IBM z/VSE
VSE Central Functions
6.4

VSE/ICCF User's Guide

SC33-6739-02

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 355.

Third Edition (September 2003)

This edition applies to Version 6 Release 4 of the IBM Virtual Storage Extended/Interactive Computing and Control Facility (VSE/ICCF), which is part of VSE Central Functions, Program Number 5686-066, and to all subsequent releases and modifications until otherwise indicated in new editions.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the addresses given below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

```
IBM Deutschland Entwicklung GmbH
Department 3282
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany
```

You may also send your comments by FAX or via the Internet:

```
Internet: s390id@de.ibm.com
FAX (Germany): 07031-16-3456
FAX (other countries): (+49)+7031-16-3456
```

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

About This Publication

This manual describes the support available to a terminal user who works with the IBM Virtual Storage Extended/Interactive Computing and Control Facility (*VSE/ICCF* for short). It shows how to use VSE/ICCF as a data editing tool and as a tool for accessing system services (such as the submission of a job for processing by VSE/POWER) from a display station.

Who Should Use This Publication

This publication is intended primarily to assist the program developer in building and testing a program.

The publication can also be used by other skilled personnel, for example by the system administrator as a supplementary source of information. The administrator's tasks are described in detail in VSE/ICCF Administration and Operation.

How to Use This Publication

This publication includes information about the double-byte character support. This support is used primarily with equipment such as the IBM 5550, which may not have been announced as available in your country.

Where to Find More Information

For further details on VSE/ICCF see VSE/ICCF Administration and Operation.

z/VSE IBM Documentation

IBM Documentation is the new home for IBM's technical information. The z/VSE IBM Documentation can be found here:

https://www.ibm.com/docs/en/zvse/6.2

You can also find VSE user examples (in zipped format) at

https://public.dhe.ibm.com/eserver/zseries/zos/vse/pdf3/zVSE_Samples.pdf

Summary of Changes

Changes for Third Edition (September 2003)

The third edition of this documentation contains these changes, which were first included in z/VSE Version 2 Release 5:

- VSE/ICCF has been enhanced to improve the access to VSE/POWER queue entries by providing direct access to queue entries, access to active queue entries, and access to in-creation queue entries. For this purpose the VSE/ICCF commands /LISTP (described on page "/LISTP Command" on page 74) and /SKIP (described on page "/SKIP Command" on page 120) have been changed.
- Program DTSGETQ (which is used to access the VSE/POWER queue entries) has been modified. As a result, procedures GETL (described on page "GETL Procedure" on page 54), GETP (described on page "GETP Procedure" on page 56), and GETR (described on page "GETR Procedure" on page 58), have a new operand QN (queue number). This operand can be used to specify the VSE/POWER queue entry number.
- The documentation has been updated with other minor corrections resulting from APARs.

Changes for Second Edition (June 1999)

The second edition of this documentation, which was published as VSE/ICCF Version 6 Release 4 first became available, contained these changes:

- The following compiler support was removed from VSE/ICCF:
  - VS BASIC
  - DOS/VS COBOL
  - DOS/PLI Optimizing Compiler
- VSE/ICCF transaction modules were converted from CICS Macro to CICS command level.
- VSE/ESA began to support two security components, BSM and ESM. For details of how this affects VSE/ICCF, see "Access Control" on page 10.
- The BTAM examples for the Terminal Error Programs (TEPs) were removed.

Chapter 1. Introduction

This section gives an overview of the major functions of VSE/ICCF. Here are some of the operations for which you can use VSE/ICCF:

- Enter programs and data and save them in a library.
- Display and edit these programs, compile, run and debug them.
- Write programs which interact with the terminal and which use the screen of display terminals.
- Build streams of interrelated job steps and save them in libraries.
- Build procedures or macros including variable parameters and run them, or save them in a library.
- Define procedures with logic and loop control facilities.
- Submit VSE/ICCF or VSE* JCL job streams to VSE/POWER to be run in another VSE partition, and display the print output.
- Submit jobs via VSE/POWER to be run at another node, or direct output from jobs to another node, and receive job ending messages for both local and remote jobs.
- Save print output from interactive partitions in a library member and later display it in print format.
- Communicate with the system operator, with the VSE/ICCF administrator, or with other terminal users via messages.

The HELP Facility

When you work with VSE/ICCF, you can type HELP on your terminal and press ENTER. You then receive a display of information about VSE/ICCF commands, macros, procedures, and job entry statements. For information about setting up your terminal and using VSE/ICCF, see also Chapter 2, “Terminal Considerations,” on page 19.

Modes of Operation

During a session, your terminal is always in one of six modes of operation. The current mode is shown on the scale line of your screen. To a certain extent, your mode of operation determines what commands you can enter at any given time. For example, the /INPUT command, which places the terminal in input mode, can be entered only when your terminal is in command mode. On the other hand, the /END command can be entered in input mode but is invalid if entered when your terminal is in command mode. Here is a summary of the modes of operation. The display that appears on the scale line of your screen is given within parentheses.

- Command mode (CM)

  This is the basic mode. It is usually indicated by the *READY message.

- Input mode (IN)

  This mode is set by the /INPUT command. After the /INPUT command has been entered, you can type data into your input area. To return from input mode to command mode, enter one of the commands / CANCEL, /SAVE, /ENDRUN, and /END

- Edit mode (FS) or (ED)

  VSE/ICCF gives you the choice between two editors: full screen (FS) and context (ED).

  Your terminal operates in FS mode if you call the editor via the ED macro. This mode allows you to modify your data anywhere on the screen. In addition, it allows you to use line commands to do such things as adding, deleting, or duplicating lines in your file. From edit mode, you can place your terminal into editor input mode by issuing the INPUT command. You leave edit mode by issuing the QUIT, END or FILE command.

You use the context editor (in ED mode) if you call the editor by issuing the /ED command. For a summary of the difference between using the full screen and the context editors, see Appendix B, “Context Editor,” on page 341.

If necessary, the indication "full screen" and "context" is used in front of the terms edit mode and editor-input mode.

- Execution mode (EX), (SP) and (RD)

  This mode is entered whenever a program execution is in progress or queued for execution. Execution mode remains in effect until the background job ends and completes printing or until you issue the / CANCEL command.

  While in EX mode, you can do editing and other work (see also “Asynchronous Execution Mode” on page 3). While print output from the executing program is being spooled to the terminal, 'SP' is being displayed. If the program in execution requests input from the terminal, 'RD' is displayed.

- List mode (LS)

  Your terminal is in this mode while a display is in progress. For example, if a /LIST command is issued for a member and the displayed data could not be contained completely on one screen, your terminal is placed into list mode. Each time you press ENTER, the next screen will be displayed.

- Message mode (MS)

  This mode is in effect while messages are being displayed.

The Command Language

VSE/ICCF has the following command groups:

- System commands

  System commands direct general system functions. The /LOGON and /LOGOFF commands, for example, begin and end a session; the /INSERT command inserts library members into your input area or into another library member; the /SAVE command instructs the system to save the contents of the input area in the library. With the /SEND command you can send messages to the system operator, or to other terminal users.

- Full-screen editor commands

  Some of the full-screen editor commands (such as VIEW, SCREEN and FORMAT) control what is being displayed on the screen and how it is displayed. Other commands, for example NEXT, LOCATE, or CHANGE, relate to moving the line pointer, locating an area within the file, or changing or adding data within the current line.

  Multiple editor commands can be entered in the command area by separating individual commands with logical end-of-line characters.

  The editor **line** commands are a subgroup of the editor commands. They manipulate lines as a whole: add, delete, move or copy single or multiple lines.

- Context Editor Commands

  These commands are identical to the full-screen editor commands summarized above. However, some of these commands have a slightly different function. These differences are described in Appendix B, “Context Editor,” on page 341.

  Context editor commands are required when you work with a typewriter terminal or when you use editor functions in macros and procedures or in the DTSBATCH utility.

- Job Entry Statements

  Job entry statements direct and control the execution of jobs. They look very much like system commands, except that a system command is put into effect immediately. Job entry statements are placed into job streams, and the functions that they request are not carried out until the job is actually run. Some examples of these statements are:

  - The /LOAD statement – It defines the start of a job and indicates which compiler or program phase is to be loaded.
  - The /INCLUDE statement – It instructs the system to include specified members in the job stream.
  - The /OPTION statement – It sets various job-processing options.
- Dump Commands

  These commands enable you to display information from programs that have ended abnormally, provided you have specified the DUMP option in your /OPTION statement. For example, the DISPLAY command shows the contents of program storage and the general or floating-point registers. LOCATE, another dump command, is used to locate characters within storage.
- Procedural and Macro Commands

  VSE/ICCF includes some procedures and macros that provide you with additional functional support. A procedure or macro is a library member that contains executable statements and commands and/or data. A procedure is invoked by specifying a file name as a command, for example SUBMIT. The same is also true for macros -- but with one exception. Macros invoked in edit mode must be specified with the prefix '@', for example: @COPY. In both cases, all statements and commands contained in the specified file are then executed. Using the IBM-supplied procedures and macros is described in Chapters 3 and 4 in alphabetical order.

  For more information on writing your own procedures and macros see Chapter 9, “Writing a Procedure or a Macro,” on page 273.

Foreground Versus Background

All VSE/ICCF system and context editor commands are executed in what is called a foreground task, whereas all compilations and executions are performed as a background task.

The commands that are processed in the foreground have a higher priority than the functions performed in the background. This is to maintain an adequate level of response to terminal users who are doing command work.

The number of execution requests that can run concurrently is decided at your location when VSE/ICCF is being tailored. An execution request is initiated whenever you issue the /RUN or /EXEC command.

When a /RUN or /EXEC command is issued, the job is queued for execution. If a block of virtual storage (called an interactive partition) is available in which to run the job, the job is scheduled and becomes a background job. You can have only one execution request running in the background at a time. You can carry on with foreground command work while the execution request is running.

Interactive partitions are associated with up to four classes. This allows you to direct your job to a partition that has the characteristics that you need for the job. Examples of such characteristics are: preallocated work files, large size, and so on.

The terms 'foreground' and 'background' relate only to the VSE/ICCF environment and should not be confused with the terms 'foreground' and 'background' as they apply to VSE, or to the submitting of jobs for execution under VSE/POWER.

Asynchronous Execution Mode

When you have initiated an interactive partition execution, you can issue the /ASYNCH command. This causes asynchronous execution mode to be set and command mode to be entered. In this mode you can perform functions such as editing while your execution is in progress. Entering the /SYNCH command will resynchronize your terminal with your execution in the background.

While in asynchronous execution mode, you cannot issue the /INPUT command. However, if you want to enter data while the execution is in progress, you can place the data into a dummy library member by using the editor. Also, the input area cannot be updated or edited while an asynchronous execution is in progress.

The VSE/ICCF Library File

As a user of VSE/ICCF you have access to one or more libraries. You can own a library exclusively or you can share it with other users. Each library has a directory, and this directory contains an entry for each of the members in the library.

A single file of data in the library is called a 'member'. Each member is represented by a member name and location in the directory. You can get a display of a library's directory by issuing the LIBRARY command.

Note: For reasons of efficiency and space management, it is of advantage to keep the number of members in your library as small as possible. Use the /PURGE command to remove entries from the library which are no longer needed.

Each member in the library consists of one or more 80-character records. The records can be programs, data, object decks, procedures, VSE/ICCF job streams, documentation, VSE JCL or any combination of these. For performance reasons, your administrator may have set a limit for the number of statements that can be contained in a member. However, if you must exceed this limit, simply start another member.

It is possible to logically connect more than one member by using the /INCLUDE job entry statement. To VSE/ICCF, there is no difference between having all of the data in one member or in several members. In fact, it is more efficient for you to update several small members rather than one large member. Each member that you save in your library must be identified with a unique name of one to eight characters, and the first character of this name must be alphabetic.

You can transfer library members from the VSE/ICCF library file to a sublibrary of your VSE system by invoking the LIBRC macro. By using the LIBRL and LIBRP macro you can copy a member from a VSE sublibrary to a VSE/ICCF library.

Types of User Library

Basically there are three types of a library in VSE/ICCF: PRIVATE, PUBLIC and COMMON. The main difference between these three types is the degree of security that each type offers. This is discussed in more detail further below.

These libraries can be referred to in other ways, depending on how they are allocated to you, and how you use them. For example, as a user of VSE/ICCF you may be given access to more than one private library (specified in your user profile), in which case you must select one of these libraries as your main library and the others -- including the public libraries to which you have access -- will become your alternate libraries. In addition to your main and alternate (private and public) libraries you will also have access to the COMMON library.

Another way of looking at libraries is how you use them during terminal operations. At any given time during a session you will have access to three libraries: two private or public libraries, and the common library, if you so desire. VSE/ICCF searches these libraries in a set order and not according to their type.

To keep track of which of your libraries is being searched first and which second, think of the first two libraries as your primary and secondary ones, regardless of their types. The common library is always the last one to be searched.

The order of search for your primary and secondary libraries can be changed at any time with the /SWITCH command.

Public versus Private Libraries

A public library is accessible to you via the /SWITCH or /CONNECT command, or automatically at logon as your main library if this is specified in your user profile.

A private library is accessible to you only if your user profile indicates that you have access to that library (via logon or the /SWITCH or /CONNECT commands).

Shared versus Owned Libraries

Your data has the highest level of security when you are the sole owner of one or more private libraries. That is, only you, or someone signing on with your user ID, can access libraries that you own. Likewise, as a user with an 'owned' private library, you have access only to the common library, to any public libraries, and to your own data. An exception to this might be if you have access also to an alternate private library (via the /SWITCH command) that is shared with other users. You cannot access another user's data even if it is flagged as PUBLIC.

A 'shared' private library on the other hand can be used by two or more users (with each user identified by his user ID). Shared private libraries allow two or more users working on the same material to have access to each other's data so long as that data has been entered as PUBLIC. To efficiently protect your data in a shared private library, enter this data as PRIVATE or protect it by a password.

The Alternate Library

Certain users can access more than one private library. If your user profile has been set to include one or more alternate libraries you can issue the /SWITCH command to switch access between an alternate library and your own main library. In addition, you can use the /CONNECT command to logically connect an alternate private or public library to your primary library. Members can be updated only in the current primary library, which means that commands like /EDIT, /SAVE, and /PURGE work only on members in that library.

Public versus Private Data

All data which is saved in a library is entered as public or private, depending on whether you specified public or private when you issued the /SAVE command. If you do not specify public or private in the /SAVE command the default indicated in your user profile is taken.

Private data can be read by any user who shares the library unless the 'alternate security' option has been selected. Such data can be modified only by the user who entered it.

Public data can be accessed by any user who 'shares' the library.

The public versus private designations on a given member can be changed in one of two ways:

1. By issuing the /PROTECT command.
2. By replacing the data, using the /REPLACE command.

Generation Member Group

A library member can be designated as a generation member group with two to ten entries in the group. Then, when the member is saved or replaced, the new data becomes the current member of the group and all previous members of the group except for the last member have their generation indicator increased. The last member of the group is purged so that the number of members in the group is always constant. (See “/GROUP Command” on page 59 for more information).

Having a generation member group gives you protection in case an error occurs. You can always use the previous generation of the member to correct the problem.

Common Data and the Common Library

There are two ways of making library members accessible to all users at all times: 'Common Data' and 'The Common Library'.

Common data refers to a special type of member, of which only one copy exists in the entire library. Yet, each user has an entry for this member in his or her library directory.

A common data member usually contains common subroutines or procedures needed by all users. Common data is public in the sense that all users have access to it. However, only the VSE/ICCF administrator can update it; the individual terminal user cannot normally update common data. However,

a common data member can be inserted by a user into the input area, updated, and saved in the user's library under a different name.

Rather than having all common members represented in each directory, your location may choose to implement a **common library**. If a common library exists within the system, all directory searches scan the common library directory if the desired member name is not found in your own primary or connected libraries.

It is also possible that both techniques are employed in your environment. To find common data that may be stored in your own library, use the /LIB FULL command. To find out about the presence of a common library, use the /LIB COMMON command.

Library Efficiency

The way you manage your library effects both your own efficiency and that of your computer system. Keep the following in mind when dealing with your library:

- The fewer members there are in your library, the faster the search will be for any one member. Purge unnecessary members from your library regularly, or use multiple libraries with few members rather than a single library with many members.
- The fewer records there are in a member, the easier and faster it will be to edit, update or copy it. For example, if a program is large but is modularized into logically related program segments, a given change may affect only one of the modules. Thus, making a copy of this module for editing, and the editing itself, will take less time.
- Use the /INCLUDE facility to logically connect two or more library members into a single logical file for execution.
- New members are added to the front of a directory, or as close to the front as possible. If one of your active entries works its way toward the bottom of the directory, you can move it closer to the top again by inserting it into the input area (/INSERT), purging it from the library (/PURGE), and resaving it (/SAVE).
- If a library member has reached a complete or inactive state, you can issue the /SQUEEZE command to compress that member. This usually results in a saving of disk space of 60 to 75 %. A library member that has been compressed cannot be edited, updated or executed. When you want to perform these functions, the member must be decompressed by inserting it into the input area (/INSERT command). However, the /LIST command allows you to view all or a portion of a compressed member without having to decompress it.

Temporary Storage Areas

Besides having access to one or more libraries, each terminal user has access to four storage (work) areas, which are part of the VSE/ICCF library:

- The input area

  The area is set up for you when you enter the /INPUT command. All data that you type in after having issued the command is placed into this area. You can use the /INSERT command to insert all or a portion of a library member into your input area.

  Once data exists in the input area, you can modify this data after having issued the ED macro. The editor input mode also provides a means of typing in more data without adding an /INSERT command on every line, and without having to leave edit mode and enter input mode.

  You can clear the input area by entering the /INPUT command followed by /CANCEL. The /CANCEL command, while in input mode, always clears all data in the input area. Another way of clearing the input area is to save its contents in the library. You do this by issuing the /SAVE or /REPLACE command, or the editor SAVE command.

  The contents of the input area are also changed by /EXEC and /RUN when these commands are specified with a name operand, or a procedure is invoked.
- The punch area

It contains the punch output of jobs run in the background. The output of programs that use the VSE symbolic units SYSPCH or SYSnnn (where nnn is defined by the installation as the punch programmer unit) is directed to the punch area. For example, when a compilation is performed, the object program is placed in the punch area. From this area it can be saved in a library member or loaded into storage for execution by the LINKNGO program.

But compilers are not the only programs which can use the punch area. You can write 80-character records to the punch area from your own programs. Again, the records thus written can be read back in another program or saved as a member of the library.

You can transfer the contents of the punch area to the input area by issuing a special form of the /INSERT command. You can also logically include the contents of the punch area as input via a similar form of the /INCLUDE statement. Punched output may be directed to a library member rather than to the punch area. This avoids lost time in transferring the contents of the punch area to the input area when this is to be saved as a library member.

- The print area

  This area, sometimes called the "spool" area, is used to hold print output from background executions until the output has been displayed at your terminal. The default number of print lines that can be held in this area is controlled locally. However, you may modify this value (to hold larger listings, for example) with the /SET command. A parameter in your user profile sets the limit to which you can increase this area.

  Similar to the punch area, the contents of the print area can be transferred to the input area via the /INSERT command. This data can also be included logically as input by issuing the /INCLUDE statement. Printed output may be directed to a library member rather than to the print area.

