SC26-3988-0
File No. S370-25

Program Product

VS FORTRAN Application Programming: System Services Reference Supplement

**Program Numbers 5748-FO3 (Compiler and Library)
5748-LM3 (Library Only)**

Release 1

IBM

TNL SN26-0876 (03 June 1981) to SC26-3988-0

This publication was produced using the
IBM Document Composition Facility
(program number 5748-XX9) and
the master was printed on the IBM 3800 Printing Subsystem.

First Edition (February 1981)

This edition, as amended by technical newsletters SN26-0845 and SN26-0876, applies to Release 1 of VS FORTRAN, Program Products 5748-F03 (Compiler and Library) and 5748-LM3 (Library only), and to any subsequent releases until otherwise indicated in new editions or technical newsletters. Information concerning the IBM 3375 and 3380 direct access devices is for planning purposes only until the availability of the devices.

The changes for this edition are summarized under "Summary of Amendments" following the preface. Specific changes are indicated by a vertical bar to the left of the change. These bars will be deleted at any subsequent republication of the page affected. Editorial changes that have no technical significance are not noted.

Changes are periodically made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/370 and 4300 Processors Bibliography, GC20-0001, for the editions that are applicable and current.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Publications are not stocked at the address given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, P.O. Box 50020, Programming Publishing, San Jose, California, U.S.A. 95150. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

PREFACE

This manual is intended for FORTRAN application programmers who need VS FORTRAN specific reference material to run VS FORTRAN programs under OS/VS1, MVS (with or without TSO), DOS/VSE and VM/370-CMS. It describes the use of these different operating systems with the VS FORTRAN compiler to process FORTRAN programs.

INDUSTRY STANDARDS

The VS FORTRAN Compiler and Library program product is designed according to the specifications of the following industry standards, as understood and interpreted by IBM as of June, 1980:

American National Standard Programming Language FORTRAN, ANSI X3.9-1978 (also known as FORTRAN 77)

International Organization for Standardization ISO 1539-1980 Programming Languages-FORTRAN

These two standards are technically equivalent. In this manual, references to the current standard are references to these two standards.

American Standard FORTRAN, X3.9-1966

International Organization for Standardization ISO R 1539-1972 Programming Languages-FORTRAN

These two standards are technically equivalent. In this manual, references to the old standard are references to these two standards.

For both current and old standard language, a number of IBM extensions are also included in VS FORTRAN. In this book, references to current FORTRAN are references to the current standard plus the IBM extensions valid with it; references to old FORTRAN are references to the old standard plus the IBM extensions valid with it.

MANUAL ORGANIZATION

This manual is organized for reference purposes. It is organized into five main parts; each of the first four parts describes FORTRAN-specific system reference considerations for source programming, compilation, link-editing and loading, execution, and cataloged procedures. Part V, the Appendixes, contains useful supplementary information. The five parts are:

- Part I. OS/VS System Services
- Part II. DOS/VSE System Services
- Part III. VM/370-CMS System Services
- Part IV. OS/VS2 TSO System Services
- Part V. Appendixes

VS FORTRAN PUBLICATIONS

The VS FORTRAN publications are designed to help you develop your programs with a minimum of wasted effort.

This book, the VS FORTRAN Application Programming: System Services Reference Supplement, provides you with FORTRAN-specific reference documentation for compiling, link-editing, and executing VS FORTRAN programs.

The complete set of VS FORTRAN application programming publications follows.

VS FORTRAN Application Programming:

- Guide, SC26-3985—contains guidance information on designing, coding, debugging, testing, and executing VS FORTRAN programs written at the current language level.
- Language Reference, GC26-3986—gives you the semantic rules for coding VS FORTRAN programs when you're using current FORTRAN.
- Library Reference, SC26-3989—gives you detailed information about the execution-time library subroutines.
- System Services Reference Supplement, SC26-3988—contains FORTRAN-specific documentation.
- Source-Time Reference Summary, GX26-3731—is a pocket-sized reference card containing current language formats and brief descriptions of the compiler options.

System/360 and System/370 FORTRAN IV Language, GC28-6515—gives you the rules for writing VS FORTRAN programs when you're using old FORTRAN.

Figure 1 shows how these manuals should be used together.

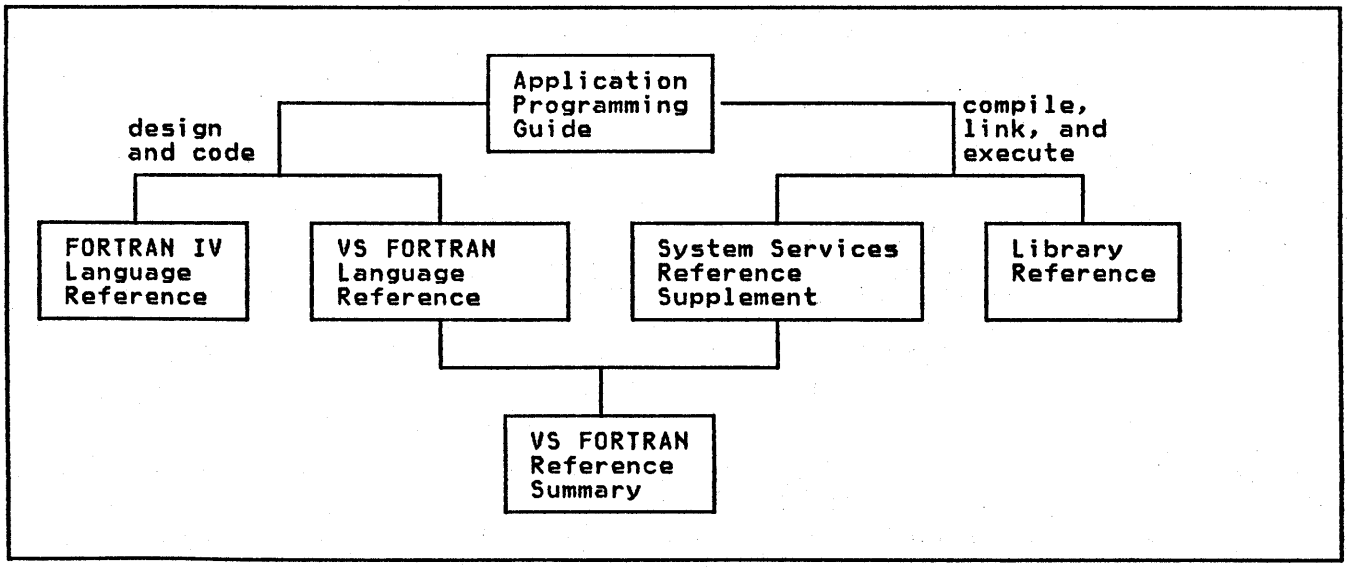


Figure 1. VS FORTRAN Application Programming Publications

RELATED PUBLICATIONS

Detailed system information has been eliminated from the VS FORTRAN publications. Therefore, in order to use this set of manuals effectively, you should have on hand one of the following sets of manuals depending on the system you are using.

OS/VS Publications

OS/VS1 JCL Reference, GC24-5099

OS/VS1 JCL Services, GC24-5100

OS/VS1 Access Method Services, GC26-3840

MVS Publications: If you are using MVS, the following additional manuals are needed.

OS/VS2 Access Method Services, GC26-3841

OS/VS2 MVS JCL, GC28-0692

TSO Publications: If you are using TSO, you will need the following publications:

OS/VS2 TSO Terminal User's Guide, GC28-0645

OS/VS2 TSO Command Language Reference, GC28-0646

OS/VS2 TSO Terminal Monitor Program and Service Routines Logic, SY28-0650

DOS/VSE Publications

DOS/VSE System Control Statements, GC33-5376

DOS/VSE System Management Guide, GC33-5371

Using VSE/VSAM Commands and Macros, SC24-5144

VM/370-CMS Publications

IBM Virtual Machine Facility/370: CMS Commands and Macro Reference, GC20-1818

IBM Virtual Machine Facility/370: Terminal User's Guide, GC20-1810

IBM Virtual Machine Facility/370: CP Command Reference for General Users, GC20-1820

SUMMARY OF AMENDMENTS

3 JUNE 1981

MISCELLANEOUS CHANGES

- Additions have been made to "Source Programming Considerations."
- The figure "Load Module Logical Units—DOS/VSE" has been redone.
- DOS/VSE JCL has been corrected.
- In the appendix, compiler option abbreviations were added.

30 APRIL 1981

OPTIMIZE COMPILER OPTION

The compiler option, OPTIMIZE(0), is available with Release 1 of VS FORTRAN. Additional optimization features (OPTIMIZE(1/2/3)), discussed in this manual, are planned for availability at a later time.

CONTENTS

FIGURES

FORMAT NOTATION

How job control statement and command formats are to be interpreted in this publication is discussed in the following paragraphs.

- **Truncations and Abbreviations of Commands**: Where truncation or abbreviation of a command name is allowed, the shortest acceptable version is in uppercase letters. However, the CMS commands can be entered with any combination of uppercase or lowercase letters.
- **UPPERCASE LETTERS, WORDS, AND NUMBERS**: Must be coded in the statement exactly as shown.
- **LOWERCASE LETTERS AND WORDS**: Represent variables, for which programmer-supplied information is substituted.
- **SYMBOLS** in the following list must be coded exactly as shown:

apostrophe	'
asterisk	*
comma	,
equal sign	=
parentheses	( )
period	.
slash	/

- **HYPHENS (-)**: Join lowercase letters and words and symbols to form a single variable.
- **LOGICAL OR SYMBOLS (|)**: Represent alternative choices from which one choice can be made.
- **BRACES ({ })**: Group required related items (such as alternative choices from which one choice is made).
- **SQUARE BRACKETS ([ ])**: Group optional related items (such as alternative choices from which one choice can be made).
- **UNDERSCORES (_)**: Indicate the standard default option. (If the option is omitted, this is the option chosen.)
- **ELLIPSES (...)**: Specify that the preceding syntactical unit can optionally be repeated. (A syntactical unit is a single syntactical item, or a group of syntactical items, enclosed in braces or brackets.)
- **SUPERSCRIPTS ([1])**: Refer to a description in a footnote.
- **BLANKS**: Are used to improve the readability of the control statement definitions. They must not appear in an operand field, unless a definition explicitly states otherwise.

SOURCE PROGRAM CONSIDERATIONS

The OPEN statement is required for VSAM and BDAM direct access files. What you specify in your job control statements must match what is specified in the OPEN statement.

VSAM FILE DEFINITIONS

VSAM is a high-performance access method of OS/VS for use with direct-access storage. VSAM entry-sequenced and relative-record data sets can be processed by the VS FORTRAN programmer after they have been defined through the use of the VSAM utility program known as Access Method Services.

A number of commands initiate the Access Method Services programs. However, only the DEFINE CLUSTER command is described here. For more detailed information on the other commands, see OS/VS1 Access Method Services.

VS FORTRAN VSAM entry-sequenced files are equivalent to sequential data sets. A VSAM relative-record is equivalent to a direct access data set.

DEFINE CLUSTER

The DEFINE CLUSTER command defines the attributes of all VSAM data sets and catalogs the data sets in the VSAM catalog. The format that follows applies only to the parameters for entry-sequenced and relative-record files. Only the required parameters and those of special interest to the VS FORTRAN programmer are given. For more detailed information regarding the DEFINE CLUSTER command, see OS/VS Access Method Services.

Format

Command	Parameters
DEFINE CLUSTER	CLUSTER (NAME(entryname) [FILE(dname)] VOLUMES(volser[volser...] [NONINDEXED|NUMBERED] RECORDS(primary [secondary])})

DEFINE CLUSTER Parameters

CLUSTER Is required and specifies that a cluster is to be defined. CLUSTER is followed by the parameters specified for the cluster as a whole; these parameters are enclosed in parentheses.

NAME Specifies the name of the cluster and is required.

The name may contain from 1 through 44 alphameric characters, extended characters (@, #, and &), and the two special characters (the hyphen and the 12-0 overpunch). Names containing more than eight characters must be segmented by periods; one to eight characters may be specified between periods. The first character of any name or name segment must be either an alphabetic or extended character.

FILE Specifies the name of the DD statement that identifies the devices and volumes to be used for space allocation.

When the cluster is defined in a recoverable catalog, and FILE is specified as a parameter of CLUSTER, the FILE statement can identify all volumes on which space is to be allocated.

VOLUMES Specifies the volumes to contain the cluster or component, and is required. It can be specified for the cluster as a whole.

NONINDEXED Specifies that the cluster being defined is for an entry-sequenced file. The data records can be accessed sequentially or by relative-byte address.

NUMBERED Specifies that the cluster's organization is for relative record data. The data records are accessed by relative-record number.

RECORDS Specifies the amount of space to be allocated to the cluster from the volume's available space. The amount of space allocated is the minimum number of tracks that are required to contain the specified number of records.

For information on how and when to use DEFINE CLUSTER, see VS FORTRAN Application Programming: Guide.

JOB PROCESSING—OS/VS

Three basic steps are taken to process a FORTRAN program:

1. Compiling
2. Linkage editing
3. Load module execution (go step)

The input to the compile step is called the source module. The output from the compile step is called an object module, which is the input to the link-edit step. The output of the link-edit step, the load module, is one or more object modules with all external references resolved. The load module is the program that is executed in the go step. If the loader is used in place of the linkage editor, the last two steps (link-edit and load module execution) are combined into one step.

Each step is called a job step—the execution of one program. Each job step may be executed alone or in combination with other job steps as a job—an application involving one or more job steps. Hence, a job may consist of one step, such as FORTRAN compiler execution, or of many steps, such as compiler execution followed by linkage editor execution and load module execution.

The programmer defines the requirements of each job to the operating system through job control statements.

JOB CONTROL STATEMENTS—OS/VS

Job control statements provide a communications link between the FORTRAN programmer and the operating system. The FORTRAN programmer uses these statements to define a job, a job step within a job, and data sets required by the job. For a complete description of job control language, see OS/VS JCL Reference.

This section describes the OS/VS job control statements.

JOB
DD
EXEC
PROC
Delimiter
Null (or end-of-job)
Comment

OS/VS Job Control Statement Sequence

The order in which job control statements are used follows:

```
// JOB statement
// EXEC statement          (VS FORTRAN compiler)
// DD statements           (as required for compilation)
   (source program to be compiled)
/* End-of-Data statement  (if source program is on cards)
// EXEC statement          (linkage editor)
// DD statements           (as required for link editing)
   (Link edit is performed)
// EXEC statement          (load module)
// DD statements           (as required for load module execution)
   (Input data to be processed)
/* End-of-Data statement  (if input data is on cards)
// End-of-Job statement
```

However, for easy reference, the job control statements are presented within this section in alphabetic order.

SPECIFYING JOB CONTROL STATEMENTS

Job control statements are identified by the characters // in columns 1 and 2, except for the delimiter statement, which is identified by the characters /* in columns 1 and 2, and the comments statement, identified by the characters //* in columns 1, 2, and 3. The delimiter statement may contain optional comments preceded by one or more blanks. The null statement may contain only the two slashes; the remainder of the statement must be blank. The comments statement contains notes written by the programmer. The other three statements: JOB, EXEC, and DD, can contain up to four fields; name, operation, operand, and comments.

Figure 2 illustrates the format of job control statements. The brackets around some of the items shown indicate that those items are optional when using the statement.

Format	Applicable Control Statements
//Name Operation [Operand] [Comment]	JOB
//[Name] Operation Operand [Comment]	EXEC,DD
/* [Comment]	delimiter
//	null (or end-of-job)
//* Comment	comments

Figure 2. Job Control Statement Formats

name Assigns a name to the statement, identifying it to other statements and to the operating system. The name field begins in column 3 immediately following the //.

operation Identifies the type of control statement, for example, JOB, DD, or EXEC. The operation field begins in any column after the name field and is preceded and followed by one or more blanks.

operand Identifies system options requested by the programmer. The operand field begins in any column after the operation field and is preceded and followed by one or more blanks. Options are specified through one or more parameters separated by commas. Parameters may be either positional or keyword.

> Positional parameters must appear in a fixed order and are identified, or given meaning, by their position in that order.
>
> Keyword parameters are composed of an identifying keyword, an equal sign (=), and a value.

Parameters may also comprise a number of subparameters, which can be either positional or keyword.

comments The comments field may contain any information that is considered helpful. There is no required syntax for comments. Comments begin in any column after the operand field (or the /* in the delimiter statement) and are preceded by one or more blanks.

All statements (except null) may be continued onto the succeeding card or card image, using the following rules:

- Interrupt the operands after a completed parameter or subparameter, including the comma that follows it, at or before column 71.

- If comments are desired in the same statement as an interrupted operand field and there is sufficient room, leave one or more spaces after the comma that follows the last parameter, then code the comments.
- For an interrupted comment, code any nonblank character in column 72. For an interrupted operand, the nonblank character in column 72 is optional. If a nonblank character is not coded in column 72 of an interrupted operand, but the conventions outlined in the next two items are followed, the next statement is treated as a continuation statement.
- Code the identifying characters // in columns 1 and 2 of the following card or card image.
- Continue the interrupted operand beginning in any column from 4 through 16.

