13.5

b. Special Considerations

There are 17 input records. The data can be selected off the population file or generated by the user (the data records are listed in the source program as comments).

There are 36 visual tests. The output consists of pairs of lines. The first line in each group is the actual result. The second line is produced by the H-edit descriptor. This is the expected result. A test is successful if the first line in the pair agrees with the second line. The optional zero to the left of the decimal point and the form of the exponent may differ, as permitted in the standard.

Related Functions, Subroutines or Programs: None

X-Numbers used: 01 and 02

D.3.52 FM901 (Full)

a. Features Tested

This segment tests simple formats and formatted data transfer statements in external sequential I/O. The tests in this segment are performed on character data types. Read and write on substrings are included in this segment.

ANS X3.9-1978 Reference: 12.9.5.2, 13.3, and 13.5.11

b. Special Considerations

There are 5 input records. The data can be selected off the population file or generated by the user (the data records are listed in the source program as comments).

There are 4 visual tests.

Related Functions, Subroutines or Programs: None.

X-Numbers used: 01 and 02.

D.3.53 FM903 (Full)

a. Features Tested

This segment tests additional features of READ and WRITE statements, formatted records, and format statements for double precision and complex data types. This segment also tests all forms of character expressions as format specifiers.

Formatted reads are used to transfer values from the systems input file to local variables. Formatted writes are used to transfer values from local variables to systems output file. The external subroutine SN904 is used to test that a character constant may be passed as a parameter to a subroutine, and used as a format specifier.

ANS X3.9-1978 Reference: 12.9.5.2; 13.1; 13.5

b. Special Considerations

There are 14 input records. The data can be selected off the population file or generated by the user (the data records are listed in the source program as comments)

There are 13 visual tests.

Related Functions, Subroutines or Programs: FM904

X-Numbers used: 01, 02

D.3.54 FM90[illegible] (Full)

a. Features Tested

This segment tests list-directed output for integer, real, logical, and character data types. A list-directed WRITE statement is used to move values from local variables to the systems output file. The variables are initialized with predetermined data.

ANS X3.9-1978 Reference: 13.6, 12.4

b. Special Considerations

For each test, two lines are printed. The first line in each pair is the actual results obtained by using a list-directed WRITE. The second line is the expected results obtained from a formatted WRITE statement, using the H-edit descriptor. The actual results are compared visually with the expected results in the printed output. The particular form of the actual results produced depends on the processor being tested. Column spacing and line breaks are processor dependent. For real numbers, either E or F format may be used and the number of decimal places printed is processor dependent. A test is successful if both lines in the pair agree, taking into account possible variations described above. If a subset processor contains list-directed output for integer, real, and logical data types, this segment may be used in a subset test to insure the proper behavior of the extension.

There are 10 visual tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.55 FM906 (Full)

a. Features Tested

This segment test list-directed input for double precision and complex data types. A list-directed READ statement is used to move values from the systems input file to local variables.

ANS X3.9-1978 Reference: 13.6, 12.4

b. Special Considerations

There are 12 input records. The data can be selected off the population file or generated by the user (the data records are listed in the source program as comments).

There are 28 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 01, 02

D.3.56 FM907 (Full)

a. Features Tested

This segment tests list-directed output for double precision and complex data types. This segment also tests list-directed output of expressions. A list-directed WRITE statement is used to move values from local variables to the systems output file. The variables are initialized with predetermined data.

ANS X3.9-1978 Reference: 13.6, 12.4

b. Special Considerations

For each test, two lines are printed. The first line in each pair is the actual results obtained by using a list-directed WRITE. The second line is the expected results obtained from a formatted WRITE statement, using the H-edit descriptor. The actual results are compared visually with the expected results in the printed output. The particular form of the actual results produced depends on the processor being tested. Column spacing and line breaks are processor dependent. For double precision and complex numbers, either E or F format may be used and the number of decimal places printed is processor dependent. A test is successful if both lines in the pair agree, taking into account possible variations described above.

There are 8 visual tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.57 FM908 (Full)

a. Features Tested

This segment tests Internal file Input for Full language concepts, with data types Double Precision, Complex, Integer, Real, Logical, and Character, and with a sampling of Edit descriptors, particularly those that are not allowed in the Subset. The Internal file forms are both character arrays and strings. For Input, the READ statement is used, moving data from the Internal files to a list of data variables. The file is initialized to a preset character sequence. After the READ, the values in the data variables are compared to the expected results.