- The log area

  You can use this area to record your terminal activity. To specify logging for your terminal you would use the /SET LOG command. This causes the input and output from and to your terminal to be saved in the log area. Then, by entering the /LIST $$LOG command, you can display the commands you have just entered. The most basic level of logging saves only commands with the time and mode of entry. Other logging options provide a more comprehensive record, including all input (commands and data) and/or all system responses.

Languages Supported

VSE/ICCF attempts to create a background processing environment which should allow most "batch" (card-input, print output) type compilers to operate without modification as long as they conform to VSE programming standards. The IBM compilers that can be used to compile programs are:

Full Product Name	Short References (See Note below)
IBM DOS/VS Report Program Generator II	RPG II
IBM VS FORTRAN Compiler	FORTRAN
IBM High Level Assembler for VSE	Assembler

Note: The column gives the reference used in this publication to refer to the various IBM compilers.

Ask your system administrator to find out which language support is available to you.

Writing an Interactive Program

An interactive program is one which accepts data from the terminal and displays responses based on the input data back to the terminal. You can write interactive programs in any of the languages mentioned above.

To write an interactive program, merely pretend that you are requesting information from the system console (SYSLOG) and that this information be displayed on this console. When the program runs, VSE/ICCF converts the reads from your terminal to reads from the console; it converts the writes to the console to writes to your terminal.

If your programming language does not support interaction with the system console (for example, FORTRAN), use the logical units SYSIPT and SYSLST for terminal input and output. The option INCON of the /OPTION and /DATA statements causes any SYSIPT directed input (FORTRAN unit number 5) to be requested from the terminal. Thus, any program written to read cards and to write on a printer can be made interactive.

The LINKNGO Program

VSE/ICCF has its own link, load, and go program called LINKNGO. It converts the output of a language compiler (except any compiler that performs its own load) into an executable program.

Normally, you do not have to request execution of the LINKNGO program because it is assumed that you want to run your program after it is compiled. If you do not want LINKNGO invoked, use the NOGO option to avoid that the compiled program is executed.

Occasionally, you may want to invoke LINKNGO explicitly (by /LOAD LINKNGO). For example, a job with two compiles and executes would require a specific invocation of LINKNGO, because LINKNGO normally executes at the end of all compiles on the assumption that all compiles are to be linked together as one loadable program.

The following example shows the coding required if you want each of a number of compiles to be loaded and run as each of these compiles completes.

```
/LOAD VFORTRAN
  ...
  ...                   (FORTRAN source program)
  ...
/LOAD LINKNGO           (Load and run the program)
/LOAD VFORTRAN
  ...
  ...                   (FORTRAN source program)
  ...
/LOAD LINKNGO           (Load and run the program)
/LOAD VFORTRAN
  ...
```

The last /LOAD LINKNGO is optional because LINKNGO is run automatically upon reaching the end of the job. Any data to be read could have been inserted following the /LOAD LINKNGO.

The /DATA statement in the job stream following a compile forces the LINKNGO program to be executed, except when /OPTION NOGO was specified.

The Include Facility

When building jobs to be run, the programs to be compiled or the data itself, or both are usually part of the job stream. However, it is often more convenient to have programs and/or data broken down into smaller, separate modules and grouped them together with successive /INCLUDE statements.

Normally you might enter a FORTRAN compile and run with job entry statements and 80-character records as follows:

```
/LOAD VFORTRAN
  ...
  ...              (FORTRAN Source Program)
  ...
/DATA
  ...
  ...              (Input 80-character records)
  ...
```

However, if the data was in a separate member of the library named FORTDATA, the job stream might look like this:

```
/LOAD VFORTRAN
  ...
  ...              (FORTRAN Source Program)
  ...
/DATA
/INCLUDE FORTDATA
```

Or, the program itself may be in yet another member named FORTPROG in which case the job entry statements would look like this:

```
/LOAD VFORTRAN
/INCLUDE FORTPROG
/DATA
/INCLUDE FORTDATA
```

Now, if the FORTRAN program were large and consisted of three library members named FORTPR1, FORTPR2 and FORTPR3, the job stream would be:

```
/LOAD VFORTRAN
/INCLUDE FORTPR1
/INCLUDE FORTPR2
/INCLUDE FORTPR3
/DATA
/INCLUDE FORTDATA
```

/INCLUDE requests can be nested eight levels deep. In other words, a library member that itself is the object of an /INCLUDE can have /INCLUDE statements.

VSE/ICCF Job Streams

You can build a VSE/ICCF job stream in the input area. To run the job stream, you can do either of the following:

- Issue the /RUN command.
- Save the job stream as a member of the library and run it with the /EXEC command.

A job stream consists of job entry statements (such as /LOAD, /FILE, /OPTION, /UPSI and /ASSGN) and can include source programs or data or both. If the source programs and data to be processed within a job stream are in separate members of the library, they can be logically included in the job stream by using the /INCLUDE statement.

A job stream can consist of a single program execution (single step job) or multiple program executions (multi-step job). Each step in a job stream begins with the /LOAD statement, except when an execution of the LINKNGO utility is implicit. In that case a /LOAD statement is not required.

You can define existing VSE files (SAM, DAM, VSE/VSAM) in your VSE/ICCF job streams via the /FILE statement, thus eliminating the need to have all DLBL/EXTENT cards present in the VSE/ICCF startup deck. In addition to being able to define VSE files via the /FILE statement, you can define dynamically allocated files and assign unit record devices to various 'SYS' units rather than having to use the system defaults. You can also cause a job stream to conversationally prompt the terminal for a job entry statement when the job is executed.

If VSE/ICCF detects an error in a job entry statement, the terminal is prompted for reentry of that statement. VSE/ICCF allows a program running in an interactive partition to set the VSE operator communication exit. This exit is activated in one of the following ways:

By the /ATTEN command
By the ATTN key of the IBM 2741
By the PA1 key of the IBM 3270.

The User Profile

Each user of the system has a unique, four-character user ID. Associated with this user ID are two 80-character records called the user profile. Some typical fields in your user profile are:

- Your user ID, logon password, and the name of your logon procedure, if you have one.
- The identification of your library, including the ID of any alternate private libraries to which you have access.
- The maximum number of lines that you can enter in any given library member, or in your punch and print areas.
- Your security level and time limits for any given execution.
- Accounting information, such as number of logon requests, number of commands entered, number of execution (background) requests and execution time.
- Direct access storage (library) usage information.

The user profile thus contains the information that informs the system about your requirements and security level.

Access Control

VSE/ICCF has several means to protect the VSE/ICCF libraries and data files and to protect phases against unauthorized execution in interactive partitions. Phases and data files are protected by VSE rather than by VSE/ICCF if:

1. Your system is IPLed with VSE/ESA access control set active in the IPL command SYS,
2. All jobs submitted for execution in VSE partitions are "logged on" by a special security identification, such as the job control ID statement or the VSE/POWER JECL parameter SEC.

Access control under VSE/ICCF is implemented as follows:

- User ID

  Each VSE/ICCF user has a four-character user ID which must be specified when logging on to the system. This ID also identifies the owner of members in a library that is shared by other users.

- Logon password

  A three to six character password must be specified when a user logs on. This password should be changed from time to time (by the administrator) to ensure continued protection. In certain cases, you may be given the right to change your own password via a system command.

- Shared versus owned libraries

  The highest level of data protection is attained when you own a library. Access to an owned library, for any purpose whatever, is possible only via the owner's user ID. A shared library, on the other hand, is accessible to a number of users as identified by their user IDs. Data in a shared library is accessible to other users of the library, provided the data has been entered as public. To secure data in a shared library, enter that data as private or protect it with a password.

- Alternate security

  If your administrator has implemented the "alternate security" option, the right to access public and private data is determined differently. With normal (default) protection, public data can be read or written by any one who shares the library, while private data can be read by any user of the library but only updated by the user who entered the data. Under the alternate-security approach, public data can be read by any user but updated only by the user who entered the data, while private data cannot be accessed for any purpose except by the user who entered the data.

- Member passwords

  Any user can save members in the library with a four-character password. The password must not be PRIV or PUBL, and it must begin with an alphabetic character. Any later reference to this member name

must include the password as a suffix. For example, /LIST FORTPROG PASS would display the member FORTPROG which was protected with the password PASS.

If a password-protected member is flagged as private, only the user who entered the data (or someone using the same user ID and who also knows the password) can modify that data. If a password protected member is flagged as public, any user who has access to the library and who knows the password can modify the data.

The password of a member in a library can be added, changed, or deleted by using the /PROTECT command. The use of a password reserves all forms of access (read and modify) for the user who entered the member.

- Controlled access to user files and programs

  VSE/ICCF allows you to define files and programs to which only authorized users may have access.

  If the VSE access control function is set active, the access control function of VSE/ICCF is replaced by a system-wide access control function that offers control for batch and interactive applications. This system-wide access control can be extended using the VSE/Access Control - Logging and Reporting program. For more information about these protection functions refer to:

  – z/VSE Guide to System Functions
  – z/VSE Administration

- Security manager and VSE/ICCF

  VSE/ESA supports two security components. Either the Basic Security Manager (BSM) or the External Security Manager (ESM). Both components keep and maintain user IDs and passwords on their own. VSE/ICCF still maintains user IDs and passwords in its DTSFILE. When a user signs on to VSE/ICCF, the following applies:

  – Logon from the Interactive Interface is always allowed.
  – Native logon to VSE/ICCF requires either the password from the DTSFILE or the central password from the BSM or ESM.

Dynamic Space Allocation

The dynamic space allocation function of VSE/ICCF allows you to allocate disk work file space when you are scheduling a job for execution. Thus you do not have to preallocate this space during start up of VSE/ICCF. Dynamically allocated files can be specified for sequential and direct access files as permanent or temporary. That is, you can retain them from day to day, or you can make them accessible only for the duration of the job or job step that allocates them. The retention of these files is subject to local procedures. For example, if all dynamic space areas are "cold-started" each day, no user files can be retained.

Job Execution Under VSE/POWER

VSE/ICCF allows you to run both VSE/ICCF and VSE jobs in VSE partitions. The VSE batch partition can be either a *static partition* or a *dynamic partition*. You submit the jobs to the VSE/POWER reader queue just as you would present normal VSE jobs for processing under control of VSE/POWER.

The print output from such a job can be disposed of in several ways. It can be routed to your terminal, into your print area, or into a library member. It can be routed to a printer associated with your terminal, to the system printer, or to a VSE/POWER RJE printer. Punch output can be placed into a library member or into your punch area. If a password is added to the SUBMIT procedure, or to the VSE/POWER JOB, LST or PUN statements for jobs submitted to VSE/POWER, the output from these jobs is protected against unauthorized retrieval. The status of submitted jobs can be requested at any time during execution.

If your VSE system is part of a VSE/POWER controlled network (PNET), you can transmit job streams to other nodes. You can control the execution of these jobs and have the results returned to your terminal. You can also exchange messages with users at other VSE/POWER controlled nodes of the network.

The following commands, macros, and procedures help you to handle VSE/POWER jobs:

- **Commands**

  /CTLP
  To issue a VSE/POWER command (VSE/ICCF administrator or user with special authority).

  /DQ
  To display VSE/POWER queues. This command provides functions of the VSE/POWER command / DISPLAY.

  /ERASEP
  To erase a VSE/POWER queue element. This command provides functions of the VSE/POWER command /DELETE.

  /LISTP
  To display output data from the VSE/POWER LST queue.

  /LOCP
  To locate a character string in the displayed VSE/POWER list output.

  /MSG
  To review the VSE/POWER messages sent for you.

  /ROUTEP
  To route a VSE/POWER queue element. This command provides functions of the VSE/POWER command /ALTER.

  /SET MSGAUTO ON
  To get the incoming VSE/POWER messages displayed automatically.

  /SKIP
  To skip forward and backward in a VSE/POWER print file (in a /LISTP operation).

  /STATUSP
  To display the status of a VSE/POWER job.

- **Macros**

  RELIST
  To transfer the contents of the print area or a library member to the printer.

- **Procedures**

  GETL
  to retrieve VSE/POWER list queue output.

  GETP
  To retrieve VSE/POWER punch queue output.

  GETR
  To retrieve a VSE/POWER reader queue entry.

  SUBMIT
  To submit jobs to VSE/POWER.

Message Transmission Facilities

VSE/ICCF users, including the administrator and the system operator, can communicate with each other by using one of these functions:

- The **BROADCAST** function

  It allows the administrator to send information to all terminal users in the form of messages. Broadcast messages are displayed when a user logs on. These messages are usually short and of general interest to all users, for example:

```
*   VSE/ESA  VERSION %  INSTALLED  12/29/1996
*   VSE/ICCF WILL BE UP UNTIL  7:00 PM
*   SEE MAIL FOR NEW SUBMIT VSE/POWER JCL DEFAULTS
```

A broadcast message is set up by the administrator by way of the ADD BROADCAST command of the VSE/ICCF utility DTSUTIL. The message is retained by VSE/ICCF until replaced manually or deleted explicitly.

- The **MAIL** function

  It can be used by the administrator to make larger amounts of information available to users of the system. This information is displayed only on request. An example would be the description of locally defined controls like VSE/POWER defaults for a SUBMIT request.

  A mail-type message is set up by the administrator in one of the following ways:

  – By ADD MEMBER libno A$MAIL, a command of the VSE/ICCF utility DTSUTIL.

  – By /EDIT A$MAIL, an editor command.

  The message is retained by VSE/ICCF until it is deleted manually.

- The **NOTIFICATION** function

  It is the usual means by which terminal users and the system operator communicate with each other. Either can send and receive messages that are typically only of short-term interest. They are displayed according to the option set in the receiver's user profile (either automatically or on request via the /MSG command). They are deleted after having been displayed. Some examples are:

```
- 13:20 MSG FROM AAAA PLEASE SEE ME BEFORE YOU LEAVE
- 14:00 MSG FROM COPER PLEASE LOGOFF
```

The Editors

VSE/ICCF includes two editors:

- The context editor

  This editor can be used with any terminal; it must be used with an IBM 274x terminal. For more details about this editor, refer to Appendix B, “Context Editor,” on page 341.

- The full-screen editor

  This editor can be used only on an IBM 3270 display system. Editing is more efficient with this editor than with the context editor. Its major functions and advantages are briefly summarized below.

The full-screen editor supports all of the features that the IBM 3270 family of terminals offer. These are features such as user-defined PF key options, and minimum data transmission (changed data only). This editor allows you to change and insert data anywhere on the screen simply by positioning the cursor and making the desired change.

The editor's split screen function allows you to display and edit up to eight files (or different areas of the same file) on the screen at one time. You can move or copy data within the same file, or between files, either on the present screen or onto subsequent screens. If extra room is needed to view a certain file, the displays for the other files can be switched off the screen and restored when they are needed.

Commands and macros are available to efficiently locate and modify data within a file and to perform control functions such as setting logical tabs or limiting editor operation to certain columns.

Any number of files can be edited during a given editing session. New library members can also be created without having to leave edit mode.

For more information about this editor, see Chapter 4, “General Information about the Editor,” on page 133.

Procedures and Macros

Procedures (also known as command lists) and macros are library members. A procedure or macro contains a sequence of VSE/ICCF statements that performs a frequently used function such as the compilation, loading, and execution of programs, or the saving of object decks and sorting of library

members. The statements in a procedure or macro can be system commands, editor commands, job entry statements, procedure or macro orders or data.

A procedure or macro saves you the effort of having to re-enter the same series of commands and statements each time you want a certain function to be performed. Once the procedure or macro has been created and saved, you can later invoke it at any time simply by issuing a single command. Variable parameter substitution is possible for both procedures and macros. This allows you to supply them with variable data, thus increasing their flexibility.

The main difference between a procedure and a macro is that a procedure can contain logical statements that determine the flow and execution of the commands, while a macro has only a limited logic capability.

Another difference is that a macro is run in the foreground like a normal command and can be invoked while in edit mode. A procedure is run in the background; it can be invoked only in command mode.

Procedures require the procedure processor program to be executed in an interactive partition, which means that a procedure performs slower than a macro.

The VSE/ICCF procedure processor includes the following capabilities:

- Branching within a procedure

  A procedure can be built such that statements are skipped or repeated. The &&GOTO order causes the flow of lines in the procedure to be altered forward or backward.
- Conditional Statements

  A conditional statement (&&IF order) can be included to cause a condition to be evaluated. Depending on the result of the evaluation, the flow of the procedure can be altered.
- Internal variable symbols

  In addition to the nine parameters which can be passed from procedure invocation (&&PARAM1 through &&PARAM9), nineteen internal alphameric variables (&&VARBLn) and nineteen internal numeric variables (&&COUNTn) can be used to store data or remember conditions. A numeric variable symbol can be incremented.
- Prompting in a procedure

  The &&TYPE order allows a procedure to write to a terminal; the &&READ order allows a procedure to conversationally obtain data from the terminal.
- Reading data into a procedure

  The &&READ order can be used to:

  – Read an entire line to be passed to the command processor as if it had come from the procedure itself.
  – Assign a value to a variable (&&VARBLn) or a count field (&&COUNTn).
  – Read in new parameters to replace the original parameter (&&PARAMn) values.
- Checking of return codes

  A procedure can check the result of processing caused by a VSE/ICCF command in an interactive partition.

- Punching data from a procedure

  The &&PUNCH order allows a procedure to place data into the punch area.

A single macro can be created that contains groups of system or editor commands. This macro can run with one terminal operation, thus greatly extending editor flexibility. For example, a macro can be set up to:

1. Place your terminal in edit mode
2. Set tabs, indexing, verification, and other defaults
3. Return to the terminal for you to continue the edit session.

A temporary macro can be created during editing and run during the same session. The commands to be used are placed in an area called the 'stack' area and run from there. Commands in the stack can be rerun any number of times.

VSE/ICCF includes IBM-supplied macros and procedures, but it includes also services that help you write your own procedures and macros. A typical IBM procedure is ASSEMBLE. It assembles a source program written in assembler language and produces an object module. Examples of IBM-supplied macros are: HC, which can be used to route the output to a hardcopy printer, or CPYLIB, which can be used to copy a member from one library to another. For a complete description of IBM-supplied procedures and macros, see Chapter 3, “System Commands, Procedures, and Macros,” on page 31.

Utility Programs

VSE/ICCF includes specialized utility programs that perform certain frequently used, standard functions. Examples of such programs are:

- LINKNGO

  The utility processes the output of language compilers and links programs and subprograms together to form an executable program.
- DTSCOPY

  The utility copies one sequential file to another.
- DTSSORT

  The utility performs a fast 'in storage' sort on data from the input or punch area.
- DTSAUDIT

  The utility allows you to trace changes made to the VSE/ICCF library file on the basis of a member, a group of members, an entire library, a group of libraries or all libraries.

  For details about the available utilities, see Chapter 10, “Utility Programs,” on page 295.

Debugging Facilities

In addition to making use of the debugging aids included in the language compilers, you can investigate program failures by using the VSE/ICCF Dump Program. This program is automatically invoked by the /OPTION DUMP job entry statement whenever a program ends abnormally. The program displays information about the abnormal end such as the PSW, the contents of the general registers, and data fields associated with the failing instruction. You can then enter other commands to display various storage areas within your program to locate the source of the problem. If desired, a hardcopy dump of an interactive partition can be obtained for later problem determination. For more information, see Chapter 8, “Dump Commands,” on page 257.