DD Control Statement

The DD statement describes data set control device and volume assignment. The DD statement is concerned with the data set, the records within the data set, and the location of the data set.

Format

Name	Operation	Operand
// {ddname} {procstep.ddname}	DD	Positional Parameters [*] [DATA] [DUMMY] Keyword Parameters [{VOLUME} {SER=serial-number}] = [{VOL} {REF=ddname}] [DDNAME=ddname] [{DSNAME} =data-set-name] [{DSN} [DISP=(subparameter-list)] [SYSOUT=x] [LABEL=(subparameter-list)] [COPIES=number] [SEP=(ddname[,ddname]...)] [DLM=delimiter] [UNIT=(device[,SEP=(ddname,...)])] [SPACE=(subparameter-list)] [DCB=(subparameter-list)]

The DD statement has all the operands shown in the format above; however, only the required and FORTRAN specific operands are described here. For additional information see OS/VS1 JCL Reference.

DD Statement Operands

ddname Identifies the DD statement to other job control statements, and relates the data set to input/output statements in the FORTRAN source module. It is specified as 1 to 8 alphameric characters, the first of which must be alphabetic or one of the extended characters #, @, or $.

The DD statement defines VS FORTRAN load module data sets. The ddname format for data sets used in FORTRAN load module execution is:

FTxxFyyy

where xx is the data set reference number and yyy is a FORTRAN sequence number (usually 001). The IBM-supplied ddnames are:

FT05F001 - input to the programmer's load module

FT06F001 - printed output data

FT07F001 - punched output data

The remaining data set reference numbers are available to the programmer. The number of units available is installation dependent.

The data set reference number is a numeric means of equating VS FORTRAN input/output statements with the proper data set definition. If the programmer uses cataloged procedures and the standard data set reference numbers there is no need to supply DD statements for those data sets; they are contained in the cataloged procedures.

The FORTRAN sequence number refers to separate data sets that are read or written using the same data set reference number. The sequence number is incremented when (1) an END FILE statement without an intervening REWIND is executed and a subsequent WRITE is issued with the same data set reference number or (2) the "END =" exit is taken following a READ and a subsequent READ or WRITE is issued with the same data set reference number.

procstep.ddname Identifies the DD statement in the particular jobstep identified as procstep. (Procstep identifies a step in a cataloged procedure.)

Procstep and ddname are each specified as 1 to 8 alphameric characters, the first of which must be alphabetic or one of the extended characters #, @, or $. Procstep and ddname are separated by a period (.).

***** Indicates that the data appears in the input stream.

When * is used,

- The data must appear immediately after the DD * statement.
- Only the DCB parameter has meaning on the DD statement.

DATA Indicates that the data appears in the input stream and may contain job control statements that are to be read as data and not executed.

When data is used,

- The data must immediately follow the DD DATA statement.
- Only the DCB parameter has meaning on the DD statement.

DUMMY Identifies a data set on which no input/output operations are to be performed (such as to eliminate writing an output data set in a program segment that has already been tested). A READ to a DD DUMMY file returns an end-of-file condition on the first READ.

DISP Indicates whether the data set is new or old, and whether it is to be kept or released at the end of the job step.

DISP=(subparameter-list), where subparameter-list indicates:

- The disposition of the data set at the beginning of the job step, that is, whether OLD, NEW, SHR, or MOD. For example, a FORTRAN programmer may use SHR for source member input during the compile step, or MOD may be used to add to a sequential file.
- The disposition to be made of the data set at the end of the job step; for example, whether to be kept, deleted, or passed to another job step,
- The disposition to be made of the data set if abnormal termination occurs; for example, whether to be kept or deleted.

Two subparameters that may be used by the FORTRAN programmer are:

SHR—describes existing data that resides on a direct-access volume and that is also available to other concurrently operating jobs.

MOD—describes a sequential data set that is to be added to. Before the first input/output operation occurs, the data set will be automatically positioned after the last record.

LABEL

Assigns such information as labels, whether the data set is protected from unauthorized processing, and how long the data set should be kept. The subparameters of this operand of specific interest to the FORTRAN programmer are IN and OUT. The format and definition of these subparameters follow:

Format

```
LABEL = (,SL, [,IN|,OUT])
```

SL—specifies that the data set has standard system-created labels.

IN—specifies that the data set is to be processed for sequential input operations only. IN is recognized only if the first input/output operation is a READ. If it is not a READ, IN is ignored and both input and output operations are permitted; if it is a READ, any subsequent WRITE is treated as an error and job processing is terminated.

OUT—specifies that the data set is to be processed for sequential output operations only. OUT is recognized only if the first input/output operation is a WRITE. If it is not a WRITE, OUT is ignored and both input and output operations are permitted. If it is a WRITE, any subsequent READ is treated as an error and job processing is terminated.

UNIT

Identifies the input/output device or device class on which the data set resides.

UNIT=(device[,SEP=(ddname,...)]), where device is a number or name identifying the device, and ddname is the name of another DD statement. SEP=ddname... is used only when the data set is not to share device access areas with the data set defined in ddname (see Appendix A).

DCB Identifies the characteristics of the records in the data set.

The format of the DCB subparameter is:

Format

```
DCB=([data-set-name]
     [,RECFM=record-format]
     [,LRECL=record-length]
     [,BLKSIZE=block-length]
     [,BUFNO=number-of-buffers]
     [,BUFOFF=block-prefix]
     [,DEN=tape-density]
     [,TRTCH=tape-recording-techniques]
     [,DSORG=direct-access-organization]
     [,OPTCD=optional-services]
```

For a complete discussion of the operands described here and other operands shown in the format, see OS/VS1 JCL Reference.

Direct-Access Data Set Considerations

Any data set to be used during a direct-access, VSAM relative-record, or entry-sequenced file input/output operation must be described by an OPEN statement. Record characteristics described in the OPEN statement must agree with the space allocation requested in the DD statement.

The data set reference number specified in any OPEN statement may refer to only one data set. In other words, references to separate data sets that are read or written using the same data set reference number are not allowed. (For a more detailed explanation of the OPEN statement, see VS FORTRAN Application Programming: Language Reference.)

EXEC Control Statement

An EXEC statement indicates the beginning of a job step and names the program or cataloged procedure to be executed. It is used for compilation, the linkage editor or loader, and load module execution.

Format

Name	Operation	Operand
//[stepname]	EXEC	Positional Parameter [PROC=]procedure name PGM=program-name Keyword Parameters [PARM='option[,option]..' [ACCT=(accounting-information)] [COND=((code,operator[,stepname])(...)) [DPRTY=step-priority] [TIME=(minutes,seconds)] [REGION=region-size] [ADDRSPC=REAL|VIRT]

The EXEC statement has all the operands shown in the format above; however, only the required and FORTRAN specific operands are described here. For additional information on the EXEC statement, see OS/VS1 JCL Reference.

Where:

procedure-name Names the cataloged procedure to be executed. A procedure-name initiates the processing of a series of job control statements that has been previously written in the system library, SYS1.PROCLIB, and cataloged in the system catalog.

PROC=procedure-name to be executed, or procedure-name indicates the name of the cataloged procedure. To specify the VS FORTRAN compiler, you specify:

```
PROC=FORTVS
```

program-name Indicates the name of the program to be executed. PGM=program-name, where program-name is the name of the program.

PARM Specifies program options. Option names a particular program option that is to be in effect during processing. For a complete list of the compiler options that can be specified, see Appendix B.

JOB Control Statement

The JOB statement indicates the beginning of a new job and describes that job.

Format

Name	Operation	Operand
//jobname	JOB	Positional Parameters [,accounting-information] [,programmer-name] Keyword Parameters [MSGLEVEL=(x,y)] [COND=((code,operator)...)] [CLASS=job-class] [PRTY=job-priority] [MSGCLASS=x] [REGION=region-size] [TIME=(minutes,seconds)] [ADDRSPC=REAL|VIRT]

The job statement has all the operands shown above; however, only the required and FORTRAN specific operands are discussed here. For additional information on the JOB statement, see OS/VS1 JCL Reference.

where:

jobname Identifies the job to the operating system. It is one to eight alphameric characters, the first of which must be alphabetic or one of the extended alphabetic characters #, @, or $.

accounting-info Identifies the account number or other accounting information relating to the job. Up to 142 characters can be specified; any special characters other than the hyphen, must be enclosed in apostrophes.

programmer-name Identifies the person submitting the job. Up to 20 characters can be specified; any special characters other than the hyphen must be enclosed in apostrophes.

MSGLEVEL Specifies the type of system messages (for example, job control statements, diagnostic messages, termination messages) to be written as part of the output listing.

MSGLEVEL=(x,y), where x is 0, 1, or 2, to indicate which job control statements and diagnostic messages are to be written, and y is 0 or 1 to indicate whether termination messages are to be written.

COND Specifies which condition codes terminate processing. It helps the programmer reduce computing time by making job continuation dependent upon successful completion of a previous job step.

The code parameter is a number between 0 and 4095 (usually 0, 4, 8, 12, or 16) and operator is a 2-character value that represents a comparison to be made between code and return code issued by the operating system. Up to eight sets of codes and operators may be specified.

The system issues a number, called a return code, at the end of each job step. The return code indicates how well the job step ran; for example, whether it completed processing normally or whether error conditions were detected. The COND parameter indicates a comparison to be made between the return code and the condition code specified in COND; if the condition is met, the job is terminated.

PROC Statement

The PROC statement is used to assign default values to symbolic parameters.

A symbolic parameter appears in the operand field of a statement and stands as a symbol for a parameter, a subparameter, or a value. Since the PROC statement deals only with symbolic parameters, the ampersand (&) that usually precedes symbolic parameters is omitted.

Symbolic parameters are used to make a cataloged procedure easily modified when it is called. Values may be assigned to symbolic parameters when calling a cataloged procedure, or the default value assigned by the PROC statement may remain in effect.

Format

Name	Operation	Operand
// [name]	PROC	symbolic-parameter=value[,...]

where:

// [name] Is the name of a cataloged procedure and is optional.

symbolic-parameter=value Identifies the values assigned to the symbolic parameter. See VS FORTRAN Application Programming: Guide for information on how to use symbolic parameters.

LINKAGE EDITOR CONTROL STATEMENTS—OS/VS

Linkage editor control statements specify an operation and one or more operands.

The first column of a control statement must be left blank. The operation field begins in column 2 and specifies the name of the operation to be performed. The operand field must be separated from the operation field by at least one blank. The operand field specifies one or more operands separated by commas. No embedded blanks may appear in the field. Linkage editor control statements may be placed before, between, or after either modules or secondary input data sets.

The INCLUDE and LIBRARY control statements specify secondary input.

INCLUDE Linkage Editor Control Statement

The INCLUDE statement specifies additional programs to be included as part of the load module.

Format

Operation	Operand
INCLUDE	ddname[(member-name [,member-name]...)] [,ddname[(member-name [,member-name]...)]...]

where:

ddname Indicates the name of a DD statement specifying a library or a sequential data set.

member-name Indicates the name of the member to be included. When sequential data sets are specified, member-name is omitted.

LIBRARY Linkage Editor Control Statement

The LIBRARY statement specifies additional libraries to be searched for object modules to be included in the load module.

The LIBRARY statement differs from the INCLUDE statement in that libraries specified in the LIBRARY statement are not searched until all other references (except those reserved for the automatic call library) are completed by the linkage editor. A module specified in the INCLUDE statement is included immediately.

Format:

Operation	Operand
LIBRARY	ddname (member-name [,member-name]..) [,ddname(member-name [,member-name]...)]...]

where:

ddname indicates the name of a DD statement specifying a library.

member-name is the name of a member of the library.

COMPILER, LINKAGE EDITOR, AND LOADER DATA SETS—OS/VS

COMPILER DATA SETS—OS/VS

The compiler uses up to eight system data sets. Many of these data sets are defined in cataloged procedures. Figure 3 lists the function, device types, and allowable device classes for each data set.

ddname	Function	Device Types	Applicable Device Class	Defined in Cataloged Procedures Calling the Compiler
SYSIN	Reading input source data set	Card reader Magnetic tape Direct access	Input stream (defined as DD *, or DD DATA)	No
SYSLIB	Storing INCLUDE data sets (Required if INCLUDE is specified)	Direct access	SYSDA	No
SYSLIN	Creating an object module data set as compiler output and linkage editor input (Required if OBJECT is specified)	Direct access Magnetic tape Card punch	SYSDA SYSSQ SYSCP	Yes
SYSPRINT	Writing source, object, and cross reference listings, storage maps, messages	Printer Magnetic tape Direct access	A SYSSQ	Yes
SYSPUNCH	Punching the object module deck (Required if DECK is specified)	Card punch Magnetic tape Direct access	B SYSCP SYSSQ SYSDA	Yes
SYSUDUMP or SYSABEND	Writing dump in event of abnormal termination (Required if DUMP is specified)	Printer Magnetic tape Direct access	A SYSSQ	No
SYSTERM	Writing error message and compiler statistics (Required if TERM is specified) listing	Printer Magnetic tape Direct access	A SYSSQ	Yes

Figure 3. Compiler Data Sets

Note: SYSPUNCH and SYSLIN may not be directed to the same card punch.

The DCB subparameters define record characteristics of a data set. The following table lists the DCB default values for compiler data set characteristics.

ddname	LRECL	RECFM	BLKSIZE	BUFNO
SYSIN	80	FB	80[1]	2
SYSPRINT	137	VBA	3429[1]	2
SYSLIN	80	FB	3200[1]	2
SYSPUNCH	80	FB	3440[1]	2
SYSTERM	121	FB	121	2

[1]These default blocksize values correspond to the BLKSIZE values specified on the DD statements in the distributed cataloged procedures. The compiler defaults to BLKSIZE = LRECL.

Figure 4. DCB Default Table

LINKAGE EDITOR DATA SETS—OS/VS

The linkage editor generally uses five system data sets; others may be necessary if secondary input is specified. Secondary input is defined by the programmer; cataloged procedures do not supply the secondary input DD statements.

Figure 5 lists the function, device types, and allowable device classes for each data set.

ddname	Function	Device Types	Applicable Device Class	Defined in Cataloged Procedures Calling the Linkage Editor
SYSLIN	Primary input data, generally output of the compiler	Direct access Magnetic tape Card reader	SYSDA SYSSQ input stream (defined as DD * or DD DATA)	Yes
SYSLIB	Automatic call library (SYS1.FORTLIB)	Direct access	SYSDA	Yes
SYSLMOD	Link-edit output (load module)	Direct access	SYSDA	Yes
SYSPRINT	Writing listings, messages	Printer Magnetic tape Direct access	A SYSSQ	Yes
User-defined	Additional libraries and object modules	Direct access Magnetic tape	SYSDA SYSSQ	No

Figure 5. Linkage Editor Data Sets

LOADER DATA SETS—OS/VS

The loader normally uses six system data sets; other data sets may be defined to describe libraries and load module data sets. Figure 6 lists the function, device types, and allowable device classes for each data set.

ddname	Function	Device Types	Applicable Device Class	Defined in Cataloged Procedures Calling the Loader
SYSLIN	Input data to linkage function, normally output of the compiler	Direct access Magnetic tape Card reader	SYSDA SYSSQ Input stream (defined as DD *)	Yes
SYSLIB	Automatic call library (SYS1.FORTLIB)	Direct access	SYSDA	Yes
SYSLOUT	Writing listings	Printer Magnetic tape Direct access	A SYSSQ	Yes
SYSPRINT	Writing messages	Printer Magnetic tape Direct access	A SYSSQ	No
SYSIN	Input data to load module function	Card reader Magnetic tape Direct access	Input stream (defined as DD *) SYSSQ SYSDA	No
FT07F001	Punched output data	Card punch	B	Yes
FTnnFnnn[1]	User-defined data set	Unit record Magnetic tape Direct access	SYSSQ A,B SYSDA	No

Figure 6. Loader Data Sets

Note to Figure 6:

[1]nn and nnn cannot be set to 0.

LOAD MODULE EXECUTION DATA SETS—OS/VS

The load module execution job step executes a load module. The load module may be passed directly from a preceding link-edit job step, or it may be called from a library of programs, or it may form part of the loader job step.

The load module execution job step may use many data sets. Figure 7 lists the function, device types, and usage for each data set.

FORTRAN Ref. Number	ddname	Function	Device Type	Usage
5	SYSIN	Input data set to load module	Card reader Magnetic tape Direct access	Load module Input data
5	FT05F001	Input data set to load module	Card reader Magnetic tape Direct access	Load module Input data
6	FT06F001	Printed output data	Printer Magnetic tape Direct access	Load module Output data
7	FT07F001	Punched output data	Card punch Magnetic tape Direct access	Load module Output data
0-4, 8-99	FTnnFnnn	Sequential data set	Unit record Magnetic tape Direct access	Program data
0-4, 8-99	FTnnFnnn	Direct access data set	Direct access	Program data
0-4, 8-99	FTnnFnnn	Partitioned data set member using sequential access	Direct access	Load module Input data

Figure 7. Load Module Execution Data Sets

Programmer-Defined Data Sets

The SYSIN DD statements must be defined by the programmer to complete the description of the input data set begun by the DD statement FT05F001. If no input data set is to be submitted, the SYSIN DD statement is omitted and the operating system treats the FT05F001 DD statement as though DD DUMMY had been specified.