ANS X3.9-1978 Reference: 12.2.5

b. Special Considerations

There are 54 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.58 FM909 (Full)

a. Features Tested

This program tests internal file output for Full language concepts. Data types tested are double precision, complex, integer, real, logical, and character. A sampling of edit descriptors are used, particularly those not allowed in the Subset. The internal file forms used are both character arrays and strings. Internal file output is produced by the WRITE statement. Variables are initialized with predetermined values and then, via the WRITE statement, the values are moved from the data variable list to the internal file.

ANS X3.9-1978 Reference: 12.2.5

b. Special Considerations

Computed values with absolute value less than one are permitted to contain either a leading zero or a leading blank. Computed values requiring "E" format may be expressed as Ae+nn or a+0nn, where a represents the mantissa, and nn represents the exponential power. Computed values requiring "D" format may be expressed as Ad+nn, Ae+nn, or a+0nn. A leading plus sign is also permitted for computed values which are positive. These additional options have been incorporated in the test code if the number of possible combinations of results is ten or less. Where the number of possible results exceeds ten, the computed value is compared with only one value, and, if the test fails, the number of possible results are printed. The results contain character values, representing the result of testing character data types.

There are 27 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02

D.3.59 FM910 (Full)

a. Features Tested

This segment tests unformatted READs and WRITEs with both direct and sequential accesses to the same file. This segment tests COMPLEX and DOUBLE PRECISION data types, in addition to INTEGER, REAL, LOGICAL, and CHARACTER. The INQUIRE statement is also tested to ensure that the RECL and NEXTREC specifiers are handled properly.

The primary purpose of this segment is to test if READ and WRITE operations with direct and sequential access correctly position the file to the proper record number. Subroutine SN911 is called to initialize six arrays of ten elements each. The arrays are of type INTEGER, REAL, LOGICAL, CHARACTER, COMPLEX, and DOUBLE PRECISION. A file is first opened for direct access, and ten records are written in sequential order. Each record contains an element of each of the arrays. The file is INQUIRE'd to see if sequential access is allowed. If so, the file is reopened for sequential access, and ten records read in and compared to the values that were written. The file is reopened for direct access, and the records read in both sequential and non-sequential order. The file is then reopened as a scratch file, and the tests repeated.

ANS X3.9-1978 Reference: 12.5

b. Special Considerations

The default unit numbers (24 and 25) are used, and must be changed if the units are not capable of being opened as a direct, unformatted files.

There are 6 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: FM911

X-Numbers used: 02, 10, 11, 20

D.3.60 FM912 (Full)

a. Features Tested

This segment tests formatted READs and WRITEs with both direct and sequential accesses to the same file. This segment tests DOUBLE PRECISION, INTEGER, REAL, LOGICAL, and CHARACTER. data types. The INQUIRE statement is also tested to ensure that the RECL and NEXTREC specifiers are handled properly.

The primary purpose of this segment is to test if READ and WRITE operations with direct and sequential access correctly position the file to the proper record number. Subroutine SN913 is called to initialize values for the arrays.

Each record contains an element of each of the arrays. The file is INQUIRE'd to see if sequential access is allowed. If so, the file is reopened for sequential access, and five records read in and compared to the values that were written. The file is reopened for direct access, and the records read in both sequential and non-sequential order.

ANS X3.9-1978 Reference: 12.5

b. Special Considerations

This test routine should be executed using LUN I13 assigned to disk.

There are 26 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: FM913

X-Numbers used: 02, 13, 20

D.3.61 FM914 (Full)

a. Features Tested

This segment tests INQUIRE by unit on a sequential, formatted file. The INQUIRE specifiers used are UNIT, EXIST, OPENED, NUMBER, ACCESS, SEQUENTIAL, FORM, FORMATTED, BLANK, ERR, IOSTAT.

An OPEN statement is used to connect a unit for sequential, formatted access. Next, an INQUIRE is performed, and the returned specifier values are compared to the expected values. Finally, a CLOSE statement with STATUS='DELETE' is performed, so that the unit may be reused in another test.

ANS X3.9-1978 Reference: 12.10.3

b. Special Considerations

A default unit number (number 14) is used, and must be changed if the unit is not capable of being opened as a sequential, formatted file.

Comparisons of the actual vs the expected values for the UNIT, EXIST, OPENED, NUMBER, ACCESS, SEQUENTIAL, FORM, FORMATTED, BLANK, ERR and IOSTAT specifiers of the INQUIRE are made and the results reported on the report as one test. If any one of the comparisons fails, all the specifiers that appear in the failed INQUIRE statement are printed, first with the actual values, and then with the expected values.