Print-Type Members

Printed output from batch or interactive executions can be placed into the VSE/ICCF library. If the members containing such data are saved with names ending in '.P', they are considered print-type members (no other members should therefore end in .P). These members can be displayed or printed in print format with the /DISPLAY and /LIST commands, the PRINT macro, and the RELIST macro. The print area ($$PRINT) is also considered to be a print-type member.

Members of this type contain control characters in the first two bytes of each record. If a member contains such characters and its name does not end in '.P', VSE/ICCF ignores these characters, and each record is displayed on a new line, which means a line size of 80 characters. When a print-type member is printed on a hardcopy printer, or displayed on the IBM 3278 Model 5, up to 132 characters per line can be printed or displayed.

DBCS Support

If an IBM 5550 with 3270 Emulation is installed at your location, you can edit and display also characters that have a two-byte representation like Japanese Kanji characters. If a character string contains characters of both a one-byte and a two-byte representation, we speak about *mixed data*.

Mixed data are structured into subfields of one-byte characters and two-byte characters. The beginning of a subfield of two-byte characters is indicated by a control character, which we call *shift-out (SO) character*, and the end by a control character, which we call *shift-in (SI) character*. When mixed data is displayed on the screen, the SO and SI characters occupy one position and the double-byte characters two positions. Depending on the terminal setup, the SO and SI characters are displayed as a blank or as any other special character. Internally, they are represented by X'0E' and by X'0F'.

VSE/ICCF allows you to process mixed data in various ways. You can:

- Display on the screen and print on a hardcopy printer
  - VSE/ICCF library members
  - VSE/POWER list queue data
  - Print-type data produced by programs running in interactive partitions
- Edit mixed data with the full-screen editor
- Do full-screen read/writes from interactive partitions using the full-screen macro DTSWRTRD
- Prepare mixed data for printing on an IBM 3200 printer.

If you want to create a library member that contains mixed data, specify the attribute DBCS for this member. The system command /PROTECT and the editor command SET allow you to set the DBCS attribute. Mixed data is recognized and interpreted as mixed data under the following conditions:

- *VSE/ICCF library members* are treated as mixed data if the DBCS attribute has been set for the member and if the terminal is an IBM 5550 with 3270 Emulation.
- *The print area* is treated as a DBCS member if it is displayed on an IBM 5550 with 3270 Emulation.
- Data in *the input area* is generally treated as alphanumeric data. You can, however, edit mixed data in the input area. To do this, you have to
  - Work at an IBM 5550 with 3270 Emulation
  - Edit the input area with the full-screen editor
  - Temporarily SET the DBCS attribute for the input area.

  When you have finished editing and you save the input area as a member, the DBCS attribute becomes permanent for the saved member.
- VSE/POWER list queue entries and data for hardcopy output are treated as mixed data, if they are displayed/printed on an IBM 5550 with 3270 Emulation.

Mixed data may be misinterpreted and displayed as alphanumeric data, if:

- An SO or SI character is not matched by its counterpart.
- The number of bytes between a pair of SO and SI characters is not even.

If mixed data is not displayed on an IBM 5550 with 3270 Emulation, double-byte characters will be displayed by meaningless symbols.

Display and Hardcopy Facilities

If you are using a display terminal you have various ways of displaying and printing the output from jobs in interactive partitions.

One way is to write the total print output from a job, or job step, to a library member while it is being displayed. Such a member can be a print-type member, which would also allow you to display, or print it in print format rather than in 80-character record format.

Another way of obtaining print output is to first save it in your library, after which you can direct it to the printer via the RELIST macro. The data is queued on disk and printed automatically, thus freeing your terminal for other work during printing. Other possibilities are to:

- Direct the output to a private queue or destination and have this output printed later.
- Direct the output to a hardcopy printer associated with the terminal; to do this, use the @PRINT macro. The printer destination must be indicated by a /HARDCPY command issued during the session.

For a 328x terminal printer with the form feed feature, a simple form feed support (skip to next page) helps you format the print output that you receive from VSE/POWER. A CICS Transaction Server* service is used to print data on a hardcopy printer and VSE/POWER is used to route data to the system printer.

To simply display print data, the /LIST and /DISPLAY commands can be used. All of these commands, procedures and macros allow you to display, or print, data in print format – provided the member has been created as a print-type member.

Chapter 2. Terminal Considerations

This section tells you about setting up your terminal for operation and how to log on and log off. It also tells you about controlling output at the terminal and discusses considerations that apply to certain terminal types.

There are basically two types of terminals that can be used with VSE/ICCF: typewriter terminals and display terminals. Within these types of terminals there are many different models including the following:

- IBM 3270 Display System via local attachment.
- IBM 3270 Display System via remote attachment.
- IBM printers attached to the above configurations and supported for buffered hardcopy output. Examples are the following IBM printers: 3288 and 3289.
- The IBM 2740 and 2741 Selectric* Typewriter terminal, supported via leased or switched line remote attachment.
- The IBM 3767 Terminal, supported in 2740 or 2741 mode.

The IBM 2740 (or 3767 in 2740 Mode)

Terminal Setup

Before you log on to the system, do the following:

1. Turn the terminal power on.
2. Set the LOCAL vs. COM switch to COM.
3. If the terminal is connected to a telephone, then:

   Set up the dial connection as instructed by your VSE/ICCF administrator. When you hear the data tone, press the Data button on the modem or place the hand set in the acoustic coupler and turn it on.

   If the terminal is not connected to a telephone, then:

   Ready the modem.

   The Standby light should now be on.
4. On an unbuffered terminal (model 1), press the BID key and wait for the Transmit light to turn on. Then press the EOB key. (If the auto EOB feature has been installed, the Carriage Return key can be used in place of the EOB key).

   On a buffered terminal (model 2), press the BID key.

You can now log on as described under “Logging On and Off” on page 21.

Data Entry

The Backspace key can be used to correct data that has been typed incorrectly. However, all characters that have been backspaced over must be retyped.

The Tab key can be used to request logical tabbing as specified in the /TABSET command. If a logical tab character has not been set via the /SET command, VSE/ICCF considers the character transmitted when the Tab key is pressed as the logical tab character.

If the auto EOB feature is installed on your terminal, you need to press only the Carriage Return key to enter a line of data.

If the feature is not installed:

- Press the Carriage Return key and the EOB key to signal the end of a line of input on an unbuffered terminal.
- Press the BID key to signal end of input on a buffered terminal.

On an unbuffered (model 1) terminal, do the following to enter a command or a line of data.

1. Press the BID key and wait for the Transmit light to turn on.
2. Type your command or data line.
3. Press EOB (or the Carriage Return key if auto EOB is installed) to complete the transmission.

The Transmit light should turn off and the Standby light should turn on.

On a buffered terminal (model 2), proceed as follows:

1. Press ENTER and wait for the Enter light to turn on.
2. Type your command or data line.
3. Press BID to signal that the buffer should be transmitted.

The IBM 3270

Terminal Setup

Before you log on to the system, do the following:

1. Turn the terminal power on. If you are using a printer, also turn that on. The presence of the cursor on the screen indicates that the power is on.

   For local terminals, no other preparation is required.
2. For remote IBM 3277 or IBM 3275 terminals attached to leased lines (no telephone), ensure that the modem is on and ready.
3. If the terminal is connected to a telephone, establish the dial connection as instructed by your administrator (dial the computer; when you hear the data tone, press the Data button on the modem, and so on).
4. Wait until the *z/VSE Online* panel appears.

You can now log on as described under “Logging On and Off” on page 21.

Data Entry

Data is always entered at the cursor position which normally appears on the first line of the screen. To correct an error, simply position the cursor to the incorrect data and retype.

The default logical tab character on the 3270 keyboard is represented by the Field Mark key. On a normal typewriter keyboard, this default character is represented by the upper-shift PA2 key, which is in a rather difficult location for use with VSE/ICCF. The lower shift PA2 key is used as a Cancel key, which makes it rather dangerous to use the key also for a Tab key. Therefore, on the 3270 terminal equipped with a typewriter keyboard, set the logical tab character (via the /SET command) to some other character. The semicolon or the 'not' character are good choices.

To transmit data or commands via the 3270, you type in your data and then press ENTER.

If the INPUT INHIBITED light has been turned on (keyboard locked) because of some incorrect action, you can clear the light by pressing RESET.

You can use the ERASE INPUT key to clear any data that you have just typed in. Pressing CLEAR, however, causes the entire screen to be erased.

After you have typed the data and pressed ENTER, the data is sent to the computer and processed by VSE/ICCF. Depending on the kind of input, VSE/ICCF responds by writing one or more lines back to the screen. Between the time you pressed ENTER and the response appears on the screen, the INPUT INHIBITED light is turned on. No data or commands can be entered at this time.

IBM 3270-Specific Functions

The Program Access keys (PA1, PA2, and PA3) and the Program Function keys (PF1 through PF12, or sometimes through PF24) are used for specific purposes in VSE/ICCF. The association of functions to the PA keys is done in the VSE/ICCF tailoring options. The default association is described here:

- The Display key (PA3)

  The key displays the last ten lines before your current line in the input area or in a library member.

  When in edit mode, the key displays the ten lines up to and including the current line. In any other non-execution mode, the key displays the last ten lines of the input area.

  In execution mode, the key is used as an Attention key if the program in execution has set an operator communication exit via the VSE STXIT OC macro. If no operator communication is in effect, the key forces any output in the print area.

  When the key is pressed during the display of output, the remainder of the data in the print area will be lost.
- The Cancel key (PA2)

  This key can always be used to end the current mode of your terminal. The PA2 key serves no function when in command mode. However:

  – When in input mode, the key performs the functions of the /END command and returns you to command mode.
  – When in edit mode, the key performs the functions of the QUIT command and returns you to command mode.
  – When in input submode of the editor, the key returns the terminal to the edit submode.
  – When in execution mode, the key functions like the /CANCEL command.
  – When a /DISPLAY or /LIST is in progress, the key terminates the printout.
  – When your job is running in an interactive partition, the first PA2 request causes that job to be canceled. Even though the job cancels, the remaining print output can still be displayed. However, pressing the PA2 key a second time while in this mode causes the remainder of the print output to be canceled.

    Note: For certain IBM-supplied subsystems (for example ISPF), pressing the CANCEL key may not have the described effect. Instead the exit routine of the subsystem gets control.

At your location, these functions may be assigned to different keys. Check with your administrator.

Logging On and Off

The first step in using VSE/ICCF is to log on. The logon procedure starts the terminal session and identifies you to VSE/ICCF.

If you use a dial-up terminal facility, first establish the connection with the computer system as you were instructed by your administrator. Usually, this consists of:

1. Dialing a certain phone number.
2. Waiting for a high-pitched signal indicating that the computer has answered.
3. Either placing your modem in data mode or placing the telephone handset into the cradle of the acoustic coupler and switching it on.

Then you may be required to type in a 4-character terminal identifier to establish the connection between your terminal and CICS, the terminal control program. Again, obtain these procedures from your administrator.

On a **Typewriter Terminal**:

To log on, type the following four characters and press ENTER or the Carriage Return key:

```
iccf
```

The following messages should be displayed:

```
*VSE/INTERACTIVE COMPUTING AND CONTROL FACILITY
*PLEASE ENTER /LOGON WITH YOUR USERID
```

Now type the following command (where xxxx is your user ID) and press the Carriage Return key:

```
/logon xxxx
```

If the user ID is valid and known to VSE/ICCF, the following message will be displayed:

```
*ENTER PASSWORD
```

Now enter your password. If the password is valid and is accepted, any messages that have been sent to you will be displayed. The logon is completed by messages as shown below

```
*LOGON COMPLETE - DATE 12/29/1996 TIME 01:30
*READY
```

On an **IBM 3270 Display System**:

Wait until the *z/VSE Online* panel appears. Then type in your user ID and directly underneath your password.

If your entries are accepted as valid by z/VSE, the *Function Selection* panel is displayed. Here you must select *command mode*.

An alternate (but highly uncommon) method of logging on is by using the CICS Escape Facility. As a prerequisite the administrator must have set up the z/VSE Online panel for use of the Escape Facility. After the z/VSE Online panel is displayed, press the PF6 or PF9 key and then proceed as described above for a typewriter terminal. As your first action therefore, you would enter the transaction ID iccf.

Your terminal is now in command mode and you can proceed with your terminal session.

Proceeding through Logon

Your user profile may indicate a macro or procedure that can be executed at logon time. (To check your user profile, issue the /USERS command with the PROfile option.) Such a procedure can be used to set defaults, to display mail, identifiers or certain instructions, or to carry out functions related to your logon procedure. You find a skeleton example in “Invoking the Editor” on page 141.

Getting Acquainted with Your VSE/ICCF

If you have logged on for your first productive session, you might want to know about the contents of your library. Enter:

```
/library
```

This will give you a list of the member names in your library. If you enter:

```
/lib common
```

You get a list of member names in the common library, if one exists. You might pick one of the names and enter

```
/list aaaa
```

where aaaa is the name you selected. If your display does not end with the *READY message at this point, press ENTER (or EOB), or enter the /CONTINU command.

Problems During Logon

Here are typical problems that could occur during logon:

- VSE/ICCF, or the system itself, is not up. This condition is generally indicated by no response to any ENTER or carriage return request from the terminal. Check with someone in the computer room.
- Another user has left the terminal without logging off. You would receive the message

  ```
  *INVALID COMMAND
  ```

  or some other VSE/ICCF response, when you attempt to enter your /LOGON command. Enter the /LOGOFF command to log the previous user off, then repeat the logon procedure.
- You have entered an invalid user ID. This is indicated by one of the following messages:

  ```
  USER ID 'xxxxxxxx' IS NOT DEFINED TO THE ONLINE SYSTEM
           (if you tried to log on to the z/VSE Interactive Interface)

  *INVALID USER ID
            (if you tried logon outside of the Interactive Interface)
  ```

  Enter /LOGOFF or /CANCEL, or press PA2. This ends processing by VSE/ICCF. Then attempt to log on again with a valid user ID.
- After your user ID has been accepted, you must enter a valid password to complete the logon procedure. If you have forgotten your password, enter /LOGOFF to end processing by VSE/ICCF and contact your administrator.

Logging Off

It is important that you log off after having completed a terminal session. You might get charged for more time than you have actually used if you do not log off.

To log off, simply enter the /LOGOFF command. Because this command is not valid in edit mode, you must first end your editor session. Save the contents of your input, punch or print areas before you issue the /LOGOFF command; these areas will be cleared.

Terminal Input

All terminal input lines have a maximum length of 80 characters. However, multiple lines can be entered in a single terminal transmission. If you are using an IBM 2741, 2740 or 3767, the Backspace key allows you to correct typing mistakes. You can also define a delete character (by /SET DEL=x). When the character so defined is keyed in as the last character of a line, VSE/ICCF ignores the entire line.

A Backspace key is not needed for a 3270, because you can position your cursor back over the mistake and rekey only the characters in error. You can also enter a delete character, but it is more convenient to simply erase the entire input line before it is entered using the ERASE INPUT key.

The /RETRIEV command allows you to retrieve previously entered commands. A retrieved command is placed in the terminal input area. You can reissue the command by simply pressing ENTER. Also, you can modify the command before you press ENTER.

Prompting

After you have entered the /INPUT command, your terminal is in input mode. When you are in this mode and you have set input verification off (by /SET VERIFY), you have the option of being prompted with the line number of the next line to be entered. The prompting number consists of four digits followed by a

space. On a typewriter terminal it prints to the left of the line being entered. On the 3270, the prompting number is displayed below the data to be entered.

If inclusion prompting is specified on the /INPUT command (or is the default), the prompt will be a five-digit number. Inclusion prompting causes the prompt to become part of the input data (in columns 1 to 5). The data which you type begins in column 7 of each line.

Normal or inclusion prompting are options which can be set on and off via the /PROMPT command. Or the prompt option may have been set up as a default in your user profile, or entered on the /INPUT command.

Tabbing

Logical tab positions are defined via the /TABSET (or edit TABSET) command. This allows you to press the Tab key (IBM 2740, 2741 and 3767) or enter a logical tab character for internal tabbing of the input data.

On a typewriter terminal, it is of advantage to have the physical tab stops on the unit set to the same positions as the logical tabs entered via the /TABSET command. You would use the /SET command to set the logical tab character. The following example illustrates the use of tabbing and the logical tab character (on an IBM 2740, 2741, or 3767 terminal, the physical Tab keys can be used instead of a logical tab character):

```
*READY
/input
/tabset 7,73                  (Sets tabs for columns 7 and 73)
/set tab=;
;write (3,10)
10;format (' a = ?')
;read (1,20) a
20;format (f8.3)
;x=a**2
;write (3,25) a,x
25;format (' a = 'f8.3,' x = 'f20.3)
;end
/end
*READY
/list
       WRITE (3,10)           (First line of the output of /LIST)
10     FORMAT (' A = ?')
       READ (1,20) A
20     FORMAT (F8.3)
       X=A**2
       WRITE (3,25) A,X
25     FORMAT ('A = 'F8.3,' X = 'F20.3)
       END
*END PRINT
*READY
```

If a backspace character is entered following a tab character, the effect of the tab is logically eliminated. The preceding example shows the use of the logical tab character (;) and the logical tab settings (column 7). The TAB key could have been used instead of the logical tab character.

Multiple Line Input

The logical line-end character can be set by means of the /SET system command or SET editor command. This allows you to enter several commands or data lines in a single terminal transmission, as in the following example:

```
*READY
/set end=:
*CONTROL SET
*READY
/input:aaa:bbb:ccc:/end:/save abcmem
*SAVED
*READY
```

In the above example, the colon (:) is used as the logical line-end character. The /INPUT command is followed by three data lines (aaa, bbb, and ccc) and the /end command that ends the input mode; the lines are saved in the library under the name ABCMEM. To enter two or more lines at a time has benefits:

- The system has less work to do.
- You can work more efficiently because you do not have to wait for a terminal response to each line.

Following the line-end character with a backspace causes the line-end character to be erased. However, once data characters (not backspaces) have been entered following a line-end character, it is not possible to backspace to a point before the line-end character.

Input of Hexadecimal Data

You may sometimes need to enter data characters for which no keys exist on your terminal keyboard. These characters must be entered as hexadecimal digits – two hexadecimal digits per character. Edit mode provides two commands (ALTER and OVERLAYX) that allow the entry of hexadecimal data. In edit mode you may view all or parts of your file in hexadecimal format (VIEW command) and you may apply your changes in hexadecimal format.

It is also possible (in any mode of operation except full-screen edit mode) to enter hexadecimal data by defining the HEX control character as in the following example:

```
*READY
/set hex=#
/input
/insert objmod
/delete
#02REP 000124 00147F0
#02END
/end
*READY
```

In this example, the pound sign (#) is defined as the hex control character. When the hex character occurs in an input line, the two succeeding characters are used to form a single EBCDIC character. Thus in the preceding example, the string #02 is interpreted as hex 02. The character placed in column 1 of the input line will thus be the EBCDIC equivalent of hex 02. To use unprintable characters in a member name (for security reasons, for example), the characters following the hex character must be greater than X'40'.

Note: Hexadecimal characters less than X'40' are translated to non-display characters. If they were sent to the screen untranslated, they could be interpreted as screen control characters.

If you use the IBM 3270 Insert or Delete keys to change a line containing such hex characters (to change the position of these characters within the line), the non-display character can no longer be retranslated correctly and the resulting line will be incorrect. View the hex data in hex format to avoid this problem.