The programmer must also ensure that the JCL for the load module execution step includes a DD statement for error code diagnostic output. This DD statement must be present when the data set is opened for each execution of a FORTRAN load module. Usually, the FT06F001 DD statement defines this data set; however, this assignment can be modified when VS FORTRAN is installed. Check with your system administrator.

DCB Default Values for Load Module Execution Data Sets: The following tables list the DCB default values for load module execution data sets.

Sequential Data Sets

ddname	RECFM[1]	LRECL[2]	BLKSIZE	DEN	BUFNO
FT05Fyyy	F	80	80	-	2
FT06Fyyy	UA	133	133	-	2
FT07Fyyy	F	80	80	-	2
all others	U	--	800	2	2

[1]For records not under FORMAT control, the default is VS.

[2]For records not under FORMAT control, the default is 4 less than shown.

Direct Access Data Sets

ddname	RECFM	LRECL or BLKSIZE	BUFNO
FT05Fyy	F	The value specified as the maximum size of a record in the OPEN statement.	2
FT06Fyyy	F		2
FT07Fyyy	F		2
all others	F		2

PART II. DOS/VSE SYSTEM SERVICES

SOURCE PROGRAMMING CONSIDERATIONS

In a FORTRAN source program, input and output devices are referred to by data set reference numbers. With this form of reference, an object program compiled from a VS FORTRAN source program is not dependent upon the availability of any specific device at execution time.

A programmer can temporarily assign a logical unit name to a specific input/output device using an ASSGN job control statement. Programmers need not make assignments unless their requirements differ from the standard assignments.

There is a direct correlation between the logical unit names, such as SYSIPT, SYSPCH, and SYS001, and the data set reference numbers used by a VS FORTRAN programmer. The first two columns of Figure 8 show the DOS/VSE logical unit name and its associated data set reference number. Figure 8 also illustrates the relationship of logical unit names and data set reference numbers that are under systems other than DOS/VSE.

The operator may assign available input/output devices. These are assigned to meet the logical unit requirements of the operating system. One device, for example, must be assigned as SYSIPT to serve as the system's main input unit.

Hardware must support extended precision.

- When executing a program on DOS/VSE that uses extended precision and was compiled on MVS, VS1, or CMS
- When compiling on DOS/VSE a program that has extended precision constants combined in any arithmetic sense with any other kinds of constants

FORTRAN object programs with asynchronous I/O cannot run on either DOS/VSE or CMS.

FORTRAN Ref. Number	Logical Unit	DOS File Name	Function (Primary)	Device Type
0	SYS000	IJSYS00	Program data set	Unit record Magnetic tape Direct access
1	SYS001	IJSYS01	Program data set	Unit record Magnetic tape Direct access
2	SYS002	IJSYS02	Program data set	Unit record Magnetic tape Direct access
3	SYS003	IJSYS03	Program data set	Unit record Magnetic tape Direct access
4	SYS004	IJSYS04	Program data set	Magnetic tape Direct access
5	SYSIPT or SYSIN	IJSYSIP	Input data set to load module	Unit record Magnetic tape Direct access
6	SYSLST	IJYSYLS	Punched output data	Unit record Magnetic tape Direct access
7	SYSPCH	IJSYSPC	Printed output data	Printer Magnetic tape Direct access
8 thru 99	SYS005 thru SYS096	IJSYS05 thru IJSYS96	Program data set	Unit record Magnetic tape Direct access

Note: Units 9-99 may be added by reassembling the unit assignment table module (IFYUATBL). See section "Using the VSFORTL Macro" in VS FORTRAN Installation and Customization.

Figure 8. Load Module Logical Units—DOS/VSE

VSAM FILE PROCESSING

This section discusses how to define VSAM files, and how to use VSAM space management for sequential files.

VSAM FILE DEFINITIONS

A VSAM file may be suballocated, or unique. A suballocated file shares a data space with other files; a unique file has a data space to itself.

VSAM treats all files as clusters. For entry-sequenced or relative-record files, a cluster consists of a data component only. Besides setting up a catalog entry for each component of a cluster, VSAM sets up a catalog entry for the cluster as a whole. This entry is the cluster name specified in the DEFINE command.

To define a suballocated VSAM file, first define a data space, then use the DEFINE command with the CLUSTER parameter. VSAM suballocates space for the file in the data space set up and enters information about the file in a VSAM catalog. A file can be stored in more than one data space on the same volume or on different volumes.

A unique VSAM file is defined by specifying the parameter UNIQUE and assigning space to the file with a space allocation parameter and the DLBL/EXTENT job control statements. The data space is acquired and assigned to the file concurrent with the file definition. However, no other file can occupy its data space(s).

DEFINE CLUSTER

The DEFINE CLUSTER command is used to define entry-sequenced, relative-record, or VSAM-managed SAM files. For an entry-sequenced or a relative-record file, two entries are created, one for the cluster and one for the data component. For guidance information on VSAM-managed SAM files, see VS FORTRAN Application Programming: Guide.

The format that follows lists the required parameters and those of special interest to the FORTRAN programmer.

Format

Command	Parameters
DEFINE	CLUSTER (NAME(entryname) FILE(dname) VOLUMES(volser[volser...]) RECORDS RECORDSIZE [NONINDEXED|NUMBERED]

DEFINE CLUSTER Parameters

CLUSTER Specifies that a cluster is to be defined. CLUSTER is followed by the parameters specified for the cluster as a whole.

NAME Specifies the name of a cluster or component and is required. The name may contain from 1 through 44 alphameric characters, extended characters (@, #, and $), and two special characters (the hyphen and the 12-0 overpunch). Names containing more than eight characters must be segmented by periods; one to eight characters may be specified between periods. The first character of any name or name segment must be either an alphabetic or an extended character.

FILE Specifies the filename of the DLBL job control statement that, together with an EXTENT statement, identifies the logical units and volumes to be used for space allocation. The FILE parameter must be specified if UNIQUE is specified or if the cluster is cataloged in a recoverable catalog.

NONINDEXED Specifies that the cluster being defined is for a entry-sequenced file. The data records can be accessed sequentially or by relative-byte address.

NUMBERED Specifies that the cluster being defined is for a relative-record file. The records are accessed by relative-record number.

VOLUMES Specifies the volumes to contain the cluster or component. Up to 123 volumes can be specified. A volume serial number (volser) may contain one to six alphameric, extended, and special characters. See Using VSE/VSAM Commands and Macros for additional information on the DEFINE CLUSTER command.

RECORDS Specifies the number of records for which a space is to be allocated.

RECORDSIZE Specifies the average and maximum length, in bytes, of the data record. The minimum record size that can be specified is one.

JOB PROCESSING—DOS/VSE

Processing of programs under control of DOS/VSE is accomplished through job control statements that indicate whether a program is to be compiled, link-edited, and/or executed.

This chapter describes the VS FORTRAN-specific job control statements required for compiling, link-editing, and executing VS FORTRAN programs under DOS/VSE.

DOS/VSE JOB CONTROL STATEMENTS

Compilation	Linkage Editor	Execution
JOB	JOB	JOB
EXEC FORTVS	EXEC LINKEDT	EXEC program-name
ASSGN	ACTION[1]	
CLOSE	ENTRY[1]	
DLBL	INCLUDE[1]	
EXTENT	PHASE[1]	
OPTION		
TLBL		

[1]These are linkage editor control statements.

DOS/VSE Job Control Statement Sequence

The order in which job control statements are used follows:

```
// JOB statement
// OPTION LINK            (sets link option)
   or
// OPTION CATAL           (sets link option and catalogs phase)
// EXEC statement         (VS FORTRAN compiler)
   (VS FORTRAN source program)
/* Statement              (if source program is in-line)
// ASSGN Statements       (as required for linkage editing)
// DLBL/TLBL statements   (as required for linkage editing)
// EXEC statement         (linkage editor)
     (Linkage Editor execution)
// ASSGN statements       (as required for execution)
// DLBL/TLBL statements   (as required for execution)
// EXTENT statements      (as required for execution)
// EXEC statement         (load module)
     (Input Data to be processed)
/* End-of-Data Statement (if input data is in-line)
/& End-of-Job Statement
```

However, for easy reference the formats and functions of the job control statements are presented in alphabetic order. For a more detailed description of the job control statements, see DOS/VSE System Control Statements.

ASSGN Control Statement

The ASSGN statement temporarily assigns a logical input/output unit to a physical device. Multiple logical units are allowed to be assigned to one physical unit within the same partition. Only DASD can be assigned to (shared by) several partitions concurrently.

The job control statement remains in effect only until the next change in assignment or until end of job, whichever occurs first. At the completion of a job, a temporary assignment is automatically restored to the permanent assignment for the logical unit.

ASSGN is required only when you assign a logical unit different from the predefined unit table.

Format

ID	Operation	Operands	
//	ASSGN	SYSxxx,	{cuu} (address-list) {UA} {IGN} {SYSyyy} {device-class} {device-type}

The ASSGN statement has all the operands shown in the format above; however, only the required and FORTRAN specific operands are described here. For additional information on the ASSGN statement, see DOS/VSE System Control Statements.

Where:

SYSxxx Is required and indicates the name of the logical unit to be used. If one of the system logical units SYSIPT, SYSLST, or SYSPCH is assigned to a disk, the assignment must be permanent and follow the DLBL and EXTENT statements.

There is a direct correlation between the logical unit names, such as SYS001, and the data set reference numbers used by the VS FORTRAN programmer as shown in Figure 8.

cuu Indicates the channel and unit number of the device to be used:

c = channel number

uu = unit number

(address list) Specifies a list of up to seven device addresses in the form cuu, separated by commas and enclosed in parentheses.

For disks, if SHR is specified, the first unit in the list is assigned, even if previously assigned.

UA Specifies that the logical unit is not to be assigned to any device.

Any operation attempted on an unassigned device causes immediate job cancellation.

IGN Indicates that the specified logical unit is not to be assigned to any device and that all source program references for that unit are to be ignored (the job will not be cancelled). This allows you

to disable a logical unit that is used in a program without removing the code for that unit. You can then execute the program as if the unit did not exist.

Do not specify with SYSRDR, SYSIPT, SYSIN, or SYSCLB.

SYSyyy This may be any system or programmer logical unit. If this operand is specified, SYSxxx is assigned to the same device as SYSyyy is currently assigned.

device-class Specifies the following devices supported by DOS/VSE:

READER
PRINTER
PUNCH
TAPE
DISK
FBA

device-type Is any supported device as shown in Appendix A under the device-class specification.

This operand should be used only in specifying a specific type of device, and not in the physical unit.

CLOSE Control Statement

The CLOSE statement is used to close either a programmer logical unit assigned to tape or a system logical unit assigned to disk.

The logical unit can optionally be reassigned to another device, unassigned, or, in the case of a tape file, switched to an alternate unit. When SYSxxx is a system logical unit (SYSLST, SYSPCH, etc.), one of the optional operands must be specified. When closing a programmer logical unit, no optional operands need be specified. When none is specified, the programmer logical unit is closed and the assignment remains unchanged.

Format

ID	Operation	Operands
[//]	CLOSE	SYSxxx, [,cuu] [,ss] [,UA] [,IGN] [,ALT] [,SYSyyy]

The CLOSE statement has all the operands shown in the format above; however, only the required and FORTRAN specific operands are described here. For additional information on the CLOSE statement, see DOS/VSE System Control Statements.

Where:

[//] CLOSE Is specified as shown.

SYSxxx, Indicates the name of the logical unit to be closed. It can be SYSLST, SYSPCH, or programmer logical units.

cuu Indicates, in hexadecimal, the channel and unit number of the device to be used after the specified unit is closed. In the case of a system logical unit, the new unit will be opened if it is either a disk, or a magnetic tape at load point.

ss Indicates tape mode specifications. If the field is not specified, the system assumes 90 for 7-track tapes and C0 for 9-track tapes.

UA Specifies that the logical unit is not to be assigned to any device after the file has been closed.

Any operation attempted on this device causes immediate job cancellation.

IGN Indicates that the logical unit is not to be assigned to any device and that all program references to it are to be ignored until a new ASSGN is given for the unit, or IPL is performed.

This operand is invalid for SYSLST.

DLBL Control Statement

The DLBL statement contains file label information for disk label checking and creation. The file name must be specified on the DLBL statement. All other entries are optional. However, if any of the subsequent operands are used, a comma must be inserted for each positional operand that is omitted.

Format

ID	Operation	Operand
//	DLBL	filename,['file-ID'], [date],[codes] [,DSF][,BUFSP=n][,CAT=filename] [BLKSIZE=n][,CISIZE=n]

The DLBL statement has all the operands shown in the format above; however, only the required and FORTRAN specific operands will be described here. For additional information on the DLBL statement see DOS/VSE System Control Statements.

where:

filename Is required and must correspond to the DOS/VSE logical unit. (See DOS/VSE Logical Unit Name column in Figure 8.) It can be from one to seven alphameric characters, the first of which must be alphabetic.

'file-ID' Indicates the unique name associated with the file on the volume. If the field is omitted, the filename is used.

It can be from 1 to 44 alphameric characters enclosed in apostrophes, including:

- 'file-ID' (1 through 35 characters)
- If used, generation number and version number
- If fewer than 44 characters, the field is left-justified and padded with blanks.

codes Indicates the type of file label to be used. It is a 2- to 4- character field that can be specified as follows:

SD - for seqential disk

DA - for direct access

VSAM - for all Virtual Storage Access Method files

If the field is omitted, sequential disk (SD) is assumed.

CAT=filename This operand is only valid in a DLBL statement for a VSAM file. It specifies the filename (1 to 7 alphameric characters) of the DLBL statement for the catalog owning this VSAM file. The system searches only this catalog for the file-ID when the VSAM file is to be opened. Specify this operand only if you want to override the system's assumption that the job catalog or, if there is no job catalog, that the master catalog owns the file.

EXEC Control Statement

The EXEC statement indicates either:

- The end of control information for job step and the beginning of execution of a program, in which case it must be the last statement processed before a job step is executed, or
- That a cataloged procedure is to be retrieved from the procedure library by job control. In this case, other statements may follow EXEC.

Format

ID	Operation	Operand
[//]	EXEC	[[PGM=]progname] [,REAL] [,SIZE=size][,GO] [,PARM='value']
[//]	EXEC	PROC=procname[,OV]

The EXEC statement has all the operands shown in the format above; however, only the required and FORTRAN specific operands are described here. For additional information on the EXEC statement see DOS/VSE System Control Statements.

Where:

PGM=progname Represents the name of the program in the core image library to be executed. The program name corresponds to the first or only phase of the program in the library. If the program to be executed has just been processed by the linkage editor, the program name is omitted and the PGM keyword cannot be used.

To execute the VS FORTRAN compiler, you specify:

```
PGM=VFORTRAN, SIZE=AUTO
```

REAL Indicates that the job step started by EXEC will be executed in real mode. If REAL is not specified, the job step is always executed in virtual mode.

SIZE=size Defines how large a partition is needed for the program to be executed. The SIZE parameter can be specified in combination with REAL or without REAL.

If specified with REAL, it gives the size of that part of the real address area of the partition that will be needed by the job step.

If the SIZE parameter is omitted and REAL is specified, the entire real address area of the partition is reserved for the job step.

If SIZE is used without REAL, it specifies the size of that part of the virtual partition that will be directly available to the job step.

SIZE (without REAL) must always be specified for VSAM files.

AUTO indicates that the program size, as calculated by the system from information in the core image directory, is to be taken as the value for SIZE.

GO Specifies, for a language translator step, that the program is to be link-edited and executed automatically after it has been compiled. Only the source program data and any additional input data for the execution step are required after the language translator step. If a serious error is encountered in either the language translator step or the link-edit step, the input stream is flushed to the end-of-job (/&) statement.

PARM='value' Specifies information to be passed to the program at execution. 'value' can be up to 100 characters in length, enclosed by apostrophes. (The enclosing apostrophes are not passed to the program.) An apostrophe within value must be coded as two apostrophes.

Use this field to specify the compiler options documented in Appendix B.

PROC=procname Indicates the name of the procedure to retrieved from the procedure library.

OV Indicates that overriding statements follow EXEC. The overriding statements must be submitted on the same device as EXEC PROC.

If you execute a nonending job from the procedure library, you cannot update the procedure library.

EXTENT Control Statement

The EXTENT statement defines each data area or extent of a file. It is required for output files. For output files, one EXTENT statement must follow the DLBL statement.

Certain fields of the EXTENT statement may be omitted. Since these fields have no specific default values, the system will use information from previous EXTENT statements. However, if no previous EXTENT statement exists, an error will be produced and further processing will not occur.