There is 1 'PASS/FAIL' test.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02, 08

D.3.62 FM915 (Full)

a. Features Tested

This segment tests INQUIRE by unit on a sequential, unformatted file. The INQUIRE specifiers used are UNIT, EXIST, OPENED, NUMBER, ACCESS, SEQUENTIAL, FORM, UNFORMATTED, ERR, and IOSTAT.

An OPEN statement is used to connect a unit for sequential, unformatted access. Next, an INQUIRE is performed, and the returned specifier values are compared to the expected values. Then, a record is written to the file, and another INQUIRE is performed and tested. The file is then rewound and read, and a third INQUIRE is performed and tested. Finally, a CLOSE statement with STATUS='DELETE' is performed, so that the unit may be reused in another test.

ANS X3.9-1978 Reference: 12.10.3

b. Special Considerations

A default unit number (number 14) is used, and must be changed if the unit is not capable of being opened as a sequential, unformatted file.

There are 3 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02, 05

D.3.63 FM916 (Full)

a. Features Tested

This segment tests INQUIRE by unit on a direct-access, formatted file. The INQUIRE specifiers used are UNIT, EXIST, OPENED, NUMBER, ACCESS, DIRECT, RECL, NEXTREC, FORM, FORMATTED, BLANK, ERR, and IOSTAT.

An OPEN statement is used to connect a unit for a direct-access, formatted file. Next, an INQUIRE is performed, and the returned specifier values are compared to the expected values. Finally, a CLOSE statement with STATUS='DELETE' is performed, so that the unit may be reused in another test.

ANS X3.9-1978 Reference: 12.10.3

b. Special Considerations

A default unit number (number 14) is used, and must be changed if the unit is not capable of being opened as a direct-access, formatted file.

There is 1 'PASS/FAIL' test.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02, 14

D.3.64 FM917 (Full)

a. Features Tested

This segment tests INQUIRE by unit on a direct-access, unformatted file. The INQUIRE specifiers used are UNIT, EXIST, OPENED, NUMBER, ACCESS, DIRECT, RECL, NEXTREC, FORM, UNFORMATTED, ERR, and IOSTAT.

An OPEN statement is used to connect a unit for a direct-access, unformatted file. Next, an INQUIRE is performed, and the returned specifier values are compared to the expected values. Then, a record is written to the file, and another INQUIRE is performed and tested. The record that was written is then read from the file, and a third INQUIRE is performed and tested. Finally, a CLOSE statement with STATUS='DELETE' is performed, so that the unit may be reused in another test.

ANS X3.9-1978 Reference: 12.10.3

b. Special Considerations

A default unit number (number 14) is used, and must be changed if the unit is not capable of being opened as a direct-access, unformatted file.

There are 3 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02, 12

D.3.65 FM919 (Full)

a. Features Tested

This segment tests INQUIRE by file on a sequential, formatted file. The INQUIRE specifiers used are FILE, EXIST, OPENED, NUMBER, ACCESS, SEQUENTIAL, FORM, FORMATTED, BLANK, ERR, and IOSTAT.

An OPEN statement is used to open a sequential, formatted file. Next, an INQUIRE is performed, and the returned specifier values are compared to the expected values. Finally,

a CLOSE statement with STATUS='DELETE' is performed so that the file may be reused in another test.

ANS X3.9-1978 Reference: 12.10.3

b. Special Considerations

A default unit number (number 14) is used, and must be changed if the unit is not capable of being opened as a sequential, formatted file.

There is 1 'PASS/FAIL' test.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02, 09, 19

D.3.66 FM920 (Full)

a. Features Tested

This segment tests INQUIRE by file on a sequential, unformatted file. The INQUIRE specifiers used are FILE, EXIST, OPENED, NUMBER, ACCESS, SEQUENTIAL, FORM, UNFORMATTED, ERR, and IOSTAT.

An OPEN statement is used to open a sequential, unformatted file. Next, an INQUIRE is performed, and the returned specifier values are compared to the expected values. A record is then written to the file, and an INQUIRE performed and tested. Next, the file is rewound, and the record just written is read from the file. Again, an INQUIRE is performed and tested. Finally, a CLOSE statement with STATUS='DELETE' is performed so that the file may be reused in another test.

ANS X3.9-1978 Reference: 12.10.3

b. Special Considerations

A default unit number (number 14) is used, and must be changed if the unit is not capable of being opened as a sequential, formatted file.