Escape Character

When you have several control characters defined (tab, backspace, line end, line delete and hexadecimal), you may sometimes want to enter one of these characters as an ordinary data character. Define a logical escape character as in this example:

```
/set esc="
```

You can then use the escape character prior to any of the defined control characters to cause the character to be entered as ordinary data. For example, if the colon (:) is defined as the logical line end character, a colon character could be entered as follows:

```
14":45":30   which results in   14:45:30
```

ENTER Key in Execution Mode

If you are in execution mode and the execution is in progress (no conversational read outstanding and no queued print output waiting) and you press ENTER, then a message is displayed as follows:

```
*BG IN PROGRESS, INPUT IGNORED, aaaa, bbbb, cccc
```

where:

aaaa =
The number of execution units used by the job.

bbbb =
The number of lines printed thus far.

cccc =
The number of cards punched thus far.

This message is followed by the last ten lines (on an IBM 3270) or two lines (on a hardcopy terminal) placed in the print area.

Continuous Output Mode

When VSE/ICCF is writing output to the terminal, it stops after each screen of data has been transferred. This allows you to read your output. To receive the next screen of output, press ENTER or the Carriage Return key.

On an IBM 3270, you can avoid this inconvenience of pressing ENTER for each screen of output. To do this, place your terminal in continuous output mode by entering the /CONTINU command after having received the first output screen. The entire output to the terminal thus proceeds automatically with only short pauses between screens of data.

Output Compression and Truncation

Terminal output from most compilers and utility programs usually contains 120 character positions per line, some output reports contain up to 132 print positions in each line.

Some terminals such as the IBM 2740 and 3270 cannot accommodate so wide a print line. Therefore, the remaining portion of the print line is displayed in either of two formats:

- Print line format.

  The portion that does not fit on the screen is truncated. The line extends, invisibly, to the right of the screen.

  Your display is automatically set to print line format when you receive the display from

  – /DISPLAY a print-type member (or the print area $$PRINT)
  – /LIST a print-type member (or the print area $$PRINT)
  – /LISTP list output from a batch execution

  The display is in print line format also if you get it while in SP=spool mode (or RD=conversational read mode).

  SYSLOG messages, too, have their right-hand portion truncated when your display is set to print line format.

  /LIST or /DISPLAY of a compressed member and /LISTX **do not** cause automatic setting of print line format.

- Wrap-around format.

  In this format, also known as the 80-character format, the remaining portion is displayed on the next line. You see a print line as two lines in 80-character format if you display a member of print data that is not a print-type member (the member name does not end with .P).

Both formats tend to make the output confusing and difficult to read. You can alleviate this problem in one of three ways:

- Print line compression.

  You can issue the /COMPRES command while a list or execution is in progress. When compression is in effect, all occurrences of multiple blanks within the print line are reduced to a single blank.

- Truncation.

  If a job runs with the TRUNC option, only the first 78 characters of the print line are sent to the terminal. The remaining characters within the print line are ignored. The option is very useful with language compiler output because this output usually reduces very well into a line length of 78 characters. There is also another advantage: the option can be set up ahead of time (/OPTION TRUNC) on a job entry statement which then stays with the job on the VSE/ICCF library file. In addition, the TRUNC option causes jobs to run faster because the truncated portion of the output is written neither to disk nor to the terminal.

- Shifting a page right or left.

  You can use this method only if you use an IBM 3270 terminal. The method allows you to view all of the output while still retaining clarity. To shift a page, use the "/SHIFT Command" on page 117.

  You may use PF key functions in their default setting, as described in the following section. However, when you get a display in print line format, these default settings are suppressed and you would have to reset them by entering the /SHIFT OFF command.

  The string <==MORE==> in the scale line indicates that your display is in print line format.

IBM 3270 Program Function Keys

The 12 (or 24 for some keyboards) Program Function (PF) keys initially have default settings for list mode (/LIST and /LISTP) as shown in Figure 1 on page 28.

You can change them by setting the keys to a frequently needed data or command line. You can do this once for command mode, for edit mode, for execution and conversational-read mode and for list and spool mode.

If you do not have PF keys, you can still take advantage of the ability to set program functions by entering the /PFnn command instead of pressing the Function key.

You use the /SET PF command to set a given PF key to a given command or data line. Then when that PF key is pressed (or the /PFnn command is entered) the command or data line is processed just as if it had been typed in.

It is also possible to set a PF key to a portion of a command or data line. Then when data is entered followed by pressing the PF key, the data entered is combined with the PF key setting to form the actual command or data line that will be processed (for more details, see the "[/]SET Command" on page 100).

To set PF keys and still use the default PF key meanings described below, you can suspend your settings by issuing the /SET ... OFF command. To later reinstate your settings for the PF key settings, you need to specify only /SET ... ON.

The default settings of the PF keys in list and spool mode let you page forward and backward in your print output. They also allow you to shift pages left or right to view lines which exceed 80 characters.

You can page forward or backward and shift left or right also by using the /SKIP and /SHIFT commands, respectively. Use these commands for print data because, as was mentioned earlier, the PF-key default settings are suppressed whenever you receive the display of print data; you would have to reset them by issuing /SHIFT OFF. Also, the /SKIP and /SHIFT offer additional controls. Therefore, ignore the default PF key settings for print data and set your PF keys to /SKIP and /SHIFT command functions. Refer to "/SHIFT Command" on page 117 and "/SKIP Command" on page 120 for details.

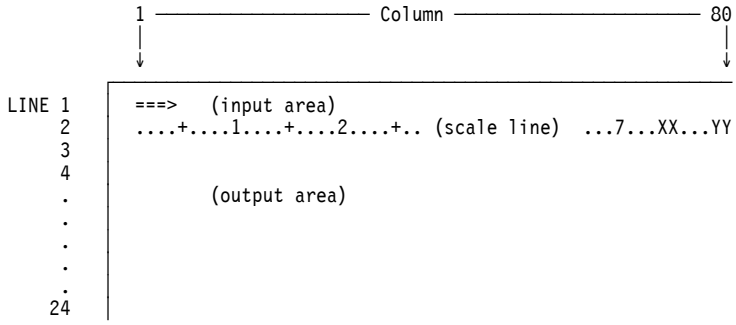
LIST MODE PRINT DISPLAY				
DISPLAY POSITIONS		PAGE		
WIDE SCRN	NORMAL SCRN	PREVIOUS	CURRENT	NEXT
1 - 132	1 - 80 ===>	PF1/PF13	PF2/PF14	PF3/PF15
31 - 162	31 - 110 ===>	PF4/PF16	PF5/PF17	PF6/PF18
31 - 162	61 - 140 ===>	PF7/PF19	PF8/PF20	PF9/PF21
31 - 162	91 - 162 ===>	PF10/PF22	PF11/PF23	PF12/PF24

Figure 1. Default PF Key Settings in List and Spool Mode

The PF1 key has another use when your terminal is not in execution mode: it can be used in place of the ENTER key. In this case the screen is not erased before the output is written back to the terminal, which can be useful if you want to keep certain information on the screen. After using the PF1 key in this way, press the ERASE INPUT key to clear the input line before typing in any data.

IBM 3270 Screen Format

The standard 3270 screen format under VSE/ICCF is shown below:



The default input area is one line, line 1 right above the scale line. However, you can expand this to 2, 3 or 4 lines so that several command or data lines (separated by end of line characters) can be entered in a single terminal transmission. To do this, use the /SET SCREEN command. The scale line always separates the input area from the output area. The scale line helps you enter data of fixed format. The scale line positions marked above by YY indicate the current mode as follows:

CM
Command mode.

IN
Input mode.

ED
Edit mode.

FS
Full screen edit mode.

LS
List-in-progress mode. Press ENTER to get the next screen.

EX
Execution is in progress in the interactive partition or the user is displaying execution mode print output.

RD
A conversational read is outstanding from the program that is running in the interactive partition.

SP
Spooling continuous list display in progress from interactive partition execution. Press ENTER to obtain the next screen.

MS
Message mode. Press ENTER to get the next screen of messages.

The character positions XX on the scale line usually contain the scale line characters .+ for column numbers. However, these positions will contain 'MR' when you are executing commands from a macro or from a multiple line sequence of input and there are more commands or lines to be processed in the macro or input sequence.

IBM 3270 Split Screen Use

The **IBM 3270** terminal lets you define (by way of the /SET SCREEN command) two or more output areas on your screen. You can thus display data in one area and switch to the other and continue processing commands. The data in the first area is retained on the screen while you are working in the second. This is useful, for example, while viewing diagnostics from a program compilation. You can keep the diagnostics in one area of the screen and switch to the other area to correct the errors.

Multiple screens are useful when PF keys are available. You can equate these keys to the various /SET SCREEN commands, thus allowing you to switch conveniently from one area of the screen to another. See the section “[/]SET Command” on page 100 for an example of multiple screens used with PF keys.

The **IBM 3278 Model 5** screen can display up to 132 characters per line, and VSE/ICCF is able to use this wide screen when the terminal is in 'SP' mode during a job execution in an interactive partition. The same is true when a /LIST or /DISPLAY command has been issued for a print-type member or for the print area. The wide screen is also used when a /LISTP command is issued to get a display of VSE/POWER list output. The VSE/ICCF Full Screen editor uses the leftmost 80 columns for data manipulation and columns 82-86 for the type 3 area.

Chapter 3. System Commands, Procedures, and Macros

Note!

The Language Environment LE/VSE and the following compilers are not supported in a VSE/ICCF interactive partition:

- COBOL for VSE/ESA
- PL/I for VSE/ESA
- C for VSE/ESA

You are recommended to instead use dynamic partitions.

The following sections describe in detail the command language that you can use in working with VSE/ICCF. The command language of VSE/ICCF consists of:

- System commands

  A system command begins with a slash (/) followed by a command name. A system command takes effect immediately; that is, its function is performed immediately after the command has been accepted. The available system commands are described in this publication.

- Macros and procedures

  Examples of macros and procedures are: PRINT, SDSERV, and SORT. Actually, a macro or a procedure consists of commands and, possibly, job entry statements, that are executed as one complete function. Therefore, the available macros and procedures are documented in this publication as if they were commands. They are referred to by their names.

- Full-screen editor commands and macros

  These can be entered only when the terminal is in full-screen edit mode. You set your terminal in full-screen edit mode by issuing the ED macro. The full-screen editor commands, including the IBM-supplied editor macros, are described in Chapter 4, “General Information about the Editor,” on page 133.

  The /EDIT command places your terminal in context editor mode. The context editor is described in Appendix B, “Context Editor,” on page 341.

- Job entry statements

  These statements are entered as if they were data. They are placed in job streams to direct the running of jobs. For example, each execution job step must be preceded by a /LOAD statement. This statement specifies what compiler or other program is to process the following data. A job entry statement begins with a slash followed by a statement name.

  The job entry statements are described in Chapter 7, “Job Entry Statements,” on page 235.

- Dump commands

  They allow you to display registers and various areas of programs that end abnormally. They are entered in response to prompting from the interactive partition dump program. To use the dump facilities of VSE/ICCF, you must specify the DUMP option.

  The dump commands are described in Chapter 8, “Dump Commands,” on page 257.

Command Syntax

Spacing: All statements, commands and procedures must begin in column 1 on their input line. One or more blank characters must follow the command verb if additional operands are coded on the line. If,

however, the logical line end character is the first character typed after an edit command, no intervening space is required (see also “Entering Multiple Commands” on page 139).

Delimiters: Operands are usually separated by spaces. Commas or parentheses are accepted as valid delimiters. For example, all of the following are equivalent:

```
/EXEC  C$BSCEXC  CLIST  PROGRAM DATA
/EXEC  C$BSCEXC  CLIST(PROGRAM,DATA)
/EXEC  C$BSCEXC,,CLIST,(PROG),(DATA)
```

Delimiters cannot be used to indicate a missing operand. Multiple delimiters have the same meaning as a single delimiter. Most commands do not require an indication of a missing operand since the nature of the operands usually indicates the presence or absence of other operands.

Abbreviations: Most system, editor, and dump commands can be abbreviated but macro names, procedure names and the majority of job entry statements cannot. The full form of system, editor and dump commands is accepted, but the minimum (shown in capital letters) is sufficient. Occasionally, an alternate command form is given that cannot be abbreviated (such as /CP for /CTLP).

For some operands, VSE/ICCF analyses no more than the required number of characters and disregards the remaining characters that may have been specified. If the analyzed characters are correct, VSE/ICCF accepts the command and performs the corresponding function. Example:

Command as Specified
Command as Accepted by VSE/ICCF

/SWITCH LIB123
/SWITCH LIB

Passwords

You can supply a four-character password when saving a member in the library. The password must begin with a non-numerical character and must not be PRIV or PUBL. Any later access to the member must include the password.

Reading the Examples

1. **Typewriter Terminal Example** The lines entered by the terminal user are shown in lowercase letters; system responses are shown in uppercase letters.

```
*READY            Shows the system's readiness to accept commands.
/switch 12        Shows the command as entered by the terminal user.
*SWITCHED         Shows the reaction of the system.
*READY
```

2. **Screen Terminal Example**: This example is in two parts. The screen on the left shows the display before ENTER is pressed. The screen on the right shows the system's response.

```
/switch 12_
 ...+....1....+....2....+....3....+....4....+....5....+ ..CM
*READY
```

```
_
...+....1....+....2....+....3....+....4....+....5....+.. ..CM
*SWITCHED
*READY
```

Summary of System Commands, Procedures, and Macros

Table 1 on page 33 lists the available commands, procedures, and macros. In Table 1 on page 33, the column "DBCS Support" indicates whether they:

Allow you to process mixed data – by Y.
Do not support the handling of mixed data – by N.
Operate independently of the type of date – by I.

The "Type" column tells whether the listed item is:

a command – C
a macro – M
a procedure – P

Table 1. Summary of System Commands, Procedures, and Macros

Operation	Function	DBCS Support	Type
$	See the /RUN command.	I	C
ASSEMBLE	Causes a library member to be processed by the VSE assembler.	I	P
/ASYNch	Places the terminal into asynchronous execution mode.	I	C
/ATten	Requests to communicate with the running job.	I	C
/CANcel	Terminates input, execution, or list mode.	I	C
/COMpres	Displays output to your terminal in compressed form.	Y	C
/CONNect	Logically connects a secondary library to the primary library.	I	C
/Continu	Places the terminal into continuous output mode.	I	C
COPYFILE	Creates a copy of a library member under a specified name in your primary library.	I	M
COPYMEM	Creates a copy of a library member under a specified name in a specified library.	I	P
/COUnt	Returns the size of a VSE/ICCF library member.	I	C
/CP	See the /CTLP command.	I	C
CPYLIB	Copies a library member from one library to another.	I	P
/CTL	See the /SET command.	I	C
/CTLP	Used to display and alter the status of jobs in VSE/POWER queues (VSE/ICCF administrator only).	I	C
/DELete	Deletes the last line entered in input mode.	I	C
/DISPC	See the /DISPLAY command.	Y	C
/Display	Displays, with line numbers, the contents of the input, punch, print, or log area, or of a library member.	Y	C
/DQ	Displays the contents of the VSE/POWER queues.	I	C
/ECHO	Displays the data specified as operand.	Y	C
ED	Invokes the full-screen editor directly.	Y	M
/EDit	Places the terminal into context-editor mode.	I	C
EDPRT	Places the terminal in full-screen edit mode for editing the contents of the print area.	Y	M
EDPUN	Places the terminal in full-screen edit mode for editing the contents of the punch (stack) area.	N	M
/END	Ends an input session normally.	I	C
/ENDRun	Ends an input session and runs the job in the input area.	I	C
/EP	See the /ERASEP command.	I	C

Table 1. Summary of System Commands, Procedures, and Macros (continued)

Operation	Function	DBCS Support	Type
/ERASEP	Removes a file from a VSE/POWER queue.	I	C
/EXec	Runs a job or a procedure.	I	C
FORTRAN	Causes a library member to be processed by the FORTRAN compiler.	I	P
GETL	Retrieves VSE/POWER list queue output and places it in a library member or in the print area.	I	P
GETP	Retrieves VSE/POWER punch queue output and places it in a library member or in the punch or the print area.	I	P
GETR	Retrieves a job enqueued in the VSE/POWER reader queue and stores it as a member of the VSE/ICCF library file.	I	P
/GRoup	Creates a generation member group in your library.	Y	C
/HARdcpy	Places an IBM 3270 into hardcopy output mode.	Y	C
HC	Switches an IBM 3270 terminal to hardcopy mode, executes a specified command, and returns to normal display mode.	Y	M
HELP	Displays how-to-use information about VSE/ICCF commands.	Y	M
INPut	Places the terminal into input mode.	N	C
/INSert	Copies all or part of a library member, or the work area, into the input area.	I	C
/LIBC	See the /LIBRARY command.	I	C
/LIBrary	Displays library directory information.	I	C
LIBRC	Catalogs a VSE/ICCF library member into a VSE library.	I	M
LIBRL	Displays a member from a VSE library.	I	M
LIBRP	Punches a member from a VSE library.	I	M
/List	Displays the contents of a work area, or a library member, without line numbers.	Y	C
/LISTC	See the /LIST command.	Y	C
/LISTP	Displays print output from the VSE/POWER list queue.	Y	C
/LISTX	See the /LIST command.	Y	C
LOAD	Causes an object program to be loaded into storage and run.	I	P
/LOCP	Locates a character string in a VSE/POWER list file.	N	P
/LOGOFF	Ends the session.	I	C
/LOGON	Starts the session.	I	C
/LP	See the /LISTP command.	Y	C
/MAil	Displays general and individual mail.	Y	C
/MSG	Displays messages.	N	C
MVLIB	Moves a library member from one library to another.	I	P
/PASswrd	Changes the logon password.	I	C
/PFnn	Simulates the effect of a PF key.	I	C

Table 1. Summary of System Commands, Procedures, and Macros (continued)

Operation	Function	DBCS Support	Type
PRINT	Routes the contents of a library member or of the input area to a hardcopy printer associated with an IBM 3270 terminal.	Y	M
/PROmpt	Sets line number or inclusion prompting on or off.	I	C
/PROTect	Adds a password to or removes it from a member, or changes the PRIVATE/PUBLIC status of a member.	Y	C
/PURge	Purges a member from the library.	I	C
RELIST	Routes the contents of the print area to the printer.	Y	M
/RENAMe	Changes the name of a library member.	I	C
/RENUM	See the /RESEQ command.	Y	C
/REPlace	Replaces a library member with the contents of the input area.	I	C
/RESeq	Changes or adds sequence numbers in or to a library member.	Y	C
/RETriev	Retrieves a previously entered command and displays it at the terminal.	I	C
/RETURN	Returns the terminal user to the Interactive Interface of z/VSE.	I	C
/ROUTEP	Routes a VSE/POWER output file to a printer or a punch.	I	C
/RP	See the /ROUTEP command.	I	C
RPGIAUTO	Invokes AUTO REPORT and compiles with RPG II.	I	P
RPGII	Causes a library member to be processed by the RPG II compiler.	I	P
RPGIXLTR	Invokes the RPG II DL/I translator.	I	P
RSEF	Calls RSEF, the RPG II Source Entry Facility.	I	P
/RUN	Runs the job in the input area.	I	C
/SAve	Saves (files) the contents of the input area as a library member.	I	C
SCRATCH	Removes DISP=KEEP files from the dynamic space area after they are no longer required.	I	P
SDSERV	Displays the (sorted) directory of a primary or connected library, or of the common library.	I	P
/SENd	Sends a message to the system operator, or to another terminal user.	N	C
/SET	Sets VSE/ICCF system controls, 3270 screen features, control characters, VSE/ICCF features, and editor auto-insert feature.	I	C
/SETIme	Sets the maximum number of execution units a job is to use.	I	C
/SHIft	Moves the display of print line data left or right.	Y	C
/SHow	Displays the settings of various VSE/ICCF control values.	I	C
/SKip	Skips forward or backward in displayed output.	Y	C
SORT	Sorts a library member and places the output as directed.	I	P
/SP	See the /STATUSP command.	I	C
$SPACE	Causes the DTSSPACE program to be run in an interactive partition to display the allocated dynamic disk areas.	I	P
/SQueeze	Converts a member to compressed format.	Y	C

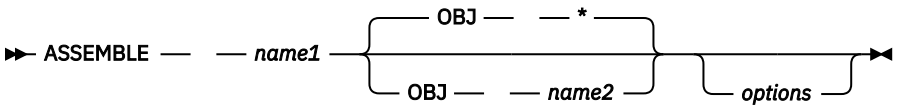
Table 1. Summary of System Commands, Procedures, and Macros (continued)			
Operation	**Function**	**DBCS Support**	**Type**
/STatus	See the /SHOW command.	I	C
/STATUSP	Displays the status of an individual VSE/POWER job.	I	C
STORE	Saves the contents of the punch area as a library member.	I	M
SUBMIT	Submits VSE/ICCF or VSE job streams to be run in a batch VSE partition.	I	P
/SUMry	Gives a summarized display of a library member.	Y	C
/SWitch	Switches from one library to another.	I	C
/SYNch	Terminates asynchronous execution mode and resets normal execution mode.	I	C
/TABset	Sets the logical tab positions.	Y	C
/Time	Displays date/time and execution units in the last background execution.	I	C
/USers	Displays the number of VSE/ICCF users logged on, and user profile information.	I	C

$ Command

See the section "/RUN or $ Command" on page 94.