ID	Operation	Operand
//	EXTENT	[logical unit] [,serial no.] [,type] [,sequence no.] [,relative track | block] [,no. of tracks | blocks] [,split cylinder tracks]

The EXTENT statement has all the operands shown in the format above; however, only the required and FORTRAN specific operands are described here. For additional information on the EXTENT statement, see DOS/VSE System Control Statements.

where:

logical unit Indicates the name of the logical unit of the volume for which this extent is effective. It is specified as six alphameric characters in the form SYSxxx. This field is not required for VS FORTRAN files.

serial no. Indicates the volume serial number for which this extent is effective. It is specified as 1 to 6 numeric characters. If fewer than six characters are used, the field is padded to the left with zeros. If the field is omitted, the volume serial number of the preceding EXTENT, if any, is used. Otherwise, the serial number is not checked, and files may be destroyed or altered if the wrong volume is mounted.

type Indicates the type of extent to be used. It is specified as one of the following numeric characters:

1=data area (no split cylinder)

8=data area (split cylinder)

If omitted, 1 is assumed.

split cylinder tracks Indicates the upper track number for the split cylinder extents (see Appendix A).

JOB Control Statement

The JOB statement specifies the beginning of a job and must be the first statement in the job deck.

Format

ID	Operation	Operand
//	JOB	jobname [accounting information]

The JOB statement has all the operands shown in the format above; however, only the required and FORTRAN specific operands are described here. For additional information on the JOB statement, see DOS/VSE System Control Statements.

where:

jobname Is the symbolic name of the job. It is required and must be from 1 to 8 alphameric characters.

accounting information If accounting information is specified, it must be separated from the job name by a single blank.

OPTION Control Statement

The OPTION statement allows the programmer to specify one or more job control options which temporarily override the system defaults. These options will remain in effect until a contrary option is encountered or until a /& control statement is read. However, if more than one linkage edit step occurs in a single job, OPTION LINK or OPTION CATAL must be specified for each execution of the linkage editor. All options are reset to the installation standard at the end of a job. If the statement is not supplied, the standard options apply.

VS FORTRAN compiler options are accepted only as a PARM field of the EXEC statement or on a @PROCESS statement.

Format

ID	Operation	Operand
//	OPTION	option1[,option2,...]

Any number of operands may be specified on the OPTION statement. If multiple options are indicated, they should be separated by commas. For a complete list of the compiler options that can be specified in the EXEC statement, see Appendix B. Some of the system options that can be specified are:

where:

CATAL A phase or program is cataloged in the core image library at the completion of a link-edit run. CATAL also sets the LINK option.

DUMP Dumps the partition, the system GETVIS area, and the SVA phase in error if the error occurred in the SVA.

PARTDUMP Dumps the partition in which the program is executing.

NODUMP Suppresses the DUMP or PARTDUMP option.

LINK Indicates that the object module is to be link-edited. When the LINK option is used, the output of the language translator is written on SYSLNK. The LINK option must always precede an EXEC LNKEDT statement in the input stream.

NOLINK Suppresses the LINK option. The language translators can also suppress the LINK option if the problem program contains an error that would preclude the successful execution of a problem program.

For a detailed description of this job control statement, see DOS/VSE System Control Statements.

TLBL Control Statement

The TLBL statement supplies file label information for the tape files with standard labels. The TLBL statement may be used with both EBCDIC or ASCII files. The only required entry is the filename. The default options described apply only to output files. If an option is omitted for an input file, the field is ignored. An omitted field is signified by a comma.

Format

ID	Operation	Operand
//	TLBL	filename [,'file-ID'] [,date] [,file-serial-no.][1] [,set-identifier][2] [,volume sequence no.] [,file-section-number] [,data set sequence no.] [,generation no.] [,version no.]

[1]Indicates EBCDIC files

[2]Indicates ASCII files

The DLBL statement has all the operands shown in the format above; however, only the required and FORTRAN specific operands are described here. For additional information, see DOS/VSE System Control Statements.

where:

filename Is required and corresponds to the DOS/VSE logical unit.

'file-ID' Indicates the name associated with the file on the volume. It is specified from 1 to 17 alphameric characters enclosed in apostrophes and may contain embedded blanks. If omitted on an output file, the filename is used.

file-serial-no. (EBCDIC) Indicates the volume serial number of the first or only reel in the file. It can be from 1 to 6 alphameric characters; if fewer than 6 characters are specified, the field is right-justified and padded with zeros.

set-identifier (ASCII) Indicates the volume serial number of the first or only reel in the file. It must be specified as 6 characters in length.

LINKAGE EDITOR CONTROL STATEMENTS—DOS/VSE

The linkage editor control statements prepare the object modules produced by the VS FORTRAN compiler for execution. This is accomplished by resolving external differences to library modules or other object modules. The linkage editor then takes the compiler output and combines it with required library modules, and with other modules, if indicated by the program.

ACTION Linkage Editor Control Statement

The ACTION statement is used to specify linkage editor options. When used, the statement must be the first linkage editor record(s) in the input stream. If multiple operands are required, they can be placed in separate ACTION statements or in one ACTION statement separated by commas.

If no ACTION statement is provided but SYSLST is assigned, ACTION MAP is assumed; if SYSLST is not assigned, ACTION NOMAP is assumed.

At least one blank must precede ACTION.

Format

Operation	Operand
ACTION	[CLEAR][,MAP|,NOMAP][,NOAUTO] [,CANCEL][,BG|,Fn][,REL|,NOREL]

The ACTION statement has all the operands shown in the format above; however, only the required and FORTRAN specific operands are described here. For additional information on the ACTION statement, see DOS/VSE System Control Statements.

Where:

CLEAR Sets the unused portion of the core image library to binary zeros prior to beginning the linkage editor function. However, clearing storage in this manner is time consuming and should be avoided.

MAP Indicates that SYSLST is available for diagnostic messages, including a listing of all linkage editor control statements processed. In addition, a map of virtual storage is printed on SYSLST, which can be used for problem determination.

If the MAP operand is specified, SYSLST must be assigned. If the ACTION statement is not used and SYSLST is assigned, MAP is assumed and a map of the virtual storage partition and any error diagnostics are printed on SYSLST. If the statement is not used and SYSLST is not assigned, NOMAP is assumed.

NOMAP Indicates that SYSLST is not available when performing the link-edit function. Storage mapping is not performed, and all linkage editor error diagnostics are listed on SYSLOG.

NOAUTO Suppresses AUTOLINK, the automatic-linking facility, during the link-editing of the entire program. AUTOLINK will be suppressed for both the private and the system relocatable libraries.

The NOAUTO operand in a PHASE statement indicates to the linkage editor that AUTOLINK is to be suppressed for that phase only. If an entire program requires NOAUTO, then specifying ACTION NOAUTO cancels AUTOLINK during link-editing of the entire program, thereby eliminating the necessity of specifying NOAUTO in each PHASE statement.

CANCEL Is used to request immediate job cancellation if certain linkage editor errors occur. If this option is not specified, the job continues.

ENTRY Linkage Editor Control Statement

The ENTRY control statement optionally completes linkage editor control statement processing.

Format

Operation	Operands
ENTRY	[entrypoint]

At least one blank must precede ENTRY.

The ENTRY statement has all the operands shown in the format above; however, only the required and FORTRAN specific operands are described here. For additional information on the ENTRY statement, see DOS/VSE System Control Statements.

where:

entrypoint Symbolic name of an entry point. It must be the name of a label definition (source ENTRY) defined in the first phase. This address is used as the transfer address to the first phase in the program. If the operand field is blank, the linkage editor uses as a transfer address the first significant address provided in an END record encountered during the generation of the first phase. If no such operand is found on the END card, the transfer address is the load address of the first phase.

It is necessary to supply the ENTRY statement only if a specific entry point is desired. When EXEC LNKEDT is read, job control writes an entry statement, with a blank operand, on SYSLNK to ensure that an ENTRY statement will be present to halt link-editing.

INCLUDE Linkage Editor Control Statement

The INCLUDE statement indicates that an object module is to be included for editing by the linkage editor.

INCLUDE is used to specify that a module from the relocatable library, or SYSIPT, is to be included in the present phase. A number of modules may be included for any one phase. The location of the INCLUDE statement determines the position of each module within the phase.

Format

Operation	Operand
INCLUDE	[modulename] [,(namelist)]

At least one blank must precede INCLUDE.

Where:

modulename Specifies the name of a module cataloged in the relocatable library. It can be from 1 to 8 alphameric characters. When the module is compiled, it must have been cataloged using the CATALR statement.

If both operands are omitted, the object module is assumed to be on SYSIPT. The module would be in deck form as a result of the compiler option DECK being specified in the OPTION statement.

For a detailed description of this linkage editor control statement, see DOS/VSE System Control Statements.

PHASE Linkage Editor Control Statement

The PHASE statement specifies a name for the phase that the linkage editor is to produce, and an origin point that specifies the address at which the phase is to be loaded. It provides a number of uses, chiefly that of supplying a name for a phase that is to be cataloged in the core image library.

The first (or only) object module input for the linkage editor should include a PHASE statement before the first ESD item. If no PHASE statement is used, or if the PHASE statement is in error, the linkage editor constructs a dummy statement. This allows testing of the program when the LINK option is used. However, the program with the dummy PHASE statement cannot be cataloged in a core image library; when the CATAL option is used, the job is cancelled.

Format

Operation	Operand
PHASE	name,origin [,NOAUTO]

At least one blank must precede PHASE.

The PHASE statement has all the operands shown in the format above; however, only the required and FORTRAN specific operands are described here. For additional information on the PHASE statement, see DOS/VSE System Control Statements.

where:

name Is required and indicates the symbolic name given to the phase. It can be from 1 to 8 alphameric characters, the first of which may not be a dollar sign ($).

For multiphase programs, 5 to 8 alphameric characters; the first four characters of each phase-name should be the same, but should not be identical to the first four characters of any other phase-name in the core image library. An asterisk (*) or a dollar sign ($) cannot be used as the first character of the phase-name.

origin Is required and specifies the load address of the phase. The forms most used for VS FORTRAN are:

- ***** Specifies that the phase be assigned the next available doubleword main storage address after the previous phase. It is specified as shown.
- **S** Specifies that the phase is to be loaded at the first available address in the problem program area. It is specified as shown.
- **ROOT** Indicates that the phase will always be resident in main storage during program execution. It is specified only on the first PHASE statement, and only on multiphase programs. It is specified as shown.
- **NOAUTO** Suppresses the automatic-linking facility for this linkage editor job step. It is specified as shown.

DOS/VSE LIBRARIAN PROGRAMS

The librarian is a group of programs used for maintaining DOS/VSE libraries and for providing printed and punched output from the libraries. The libraries are:

- Core Image Library (System and Private)

  Contains the system control components and program phases.

  To catalog a load module (phase) in this library, you must specify the following statement in your link-edit step:

  ```
  // OPTION  CATAL
  ```

- Relocatable Library (System and Private)
  relocatable library, DOS/VSE

  Contains the system control components and compiler logical input/output control system (LIOCS) object modules.

  To catalog your object module to this library, you use the CATALR statment.

- Source Statement Library (System and Private)

  Contains sequences of source statements, and macro definitions.

  To catalog your source programs to this library you use the CATALS statement.

- Procedures Library (System Only)

  Contains cataloged procedures.

  To catalog a procedure to this library, you use the CATALP statement.

Maintenance and service are the major functions performed for both the private and the system libraries by the librarian programs. Maintenance includes the addition, deletion, or copying of the items in the library. Service includes the translation of information in a library to printed or punched form. Information in a library directory and in header records can also be displayed.

For a more detailed description of these libraries and how to catalog in them, see DOS/VSE System Management Guide. For guidance information on cataloging to the different libraries, see the VS FORTRAN Application Programming: Guide.

PART III. VM/370-CMS SYSTEM SERVICES

You can create, compile, load, and execute VS FORTRAN programs using VM/370 CMS. An important aspect of FORTRAN programming under CMS is the creation and management of CMS files. Files must be created to hold your VS FORTRAN source programs. The compiler, in processing programs, creates files that contain its listing and the executable code it produces. Some of the programs, during execution, may process or create files containing data. This section describes files in a FORTRAN context. For additional information on CMS file management, see IBM VM/370: CP Command Reference for General Users.

VS FORTRAN SOURCE PROGRAMS UNDER CMS

Before invoking the VS FORTRAN compiler, a VS FORTRAN source program must be available in a CMS file on one of your disks. The source program is usually created using the CMS EDIT facilities, and may be written in either fixed or free format. The files may be created at the terminal just prior to compiling them or they may be old files, created some time ago, which need recompilation. In either case, all VS FORTRAN source files must have a CMS identifier and file characteristics that conform to VS FORTRAN compiler requirements.

For detailed information on creating and compiling your program using CMS, see VS FORTRAN Application Programming: Guide. For detailed information on using CMS commands from a terminal, see VM/370 CMS Terminal User's Guide.

VS FORTRAN SOURCE FILES

Every FORTRAN source file to be compiled requires a file identifier with the following format:

Format

```
filename   filetype [filemode]
```

where:

filename Is any valid CMS filename. A valid name consists of from 1 to 8 alphameric characters, which may be any combination of the following:

- Uppercase letters A through Z
- Lowercase letters a through z
- Numbers 0 through 9
- Extended characters @, #, or $

filetype Is one of the following:

FORTRAN - source program file

DATA - data file

EXEC - EXEC procedures

filemode Is any valid CMS filemode. For detailed information on filemodes, see IBM VM/370: Command Reference for General Users.

CMS RETURN CODES

CMS produces a return code following the execution of a VS FORTRAN compilation. It appears in the ready message and corresponds to the highest diagnostic message severity level encountered during compilation.

Example

```
R(00004);
```

Figure 9 lists the possible return codes.

Code	Meaning
00000	No errors were detected. However, there may be information messages. If an information message is produced, check your program for possible errors, as the reliability of your results may be in doubt.
00004	Possible errors were detected or warning messages were issued. Execution of your program should be successful, but the results may not be reliable.
00008	Errors were detected. Compilation continues, but execution may fail. If specified, an object module will be created, but you may not be able to execute it.
00012	Severe errors were detected. Compilation may not continue; if it does, execution of the program is impossible.
00016	Extremely severe errors were detected. Compilation terminates at the point at which the error was detected.

Figure 9. CMS Return Codes

DEFINING EXECUTION-TIME FILES

The following discussion describes how files are defined through the use of the CMS FILEDEF command, and how they are identified to the system through the use of file identifiers and VS FORTRAN input/output statements.

PREDEFINED FILES

All execution-time files that you want to use must be defined to CMS. Three files are predefined:

Sequential

- Terminal input
- Terminal output
- Punched card output

The three predefined files are provided for you by the VS FORTRAN initialization routine, which also supplies the FILEDEF for these files. These files are recognized by CMS when you refer to them in VS FORTRAN input or output statements with the VS FORTRAN data set reference number that has been assigned to them. Since they are predefined, you do not have to supply a FILEDEF command for them, unless you want to change their definition or create new files to replace them. See Figure 10.

VS FORTRAN Data Set Reference Number	Used in the Following VS FORTRAN Input and Output Statements	Identifies	Requires Records of the Following Format and Maximum Length
5	READ (5,b)list READ (5,*)list	Terminal input	Fixed-length, unblocked (F), 130 characters long.
6	WRITE (6,b)list WRITE (6,*)list	Terminal output	Fixed-length, unblocked (F), 131 characters long.
7	WRITE (7,b)list	Punched Card output	Fixed-length, unblocked (F), 80 characters long.

Figure 10. Predefined Files—Data Set Reference Numbers and Record Formats

USER-DEFINED FILES

User-defined files containing data that you will want your program to process may already exist in your system. Conversely, you may want your program to create a file to hold data that was generated during its execution.

All such files you must define with a FILEDEF command; since they are not predefined, they cannot be identified by CMS and associated with your program. FILEDEF is used in conjunction with the data set reference numbers in your VS FORTRAN input and output statements and the identifier of the file that you want to use or create.

You can, in addition, define the following files to be used in place of the system's predefined files or change them to suit your own needs:

Sequential

- Terminal input
- Terminal output
- Punched card output

Whether a file is sequential or direct access will, to a great extent, determine how a record is defined, the way it is identified, and how it is referred to in a VS FORTRAN program.

Regardless of the type of file you are using, there are several general guidelines that must be followed in defining and using files.

- Each file used in your program must be defined to the system (either through system-supplied definition or one that you supply).
- Do not use the same definition for more than one file.
- Do not use the same file on more than one type of device in the same program.
- You may refer to the same file from more than one program through different devices and access methods, if you change the file and its definition appropriately before using it.

CP COMMANDS

The commands listed in the following tables are those most frequently used by the VS FORTRAN programmer. This table is intended to be used as a reference tool by providing easy lookup of those commands used most often.