There are 3 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02, 05, 19

D.3.67 FM921 (Full)

a. Features Tested

This segment tests INQUIRE by file on a direct-access, unformatted file. The INQUIRE specifiers used are FILE, EXIST, OPENED, NUMBER, ACCESS, DIRECT, RECL, NEXTREC, FORM, UNFORMATTED, ERR, and IOSTAT.

An OPEN statement is used to open a direct-access, unformatted file. Next, an INQUIRE is performed, and the returned specifier values are compared to the expected values. A record is then written to the file, and another INQUIRE performed and tested. The record just written is then read from the file, and a third INQUIRE is performed and tested. Finally, a CLOSE statement with STATUS='DELETE' is performed so that the file may be reused in another test.

ANS X3.9-1978 Reference: 12.10.3

b. Special Considerations

A default unit number (number 24) is used, and must be changed if the unit is not capable of being opened as a direct-access, unformatted file.

There are 3 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02, 10, 20

D.3.68 FM922 (Full)

a. Features Tested

This program tests INQUIRE by file on a file that is not connected to a unit. The INQUIRE specifiers used are FILE, IOSTAT, EXIST, OPENED, SEQUENTIAL, FORMATTED and ERR.

First, an OPEN statement is used to connect a formatted sequential file to a unit. The file is written, followed by an ENDFILE statement for the file and the file rewound. Then a CLOSE statement with STATUS='KEEP' is performed to ensure that the file exists. Next, an INQUIRE statement is executed and the tests made. The file is then opened and closed with DELETE.

ANS X3.9-1978 Reference: 12.10.3

b. Special Considerations

The file should exist but not be connected.

A default unit number (number 14) and file name (composed of 15 characters - 8 spaces followed by the letters "SEQFILE") are assigned. The unit number and file name may be changed if not valid in an OPEN statement connecting a file for sequential, formatted access.

There is 1 'PASS/FAIL' test.

Related Functions, Subroutines or Programs: None

X-Numbers used: 02, 15,

D.3.69 FM923 (Full)

a. Features Tested

This program tests list-directed input for integer, real, logical, and character data types.

ANS X3.9-1978 Reference: 12.4, 13.6

b. Special Considerations

There are 34 input records. The data can be selected off the population file or generated by the user (the data records are listed in the source program as comments).

There are 28 'PASS/FAIL' tests.

Related Functions, Subroutines or Programs: None

X-Numbers: 01, 02

APPENDIX E
SUMMARY OF FEXEC CONTROL INPUTS

E. SUMMARY OF FEXEC CONTROL INPUTS

CARD	FUNCTION
*ENVIR S	Select all subset audit routines
*ENVIR F	Select both full and subset audit routines
PFMnnn	Select a program
PFMnnn,PFMnnn	Select a series of audit routines
MFMnnn	Exclude a program
MFMnnn,MFMnnn	Exclude a series of programs
*DATE	Override date supplied by FEXEC
*COMPILER	Override compiler ID supplied by FEXEC
*PROJECT	Override project code supplied by FEXEC
*OPT1 D	Optional code selection
*TPF Y	Include all TPFs
*TPF N	Exclude TPFs
*LIST	List FEXEC control and update information
I-kk	(JCL) run initialization card
B-kk	(JCL) card preceding program
D-01nn	(JCL) card preceding data
D-02nn	(JCL) card following data (if required)
E-kk	(JCL) card following program
E-kk DATA****	Include data for source programs
T-kk	(JCL) run termination card
X-kk I02 = 6	X-card (assign logical unit number)
X-19 CHARACTER*10 CSEQ	| Override default file name
X-191 CSEQ = ' CSEQ'	| used in the OPEN and INQUIRE
X-20 CHARACTER*5 CDIR	| statements
X-201 CDIR = ' CDIR'	|
*END-MONITOR	Terminate Monitor section input
*BEGIN-UPDATE	Begin Update section input
*START,FMnnn	Begin updates for program FMnnn
=iiiiiT	Delete Test mmmm
=iiiiiTjjjjj	Delete a test series
=nnnnn	Insert new source code after line iiiii
=nnnnn,nnnnn	Delete a card image
=nnnnn,mmmmm	Replace a series of card images
=nnnnnC	Change line to comment
=nnnnnCmmmmm	Change series of lines to comments
=nnnnn,mmmmm	Delete a series of lines
*END-UPDATE	End Update section
*END-INPUT	End of all input