ASSEMBLE Procedure

The procedure causes a library member to be processed by the VSE assembler.



name1
Is the name of a member in the library containing the assembler language source program to be assembled.

OBJ name2
OBJ *
For name2, specify the library member name of the object module resulting from the assembly. This name can have up to eight characters, and its first character must be alphabetic. If you specify a name of more than eight characters, only the first eight are taken.

If the member is to be submitted for processing under control of VSE/POWER, certain names should not be used. For details, see the restriction "6" on page 331.

The member need not initially exist in your library. If a member by the specified name exists, you will be prompted whether the object module is to overlay the existing member.

Specify OBJ * or omit the operand if the generated object module is to be placed into your punch area.

options
Are one or more /OPTION job entry statement options that alter the way the assembler is processed (LIST, NODECK, XREF, and so on).

If you specify the NODECK option, no object module will be produced even if the OBJ operand is present. Specify the NORESET option if multiple object decks are to be stacked in the punch area.

Use the LOAD procedure to load and run the object module produced by the ASSEMBLE procedure.

This procedure still uses

```
/LOAD ASSEMBLY
```

This will automatically be interpreted as

```
/LOAD ASMA90
```

with

```
PARM='SIZE(MAX),CPAT(SYSL),EX(LBX(EDECKXIT)),FOLD'
```

To exploit the full High Level Assembler instruction set

```
/LOAD ASMA90
```

must be specified. Please consult *High Level Assembler User's Guide* for recommended settings.

Note: Samples throughout the VSE/ICCF literature may show:

```
// EXEC ASSEMBLY

or

/LOAD ASSEMBLY
```

Examples

```
1.  assemble proga
    load * data *

2.  assemble progb obj objmem xref norld
    load objmem jes fildef data inpdata
```

/ASYNCH Command

The command places your terminal into asynchronous execution mode while a job is running in an interactive partition. You can thus do other terminal work such as entering or editing data while your job is running.

This command is effective only in execution mode. You cannot use it when a conversational read is pending, nor while displaying print output.

Normally your terminal is blocked while you have a job running in an interactive partition. However, with the /ASYNCH command you can disconnect your terminal from this execution and carry on with other terminal work.

For example, you may have a long compilation running. By issuing the /ASYNCH command, you can return your terminal to command mode and carry on with other work, such as editing. However, do not attempt to edit the input area, or any member involved in the execution job stream.

When you have finished editing, issue the /SYNCH command. If the compilation is finished, the display will start. It is not possible to run a second job while your terminal is in execution mode.

While in asynchronous execution mode you can enter the /SHOW EXEC command. In response to this command, VSE/ICCF will tell you whether your job has reached a point where it requires the terminal.

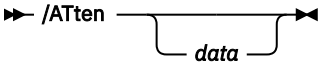
Example

```
*READY
/ex compile
```

```
*RUN REQUEST SCHEDULED FOR CLASS=A
/asyn
*ASYNCHRONOUS EXECUTION MODE ENTERED
ed memba
 ...
 ...                    (Edit commands)
 ...
*READY
/syn
 ...
 ...                    (Job output)
 ...
```

/ATTEN Command

The command signals that you want to communicate with a job that is running in an interactive partition.



data
The data to be passed to the program.

This command is effective only:

In execution mode while a program is running.
While a conversational read is outstanding.
While job output is being displayed.

On an IBM 3270 terminal, the command can be invoked also by pressing the Display/Attention key. [1]. On an IBM 2741 terminal, the Attention key has the same function.

If the program has an operator communication exit (VSE STXIT OC macro), the /ATTEN command causes a program interrupt, and the exit routine receives control. If the /ATTEN command is entered while print output is being displayed, any data remaining in the print area will be bypassed. If the /ATTEN command is entered while a conversational read is outstanding, any data supplied to the command as an operand is used to satisfy the conversational read (this does not apply to the PA1 key, since no data can be passed with a PA key).

If (1) the /ATTEN command is entered while a program without an operator communication exit is producing print data and (2) that data has not yet been displayed, the accumulated print output is displayed. However, this reduces the amount of data retained in the print area. Later print output will be written from the beginning, thus overlaying part or all of the print area's previous contents.

Example

```
*READY
/run ditto
*RUN REQUEST SCHEDULED FOR CLASS=A
DITTO FUNCTION
?                           (Request conversational input)
fsr,280,1000
/atten                      (Interrupt execution)
DITTO FUNCTION
?
```

/CANCEL Command

The command terminates whatever activity is in progress at your terminal. The command is effective in input, list and execution mode.

[1] The definition of the Display/Attention key is a VSE/ICCF tailoring option. The default is PA3. It may be different for your system

ABort
Need be specified only for a program that uses the /CANCEL command as an attention signal. The operand indicates that a cancel is requested instead of the attention signal.

Depending on the mode in effect when the command is entered, /CANCEL causes the following:

- Input mode – the input session is terminated, the input area is cleared, and command mode is entered.
- List mode – when a foreground job is in progress, printing is terminated, and the terminal is returned to the mode in effect before the display request. For example, if your terminal was in command mode when you entered the /LIST command, the terminal is returned to command mode.
- Execution mode – when a background job is in progress, or when print output is being displayed, the first /CANCEL command cancels it. However, the remaining data from the print area continues to be displayed. A second /CANCEL terminates also the display.

The terminal halts after each screen has been transferred. You can issue a /CANCEL command at this time. Even if the terminal is in continuous output mode (see /CONTINU command), there should be enough time between screens to enter the command.

Note:

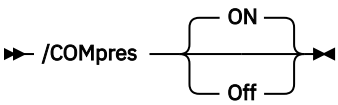
1. The /CANCEL command can also be entered as the response to a request for terminal input from a conversational program. In this situation, the background job that issued the request is canceled.
2. If /CANCEL is issued while output from a procedure is being displayed, you may have to skip to the end (/SKIP END) and then issue /CANCEL. This would bypass the print output.

Example

```
*READY
/list longmem
FIRST OUTPUT LINE
SECOND OUTPUT LINE
(pause)
 ...
/cancel
*END PRINT
*READY
```

/COMPRES Command

The /COMPRES command compresses output before it is displayed or printed. It may only be entered in list or execution mode.



ON
Sets compression on. Specifying no operand (or any operand except 'OFF') has the same effect.

Off
Sets off space compression.

When alphanumeric data is being processed, space compression reduces all multiple space characters (blanks) to a single blank. When mixed data is being processed, space compression reduces all spaces that occur in subfields of double-byte characters to a double space. The /COMPRES command, however, has no effect if a DBCS member or the print area is displayed with the /LISTX command on an IBM 5550 with 3270 emulation.

This command is useful for terminals whose line widths do not accommodate a full print line. Some printouts lend themselves well to space compression; others may yield better results with the TRUNC option of the /OPTION statement.

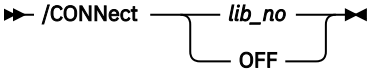
Even if some of the data has already been printed, you can still print a report or display data in compressed format. Just enter the /COMPRES command followed by a /SKIP command with a negative operand. This causes print output to start from the beginning.

Example

```
*READY
/list anymemb
A   B   C
D   E   F
/compres
G H I
J K L
```

/CONNECT Command

The /CONNECT command logically connects another library to your primary library. It is effective in command mode.



lib-no
Is the library identification number of the library to be connected to your primary library.

A connected library is scanned whenever you request access to a member that cannot be found in your primary library. However, you cannot update members in a connected library. Thus commands such as /EDIT and /RESEQ will fail if the member is in a connected library. To update these members, use the /SWITCH LIBS command to make the connected library your primary library.

You can connect only to public libraries and to private libraries that are specified as alternate ones in your user profile. The VSE/ICCF administrator can connect to any library.

OFF
Cancels the effects of a previous /CONNECT, thus leaving no library connected.

/LIBRARY or /LIBRARY CONN commands display the identification numbers of your primary library and your connected libraries (if any).

Note:

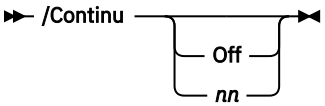
1. If the /CONNECT command is used within a procedure, it is effective until the procedure has ended.
2. If a procedure includes a member from a connected library via a /INCLUDE, you must issue the /CONNECT command before you call the procedure.

Example

```
*READY
/list cmemb           (Member CMEMB is in library 5)
*FILE NOT IN LIB
/connect 5
*CONNECTED
*READY
/list cmemb
RECORD 1
RECORD 2
*END PRINT
*READY
```

/CONTINU Command

The command causes output to the terminal to be displayed continuously. The command is valid in list and execution modes; it ends when you leave these modes.



Off
Terminates continuous display.

nn
Sets the pause between screens to 'nn' seconds, where 'nn' can be a value from 1 to 255.

The default is:

1 second for a terminal other than IBM 3270
6 seconds for an IBM 3270 terminal

Normally, the terminal halts while each screen of data is being displayed. To receive the next screen, press ENTER or the Carriage Return key. During one of these pauses you can enter the /CONTINU command.

Continuous output display with a time delay factor is useful with the IBM 3270 Display terminal. You can adjust the rate of paging to your own convenience.

During continuous output, a short pause occurs between screens.

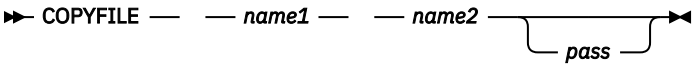
To stop continuous output, issue the /CONTINU OFF or /CANCEL command. Continuous output is also stopped if you change to edit mode or if you enter the /LIST or /LIB command while in asynchronous execution mode.

Example

```
*READY
/exec printjob
*RUN REQUEST SCHEDULED
 ...                        (Output from your job)
/con                        (Entered at first halt, or before
 ...                        the first data is displayed)
 ...
 ...                        Continuous output proceeds)
*****JOB TERMINATED - ICCF RC 00, EXEC RC 0000, NORMAL EOJ
*READY
```

COPYFILE Macro

The COPYFILE macro creates a copy of a VSE/ICCF library member and saves that copy in your primary library under another name. It is valid only in command mode.



name1
Is the name of the member which is to be copied.

name2
Is the name which will be given to the copy of the original member. If the member is to be submitted for processing under control of VSE/POWER, certain names should not be used. For details, see the restriction “6” on page 331.

pass
Is a four-character password which applies to name1 or to name2 or both.

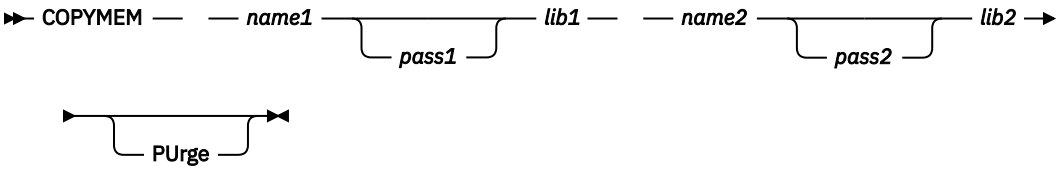
Example

Make a copy of the member called STATFIL and save this copy as a member named STATFILB:

```
COPYFILE STATFIL STATFILB
```

COPYMEM Procedure

The COPYMEM procedure creates a copy of a VSE/ICCF library member and saves it under the specified name in the specified library.



name1
Is the name of the member which is to be copied.

pass1
Is the password assigned to the member that is to be copied.

lib1
Is the number of the library which contains the member that is to be copied.

name2
Is the name which will be given to the copy of the original member. If the member is to be submitted for processing under control of VSE/POWER, certain names should not be used. For details, see the restriction “6” on page 331.

pass2
Is the four-character password for the copy of the original member.

lib2
Is the number of the library into which the original member is to be copied.

PURGE
Causes the original member to be purged from its library.

You must issue the /CONN OFF command before you can copy a member from or to a connected library. The specified libraries must be public or private ones to which you have access according to your user profile.

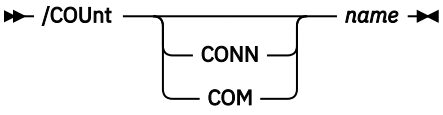
Example

1. Copy member MEMA with password PAS1 from library 3 to library 7.
2. Name the new member MEMB and give it the password PAS2.
3. Purge member MEMA from library 3.

```
COPYMEM MEMA PAS1 3 MEMB PAS2 7 PU
```

/COUNT Command

The command lets you know how many records a VSE/ICCF library member consists of. It is valid in command mode only.



CONN
The connected library is to be searched for the specified member.

COM
The common library is to be searched for the specified member.

name
Specifies the name of the member. The names of the temporary areas $$PRINT, $$PUNCH, $$LOG, and $$STACK must not be specified as 'name'.

If neither CONN nor COM is specified, your primary library will be searched for the specified member. When the member is found, VSE/ICCF displays the following message:

```
nnnnnn RECORDS IN MEMBER xxxxxxxx   [text]
```

where 'text' is one of the following:

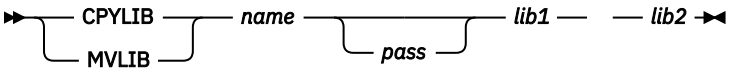
```
** COMPRESSED **
**  UPDATE IN PROGRESS **
blank
```

/CP Command

See the the section "/CTLP and /CP Commands" on page 44.

CPYLIB and MVLIB Procedures

The CPYLIB procedure copies a member from one library to another. The MVLIB procedure performs the same function except that the member is purged from the original library.



name
Is the name of the library member to be copied or moved from one to another library. The attributes of the already existing member are not transferred to the new member.

pass
Is the password to be specified if the member is password-protected.

lib1
Is the number of the library which contains the member that is to be copied or moved.

lib2
Is the number of the library into which the member is to be copied or moved.

The libraries specified must be public or private libraries to which you have access according to your user profile. If library lib2 already has a member with the specified name, the existing member will not be overwritten.

You must issue the /CONN OFF command before you can copy or move a member from or to a connected library.

Example

Copy member MEMBRZ from library 3 to library 7:

```
cpylib membrz 3 7
```

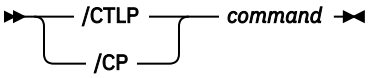
[/]CTL Command

See the [/]SET command.

/CTLP and /CP Commands

The /CTLP command passes VSE/POWER commands to VSE/POWER.

It can only be used in command mode and by the VSE/ICCF administrator or a user with special authority.



command
Is a VSE/POWER command which can be passed to VSE/POWER via the spool access support of that program.

For a list of the VSE/POWER commands that can be passed, see the description of the PWRSPL macro in the VSE/POWER Application Programming. For a description of the VSE/POWER commands, refer to VSE/POWER Administration and Operation.

Note: Except for the display of queue entries, a response of VSE/POWER to this command is limited to a single line.

Examples

1. The command passed to VSE/POWER causes a display of all list queue jobs in output class S.

```
    /ctlp pdisplay lst,cclass=s
```

2. If your VSE system is a node of a VSE/POWER controlled network, this transmits a command to the node NODEX; the transmitted command causes all jobs submitted by user ID UUUU to be deleted. The system NODEX must run with VSE/POWER.

```
    /cp pxmit nodex,pdelete rdr,cuser=uuuu
```

/DELETE Command

The command deletes the input line most recently entered into the input area.

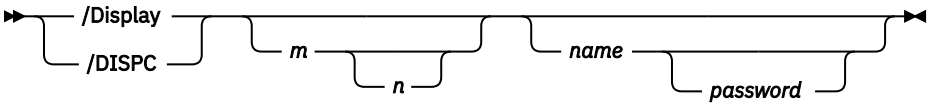
The /DELETE command is valid only in input mode. Successive /DELETE commands erase successive previous input lines.

Example:

```
    *READY
    /input
    aaaaa
    bbbbb
    /delete
    ccccc
    ddddd
    eeeee
    /delete
    /delete
    /end
    *READY
    /list
    AAAAA
    CCCCC
    *END PRINT
    *READY
```

/DISPC and /DISPLAY Commands

Each of the commands allows you to display the contents of the input, punch, print, or log areas, or of a non-compressed library member. The data to be displayed can be alphanumeric or mixed data. The commands are valid in command and input mode.



m
Is the first line number to be displayed. If omitted, '1' is assumed. The maximum is 99999.

n
Is the last line number to be displayed. If omitted, or if it exceeds the number of lines in the file, the display will proceed through end of file. The maximum is 99999.

name
Is the name of the library member to be displayed. If name is omitted, the input area is displayed. Name can also be $$PUNCH, $$PRINT or $$LOG to display these areas.

password
Is the password for the member. It need not be specified if the member is not password-protected.

The /DISPC command displays the file (or member) in continuous mode without requiring the /CONTINU command.

Line numbers appear at the left of each line. To display a part of a file, specify the first and last line number of the part to be displayed. If only one number is specified, it is assumed to be the starting line number. The file will then be displayed from this line to the end of the file.

For a print-type member, the first two bytes of each record are interpreted as print control characters, and the member is displayed in print line format. The default PF key settings (as described in Figure 1 on page 28) are **not active**; you can use your own PF key settings, instead.

The print-type member is displayed using all 132 characters of the IBM 3278 Model 5 wide screen. On a hardcopy printer, such a member is printed using the total line size. Moreover, line numbers refer to print records, which are not identical with physical records; if a record of the member does not contain valid print-control characters in the first two columns, this record is printed as an 80-byte record.

Note:

1. You can use the /SKIP command to skip over certain parts of the file.
2. The /DISPLAY and /LIST commands have identical formats and nearly identical functions, except that the /DISPLAY command also shows line numbers.
3. If columns 73-80 contain sequence numbers, the print lines are truncated at column 72, unless the display width (see /SET LINESIZE) has been altered from the default width of 72. If any other data appears in 73-80, it will be displayed regardless of the line size setting. For print-type members the line numbers will be treated as print data.
4. When this command is used in input mode, your terminal will go into list mode until the display is complete. Then input mode will be re-instated.