CP Commands	Function
BEGIN	Returns the system to the CMS environment and resumes execution of a program.
IPL	Simulates an initial program load for the user's virtual machine.
LOGON	Identifies the user to VM/370.
LOGOFF	Terminates the terminal session.
QUERY[1]	Types out the status of the user's virtual configuration and user-defined parameters.
SET[1]	Sets operational characteristics for the user's virtual machine.
TERMINAL	Sets operational characteristics for the user's terminal.

Note: These commands are both CP and CMS commands.

CMS COMMANDS

This section contains reference information for the CMS commands used most often by the VS FORTRAN programmer. Each command description gives the format, operands, and options available.

For a more complete description of the commands that follow, see IBM VM/370: CMS Command and Macro Reference.

AMSERV COMMAND

The AMSERV command invokes Access Method Services to:

- Define VSAM catalogs, data spaces, or clusters
- Alter, list, copy, delete, export, or import VSAM catalogs and data sets

Format

Command	Operand
AMserv	fn1 [fn2] [(options...[)]] [fn1]

For a description of all the options that can be specified and additional information on the use of this command, see IBM Virtual Machine Facility/370: CMS Command and Macro Reference.

where:

fn1 Specifies the filename of a CMS file with a filetype of AMSERV that contains the Access Method Services control statements to be executed. CMS searches all your accessed disks, using the standard search order, to locate the file.

fn2 Specifies the filename of the CMS file that is to contain the Access Method Services listing; the filetype is always LISTING. If fn2 is not specified, the LISTING file will have the same name as the AMSERV input (fn1).

The LISTING file is written to the first read/write disk in the standard search order, usually your A-disk. If a LISTING file with the same name already exists, it is replaced.

options The following options may be specified:

PRINT
TAPIN
TAPOUT

DLBL COMMAND

In CMS, the DLBL command defines and identifies VSAM catalogs, clusters, and data spaces. DLBL also identifies VSAM files used for program input/output, and identifies input/output files for AMSERV.

Format

Command	Operands
DLBL	[ddname{mode } [DSN qual1...qualn]] {DUMMY } [DSN ?] [(options...[)]]]

For a description of all the options that can be specified and for further information on the use of this command, see the IBM Virtual Machine Facility/370: CMS Command and Macro Reference.

Where:

ddname Specifies a 1- to 7-character program ddname (OS/VS) or filename (DOS/VSE).

mode Specifies a valid CMS disk mode letter and optionally, a filemode number. A letter must be specified; if a number is not specified it defaults to 1. The disk must be accessed when the DLBL command is issued.

DUMMY Indicates that no real input/output is to be performed. A read results in an end-of-file condition and a write results in a successful return code. DUMMY should not be used for OS/VS VSAM data sets (a dummy data set does not return, on open, an end-of-file indication).

DSN Indicates non-CMS files.

? Indicates that the data set name will be entered interactively.

qual1...qualn Indicates an OS/VS data set name or DOS/VSE file-id.

options The following options may be specified:

VSAM - indicates a VSAM file. This option must be specified for VSAM functions except when the EXTENT, CAT, or BUFSP options are specified.

EXTENT - specifies extent information, and indicates that Access Method Services will be used to define a VSAM catalog, or data space.

MULT - indicates an existing multivolume file specification.

CAT catdd - identifies the VSAM catalog, defined by a previous DLBL, which contains the entry for the file.

BUFSP nnnnnn - specifies the number of bytes to be used for I/O buffers by VSAM during program execution.

EDIT COMMAND

The EDIT command invokes the CMS editor to create, modify, and manipulate CMS disk files. Once the editor is invoked, only the EDIT subcommands, edit macro requests, and input data to the disk file can be executed. However, a limited number of CMS commands may be executed in the CMS subset mode, entered from the edit environment.

Control is returned to the CMS environment by issuing the EDIT subcommands FILE or QUIT. For a complete list of the EDIT subcommands, see the "EDIT Subcommands" section.

Format

Command	Operands
Edit	fn ft [fm] [(options...[)]]

Additional information on the use of this command and the EDIT subcommands can be found in IBM Virtual Machine Facility/370: CMS Command and Macro Reference.

where:

fn Specifies the filename of the file to be created or edited. If the file specified is a new file, you issue the INPUT subcommand, and all data lines entered become input to the new file. If the file specified exists, you issue the EDIT subcommands to modify the file.

ft Specifies the filetype of the file to be created or edited. Filetype may be specified as :

FORTRAN - for FORTRAN source files

AMSERV - VSAM control commands

EXEC - for EXEC procedures

DATA - for data files

fm Specifies the filemode of the file to be edited, indicating the disk on which the file resides. The editor determines the filemode of the edited file as follows:

If fm is specified as an asterisk (*), all accessed disks are searched for the specified file.

options The following options may be specified:

LRECL nn is the record length of the file to be created or edited. This option is used to override the default value supplied by the editor.

NODISP forces a display terminal into line (typewriter) mode. When this option is in effect, all subcommands such as SCROLL, SCROLLUP, FORMAT, and CHANGE (with no operands) that control the display on a terminal are made invalid for the edit session.

EDIT Subcommands

The EDIT command allows you to enter EDIT mode and makes the following subcommands available to you for file creation and alteration. Entering a null line puts the editor into INPUT mode.

EDIT Subcommands	Usage
BOTTOM	Moves the editor's pointer to the last line of the file.
CHANGE	Replaces a string of characters with another in one or more lines.
DELETE	Deletes one or more lines from a file.
DOWN	Moves the editor's pointer to a subsequent line.
FILE	Places a file on the user's disk and leaves EDIT mode.
FIND	Performs a string search, which is column-dependent, for the specified group of characters.
FMODE	Changes the filemode of a file.
FNAME	Changes the filename of a file.
GETFILE	Includes a part of an existing file in the file being created.
INPUT	Enters INPUT mode and accepts subsequent lines as part of the file being created.
LOCATE	Performs a string search, which is not column-dependent, for the specified group of characters. The search begins with the current position of the editor's pointer.
NEXT	Moves the editor's pointer to the next line of a file.
QUIT	Terminates the operation of the editor without effecting any modifications.
REPLACE	Replaces a line with one or more lines.
TOP	Moves the editor's pointer to the null line at the beginning of a file.
TYPE	Types all or part of a file being created.
UP	Moves the editor's pointer to a previous line of a file.
VERIFY	Controls the typing of any lines that have been changed or replaced.

EXEC COMMAND

The EXEC command executes one or more CMS commands or EXEC control statements contained in a specified EXEC file.

Format

Command	Operands
[EXec]	fn [args...]

where:

[EXEC] Indicates that the EXEC command may be omitted if executing the EXEC procedure from the CMS command environment and the SET IMPEX OFF command has not been issued.

fn Indicates the filename of a file containing one or more CMS commands and/or EXEC control statements to be executed. The filetype of the file must be EXEC and the file can have either fixed- or variable-length records with a logical record length not exceeding 130 characters. EXEC files can be created with the EDIT command or by a user program. EXEC files created by the CMS editor have, by default, variable-length, 80-character records.

args Indicates any arguments you may pass to the EXEC. The arguments are assigned to the special variables &1 through &30 in the order in which they appear in the argument list.

For additional information on the use of this command, see IBM Virtual Machine Facility/370: CMS Commands and Macro Reference.

FILEDEF COMMAND

The FILEDEF command defines a file and its input and output devices.

Format

Command	Operands
FIledef	{ddname} {device} ([option|option]...) {xx}

Where:

ddname The data definition name provides the link between your VS FORTRAN input and output statements and the file that you want to process or create. The standard VS FORTRAN data definition name has the following format:

FTxxFyyy

Where:

xx is a VS FORTRAN data set reference number. This number must be used in any VS FORTRAN input or output statements that refer to the file being defined. The number 00 must not be specified as a data set reference number. The data set reference number may range from 01 to a maximum value determined at the time the sytem was installed.

yyy Is a sequence number (ranging from 001 to 999) that identifies multiple files under the same data set reference number. For direct access files, this number will vary depending upon the order in which the file is referred to in your program.

xx A VS FORTRAN data set reference number used alone performs the same function as the ddname; however, it cannot be used for multifiles, since it generates a default ddname of FTxxF001. The data set reference number specified must be used in your VS FORTRAN input and output statements that refer to the file being defined.

device Specifies the input/output device on which the file will reside. The following devices may be specified:

TERMINAL Specifies that your user-defined file is to be read or written at the terminal. With this option you can either change the characteristics of the system's predefined file or create a new file for the terminal.

UPCASE—Specifies that the data entered at the terminal will appear on the printout in uppercase characters. This is the default for the option.

LOWCASE—Specifies that the data entered at the terminal will appear on the printout in lowercase characters.

PRINTER Indicates that your user-defined file is to be written on an offline printer. Files defined as PRINTER may only be used for output.

PUNCH Indicates that your user-defined file is to be punched into a card deck. Files defined as PUNCH may only be used for output.

READER Indicates that you want to read a user-defined file consisting of a deck of cards. Files defined as READER may only be used for input.

DISK Indicates you want to use an existing disk file or create a new one.

If you are reading an existing file, you must include its file identifier (fn ft [fm]). If you are creating a new disk file, you must supply your own file identifier; if you do not, the system will supply the following default file identifier:

FILE FTxxFyyy A1

For files defined as spanned (VS or VBS) the file mode must be specified as 4.

XTENT{nn|50}

Must be included in each FILEDEF command that defines a VS FORTRAN direct access file. The variable nn should correspond to the number of records that you specified in the OPEN statement. If you do not include this option, a value of 50 records is assumed.

DISP MOD

Indicates that the read/write pointer is to be positioned after the last record in the disk file.

DUMMY Indicates that no real input or output operation is to be performed for a disk file. This option may be used in place of the DISK option only. You may specify any of the disk options; however, they will be ignored.

FILEDEF Tape Options

TAPn Indicates that you want to use an existing tape file or create a new one. The variable n represents the tape unit number and can range from 1 through 4. The numbers correspond to tape units attached to your virtual machine addresses 181 through 184.

Specify these valid FILEDEF options for tape files.

nTRACK Indicates the type of tape device being used. The variable n represents the number of tracks that the tape device records on. It can be specified as 7 or 9.

TRTCH aa Used for 7-track tapes to indicate the recording technique being used. The variable aa is a code that specifies the parity, converter, and translator settings. Figure 11 lists the possible tape recording specifications available for this option.

aa	parity	converter	translator
OC	odd	on	off
OT	odd	off	on
O	odd	off	off
ET	even	off	on
E	even	off	off

Figure 11. Tape Recording Specifications

DEN nnnn Indicates the density of the tape being used. For additional information on tape densities see Appendix A.

FILEDEF Options Specifying Format and Logical Characteristics

RECFM aaa[a] Indicates the format of a set of VS FORTRAN records being read or written and whether they contain carriage and print control characters. The control characters A and M can be used with any valid RECFM setting (for example, FA, FBA, VA or VBA). The variables aaa and a are codes that represent the possible record formats and control characters. Figure 12 and Figure 13 list the possible record formats and control characters that can be specified.

aaa	Record Format
F	Fixed-length records (must be specified for direct files)
FB	Fixed-length, blocked records
V	Variable-length records
VB	Variable-length, blocked records
U	Undefined-length records
FS	Fixed-length, standard block records
FBS	Fixed-length, blocked, standard records
VS	Variable-length, spanned records
VBS	Variable-length, blocked, spanned records

Figure 12. Record Formats for the RECFM Option

a	Control Characters
A	ANSI carriage control character
M	machine control characters

Figure 13. Control Character Specifications for the RECFM Option

LRECL nn Indicates the the maximum length of a VS FORTRAN record. The record format, which you specified in RECFM, determines how you must specify LRECL. The criteria for determining logical record lengths are listed in Figure 14.

For RECFM	LRECL Must
F, FB, FS, or FBS	Specify the actual size of the records (must be specified for direct files)
V, VB, VS, or VBS	Specify the size of the longest record
U	Be omitted

Note: The maximum LRECL that can be specified is 65K bytes.

Figure 14. Criteria for Determining a Value for LRECL

BLOCK BLOCK or BLKSIZE indicates the maximum amount of space required by one or more VS FORTRAN records that are to be read or written by a single input or output statement. Figure 15 lists the criteria for determining block size.

For RECFM	BLOCK must
F or FS	Specify the same value as LRECL.
FB or FBS	Specify a multiple of LRECL.
V or VS	Specify the value of LRECL plus 4 bytes for a segment descriptor word.
VB or VBS	Specify the value of LRECL plus 4 bytes for the segment descriptor word of each record that can be contained in the block plus 4 bytes for a block descriptor word.
U	Specify the greatest amount of space required to hold all the records that are to be grouped together.

Note: The maximum block value that can be specified is 65K bytes.

Figure 15. Criteria for Determining a Value for the BLOCK Option

PERM Indicates that the file characteristics specified in a FILEDEF command are to remain in effect until they are either explicitly cleared, or changed with a new FILEDEF command that has the CHANGE option.

CHANGE Indicates that, if a file definition exists for the ddname that is specified in this command, the options that are included will replace the corresponding options in the old FILEDEF command. This is the default if PERM, CHANGE, or NOCHANGE is not specified.

NOCHANGE Indicates that, if a file definition exists for the ddname specified in this FILEDEF command, the options that are included will not replace the corresponding options in the old FILEDEF command.

Additional information on the use of this command can be found in IBM Virtual Machine Facility/370: CMS Command and Macro Reference.

FORTVS COMMAND

The FORTVS command invokes the VS FORTRAN compiler. Your source program must be in the CMS file that you identify in the command. The FORTVS command allows you to specify a set of options governing compiler operation and output. If however, you omit one or more of the options, default values are assumed.

Format

Command	Operands
FORTVS	filename ([option|option][option|option]...)

where:

FORTVS Invokes the VS FORTRAN compiler. It must be specified as shown.

filename Specifies the name of the file containing the FORTRAN program to be compiled.

option Option can be any of the VS FORTRAN compiler options. For a complete list of the compiler options that can be specified, see Appendix B.

GLOBAL COMMAND

The GLOBAL command specifies the text or macro libraries to be searched in resolving external references in a program being loaded.

This command makes the VS FORTRAN libraries accessible to your disk storage before you load and execute your VS FORTRAN program.

Format

Command	Operands
GLobal	{TXTLIB} [libname1 ... libname8] {MACLIB}

where:

TXTLIB Precedes the specification of text libraries to be searched for missing subroutines when the LOAD or INCLUDE command is issued, or when a dynamic load occurs.

MACLIB Precedes the specification of macro libraries that are to be searched for macros and copy files during the execution of the compiler. The macro libraries may be CMS or OS/VS files. For OS/VS files, a FILEDEF command must be issued for the file before you issue the GLOBAL command.

libname1 Indicates the filename of a library. As many as eight libraries may be given. Filetypes must be TXTLIB or MACLIB. The libraries are searched in the order they are named. If no library names are specified, the command cancels the effect of any previous GLOBAL command.

Check with your system administrator for the libraries available.

For more detailed information on the use of the GLOBAL command, see IBM Virtual Machine Facility/370: CMS Command and Macro Reference.

INCLUDE COMMAND

The INCLUDE command reads one or more TEXT files (containing relocatable object code) from disk and loads them into virtual storage, establishing the proper linkages between the files. In order for the INCLUDE command to produce the desired results, a LOAD command must have been previously issued.

The INCLUDE and LOAD commands are used to create a load module from one or more object modules (plus any needed VS FORTRAN library modules) when loading and executing your program under CMS.

Format

Command	Operands
INclude	fn... [(options...[)]]

where:

fn... Is the name of the files to be loaded into storage. Files must have a filetype of TEXT and consist of relocatable object code. If a GLOBAL TXTLIB command has identified one or more TXTLIBs, fn may indicate the TXTLIB member.

options If options were specified with a previous LOAD or INCLUDE command and if SAME is specified when INCLUDE is issued, these options (with the exception of CLEAR and ORIGIN) remain set.

The options that may be specified are:

CLEAR/NOCLEAR
clears the load area in storage to binary zeroes.

MAP/NOMAP
adds information to the load map.

TYPE/NOTYPE
displays the load map of the files at the terminal, in addition to writing it to disk.

AUTO/NOAUTO
searches your disk for TEXT files to resolve undefined references.

LIBE/NOLIBE
Searches the text libraries defined by the GLOBAL command for missing subroutines.

START
retains the same options (except ORIGIN and CLEAR) that were used by a previous INCLUDE or LOAD command.

For a description of the options that can be specified and for additional information on the INCLUDE command see IBM Virtual Machine Facility/370: CMS Command and Macro Reference.

LOAD COMMAND

The LOAD command reads one or more CMS or OS/VS TEXT files from disk and loads them into virtual storage, establishing the linkage between the files for execution.

The LOAD command is used to create and, optionally, execute a load module. Your input consist of your object module, VS FORTRAN library routines, and secondary input such as TEXT files of called subprograms.

Format

Command	Operands
LOAD	fn ... [(options...[)]]

where:

fn Specifies the names of the files to be loaded into storage. The files must have a filetype of TEXT and consist of relocatable object code. If a GLOBAL TXTLIB command has been issued, fn may indicate the name of a TXTLIB member.

options If conflicting options are specified, the last one entered is in effect. Options may be overridden or added when you use the INCLUDE command to load additional TEXT files.