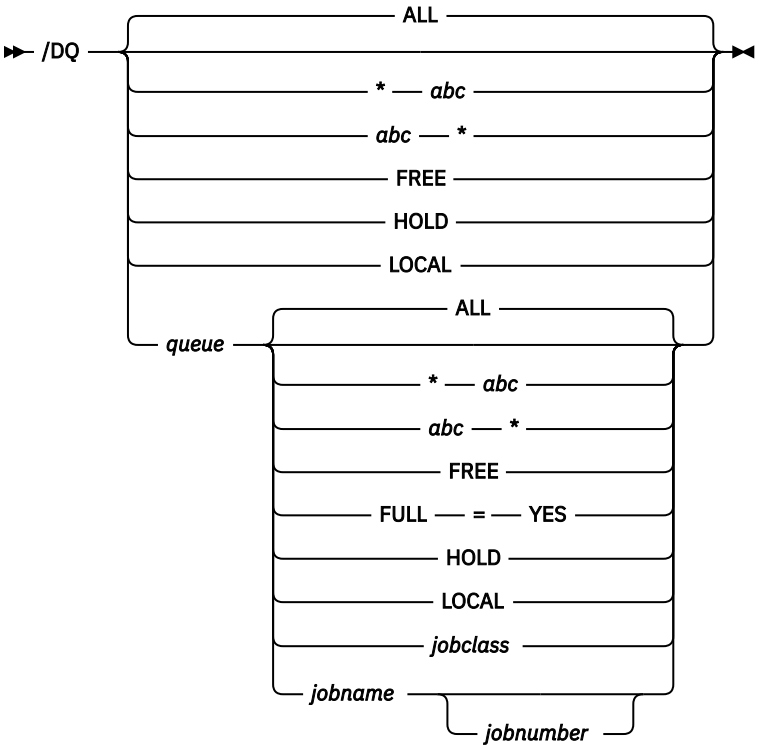
Example

```
*READY
/input
first line
second line
third line
/display
0001 FIRST LINE
0002 SECOND LINE
0003 THIRD LINE
*END PRINT
```

```
    fourth line
    fifth line
    /end
    *READY
    /save sampl1
    *SAVED
    *READY
    /d sampl1
    0001 FIRST LINE
    0002 SECOND LINE
    0003 THIRD LINE
    0004 FOURTH LINE
    0005 FIFTH LINE
    *END PRINT
    *READY
```

/DQ Command

The command displays the contents of the specified VSE/POWER queue: list, punch, reader, or transmit.



queue
Specifies the queue for which the /DQ command is intended:

LST
For the list queue

PUN
For the punch queue

RDR
For the reader queue

XMT
For the transmit queue

If no queue is specified, status information is displayed for all jobs in all queues.

ALL
Gives the status of all jobs in the specified queue. If 'queue' is not specified, status information for all jobs in all queues is displayed. ALL is the default.

FREE
Gives the status of all jobs in the specified queue that are available for processing (that is, jobs in the *keep* or *dispatchable* state) or that are currently processing.

FULL=YES
Causes the display of all displayable information about a particular queue entry.

HOLD
Gives the status of all jobs in the specified queue that are not available for processing. These are the jobs that are in hold or leave state.

jobclass
Gives you the status of all jobs with the specified jobclass in the specified queue.

jobname
Is the job name by which the job is known to VSE/POWER.

jobnumber
Is the job number assigned to the job by VSE/POWER.

LOCAL
Gives the status of all jobs in the specified queue that were submitted from, or routed to your installation.

***abc**
abc*
Requests the status of all jobs whose names begin with the characters that you specify. For 'abc' you can specify up to eight alphanumeric characters, including the characters dollar ($), commercial at (@), hyphen (-), period (.), and slash (/).

Note:

1. The operands of the /DQ command are identical to those of the VSE/POWER PDISPLAY command. Only a subset of all possible operands is shown here; for a description of the remaining operands of the PDISPLAY command , refer to VSE/POWER Administration and Operation.
2. VSE/POWER checks for any errors in the operands of the /DQ command and displays appropriate messages.
3. The /DQ command must not be used in a procedure.

Examples

For a description of the responses shown here, refer to VSE/POWER Administration and Operation.

Example 1 – Display the contents of all VSE/POWER queues

The directory information for the reader, list, and punch queues is returned from VSE/POWER.

```
/dq_
 ...+....1....+....2....+....3....+....4....+....5....+ ..CM
*READY
_
 ...+....1....+....2....+....3....+....4....+....5....+....6. ..CM
1R46I READER QUEUE   P D C S  CARDS
1R46I CISTART  00696 3 * 2       68
1R46I JOB02    00527 3 D A      343 FROM=(AAAA)
1R46I   LIST QUEUE   P D C S  PAGES CC FORM
1R46I CISTART  00696 3 D A        3
1R46I JOB01    00439 3 L Q       12         TO=(AAAA) FROM=(AAAA)
1R46I  PUNCH QUEUE   P D C S  CARDS CC FORM
1R46I JOB01    00439 3 D A       12         TO=(AAAA) FROM=(AAAA)
*END PRINT
*READY
```

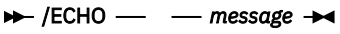
Example 2 – Display all displayable information of the VSE/POWER LST queue

```
/dq LST FULL=YES_
 ...+....1....+....2....+....3....+....4....+....5....+ ..CM
*READY
_
```

```
 ...+....1....+....2....+....3....+....4....+....5....+....6. ..CM
1R46I   LIST QUEUE     P D C S  PAGES CC FORM
1R46I CISTART  00202   3 * A        3  1
      D=12/22/1994,    L= 62
1R46I CISTART  00204   3 D A        3  1
      D=01/11/1995,    L= 60
1R46I RELIST   00211   3 D A        1  1      TO=(AAAA) FROM=(AAAA)
      D=01/12/1995,    L=  6
1R46I RELIST   00213   3 D A        1  1      TO=(AAAA) FROM=(AAAA)
      D=01/12/1995,    L=  6
1R46I RELIST   00215   3 D A        5  1      TO=(AAAA) FROM=(AAAA)
      D=01/12/1995,  S=001,L=    245
1R46I RELIST   00215   3 D A        5  1      TO=(AAAA) FROM=(AAAA)
      D=01/12/1995,  S=002,L=    300
1R46I RELIST   00215   3 D A        5  1      TO=(AAAA) FROM=(AAAA)
      D=01/12/1995,  S=003,L=    300
*END PRINT
*READY
```

[/]ECHO Command

The [/]ECHO command displays the operand data.



message
Is any text that you want to be written back to your terminal.

This command is useful in a macro to display a completion message back to the terminal. Without /, the command is effective in edit mode; with /, it is effective in any other mode.

For IBM 3270 terminals, the screen is not erased; the echo data replaces the first output line on the screen.

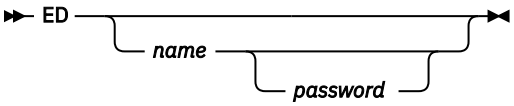
Mixed data can be displayed with the /ECHO command only if the command is invoked from a macro and if the display station is an IBM 5550 with 3270 Emulation.

Example

```
/echo operation complete
OPERATION COMPLETE
```

ED Macro

The ED macro directly invokes the full-screen editor. It is valid in command mode.



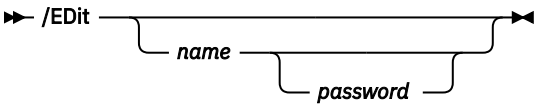
name
Is the name of the library member to be edited. If no name is specified, the input area is edited. The named member must be in your primary library.

password
Can be specified only if name is specified and the member is password-protected.

This macro includes the /EDIT command. The description of that command applies also to this macro.

/EDIT Command

The /EDIT command places the terminal in **context editor** mode. It is valid only in command mode.



name
Is the name of the library member to be edited. The member must be in your primary library. If no name is specified, it is assumed that you want to edit in the input area.

password
Is required only if the library member being edited is password-protected.

If 'name' is specified, the edit commands apply directly to the member. After you have finished editing in the input area, you can issue the SAVE command to save your edited data as a member in the library. The SAVE command returns you to command mode.

If the input area is empty when you enter the /EDIT command, you can start typing in data only after you have issued the INPUT command.

Note:

1. When editing from a local IBM 3270, verify long mode is assumed. Thus, you see a full-screen display of the part of the file you are editing. If you are using a remote IBM 3270, issue VERIFY LONG to get a full screen display.
2. To enter full-screen edit mode, issue the VERIFY FULL command after the /EDIT command.
3. Mixed data other than DBCS print-type members can be edited only in full-screen edit mode.

Example

```
*READY
/input
first card
second card
third card
/end
*READY
/edit
*EDIT MODE
locate sec
SECOND CARD
ch /seco/2
2ND CARD
save sampl2
*SAVED
*READY
```

EDPRT and EDPUN Macros

The EDPRT and EDPUN macros are used to enter edit mode in order to view or edit the print area and the punch (stack) area, respectively. These macros are valid only in command mode.

The EDPRT macro helps you find compilation errors within the print area. It places your terminal in edit mode, which allows you to use commands such as LOCATE to find certain data in the print area. You can use the same approach to modify data before routing the contents of the print area to the system printer via the RELIST macro. However, additions or deletions of entire lines cannot be made using this macro.

The punch area can also be modified in this way before saving it in the library or before using it to form part of a job stream (via the /INCLUDE $$PUNCH statement).

/END Command

The command ends an input session. It is valid only in input mode.

The /END command closes the input area and places the terminal in command mode. To add further data to the input area you must enter edit mode.

If you are using an IBM 3270 you can also use the Cancel key [2] to terminate the input session.

Example

```
*READY
/input
card 1
card 2
card 3
/end
*READY
/list
CARD 1
CARD 2
CARD 3
*END PRINT
*READY
```

/ENDRUN Command

The /ENDRUN command has the same effect as an /END command followed by a /RUN command. It can be used only in input mode.

This command terminates the input session and runs the job in the input area.

Example

It assumes that the semicolon (;) has been defined as the logical tab character.

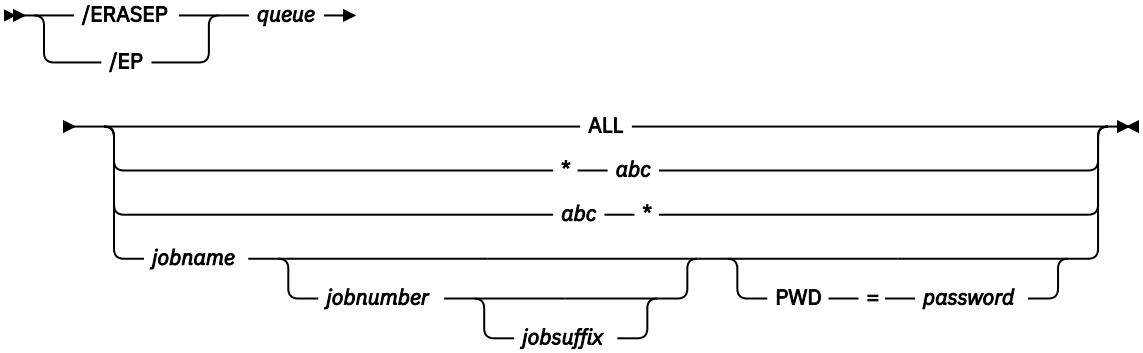
```
*READY
/input
/load vfortran
;write (3,10)
10;format (' this means the program ran')
;end
/endrun
*RUN REQUEST SCHEDULED
  (compiler output)
TOTAL MEMORY REQUIREMENTS 000154 BYTES
HIGHEST SEVERITY LEVEL OF ERRORS FOR THIS MODULE WAS 0
***** BEGIN LINKNGO
  12,236 BYTES REQUIRED FOR PROGRAM STORAGE
  PGM LOADED AT 09B100
  XFER ADDRESS 09B100
*****
THIS MEANS THE PROGRAM RAN
**** JOB TERMINATED - ICCF RC 00, EXEC RC 0000, NORMAL EOJ
*READY
```

/EP and /ERASEP Commands

The command removes one, several, or all jobs from the specified VSE/POWER queue.

[2] The definition of the Cancel key is a VSE/ICCF tailoring option. The default is PA2. It may be different for your system.

It is valid in command mode only.



queue
Specifies the VSE/POWER queue to which the command applies in the form:

RDR for reader queue
LST for list queue
PUN for punch queue
XMT for transmission queue

You may want to erase a job from the reader queue (to cancel a job that you have decided not to run). If processing for the job has already started, send a message to the system operator with the /SEND command asking him to cancel the job for you.

When specified with 'LST' or 'PUN', /ERASEP removes the print or punch output of the named job from the specified output queue. For example, if you have already viewed the output from a job, use this command to erase the output. **However**, the command does not erase a list queue entry that is available to you due to a DEST=(node,ANY) specification for the entry in the VSE/POWER JECL statement. To have such an entry erased, ask your administrator or the central operator to do this for you.

jobname
Is the name (2 to 8 characters long) of the VSE/POWER job that is to be erased.

If you request a job (identified by a job name and, possibly, by a job number) to be removed and this job name (and number) occurs more than once in the queue, the job occurring first will be removed.

jobnumber
Specifies the job number assigned to the job by VSE/POWER.

jobsuffix
Is the job suffix which designates the segment number. If this operand is omitted, the entire job will be erased.

PWD=password
Is the password given to the job when it was submitted to VSE/POWER.

ALL
Causes all jobs to be removed from the specified VSE/POWER queue. It can be specified only by the VSE/ICCF administrator.

***abc**
abc*
Causes all jobs whose names begin with the characters that you specify to be removed from the 'queue' queue. For 'abc' you can specify up to eight alphameric characters, including the characters dollar ($), commercial at (@), hyphen (-), period (.), and slash (/).

Note:

1. The operands of the /ERASEP command are identical to those of the VSE/POWER PDELETE command, except for *jobname* which must be at least 2 characters long. Contrary to the PDELETE command, the /ERASEP command does not allow *class* as positional operand.

Only a subset of all possible operands is shown here. For a description of the remaining operands of the PDELETE command , refer to VSE/POWER Administration and Operation.

2. The /ERASEP command has replaced the /PURGEP command. /PURGEP is still valid, but its use is not recommended.

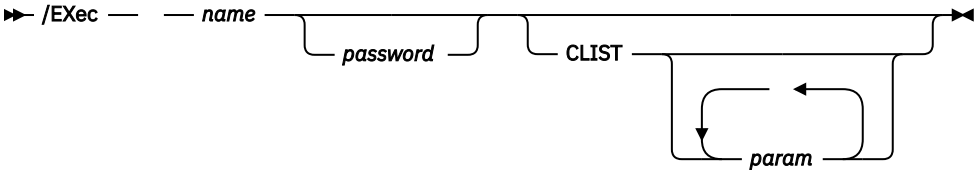
Examples

Erase the punch output of job 'MYJOB' from the VSE/POWER punch queue. The job number is 0200 and the password is 'SECRET':

```
/erasep pun myjob 0200 pwd=secret
*OK
*READY
```

/EXEC Command

The /EXEC command runs a job stream, or a procedure; either must be a member in your library. The command is effective only in command mode.



name
Is the name of a member in your library or of the punch area $$PUNCH. The member or the punch area must contain a job stream or a procedure.

A job stream to be run must begin with a /LOAD job entry statement. It can comprise just one step or two or more steps; it can consist only of job entry statements or of job entry statements and the data to be processed. If that data is not part of the job stream, it must be in a different library member and referenced by the /INCLUDE statement. If the member is password-protected the 'password' operand must be specified.

password
Need be specified only if the library member is password-protected.

CLIST
Indicates that the specified member contains a procedure which must be processed using the DTSPROCS utility.

param
Are parameters which are to be passed to the procedure being run. Specify parameters only if the procedure has been written using variable symbols in place of operands. The parameters specified on the /EXEC command will replace the variable symbols in the procedure (for more information about procedures, see Chapter 9, “Writing a Procedure or a Macro,” on page 273).

The command /EXEC $$PUNCH allows you to load and run an object deck that is in the punch area. Normally the output of a language translator is placed in a punch area from which it is retrieved by the LINKNGO program for execution. Sometimes it may be useful to re-run the object module a second or third time without recompiling the source program. To do this, issue the /EXEC $$PUNCH command.

You must use the CLIST form of the /EXEC command if the procedure to be run does not include the / LOAD DTSPROCS statement

```
/EXEC MYPROC CLIST
```

If the procedure list has been defined with variable symbols in place of operands, you must supply parameter values in addition. For example, the command

```
/EXEC BSCEXEC CLIST OBJMOD DTAMOD
```

runs the user-written procedure BSCEXEC. This procedure causes an object deck stored in the library as member OBJMOD to be loaded and run, using the input data stored in the library as member DTAMOD. If the implied execute function has not been turned off, the /EXEC command name and the CLIST operand need not be specified. Thus, the preceding example can also be specified as follows:

BSCEXEC OBJMOD DTAMOD

Note:

1. The /EXEC statement destroys the contents of the input area. To preserve the input area, enter a /SAVE or /REPLACE command before entering the /EXEC command.
2. After an /EXEC command has been entered and execution has finished, the job can be rerun with a /RUN (or $) command. No operand is needed; an /INCLUDE for the member to be run has already been placed into the input area when the /EXEC command was processed.
3. If you press ENTER while execution is in progress, *BG IN PROGRESS is displayed together with the last print lines placed into the print area. The last two lines are displayed for typewriter terminals and the last ten for the IBM 3270.

Examples:

1. Build a job stream in input area and execute it:

```
*READY
/input
/load basic
/option getvis=100
run *
010 print 'enter your first name'
020 input a$
030 print '   hello   ', a$
040 end
/data incon
/end
*READY
/save hello                 (Saves the job stream)
*SAVED
*READY
/exec hello                 (Runs the job stream)
*RUN REQUEST SCHEDULED
 ...
 ...                        (Output of compile, load, and run)
 ...
```

2. Execute the procedure SDSERV:

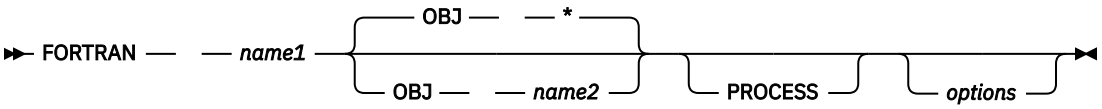
```
*READY
/exec sdserv clist          (This procedure runs and displays
*RUN REQUEST SCHEDULED      a sorted list of your library)
 ...
 ...                        (Print output)
 ...
*END PRINT
```

3. Execute a procedure and pass parameters to it. The procedure loads the object deck MYPROG from the library and runs it. MYPROG reads its input from the member named MYDATA.

```
*READY
/exec myload clist myprog mydata
*RUN REQUEST SCHEDULED
 ...
 ...                        (Printed output)
 ...
```

FORTRAN Procedure

The procedure causes a library member to be processed by the FORTRAN compiler.



name1
Is the name of a member in the library containing the FORTRAN language source program to be compiled.

OBJ name2
OBJ *
For name2, specify the library member name of the object module resulting from the compilation. This name can have up to eight characters, and its first character must be alphabetic. If the member is to be submitted for processing under control of VSE/POWER, certain names should not be used. For details, see the restriction "6" on page 331.