The START option executes the program being loaded. Upon successful completion of the loading, START must be specified only if you want execution to begin immediately.

A list of other options that can be specified follows:

CLEAR/NOCLEAR
MAP/NOMAP
TYPE/NOTYPE
AUTO/NOAUTO
LIBRE/NOLIBRE
START

For an explanation of these options, see the INCLUDE command.

For further information on how to use this command in your program, see VS FORTRAN Application Programming: Guide. Also, see the IBM Virtual Machine Facility/370: Terminal User's Guide for details on the use of this command.

RUN COMMAND

The RUN EXEC procedure initiates a series of functions on a file. The RUN command compiles, loads, and starts execution of the file specified, depending on the filetype.

Format

Command	Operands
RUN	fn [ft [fm]] [(args...[)]]

where:

fn Is the filename of the file to be compiled, link-edited, and executed.

ft Is the filetype of the file to be compiled, link-edited, and executed. If the filetype is not specified, a search is made for a file with the specified filename and the filetype of EXEC, MODULE, or TEXT. If the filetype of an input file is FORTRAN, the VS FORTRAN compiler is invoked to compile the source statements and produce a TEXT file. Then, LOAD and START are called to initiate program execution. The valid filetypes and resulting action for the command are given in Figure 16.

Filetype	Action
EXEC	The EXEC processor is called to process the file.
MODULE	The LOADMOD command is issued to load the program into storage, and the command executes the program, beginning at the entry point equal to fn.
TEXT	The LOAD command brings the file into storage in an executable format, and the START command executes the program, beginning at the entry point named by fn.
FORTRAN	The FORTRAN compiler is invoked to compile, link-edit, and execute the FORTRAN source program.

Figure 16. Valid RUN Filetypes and Resulting Actions

fm Is the filemode of the file to be compiled, link-edited, and executed. If this field is specified, a filetype must be specified. If fm is not specified, the default search order is used to search your disks for the file.

args Are arguments you want to pass to your program. Up to 13 arguments can be specified in the RUN command, provided they fit on a single input line. Each argument is left-justified, and any argument more than eight characters long is truncated on the right. If the input file is FORTRAN, args can be compiler options. See Appendix B for the options that can be specified.

For further details on the use of the RUN command, see IBM Virtual Machine Facility/370: CMS Command and Macro Reference.

The START command begins execution of a previously loaded program.

Format

Command	Operands
START	[entry [args...]] [*]

where:

entry Passes control to the control section name or entry point name at execution time. Entry may be a filename only if the filename is identical to a control section name or an entry point name.

***** Passes control to the default entry point.

args Are arguments to be passed to the started program. If user arguments are specified, entry or * must be specified; otherwise, the first argument is taken as the entry point. Arguments are passed to the program via general register 1. The entry operand and any arguments become a string of doublewords, one argument per doubleword, and the address of the list is placed in general register 1.

Note: Any undefined names or references specified in the files loaded into storage are defined as zero. Thus, if there is a call or branch to a subroutine from a main program, and if the subroutine has never been loaded, the call or branch transfers control to location zero of the virtual machine at execution time.

Additional information on the use of this command is given in IBM Virtual Machine Facility/370: CMS Command and Macro Reference.

PART IV. OS/VS2 TSO SYSTEM SERVICES

You can create, compile, load, and execute VS FORTRAN programs using OS/VS2 TSO. An important aspect of FORTRAN programming under TSO is the creation and management of TSO files. Files must be created to hold your VS FORTRAN source programs. The compiler, in processing programs, creates files that contain its listing and the executable code it produces. Some of the programs, during execution, may process or create files containing data. This section describes files in a FORTRAN context. For additional information on CMS file management, see OS/VS2 TSO: Command Language Reference.

VS FORTRAN SOURCE PROGRAMS UNDER TSO

Before invoking the VS FORTRAN compiler, a VS FORTRAN source program must be available in a TSO file. The source program is usually created using the EDIT facilities, and may be written in either fixed or free format. The files may be created at the terminal just prior to compiling them or they may be old files which need recompilation. In either case, all VS FORTRAN source files must have a TSO identifier and file characteristics that conform to VS FORTRAN compiler requirements.

For detailed information on creating and compiling your program using TSO, see VS FORTRAN Application Programming: Guide. For detailed information on using TSO commands from a terminal, see OS/VS2 TSO Terminal User's Guide.

USER-DEFINED FILES

User-defined files, containing data that you will want your program to process may already exist in your system. Conversely, you may want your program to create a file to hold data that was generated during its execution. You must define all such files with an ALLOCATE command; since they are not predefined, they cannot be identified by TSO and associated with your program. ALLOCATE is used in conjunction with the data set reference numbers in your VS FORTRAN input and output statements and the identifier of the file that you want to use or create.

Whether a file is sequential or direct access will, to a great extent, determine how a record is defined, the way it is identified, and how it is referred to in a VS FORTRAN program.

Regardless of the type of file you are using, there are several general guidelines that must be followed in defining and using files.

- Each file used in your program must be defined to the system (either through system-supplied definition or one that you supply).
- Do not use the same definition for more than one file.
- Do not use the same file on more than one type of device in the same program.
- You may refer to the same file from more than one program through different devices and access methods, if you change the file and its definition appropriately before using it.

TSO DATA SET NAMING CONVENTIONS

A data set is a collection of related data. Each data set stored in the system is identified by a unique data set name. The data

set name allows the data to be retrieved and helps protect the data from unauthorized use.

The data set naming conventions for TSO simplify the use of data set names. When a data set name conforms to the conventions, you can refer to the data set by its fully qualified name or by an abbreviated version of the name.

TSO DATA SET NAMES

A data set name must be qualified in order to conform to the TSO data set naming conventions. The qualified name must consist of at least the two required fields of the following three:

1. Your user-prefix (required; defaults to userid; may be redefined using PROFILE command).
2. A user-supplied name (optional for a partitioned data set).
3. A descriptive qualifier (required).

Normally, all three names are used:

```
USER-PREFIX.USER-SUPPLIED-NAME.DESCRIPTIVE QUALIFIER
```

The total length of the data set name must not exceed 44 characters, including periods. A typical TSO data set name is:

```
WRRID.PARTS.DATA
```

User-prefix - WRRID
User-supplied name - PARTS
Descriptive qualifier - DATA

The TSO data set naming conventions also apply to partitioned data sets. A typical TSO name for a member of a partitioned data set is:

```
WRRID.PARTS.DATA(PART14)
```

User-Prefix: The user-prefix is always the leftmost qualifier of the full data set name. For TSO, this qualifier is the prefix selected in the PROFILE command. If no prefix has been selected, the userid assigned to you by your installation will be used.

User-Supplied Name: You choose a name for the data sets that you want to identify. It can be a simple name or several simple names separated by periods.

Descriptive Qualifier: The descriptive qualifier is always the rightmost qualifier of the full data set name. To conform to the data set naming conventions, this qualifier must be one of the qualifiers listed in Figure 17.

Descriptive Qualifier	Data Set Contents
CLIST	TSO commands
DATA	Uppercase text
FORT	VS-FORTRAN statements
LINKLIST	Output listings from linkage editor
LIST	Listings
LOAD	Load module
LOADLIST	Output listing from loader
OBJ	Object module
OUTLIST	Output listing from OUTPUT command
TESTLIST	Output listing from TEST command

Figure 17. Descriptive Qualifiers Useful for VS FORTRAN

HOW TO ENTER DATA SET NAMES

The data set naming conventions simplify the use of data set names. If the data set name conforms to the conventions, you need specify only the user-supplied name field (in most cases) when you refer to the data set. The system will add the necessary qualifiers to the beginning and to the end of the name that you specify. In some cases, however, the system will prompt you for descriptive qualifiers.

When you specify an entire fully qualified data set name, as you must do if the name does not conform to the TSO data set naming conventions, you must enclose the entire name within apostrophes, as follows:

'WRRID.PROG.LIST'	where WRRID is not your user identification.
or	
'WRRID.PROG.FIRST'	where FIRST is not a valid descriptive qualifier.

The system will not append qualifiers to any name enclosed in apostrophes.

Defaults for Data Set Names: When you specify only the user-supplied name, the system adds your user identification and, whenever possible, a descriptive qualifier, which the system attempts to derive from available information. For instance, if you specified ASM as an operand for the EDIT command, the system will assign ASM as the descriptive qualifier. If the information is insufficient, the system will issue a message at your terminal requesting the required information. If you specify the name of a partitioned data set and do not include a required member name, the system will use TEMPNAME as the default member name. (If you are creating a new member, the member name will become TEMPNAME; if you are modifying an existing partitioned data set, the system will search for a member named TEMPNAME.) Figure 18 shows the descriptive qualifiers useful for VS FORTRAN that are supplied by default.

Descriptive Qualifiers			
Command	**Input**	**Output**	**Listing**
CALL	LOAD	--	--
EXEC	CLIST	--	--
FORT	FORT	OBJ	LIST
LINK	OBJ	LOAD	LINKLIST
	LOAD	--	--
LOADGO	OBJ	--	LOADLIST
	LOAD	--	--
TEST	OBJ	--	TESTLIST
	LOAD	--	--

Figure 18. Descriptive Qualifiers Supplied by Default

USING COMMANDS FOR VSAM AND NON-VSAM DATA SETS

Figure 19 gives recommended commands, by function, for VSAM and non-VSAM data sets. Numbers in parentheses after the commands indicate order of preference. Refer to OS/VS2 Access Method Services for commands not covered in this document.

Function	Non-VSAM	VSAM
Allocate new DASD space	ALLOCATE	DEFINE
Connect data set to terminal	ALLOCATE	ALLOCATE
Catalog data sets	DEFINE(1) ALLOCATE(2)	DEFINE
List contents	EDIT	PRINT
Delete	DELETE	DELETE

Figure 19. Commands Preferred for VSAM/Non-VSAM Data Sets

TSO RETURN CODES

TSO produces a return code following the execution of a VS FORTRAN compilation. It appears in the ready message and corresponds to the highest diagnostic message severity level encountered during compilation.

Example

```
R(00004);
```

Figure 20 lists the possible return codes.

Code	Meaning
00000	No errors were detected. However, there may be information messages. If an information message is produced, check your program for possible errors, as the reliability of your results may be in doubt.
00004	Possible errors were detected, or warning messages were issued. Execution of your program should be successful, but the result may not be reliable.
00008	Errors were detected. Compilation continues but execution may fail. If specified, an object module will be created, but you may not be able to execute it.
00012	Severe errors were detected. Compilation may not continue; if it does, execution of the program is impossible.
00016	Extremely severe errors were detected. Compilation terminates at the point at which the error was detected.

Figure 20. TSO Return Codes

TSO COMMANDS

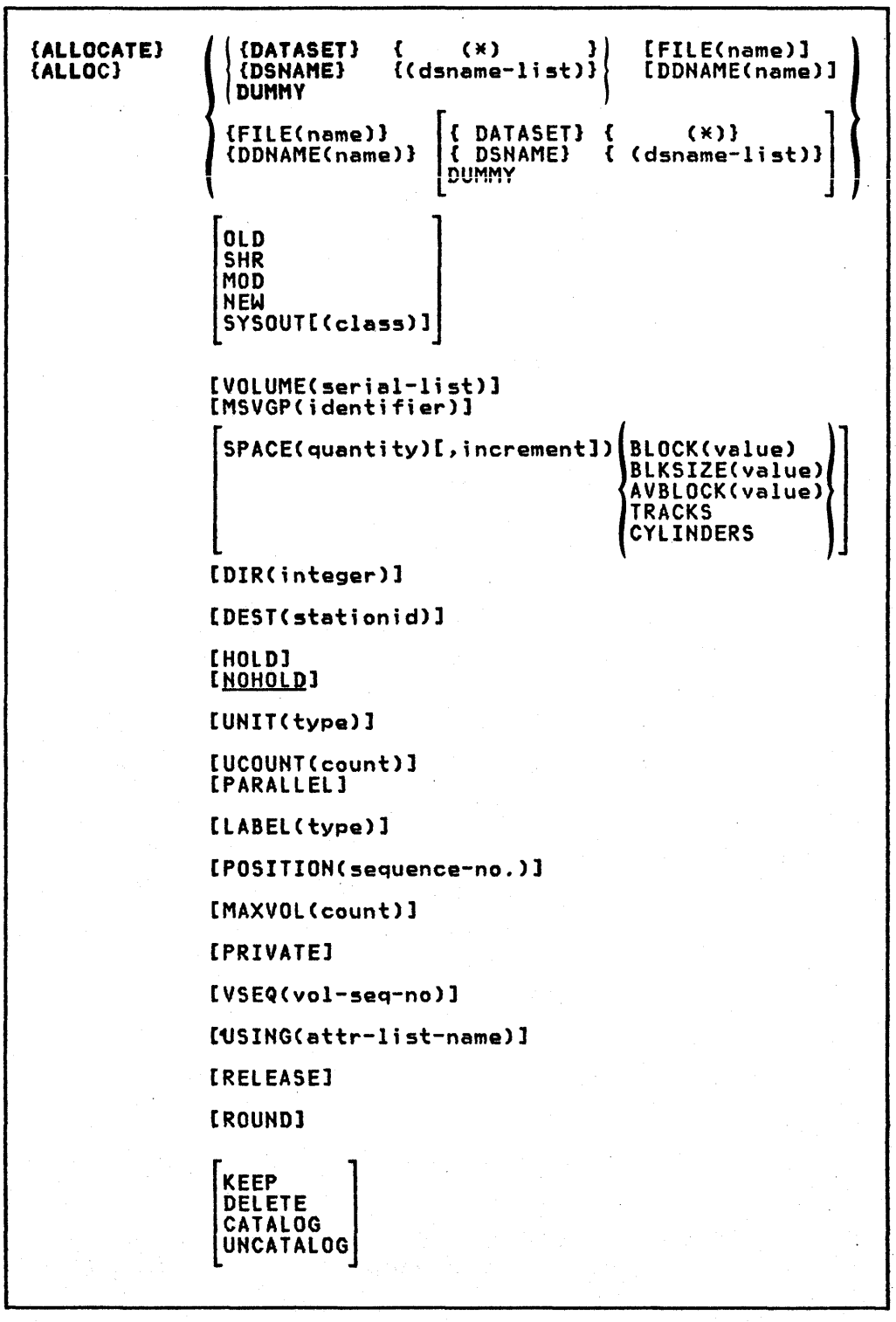
This section contains reference information for the TSO commands used most often by the VS FORTRAN programmer. Each command description gives the format, operands, and options available.

For a more complete description of the commands that follow, see OS/VS2 TSO: CMS Command Language Reference.

ALLOCATE COMMAND

Use the ALLOCATE command or the ALLOCATE subcommand of EDIT (function and syntax are identical to the ALLOCATE command) to dynamically allocate the data sets required by a program that you intend to execute.

Format



All the required and optional operands for the ALLOCATE command are shown in the format. Only those of interest to the VS FORTRAN programmer are described in the following paragraphs. For complete documentation, see OS/VS2 TSO Command Language Reference.

DATASET(dsname-list or *) or DSNAME(dsname-list or*)
specifies the name of the data set to be allocated. If a list of data set names is entered, ALLOCATE will allocate and concatenate non-VSAM data sets. The data set name must include the descriptive (rightmost) qualifier and may contain a member name in parentheses.

You may substitute an asterisk (*) for the data set name to indicate that you want to have your terminal allocated for input and output. If you use an asterisk (*), only the FILE or DDNAME, BLOCK or BLKSIZE, and USING operands should be entered. All other operands are ignored. No message is issued to notify the user.

DUMMY
specifies that no devices or external storage space is to be allocated to the data set, and no disposition processing is to be performed on the data set. Entering the DUMMY keyword will have the same effect as specifying NULLFILE as the data set name on the DATASET or DSNAME operand. If DUMMY is specified, only the FILE or DDNAME, BLOCK or BLKSIZE, and USING operands should be entered. All other operands are ignored.

FILE(name) or DDNAME(name)
specifies the name to be associated with the data set. It may contain no more than 8 characters. (This name corresponds to the name on the data definition (DD) statement in job control language and must match the ddname in the data control block (DCB) that is associated with the data set.) This name is the data set reference number that identifies a data set and has the form "FTxxFyyy"; for instance, FT06F002. If you omit this operand, the system assigns an available file name (ddname) from a data definition statement in the procedure that is invoked when you enter the LOGON command.

OLD
indicates that the data set currently exists and that you require exclusive use of the data set. The data set should be cataloged. If it is not, you must specify the VOLUME operand. OLD data sets are retained by the system when you free them from allocation. The DATASET or DSNAME parameter is required.

SHR
indicates that the data set currently exists but that you do not require exclusive use of the data set. Other tasks may use it concurrently. ALLOCATE assumes the data set is cataloged if the VOLUME operand is not entered. SHR data sets are retained by the system when you free them. The DATASET or DSNAME parameter is required.