The member need not initially exist in your library. If a member by the specified name exists already, the object module will replace that member.

Specify OBJ * or omit the operand if the generated object module is to be placed into your punch area.

PROCESS
Specifies that you are to be prompted for entry of FORTRAN compiler options such as LIST, SOURCE, and XREF. No object module will be produced if you specify the NODECK option, even if the OBJ operand is present. Specify the NORESET option if two or more object decks are to be stacked in your punch area.

options
Are one or more of the VSE/ICCF unique options of the job entry statement /OPTION, for example: ANYPHASE, CLEAR, CONTINUE. For a description of these options, see "/OPTION Job Entry Statement" on page 249.

Use the LOAD procedure to load and run the object module produced by this procedure.

Examples

1. To have an object module created, using source code stored in the library as member FORTEST, and subsequently load and run the module.

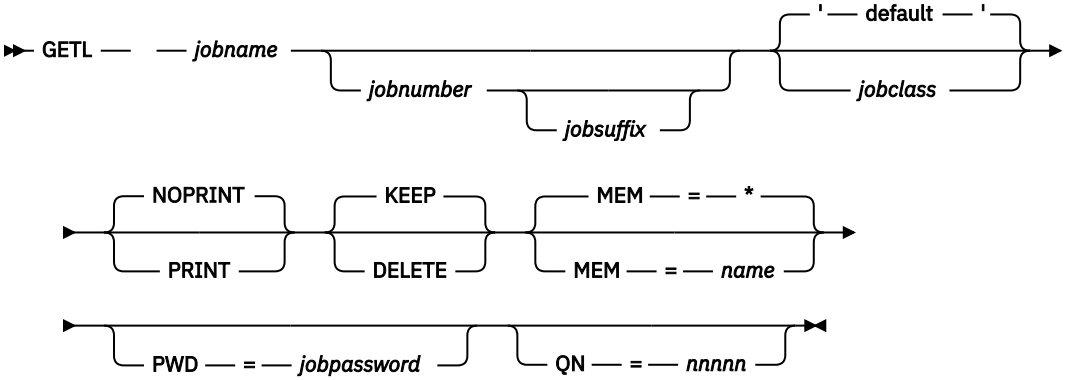
```
fortran fortest
load *
```

2. To have an object module created, using source code stored in the library as member XXPROG, store this module in the library as member OBJMEM, and subsequently load and run the module. For a description of the remaining LOAD operands, see "LOAD Procedure" on page 76.

```
fortran xxprog obj objmem
load objmem jes filder data *
```

GETL Procedure

The GETL (GET list queue) procedure retrieves output from the VSE/POWER list queue and stores this output either as a member in your library or in the print area.



jobname
Is the name of the spooled list output that you want to retrieve from the VSE/POWER list queue.

jobnumber
Is the job number assigned to the spooled list output in the VSE/POWER list queue. This number must be specified if the job name is not unique within the queue.

In general, the job number is identical to the number of the job which produced the output, but is different from that number in either of the following cases:

1. The job stream was submitted via an /INCLUDE with the ICCFSLI option and contained a * $$ LST statement.
2. The job had been run under control of the VSE/POWER PNET networking function.

jobsuffix
Is the job suffix, which designates the segment number of the output. If this operand is omitted, the first (or only) segment is assumed.

jobclass
Is the class of the spooled output in the VSE/POWER list queue. It must be a class for which no printer has been started. The default output class is Q; it may be a different class for your locality.

If the `jobclass` is numeric, the `jobnumber` and the `jobsuffix` must be specified. 0 must be specified for `jobsuffix` if the job does not have a job suffix.

NOPRINT
PRINT
Specifies whether print control characters are to be placed in the output data. PRINT causes the member to be created in the format required for processing by the“RELIST Macro” on page 85.

KEEP
DELETE
Specifies the disposition of the output in the VSE/POWER list queue after retrieval is complete.

If DELETE is specified, the job output is automatically deleted from the queue. If this output is segmented and you do not specify a job suffix, all segments of the output are deleted. An abnormal end causes the output to be kept.

MEM=*name*
MEM=*
MEM=name specifies the name of the VSE/ICCF library member into which the retrieved list output is to be stored.

If you specify MEM=* or omit the operand, the retrieved list output is placed into the print area.

By specifying '/HARDCPY devicename queuename' before invoking the GETL procedure, the requested list output can be transferred to a hardcopy printer. Under CICS Transaction Server, it can be transferred to a private print destination queue for hardcopy output.

PWD=*jobpassword*
Is the password that protects the output. This password may have been assigned to the job and its output when the job was submitted . For details see “SUBMIT Procedure” on page 127. it might have been assigned by VSE/POWER JECL. For details about password protection by VSE/POWER , see VSE/POWER Administration and Operation. You can omit this operand if there is no password protection or if you are the VSE/ICCF administrator.

Note: Jobs submitted via the card reader are protected by an unprintable password if you do not provide one. Output from such jobs can be retrieved only by the VSE/ICCF administrator.

QN=*nnnnn*
Specifies the queue entry number. The processing sequence is first the keyword parameters, and then the positional parameters. If an invalid QN=*nnnnn* is specified, keyword processing is terminated. DTSGETQ then continues to process the positional parameters.

To avoid transferring very large files, specify the point where you want retrieval to begin, and the number of lines that you want transferred. Message K871D, which appears after the GETL procedure has been invoked, allows you to do this. It tells you the size of the file that is to be transferred and asks how much you want to have transferred. For example, if you simply want to look at the errors in a large program before deciding whether to release the output to a printer, you need to view no more than the last 50 to 100 lines.

If the program DTSGETQ (the program that does the actual retrieving of the list output) is canceled for some reason, purge the library member that was created as a result of your MEM=name specification.

Note:

1. Only the submitter or the user named in the DEST operand of the VSE/POWER statement * $$ LST may access a list-output queue entry. An authorized user may specify ANY in the DEST operand (authorization is established in the VSE/ICCF user profile). This makes the list output accessible to any user.
2. The GETL procedure invokes the DTSGETQ program. This program processes the procedure based on the positional operand '`jobname`', optionally with '`jobnumber`', '`jobsuffix`', and '`jobclass`'. These operands (as far as used) must be specified in the order as shown. The remaining operands may be specified in any order.

Examples

1. Store the VSE/POWER list job 'MYJOB', jobnumber 0200 with password 'SECRET', as VSE/ICCF library member 'MEMBERA'; delete 'MYJOB' from VSE/POWER queue after successful transfer.

```
getl myjob 0200 a pwd=secret mem=membera delete
```

2. Place the VSE/POWER list job 'MYJOB' with password 'SECRET' into the print area (if the /HARDCPY command was issued beforehand, VSE/POWER transfers this output directly to a hardcopy printer).

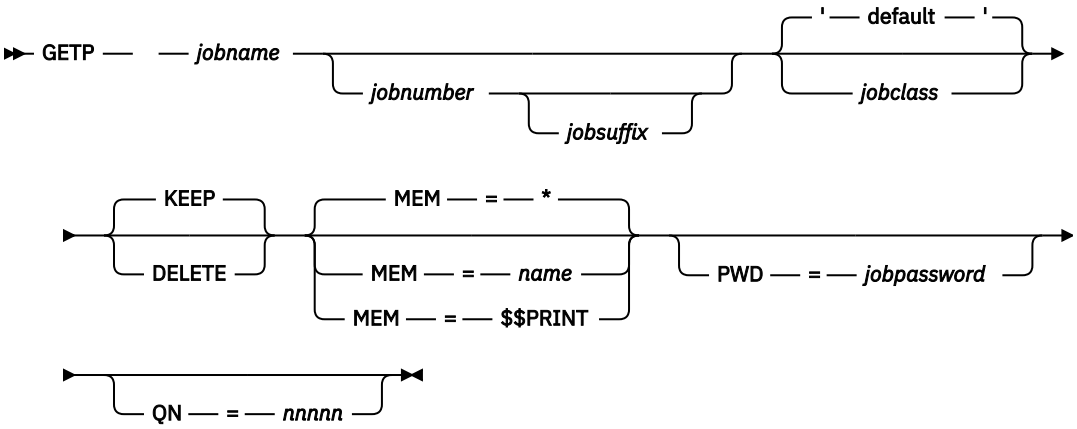
```
getl myjob mem=* pwd=secret
```

3. Store the VSE/POWER list job 'MYJOB' with password 'SECRET' as VSE/ICCF library member 'LISTJOB' with print control characters for processing by RELIST; delete the job from the VSE/POWER list queue.

```
getl myjob mem=listjob pwd=secret print delete
```

GETP Procedure

The GETP (GET PUNCH QUEUE) procedure retrieves output from the VSE/POWER punch queue and places it in a library member, in the punch area, or in the print area.



jobname
Is the name of the spooled punch output that you want to retrieve from the VSE/POWER punch queue.

jobnumber
Is the job number assigned to the spooled punch output the VSE/POWER punch queue. This number must be specified if the job name is not unique within the queue.

In general, the job number is identical to the number of the job which produced the output, but is different from that number in either of the following cases:

1. The job stream was submitted via an /INCLUDE with the ICCFSLI option and contained an * $$ PUN statement.
2. The job had been run under control of the VSE/POWER PNET networking function.

jobsuffix
Is the job suffix, which designates the segment number of the output. If this operand is omitted, the first (or only) segment is assumed.

jobclass
Is the class of the spooled output in the VSE/POWER punch queue. It must be a class for which no punch unit has been started. The default output class is Q; it may be a different class for your locality.

If the `jobclass` is numeric, the `jobnumber` and the `jobsuffix` must be specified. 0 must be specified for `jobsuffix` if the job does not have a job suffix.

KEEP
DELETE
Specifies the disposition of the output in the VSE/POWER punch queue after retrieval is complete.

If DELETE is specified, the job will automatically be deleted from queue (if the output for the job is segmented and you do not specify a jobsuffix, all segments of this output are deleted). An abnormal end causes the output to be kept.

MEM=*name*
MEM=*
MEM=$$PRINT
MEM=name specifies the name of the library member into which the retrieved punch output is to be stored. If the member may have to be submitted for processing under control of VSE/POWER, certain names should not be used. For details, see the restriction “6” on page 331.

If you specify MEM=* or nothing, VSE/ICCF places the data into the punch area.

MEM=$$PRINT causes the punch data to be placed in the print area. This allows you to view the output before deciding whether to store this output as a member of your library.

PWD=*jobpassword*
Is the password that protects the output. This password may have been assigned to the job and its output when the job was submitted . For details see “SUBMIT Procedure” on page 127. it might have been assigned by VSE/POWER JECL. For details about password protection by VSE/POWER , see VSE/

POWER Administration and Operation. You can omit this operand if there is no password protection or if you are the VSE/ICCF administrator.

Note: Jobs submitted via the card reader are protected by an unprintable password if you do not provide one. Output from such jobs can be retrieved only by the VSE/ICCF administrator.

QN=*nnnnn*
Specifies the queue entry number. The processing sequence is first the keyword parameters, and then the positional parameters. If an invalid QN=*nnnnn* is specified, keyword processing is terminated. DTSGETQ then continues to process the positional parameters.

The GETP procedure lets you do most of your compilations in a batch partition, thus balancing the work load within the system and freeing interactive partitions for interactive processing. The punch queue output can then be retrieved for loading and execution within an interactive partition.

The procedure can also be used for online updating of the VSE/ICCF library file. For example, by using a utility like OBJMAINT, you can have SYSIN input punched to VSE/POWER, have it retrieved online by GETP and stored as a library member.

Note:

1. Only the submitter or the user named in the DEST operand of the VSE/POWER statement * $$ PUN may access a punch-output queue entry. An authorized user may specify ANY in the DEST operand (authorization is established in the VSE/ICCF user profile). This makes the punch output accessible to any user.
2. The GETP procedure invokes the DTSGETQ program. This program processes the procedure based on the positional operand '`jobname`', optionally with '`jobnumber`', '`jobsuffix`', and '`jobclass`'. These operands (as far as used) must be specified in the order as shown. The remaining (keyword) operands can be specified in any order.

Examples

1. Place VSE/POWER punch job 'MYJOB' into punch area. The job has no password.

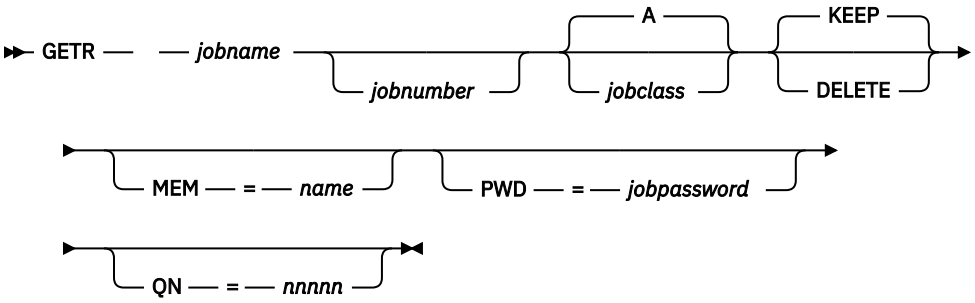
```
getp myjob
```

2. Display VSE/POWER punch job 'MYJOB'.

```
getp myjob mem=$$print
```

GETR Procedure

The GETR (GET reader queue) procedure retrieves a job from the VSE/POWER reader queue and stores it as a member of your library.



jobname
Is the name of the job that you want to retrieve from the VSE/POWER reader queue.

jobnumber
Is the job number assigned to the job in the VSE/POWER reader queue. This number must be specified if the job name is not unique within the queue. If the number is omitted, the first job in the VSE/POWER reader queue with the specified job name will be retrieved.

jobclass
Is the class assigned to the spooled input in the VSE/POWER reader queue. If this class is numeric, you must specify also a job number; else the specification will be mistaken as the job number.

KEEP
DELETE
Specifies the disposition of the VSE/POWER job after retrieval is complete. KEEP, the default, means that the disposition of the job will not be changed.

If DELETE is specified, the job will automatically be deleted from the VSE/POWER reader queue after successful retrieval. Abnormal end causes the input to be kept.

MEM=*name*
Specifies the name of the VSE/ICCF library member into which the VSE/POWER job is to be stored. If no member name is given, the specified job name will be used as member name. If the member may have to be resubmitted for processing under control of VSE/POWER, certain names should not be used. For details, see the restriction "6" on page 331.

PWD=*jobpassword*
Is the password that protects the job. This password may have been assigned to the job and its output when the job was submitted For details see "SUBMIT Procedure" on page 127. it might have been assigned by VSE/POWER JECL. For details about password protection by VSE/POWER , see VSE/POWER Administration and Operation. You can omit this operand if there is no password protection or if you are the VSE/ICCF administrator.

Jobs submitted via the card reader are protected by an unprintable password if you do not provide one. Output from such jobs can be retrieved only by the VSE/ICCF administrator.

QN=*nnnnn*
Specifies the queue entry number. The processing sequence is first the keyword parameters, and then the positional parameters. If an invalid QN=*nnnnn* is specified, keyword processing is terminated. DTSGETQ then continues to process the positional parameters.

A normal terminal user can retrieve only jobs that he originated, whereas the VSE/ICCF administrator may retrieve any job.

The retrieved job will not include any VSE/POWER JECL statements (such as * $$ JOB or * $$ EOJ). Similarly, in place of an * $$ RDR statement, the job will include the diskette data that is addressed by that statement.

Note: Jobs submitted to another node in a VSE/POWER controlled network cannot be retrieved.

Examples

1. Store the job MYJOB, jobnumber 123, in the library as member MYJOB. Keep the job after retrieval.

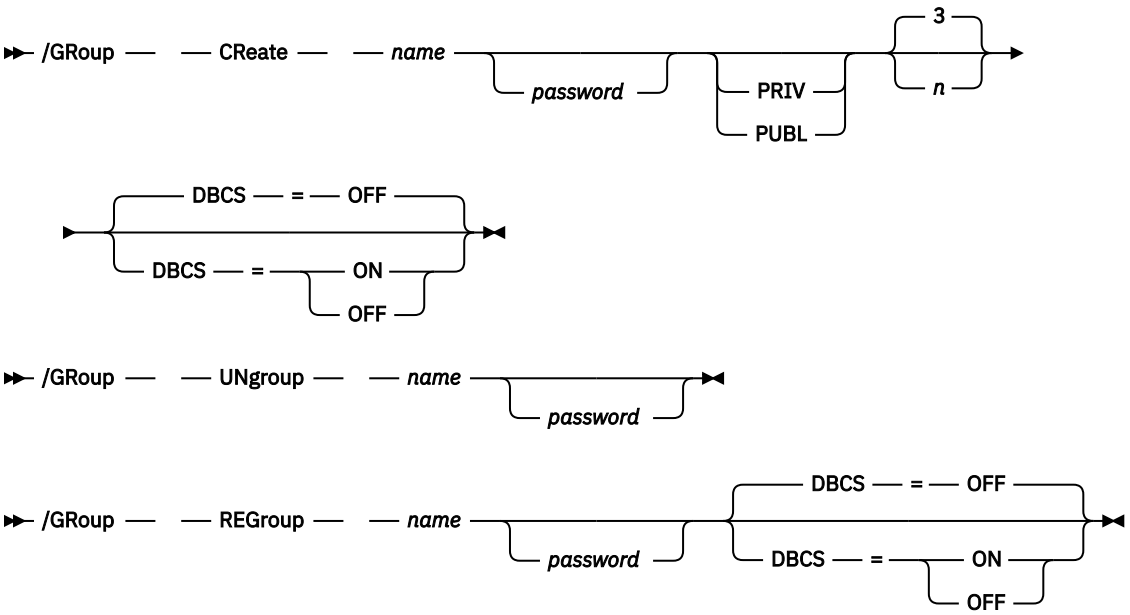
```
getr myjob 123
```

2. Place the job MYJOB, jobnumber 10557, password SECRET, into the library as member MYMEM. Delete MYJOB from the VSE/POWER reader queue after successful transfer.

```
getr myjob 10557, mem=mymem pwd=secret delete
```

/GROUP Command

The /GROUP command creates a generation member group. It can be used also to remove the group characteristic from the members of a group and to reform the members back into a group. The command is effective in command mode.



CReate
Allows you to establish a generation member group within your library.

UNgroup
Removes the generation member group characteristic from the members of the group and makes them ordinary library members. They can thus be handled as such.

REGroup
Forms a generation member group from existing members.

name
Is the unique 1 to 6 character name for a generation member group, beginning with an alphabetic character. If the (most recent) member is to be submitted for processing under control of VSE/POWER, certain names should not be used. For details, see the restriction “6” on page 331.

password
Need be specified only if one or more or all members of the group are to be password-protected.

PRIV
PUBL
Makes the members of the group either public or private. If you omit this operand, the private or public status will be set according to the default in your user profile.

n
Is a decimal number from 2 to 10, indicating the number of entries to be created in the group.

DBCS=ON
DBCS=OFF
Sets or resets the DBCS attribute for all members of a generation group.

The /GROUP CREATE command creates from 2 to 10 entries in the library. Each of these entries has a two-character suffix added to the name; this suffix is in the form -n. Whenever data in the input area is saved (by the /SAVE command) with a member-group name, this data becomes the '-0' version of the member. The previous '-0' version of the member becomes the '-1' version, the -1 version becomes the '-2' version, and so on. The oldest member in the group is purged.