MOD
indicates that you want to append data to the end of the data set. If the data set does not exist, a new data set is created. MOD data sets will be retained by the system when you free them. The DATASET or DSNAME parameter is required.

NEW
(non-VSAM only) indicates that the data set does not exist and that it is to be created. For new partitioned data sets you must specify the DIR operand. A NEW data set will be kept and cataloged if you specify a data set name. If you do not specify a data set name, it will be deleted when you free it or log off.

SYSOUT[(class)]
indicates that the data set is to be a system output data set. An optional subfield may be defined giving the output class of the data set. Output data will be initially directed to the job entry subsystem and may later be transcribed to a final output device. The final output device is associated with output class by the installation. After transcription by the job entry subsystem, SYSOUT data sets are deleted.

LABEL(type)
specifies the kind of label processing to be done. Type may be one of the following: SL, SUL, AL, AUL, NSL, NL, LTM, or BLP. These types correspond to the present JCL label-type values.

KEEP
specifies that the data set is to be retained by the system after it is freed.

DELETE
specifies that the data set is to be deleted after it is freed.

CATALOG
specifies that the data set is to be retained by the system in a catalog after it is freed.

UNCATALOG
specifies that the data set is to be removed from the catalog after it is freed. The data set is still retained by the system.

ATTRIB COMMAND

Use the ATTRIB command to build a list of attributes for non-VSAM data sets that you intend to allocate dynamically. During the remainder of your terminal session, you can have the system refer to this list for data set attributes when you enter the ALLOCATE command.

Format

```
{ATTRIB}     attr-list-name
{ATTR}
             [BLKSIZE(blocksize)]

             [BUFL(buffer-length)]

             [BUFNO(number-of-buffers)]

             [LRECL ({logical-record length})]
             [      ({            X       })]

             [NCP(no.-of-channel-programs)]

             [INPUT ]
             [OUTPUT]

             [EXPDT(year-day)    ]
             [RETPD(no.-of-days)]

             [BFALN ({F})]
             [      ({D})]

             [OPTCD(A,B,C,E,F,H,Q,R,T,W, and/or Z)]

             [EROPT({ACC|SKP|ABE})]

             [BFTEK({S|E|A|R})]

             [RECFM(A,B,D,F,M,S,T,U, and/or V)]

             [DIAGNS(TRACE)]

             [LIMCT(search-number)]

             [BUFOFF ({block-prefix-length})]
             [       ({        L         })]

             [DSORG ({DA|DAU|PO|POU|PS|PSU})]

             [DEN ({0|1|2|3|4})]

             [TRTCH({C })]
             [      ({E })]
             [      ({ET})]
             [      ({T })]
             [KEYLEN(key-length)]
```

All of the required and optional operands for the ATTRIB command are shown in the format. Only those of interest to the VS FORTRAN programmer are described in the following paragraphs. For complete documentation, see OS/VS2 TSO Command Language Reference.

where:

attr-list-name
specifies the name for the attribute list. This name can be specified later as a parameter of the ALLOCATE command. The name must consist of 1 through 8 alphameric and/or national characters. The name must begin with an alphabetic or national character, and must be different from all other attr-list-names and ddnames that are in existence for your terminal session.

BLKSIZE(blocksize)
specifies the block size for the data sets. The block size must be a decimal number and must not exceed 32,760 bytes. The block size that you specify must be consistent with the requirements of the RECFM operand. If you specify:

RECFM(F)
then the block size must be equal to or greater than the logical record length.

RECFM(F B)
then the block size must be an integral multiple of the logical record length.

RECFM(V)
then the block size must be equal to or greater than the largest block in the data set.

RECFM(V B)
then the block size must be equal to or greater than the largest block in the data set. Since the number of logical records can vary, you must estimate the optimum block size (and the average number of records for each block) based on your knowledge of the application that requires the I/O.

LRECL(logical-record-length)
specifies the length, in bytes, of the largest logical record in the data set. You must specify this operand for data sets that consist of either fixed-length or variable-length records.

Omit this operand if the data set contains undefined-length records. The logical record length must be consistent with the requirements of the RECFM operand and must not exceed the block size (BLKSIZE operand) except for variable-length-spanned records. If you specify:

RECFM(V) or RECFM(V B)
then the logical record length is the sum of the length of the actual data fields plus 4 bytes for a record descriptor word.

RECFM(F) or RECFM(F B)
then the logical record length is the length of the actual data fields.

RECFM(U)
then you should omit the LRECL operand.

INPUT
specifies that the data set will be used only as input to a processing program.

OUTPUT
specifies that the data set will be used only to contain output from a processing program.

EROPT ({ACC | SKP | ABE})
specifies the option you want executed if an error occurs when a record is read or written. The options are:

ACC
accepts the block of records in which the error was found.

SKP
skips the blocks of records in which the error was found.

ABE
to end the task abnormally.

RECFM(A,B,D,F,M,S,T,U, and/or V)
specifies the format and characteristics of the records in the data set. The format and characteristics must be completely described by one source only. If they are not available from any source, the default will be an undefined-length record.

Use the following values with the RECFM operand.

A
indicates the record contains the RECFM operand.

B
indicates the records are blocked.

D
indicates variable-length ASCII records.

F
indicates the records are of fixed-length.

M
indicates the records contain machine code control characters.

S
indicates for fixed-length records, the records are written as standard blocks (there must be no truncated blocks or unfilled tracks except for the last block or track). For variable-length records, a record may span more than one block. Exchange buffering, BFTEK(E), must not be used.

T
indicates records may be written onto overflow tracks if required. Exchange buffering, BFTEK(E), or chained scheduling, OPTCD(C), cannot be used.

U
indicates records of undefined length.

V
indicates records are of variable length.

You may specify one or more values for this operand (at least one is required).

KEYLEN(key-length)
specifies the length in bytes of each of the keys used to locate blocks of records in the data set when the data set resides on a direct access device. The key length must not exceed 255 bytes. If an existing data set has standard labels, you can omit this operand and let the system retrieve the key length from the standard label. If a key length is not supplied by any source before you issue an OPEN macro instruction, a length of zero (no keys) is assumed. This keyword is mutually exclusive with TRTCH.

In FORTRAN the key length must be:

2
for an INTEGER*2 key

4
for an INTEGER*4 key

CALL COMMAND

Use the CALL command to load and execute a program that exists in executable (load module) form. The program may be user-written, or it may be the VS FORTRAN compiler.

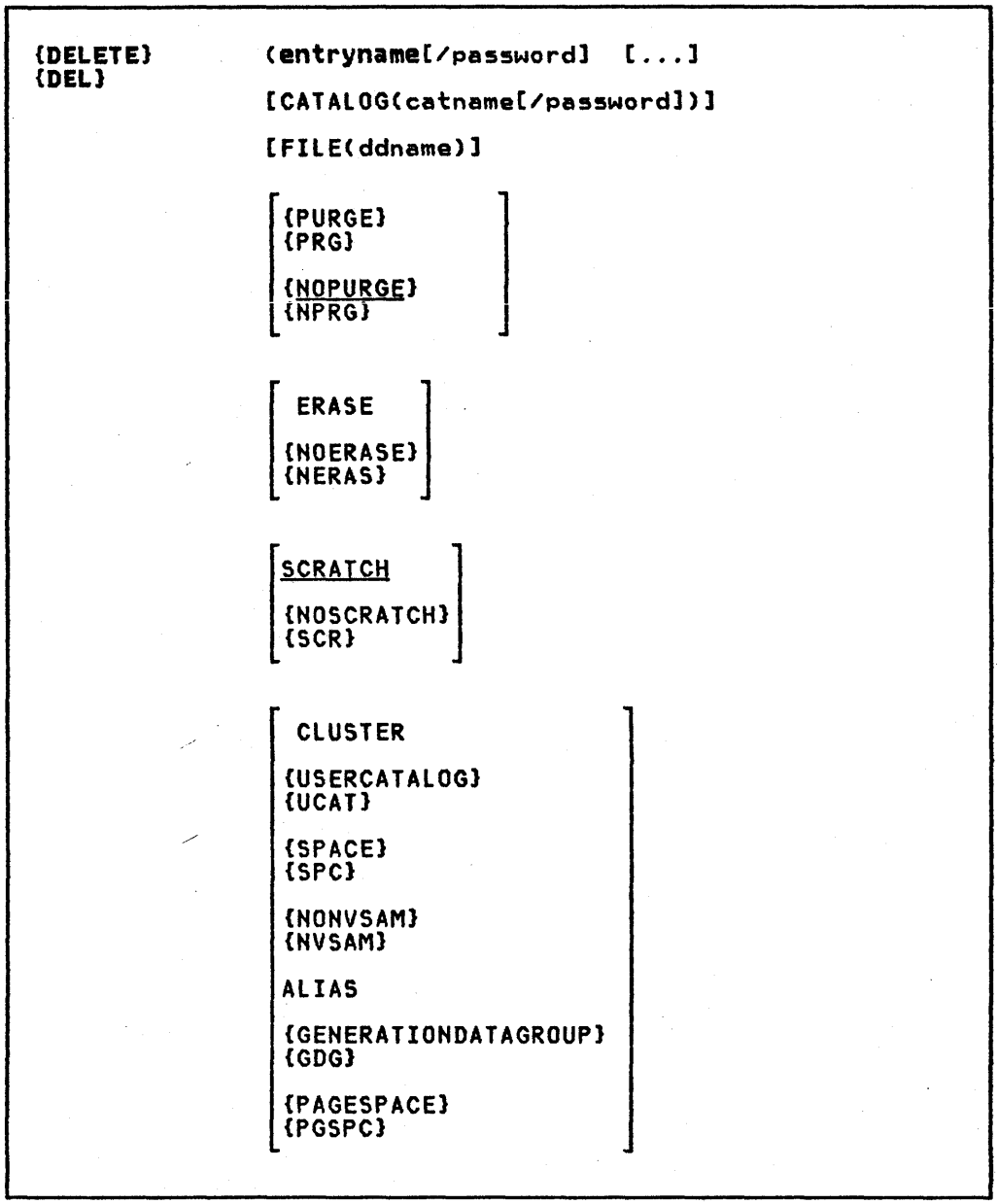
```
CALL            {dsname}
                {dsname(membername)}

                ['parameter-string']
```

dsname(membername)
specifies the name of a partitioned data set and the membername (program name) to be executed. The membername must be enclosed in parentheses.

parameter-string
specifies up to 100 characters of information that you want to pass to the program as a parameter list. When passing parameters to a program, you should use the standard linkage conventions.

DELETE COMMAND

Use the DELETE command to delete one or more data set entries or one or more members of a partitioned data set.



All the required and optional operands for the DELETE command are shown in the format. Only those of interest to the VS FORTRAN programmer are described in the following paragraphs. For complete documentation, see OS/VS2 TSO Command Language Reference.

entryname[/password][,...]
is a required parameter that names the entries to be deleted. When more than one entry is to be deleted, the list of entry names must be enclosed in parentheses. This parameter must be the first parameter following DELETE.

password
specifies a password for a password-protected entry. Passwords may be specified for each entry name, or the catalog's password may be specified through the CATALOG parameter for the catalog that contains the entries to be deleted.

CATALOG(catname[/password])
specifies the name of the catalog that contains the entries to be deleted.

catname
identifies the catalog that contains the entry to be deleted.

password
specifies the master password of the catalog that contains the entries to be deleted.

FILE(ddname)
specifies the name of the DD statement that identifies the volume that contains the data set to be deleted or identifies the entry to be deleted.

CLUSTER
specifies that the entry to be deleted is a cluster entry for a VSAM data set.

NONVSAM or NVSAM
specifies that the entry to be deleted is a non-VSAM data set entry.

EDIT COMMAND

With EDIT and its subcommands, you can create, modify, store, submit, retrieve, and delete data sets with sequential or partitioned data set organization.

```
{EDIT}      data-set-name[/password]
{E}
            [NEW]
            [OLD]

            [PLI   [([integer1 [integer2]][CHAR60])]
                   [([    2    [   72   ]][CHAR48])]

            PLIF   [([integer1 [integer2]][CHAR60])]
                   [([    2    [   72   ]][CHAR48])]

            ASM
            COBOL
            VSFORT
            TEXT
            DATA
            CLIST
            CNTL
            VSBASIC                                 ]

            [SCAN]
            [NOSCAN]

            [NUM]      [(integer1[integer2])]
            [NONUM]

            [BLOCK(integer)]
            [BLKSIZE(integer)]

            [LINE(integer)]
            [LRECL(integer)]

            [CAPS]
            [ASIS]
```

All the required and optional operands for the EDIT command are shown in the format. Only those of interest to the VS FORTRAN programmer are described in the following paragraphs. For complete documentation, see OS/VS2 TSO Command Language Reference.

data-set-name
specifies the name of the data set that you want to create or edit.

password
specifies the password associated with the data-set-name. If the password is omitted and that data set is password protected, you will be prompted for the data set's password. Read protected partitioned data sets will cause a prompt for the password twice, provided it is not entered on the EDIT command, or is not the same password as your LOGON userid password.

CLIST
specifies that the data set identified by the first operand is for a command procedure and will contain TSO commands and subcommands as statements or records in the data set. The data set will be assigned line numbers.

DATA
specifies that the data set identified by the first operand is for data that may be subsequently retrieved or used as input data for processing by an application program.

VSFORT
specifies that the data set identified by the first operand is for VS FORTRAN statements.

Modes of Operation

The EDIT command has two modes of operation: input mode and edit mode. You enter data into a data set when you are in input mode. You enter subcommands and their operands when you are in edit mode. The EDIT subcommands useful for VS FORTRAN are listed in Figure 21.

EDIT Subcommands	Function
ALLOCATE	Allocate data sets
END	Ends an edit session
HELP	Explains EDIT subcommands
INPUT	Enters INPUT mode
PROFILE	Changes library identifier
SAVE	Places selected file in a data set

Figure 21. EDIT Subcommands Useful for VS FORTRAN

Use the FREE command to release (deallocate) previously allocated data sets that you no longer need.

Format

```
FREE    { DSNAME(dataset-name-list)     } [1]  [DEST(station-id)]
        { SET(dataset-name-list)        }
        { DDNAME(file-name-list)        }
        { FILE(file-name-list)          }
        { ATTRLIST(attr-list-names)     }

        [HOLD]     [KEEP     ]                 [SYSOUT(class)]
        [NOHOLD]   [DELETE   ]
                   [CATALOG  ]
                   [UNCATALOG]
```

Notes to Format

[1]Choose one or more of the parameters within braces.

[2]DELETE is the only disposition valid for SYSOUT data sets.

All the required and optional operands for the FREE command are shown in the format. Only those of interest to the VS FORTRAN programmer are described in the following paragraphs. For complete documentation, see OS/VS2 TSO Command Language Reference.

DATASET or DSNAME(data-set-name-list)
specifies one or more data set names that identify the data sets that you want to free. The data set name must include the descriptive (rightmost) qualifier and may contain a member name in parentheses. If you omit this operand, you must specify either FILE or DSNAME or the ATTRLIST operand.

FILE or DDNAME(file-name-list)
specifies one or more file names that identify the data sets to be freed. If you omit this operand, you must specify either the DATASET, DSNAME, or ATTRLIST operand.

KEEP
specifies that the data set is to be retained by the system after it is freed.

DELETE
specifies that the data set is to be deleted by the system after it is freed. DELETE is not valid for data sets allocated SHR or for members of a PDS. Only DELETE is valid for SYSOUT data sets.

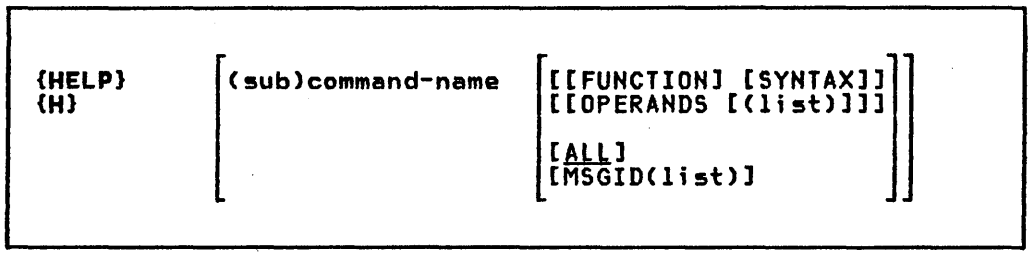
CATALOG
specifies that the data set is to be retained by the system in a catalog after it is freed.

UNCATALOG
specifies that the data set is to be removed from the catalog after it is freed. The data set is still retained by the system.

HELP COMMAND

Use the HELP command or subcommand to obtain information about the function, syntax, and operands of commands and subcommands.

Format



All the required and optional operands for the HELP command are shown in the format. Only those of interest to the VS FORTRAN programmer are described in the following paragraphs. For complete documentation, see OS/VS2 TSO Command Language Reference.

command-name or subcommand name
specifies the name of the command or subcommand that you want to know more about.

FUNCTION
specifies that you want to know more about the purpose and operation of the command or subcommand.

SYNTAX
specifies that you want to know more about the syntax required to use the command or subcommand properly.

OPERANDS(list-of-operands)
specifies that you want to see explanations of the operands for the command or subcommand. When you specify the keyword OPERANDS and omit any values, all operands will be described. You can specify particular keyword operands that you want to have described by including them as values within parentheses following the keyword. If you specify a list of more than one operand, the operands in the list must be separated by commas or blanks.

ALL
specifies that you want to see all information available concerning the command or subcommand. This is the default value if no other keyword operand is specified.

Help information
The scope of available information ranges from general to specific. The HELP command or subcommand with no operands produces a list of valid commands or subcommands and their basic functions. From the list you can select the command or subcommand most applicable to your needs. If you need more information about the selected command or subcommand, you may use HELP again, specifying the selected (sub)command name as as operand.

LINK COMMAND

Use the LINK command to invoke the linkage editor service program. Basically, the linkage editor converts one or more object modules (the output modules from compilers) into a load module suitable for execution. In doing this, the linkage editor changes all symbolic addresses in the object modules into relative addresses.

Format

```
LINK        (data-set-list)

            [LOAD[(data-set-name)]]

            [PRINT   ({*            })]
                     ({data-set-name})
            [NOPRINT                 ]

            [LIB(data-set-list)]

                        [REFR]      [TERM]
            [PLICMIX]   [NOREFR]    [NOTERM]

            [PLIBASE]   [SCTR]      [DCBS(blocksize)]
                        [NOSCTR]

            [FORTLIB]   [OVLY]      [AC(authorization-
            [COBLIB]    [NOOVLY]        code)]

            [MAP]       [RENT]
            [NOMAP]     [NORENT]

            [NCAL]      [SIZE(integer1 integer2)]
            [NONCAL]

            [LIST       [NE]
            [NOLIST]    [NONE]

            [LET]       [OL]
            [NOLET]     [NOOL]

            [XCAL]      [DC]
            [NOXCAL]    [NODC]

            [XREF]      [TEST]
            [NOXREF]    [NOTEST]

            [REUS]
            [NOREUS]
```

All the required and optional operands for the LINK command are shown in the format. Only those of interest to the VS FORTRAN programmer are described in the following paragraphs. For complete documentation, see OS/VS2 TSO Command Language Reference.

(data-set-list)
specifies the names of one or more data sets containing your object modules and/or linkage editor control statements. (See the TSO data set naming conventions.) The specified data sets will be concatenated within the output load module in the sequence they are included in this operand. If there is only a single name in the data-set-list, parentheses are not required unless the single name is a member name of a partitioned data set; then, two pairs of parentheses are required, as in:

```
link((parts))
```

You may substitute an asterisk (*) for a data set name to indicate that you will enter control statements from your terminal. The system will prompt you to enter the control statements. A null line indicates the end of your control statements.

LOAD(data-set-name)
specifies the name of the partitioned data set that will contain the load module after processing by the linkage editor (see the data set naming conventions). If you omit this operand, the system will generate a name according to the data set naming conventions.

FORTLIB
specifies that the partitioned data set named SYS1.FORTLIB is to be searched by the linkage editor to locate load modules referred to by the module being processed.

LOADGO COMMAND

Use the LOADGO command to load a compiled or assembled program into real storage and begin execution.

Format

```
{LOADGO}    (data-set-list)
{LOAD}      ['parameters']

            [PRINT  ({*            })]
                    ({data-set-name})
            [NOPRINT               ]

            [LIB(data-set-list)]

            [PLIBASE]

            [PLICMIX]

            [FORTLIB]

            [COBLIB]

            [TERM]
            [NOTERM]

            [RES]
            [NORES]

            [MAP]
            [NOMAP]

            [CALL]
            [NOCALL]

            [LET]
            [NOLET]

            [SIZE(integer)]
            [EP(entry-name)]
            [NAME(program-name)]
```

All the required and optional operands for the LOADGO command are shown in the format. Only those of interest to the VS FORTRAN programmer are described in the following paragraphs. For complete documentation, see OS/VS TSO Command Language Reference.

(data-set-list)
specifies the names of one or more object modules and/or load modules to be loaded and executed. The names may be data set names, names of members of partitioned data sets, or both (see the TSO data set naming conventions). When you specify more than one name, the names must be enclosed within parentheses and separated from each other by a standard delimiter (blank or comma).

'parameters'
specifies any parameters that you want to pass to the program to be executed.

FORTLIB
specifies that the partitioned data set named SYS1.FORTLIB is to be searched to locate load modules referred to by the module being processed.

STATUS COMMAND

Use the STATUS command to have the status of conventional batch jobs displayed at your terminal. You can obtain the status of all batch jobs, of several specific batch jobs, or of a single batch job.

This command may be used only by personnel who have been given the authority to do so by your organization.

Format

```
{STATUS}        [(jobname[(jobid)]-list)]
{ST}
```

(jobname[(jobid)]-list)
specifies the names of the conventional batch jobs for which you want to know the status. If two or more jobs have the same jobname, the system will display the status of all the jobs encountered and supply jobids for identification. When more than one jobname is included in the list, the list must be enclosed within parentheses. If you do not specify any jobnames, you will receive the status of all batch jobs in the system whose jobnames consist of your userid and one identifying character (alphameric or national).

The optional jobid subfield may consist of 1 to 8 alphameric characters (the first character must be alphabetic or national). The jobid is a unique job identifier assigned by the job entry subsystem at the time the job is submitted to the batch system.

Note: When you specify a list of jobnames, you must separate the jobnames with standard delimiters.

SUBMIT COMMAND

Use the SUBMIT command to submit one or more batch jobs for conventional processing.

Format

```
{SUBMIT}     (data-set-list)   [NOTIFY]   .
{SUB}                          [NONOTIFY]
```

(data-set-list)
specifies one or more data set names or names of members of partitioned data sets that define an input stream (JCL plus data). If you specify more than one data set name, enclose them in parentheses.

NOTIFY
specifies that you are to be notified when your job terminates in the background if a JOB statement has not been provided. If you have elected not to receive messages, the message will be placed in the broadcast data set.

NONOTIFY
specifies that a termination message will not be issued or placed in the broadcast data set. The NONOTIFY keyword is only recognized when a JOB statement has not been provided with the job that you are processing.

Note: Data sets that are dynamically allocated by the SUBMIT command processor are not automatically freed when the command processor terminates. You must explicitly free dynamically allocated data sets.

TEST COMMAND

Use the TEST command and subcommands to test a program for proper execution.

Format

```
TEST        ['data-set-name']

            ['parameters']

            [LOAD]
            [OBJECT]

            [CP]
            [NOCP]
```

All the required and optional operands for the TEST command are shown in the format. Only those of interest to the VS FORTRAN programmer are described in the following paragraphs. For complete documentation, see OS/VS2 TSO Command Language Reference.

'data-set-name'
specifies the name of the data set containing the program to be tested. The program must be a load module that is a member of a partitioned data set or it must be an object module.

A data set name must be specified to test a program that is not currently active. (A currently active program is one that has abnormally terminated or has been terminated by an attention interruption.)

Note: When specifying the data-set-name for TEST, the name should be enclosed by single quotation marks, or the LOAD or object qualifier will be added to the name specified. If no name is specified, TEMPNAME is the member searched for via the TEST request.

CAUTION: The program to be tested should not have the name TEST or the name of any existing TSO service routine. For a listing of the existing module names, see OS/VS2 TSO Terminal Monitor Program and Service Routines Logic.

'parameters'
specifies a list of parameters to be passed to the named program. The list must not exceed 100 characters including delimiters.

LOAD
specifies that the named program is a load module that has been processed by the linkage editor and is a member of a partitioned data set. This is the default value if both LOAD and OBJECT are omitted.

OBJECT
specifies that the named prorgram is an object module that has not been processed by the linkage editor. The program can be contained in a sequential data set or a member of a partitioned data set.

SUBCOMMANDS: Subcommands of the TEST command useful to the VS FORTRAN programmer are:

ASSIGNMENT OF VALUES(=)
modifies values in virtual storage and in registers.

AT
establishes breakpoints at specified locations.

CALL
initializes registers and initiates processing of the program at a specified address, using the standard subroutine linkage.

END
terminates all operations of the TEST command and the program being tested.

GO
restarts the program at the point of interruption or at a specified address.

HELP
lists the subcommands of TEST and explains their function, syntax, and operands.

LIST
displays the contents of a virtual storage area or registers.

LISTMAP
displays a map of the user's virtual storage.

LOAD
loads a program into virtual storage.

OFF
removes breakpoints.

RUN
terminates TEST and completes execution of the program.

APPENDIX A. DEVICE INFORMATION

This appendix gives information regarding specific input/output devices that can be used with a VS FORTRAN program.

MINIMUM AND MAXIMUM BLOCK SIZE VALUES

The minimum and maximum block sizes that can be specified for specific devices are shown in Figure 22.

Device Type	Fixed and Undefined Records Block Size (Bytes)		Variable Records Block Size (Bytes)	
	Minimum	Maximum	Minimum	Maximum
Card Reader	1	80	9	80
Card Punch	1	81	9	89
Printer Line Length (1403, 3800, etc.)				
120 characters	1	121	9	129
132 characters	1	133	9	141
144 characters	1	145	9	153
150 characters	1	151	9	159
Magnetic Tape	18	32760	18	32760
Direct Access				
2314	1	7294	9	7294
3330	1	13030	9	13030
3340	1	8368	9	8368
3350	1	19069	9	19069

Notes:

1. For DOS/VSE Fixed Block Architecture devices, see the manuals describing the devices you're using.
2. For direct access devices with the track overflow feature, the maximum is 32760 for each device.

Figure 22. VS FORTRAN Devices—Minimum and Maximum Block Sizes

DIRECT ACCESS DEVICE CAPACITIES

The capacities of specific direct access devices are shown in Figure 23.

Device Type	Track Capacity	Tracks per Cylinder	Number of Cylinders	Total Capacity
2314(2319)	7294	20	100	19,176,000
3330	13030	19	404	101,751,270
3330-11	13030	19	808	203,502,340
3340	8535	12	348	34,944,768
3350	19254	30	815	317,498,850

Notes:

1. For DOS/VSE fixed block architecture devices, see the manuals describing the devices you are using.
2. For the 3375 and 3380 devices, device information will be provided after the devices are available.

Figure 23. Direct Access Device Capacities

APPENDIX B. COMPILER OPTIONS

The VS FORTRAN compiler options let you specify details about the input source program and request specific forms of compilation output. How the options are specified depends upon the system you are using:

- In OS/VS, the options are specified by using the PARM field of the EXEC job control statement. See the "PARM Field Options" descriptions following the EXEC statement in the OS/VS section.
- In DOS/VSE, they are specified as values of the PARM field of the EXEC job control statement or on a @PROCESS statement.
- In VM/370-CMS, they are specified as options of the FORTVS and RUN command.

In the following paragraphs, each of the compiler options is briefly explained. Defaults are supplied with each options. Check with your system administrator for the ones in effect for your organization.

DECK|NODECK
specifies whether or not the object module in card image format is to be produced. Abbreviation: D|NOD

FIPS (S|F)|NOFIPS
specifies whether or not standard language flagging is to be performed, and, if it is, the standard language flagging level:

NOFIPS specifies no standard language flagging.

FIPS (S) specifies subset standard language flagging.

FIPS (F) specifies full standard language flagging.

FLAG (I|W|E|S)
Specifies the level of diagnostic messages to be written:

FLAG (I) specifies that all messages are to be written.

FLAG (W) specifies that error messages (return code 4) are to be written.

FLAG (E) specifies that error messages (return code 8) and severe error messages (return code 12 or higher) are to be written.

FLAG (S) specifies that only severe error messages (return code 12 or higher) are to be written.

FREE|FIXED
indicates whether the input source program is in free form or in fixed form.

GOSTMT|NOGOSTMT
specifies whether or not internal sequence numbers (for traceback purposes) are to be generated for a call sequence to a subprogram.

LANGLVL (66|77)
specifies the language level in which the input source program is written:

LANGLVL (66) specifies the old FORTRAN level.

LANGLVL (77) specifies the current FORTRAN level.

LINECOUNT (number)
specifies the maximum number of lines on each page of the printed source listing. Abbreviation: LC

LIST|NOLIST
specifies whether or not the object module listing is to be written.

MAP|NOMAP
specifies whether or not a table of source program names and statement labels is to be written.

NAME(name)
for old FORTRAN programs only, specifies the name to be given to a main program.

When NAME is omitted, the main program is named MAIN.

OBJECT|NOOBJECT
specifies whether or not the object module is to be produced. Abbreviation: OBJ|NOOBJ

OPTIMIZE (0|1|2|3) | NOOPTIMIZE
specifies the optimizing level to be used during compilation:

OPTIMIZE (0) OR NOOPTIMIZE specifies no optimization.

OPTIMIZE (1) specifies register and branch optimization.

OPTIMIZE (2) specifies partial code-movement optimization, code movement that can't introduce logic changes into the program.

OPTIMIZE (3) specifies full code-movement optimization, which can possibly introduce logic changes into the program.

Abbreviation: OPT|NOOPT

SOURCE|NOSOURCE
specifies whether or not the source listing is to be produced. Abbreviation: S|NOS

TERMINAL|NOTERMINAL
specifies whether or not statistics and messages are to be directed to the SYSTERM data set. TERMINAL also produces an indexed summary of statistics and messages for each compilation. Abbreviation: TERM|NOTERM

XREF|NOXREF
specifies whether or not a cross-reference listing is to be produced. The cross-reference listing includes all variables and labels used in the source program. Abbreviation: X|NOX

INDEX

SC26-3988-0

VS FORTRAN Application Programming: System Services Reference Supplement (File No. S370-25) Printed in U.S.A. SC26-3988-0

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

VS FORTRAN Application Programming:
System Services Reference Supplement
SC26-3988-0

Reader's Comment Form

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Note: Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.

List TNLs here:

If you have applied any technical newsletters (TNLs) to this book, please list them here:

Last TNL _____

Previous TNL _____

Previous TNL _____

Fold on two lines, tape, and mail. No postage necessary if mailed in the U.S.A. (Elsewhere, any IBM representative will be happy to forward your comments.) Thank you for your cooperation.

SC26-3988-0

Reader's Comment Form

Fold and tape | **Please do not staple** | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

Fold and tape | **Please do not staple** | Fold and tape

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

VS FORTRAN Application Programming: System Services Reference Supplement (File No. S370-25) Printed in U.S.A. SC26-3988-0

VS FORTRAN Application Programming:
System Services Reference Supplement
SC26-3988-0

Reader's Comment Form

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Note: Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.

List TNLs here:

If you have applied any technical newsletters (TNLs) to this book, please list them here:

Last TNL _____

Previous TNL _____

Previous TNL _____

Fold on two lines, tape, and mail. No postage necessary if mailed in the U.S.A. (Elsewhere, any IBM representative will be happy to forward your comments.) Thank you for your cooperation.

SC26-3988-0

Reader's Comment Form

Fold and tape | **Please do not staple** | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

Fold and tape | **Please do not staple** | Fold and tape

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

VS FORTRAN Application Programming: System Services Reference Supplement (File No. S370-25) Printed in U.S.A. SC26-3988-0

Technical Newsletter

This Newsletter No. SN26-0845
Date 30 April 1981

Base Publication No. SC26-3988-0
File No. S370-25

Prerequisite Newsletters None

VS FORTRAN Application Programming: System Services Reference Supplement

This technical newsletter, a part of Release 1.0 of the VS FORTRAN Compiler and Library, program numbers 5748-FO3 and 5748-LM3, provides information for the subject publication. This information remains in effect until further notice.

Summary of Amendments

The compiler option, OPTIMIZE(0), is available with Release 1.0 of VS FORTRAN. Additional optimization features (OPTIMIZE(1/2/3)), discussed in this manual, are planned for availability at a later time.

Note: Please file this cover letter at the back of the manual to provide a record of changes.

IBM Corporation, P.O. Box 50020, Programming Publishing, San Jose, California 95150

Printed in U.S.A.

Technical Newsletter

This Newsletter No. SN26-0876
Date 3 June 1981

Base Publication No. SC26-3988-0
File No. S370-25

Prerequisite Newsletters SN26-0845

**VS FORTRAN Application Programming:
System Services Reference Supplement**

This technical newsletter, a part of Release 1 of VS FORTRAN Program Products 5748-F03 (Compiler and Library) and 5748-LM3 (Library only), provides replacement pages for the subject publication. These replacement pages remain in effect for subsequent releases unless specifically altered. Pages to be inserted and/or removed are:

cover,edition notice
v,vi (vi added)
21—28
91,92

Each technical change is marked by a vertical line to the left of the change.

Summary of Amendments

Changes included in this newsletter are summarized under "Summary of Amendments" following the preface.

Note: Please file this cover letter at the back of the publication to provide a record of changes.

IBM Corporation, P.O. Box 50020, Programming Publishing, San Jose, California 95150

Printed in U.S.A.