All the members to be grouped with the /GROUP REGROUP command must begin with the 1 to 6 character group name followed by the '-n' suffix. The resulting generation member group will consist of all library members beginning with 'name-0' up to and including 'name-n', the highest suffix in the library. If there is a break in the suffix numbering, only the first continuously suffixed members will be grouped.

Note:

1. If the name of a generation member group ends with '.P', all members in the group are print-type members. It is not possible to have only one print-type member within a generation member group. The period should not be used as a delimiter, backspace character or line end character.
2. If a generation member group is of print type, the name part of the member name cannot exceed 4 characters.
3. The type of a member can be changed also via /RENAME, which sets the type to print or non-print, depending on the presence or absence of the .P suffix.
4. You can change the DBCS attribute for each member of the group separately with the /PROTECT command. However, it is your responsibility to keep the group consistent if you do this.

Examples

1. Create a member group consisting of the entries XXX-0, XXX-1, XXX-2, XXX-3 in your library.

```
*READY
/group create xxx 4
*GROUP CREATED
*READY
```

2. Assume that you have a group named XXX whose latest member is called XXX-0. You want to make some changes to XXX-0 and yet keep that version current. You might do this as follows:

```
*READY
/input
/insert XXX-0
*END INSERT
/end
*READY
/edit
 ...
 ...                          (Editor commands to make changes)
 ...
quit
END EDIT
*READY
/save XXX-0
*GENERATION MEMBER SAVED
*READY
```

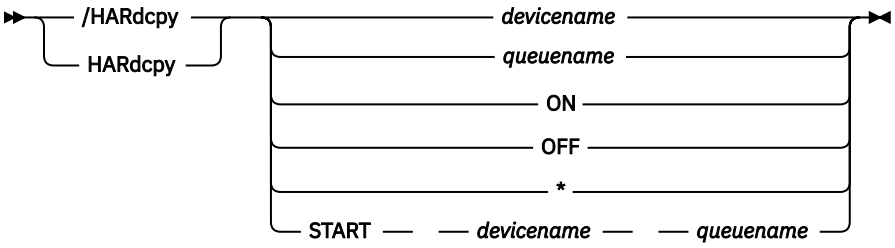
   At the conclusion of this example, the changed version will be called XXX-0 and the previous version will be called XXX-1.

3. The following example shows a group consisting of 4 members (XXX-0 through XXX-3) to which a fifth entry is added.

```
*READY
/group ungroup xxx
*GROUP FUNCTION PERFORMED
/inp
dummy record
/save xxx-4
*SAVED
*READY
/group regroup xxx
*GROUP FUNCTION PERFORMED
*READY
```

[/]HARDCPY Command

The command sets hardcopy mode on or off and directs 3270 terminal output to a hardcopy printer. It directs this output to a private print destination queue for hardcopy output.



devicename
The logical name of the printer on which the hardcopy output is to be produced. Ask your administrator for this name. Use the command with this operand for normal hardcopy output.

queuename
The logical name of a private print destination queue. Ask your administrator for this name.

Use the command with this operand to set hardcopy mode if your administrator has made a private print destination available to you. A private destination reduces the risk of your output being mixed with that of other users.

ON
Sets hardcopy mode on after the printer's logical name has been entered at least once.

OFF
Ends hardcopy mode.

*****
Applies only to the remote 3275 with a printer directly attached. The command with this operand starts non-disk-queued hardcopy operation back to your terminal printer. Each transmission of data normally sent to the display is sent to the attached printer.

START devicename queuename
Directs print output to the specified printer to print the output queued in the specified print destination queue.

This command is effective for IBM 3270 terminals only; it cannot be issued from a procedure. The command with / is effective in all modes except edit mode.

While in hardcopy mode, all output which normally is displayed will be directed to the hardcopy destination until the /HARDCPY OFF command is given. The data being sent to the printer is also displayed as verification that the print operation is in progress.

For lengthy output, enter the /HARDCPY command, wait for the first screen of output to appear, press the CLEAR key and then enter the /CONTINU command. The output will then be displayed continuously, and you will not have to press ENTER to receive each new screen of output.

All data directed to printers is queued on disk, thus allowing you to rapidly dispose of print data. The data is printed when the printer becomes available. Meanwhile you can press the CLEAR key, enter the /HARDCPY OFF command and proceed with other work.

If you are using an IBM 3275, you may nevertheless take advantage of queued-on-disk printing. To do so, specify /HARDCPY followed by some other printer name (not your own printer) or a private destination queue name, if you have one. When you have finished terminal operations, enter the /HARDCPY START command followed by the name of your terminal, and the private destination queue name. Printing will start and continue until the queue is empty.

Note:

1. When a printer-directed data stream appears on a display terminal, you must press CLEAR before you can enter a VSE/ICCF command.
2. Full screen mode cannot be invoked while hardcopy mode is in effect.
3. Print-type members are printed using the full width (up to 132 characters) of the hardcopy printer.

4. When /LISTP is used in hardcopy mode, all forms control characters are changed to 'write with single space' (X'15'), except for CICS Transaction Server controlled 328x terminal printers with the form feed feature. For these, a skip to the next print page is supported.
5. Control characters contained in data sent to the screen are not interpreted in hardcopy mode. Thus, multiple printer lines may occur on one screen line (see also the example below).

Example

Step 1: Connect a hardcopy printer.

```
/har l86p_
 ...+....1....+....2....+....3....+....4....+....5....+ ..CM
*READY
      *HARDCOPY MODE SET5*READY59_
```

- Response is in print format with print control characters (5,9) and without a scale line.
- Press CLEAR before you enter the next command.

Step 2: List the member MYMEM:

```
/list mymem_
 ...+....1....+....2....+....3....+....4....+....5....+ .CM
      LINE 15LINE 25LINE 35LINE 45*END PRINT5*READY59_
```

- Response is a display of the member in print format.
- Press CLEAR before you enter the next command.

Step 3: Disconnect the hardcopy printer:

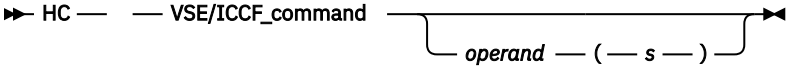
```
/har off_
 ...+....1....+....2....+....3....+....4....+....5....+ .CM
*READY
      _
       ...+....1....+....2....+....3....+....4....+....5....+.. .CM
      *HARDCOPY MODE NOW OFF
      *READY
```

Result (output on the hardcopy printer):

```
    *HARDCOPY MODE SET
    *READY
    LINE 1
    LINE 2
    LINE 3
    LINE 4
    *END PRINT
    *READY
```

HC Macro

The HC macro switches an IBM 3270 to hardcopy mode. It processes the specified command, routes the output to a hardcopy printer, and returns the terminal to normal display mode.



command
Is any VSE/ICCF system command which is to be executed in hardcopy mode.

If you enter a list-type command, use the continuous version of the command to avoid that you have to repeatedly press ENTER or have to enter the /CONTINU command.

operands
Are any operands associated with the specified command.

The macro applies only to an IBM 3270 terminal and cannot be invoked in edit mode. Also, a /HARDCPY or HARDCPY command specifying the printer must already have been issued during the session so that the HC macro knows to which hardcopy printer it should direct the output.

Examples

1. Print a member on the hardcopy printer.

```
hc /listc xfile
```

2. Print a full library display on the hardcopy printer.

```
hc /libc full all
```

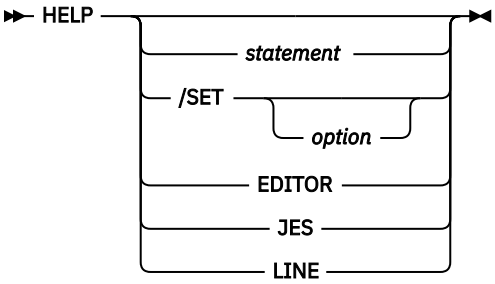
3. Display status information on the hardcopy printer:

```
hc /status
```

HELP Macro

The macro displays summary and detailed information about using VSE/ICCF commands, macros, procedures, and job entry statements.

The help information can be presented in any language, including languages that have a double-byte representation like the Japanese Kanji. The HELP macro can be invoked only in command mode.



statement
Is the name of a command, macro, procedure, or job entry statement.

/SET
/SET option
Requests HELP information on the /SET command. 'option' is any of the first operands of the /SET command.

EDITOR
Requests a display of all editor commands that are explained in the HELP facility.

JES
Requests a display of all job entry statements that are explained in the HELP facility.

LINE
Requests a display of all editor line commands that are explained in the HELP facility.

If the HELP macro is issued without an operand, an overview panel will be displayed. This panel lists all items for which HELP information is available and tells you how to retrieve that information.

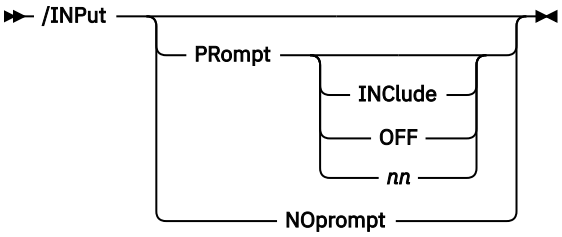
Note: HELP information about rarely used VSE/ICCF functions is not available.

Examples

```
*READY
help
 ...
 ...                          (Help information is being displayed)
 ...
*END PRINT
*READY
```

/INPUT Command

The /INPUT command causes the terminal to leave command mode and enter input mode.



PRompt
If input verification is set off (see the description of the VERIFY function of /SET on page "VERify" on page 106), this operand causes a prompt number to appear on the terminal. The number identifies the next line to be entered.

If you specify no further operands, VSE/ICCF displays the last line entered and shows the line number of the next line to be typed in. The prompt number is not added to the data you enter.

If you are using input verification, a maximum of the last ten lines is displayed and no prompt number will appear on your screen.

If you specify PROMPT with INCLUDE or with 'nn', inclusion prompting will be in effect. This causes the prompt number to become part of the typed line. If you specify INCLUDE, the first prompt will be the line number times 10 as a five-digit number in columns 1 through 5 of each input line. If you set inclusion prompting by specifying the 'nn' operand, VSE/ICCF uses your value for nn as the prompting increment. You need not supply the INCLUDE operand if your user profile indicates automatic inclusion prompting.

NOprompt
Eliminates any type of prompting even if your user profile indicates automatic prompting.

INClude
Causes the prompt number to become part of the line entered.

OFF
Eliminates any type of prompting, even if your user profile indicates automatic prompting.

nn
Is a decimal number from l to 32767 to be used as prompt increment. The default is 10.

The /INPUT command is effective only in command mode. After the command has been accepted, all lines that are not system commands recognizable by VSE/ICCF are placed into your input area. The only limit is the size of your input area as defined in your user profile.

The command places the terminal into system (not editor) input mode. Editor input mode allows any type of data, including VSE/ICCF commands, to be entered into the input area; input mode does not allow VSE/ICCF system commands to be entered into the input area.

In addition to typing in data, you can copy all or part of a library member into the input area by using the /INSERT command.

The /INPUT command does not immediately destroy the previous contents of the input area. However, after the first line of input has been entered, the previous contents of the input area will have been destroyed.

To end the system-input mode, issue the /END command.

Note:

1. To erase the contents of your input area, issue the /INPUT command followed by the /CANCEL command.
2. If you require more space than is in your input area, save the contents of the input area (/SAVE command) and start a new one (/INPUT command). After the new area has been saved, the multiple areas can be logically connected using the /INCLUDE facility.
3. Inclusion prompting (INCLUDE or 'nn' operand) is useful when you enter a program, which requires a line number at the beginning of each statement. This type of prompting saves you from having to type the statement number. (For prompting with sequence numbers in other than columns 1 to 5, see “Line-Number Editing” on page 157.)
4. See the /PROMPT command for an explanation of how to vary prompting characteristics during an input session.
5. When inclusion prompting is set on, the included prompt disregards lines beginning with a slash. Thus, job entry statements can be entered although inclusion prompting is set on.
6. The /INPUT command cannot be used while an asynchronous (see /ASYNCH command) interactive partition execution is in progress because the input area is the job stream being run.

```
*READY
/input
111
222
/end
*READY
/save group1
*SAVED
*READY
/input
333
444
/insert group1
*END INSERT
555
/end
*READY
/list
333
444
111
222
555
*END PRINT
*READY
```

/INSERT Command

The /INSERT command copies all or part of a library member, or the print, punch or log area, into the input area.

If the library member was in compressed format, the member is also decompressed. The command is effective in input mode.

name
Is the name of a library member which is to be copied into the input area. This member must exist in your primary, connected or common library.

'name' can also be $$PUNCH, $$PRINT or $$LOG for the punch, print or log area, respectively. This allows you to save the output of compilers or of utilities. By copying the contents of the punch area with the /INSERT command and then saving the input area, you can save an object program produced by a language compiler. You can also save the contents of the print or log areas for future reference.

If the name refers to a DBCS member, the included double-byte characters are not displayed correctly. However, if you save the new member and set the DBCS attribute for it (by /PROTECT), the double-byte characters will be displayed in their correct form.

The copy operation ends when it reaches the limit of the input area as specified in your user profile. To copy the remaining records, save the contents of the input area and issue another /INPUT command.

password
Need be specified only if the member is password-protected.

m
Is the number of the first line to be inserted. If 'm' is not specified, line 1 is assumed. For compressed members line 1 is assumed, regardless of what has been specified. The maximum value is 99999.

n
Is the last line to be inserted. The maximum value is 99999. If n is not specified, end of file is assumed.

To copy only a portion of a member into the input area, specify a starting and ending line number. Use the /DISPLAY command to find these numbers.

Example

See the example of the "/INPUT Command" on page 65/

/LIBC Command

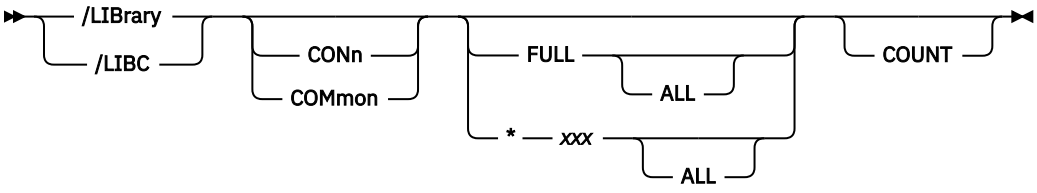
The command displays directory information in a way similar to the [/]LIBRARY command.

However, because it scrolls the directory information continuously to the end, this command is useful only if the display needs more than one screen.

For its operands, see the description of the [/]LIBRARY command.

[/]LIBRARY Command

The [/]LIBRARY command displays directory information for individual libraries to which you have access.



CONn
COMmon
Selects the library for which you want directory information displayed. CONn stands for your connected library; COMmon stands for the common library. If the operand is omitted, the directory information for your primary library is displayed (see Example 1).

FULL
FULL ALL
Displays the entire contents (see Example 2) of directory information. ALL causes directory information for all members to be displayed, not only the members that you own.

***xxx**
***xxx ALL**
Displays the FULL directory information for members that have names starting with 'xxx' (see Example 3). 'xxx' can be a string from 1 to 7 characters. ALL causes directory information to be displayed for all of the members (and not just the ones that you own) whose names start with the specified characters.

ALL is assumed if, in your user profile, OPTB bits 2, 3, or 6 are on; normally, these are VSE/ICCF administrator bits.

COUNT
Displays additionally the number of records per member. Large members can be identified easily. Member size and number of members impact the /LIB COUNT command performance.

When neither FULL nor *xxx are specified, a short form of the directory information is displayed (see Example 1).

The FULL listing is sequenced by location within the directory. The directory is maintained on a last-in/first-out basis, so that the recently added members usually appear at the beginning of the directory.

Note:

1. Use the SDSERV procedure to obtain a sorted directory listing.
2. The /LIBC command does not use the last two lines of your screen. They are reserved for messages.
3. Dates are displayed in the format active when VSE/ICCF was started.

Examples

1. You ask for a short display of the member names in your primary library.

```
/lib_
 ...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...CM
*READY
 ┌──────────────────────────────────────────────────────────────────────────────┐
 │_                                                                             │
 │ ...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...CM
 │LIBRARY NO. - 00005   01/27/1998  17/14/44   DIR ENTRIES: ACT=00009 MAX=00200 │
 │                                                                              │
 │JAC005    ********  DTSANAL1  SCALE     GROUP1    FORTPROG                   │
 │LETTERS   DIGITS    FORTDTA                                                   │
 │*END PRINT                                                                    │
 │*READY                                                                        │
```

   Member names are displayed in directory sequence. An ******** entry represents a directory slot empty either because the member was purged or because free space was requested when the library was restored.

2. Display full information of the members owned by you contained in your connected library.

```
/lib con full_
 ...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...CM
*READY
 ┌──────────────────────────────────────────────────────────────────────────────┐
 │_                                                                             │
 │ ...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...CM
 │LIBRARY NO. - 00013   02/01/1998  13/54/03   DIR ENTRIES: ACT=00012 MAX=00200 │
 │                                                                              │
 │0003 HELLO     09/13/1998 COOL                                                │
 │0006 PAYROLL  P08/14/1998 COOL PRIV CPR                                       │
 │0009 SALES-0   08/14/1998 COOL          GDG                                   │
 │0010 SALES-1   08/14/1998 COOL          GDG FL                                │
 │*END PRINT                                                                    │
 │*READY                                                                        │
```

Explanation of information in the various columns:

1st column: 0003, 0006, ... in the example.
This is the directory-entry sequence number.

2nd column: HELLO, PAYROLL, ... in the example.
This is the member name.

3rd column: P on the 0006 PAYROLL line.
Indicates that the member is password-protected.

4th column: 09/13/1996 on the 0003 HELLO line.
This is the creation or replacement date.

5th column: COOL in the example.
The member owner's user ID ('MSG*' if the member is a message).

6th column: PRIV on the 0006 PAYROLL line, blank on all others.
PRIV indicates that the member is private; blank (no display) means that the member is public.

7th column: CPR on the 0006 PAYROLL line.
The member is stored compressed.

8th column: GDG on the 0009 SALES-0 line.
Indicates that the member is part of a group.

9th column: FL on the 0010 SALES-1 line.
The member has been flagged as changed by the editor.

3. Display full information of the members you own, that are contained in your primary library, and whose names start with the letter F.

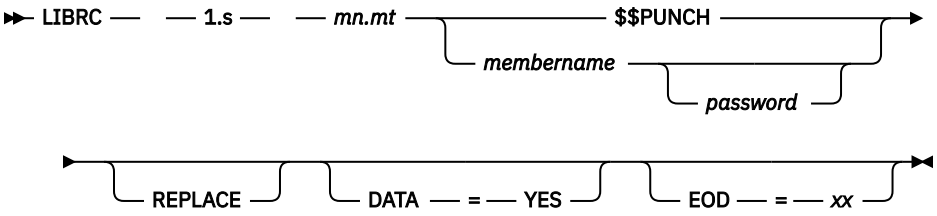
```
/lib *f_
 ...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...CM
*READY
```

```
_
 ...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...CM
LIBRARY NO. - 00005   01/27/1998  17/14/44   DIR ENTRIES: ACT=00009 MAX=00200

0006 FORTPROG  02/14/1998 COOL PRIV
0009 FORTDTA   02/14/1998 COOL PRIV
*END PRINT
*READY
```

LIBRC Macro

The LIBRC macro is used to catalog a VSE/ICCF library member or the punch area into a VSE sublibrary.

The macro requires an interactive partition of a size of 384K bytes. It requires the partition to have a size of 512K bytes if the accessed VSE library resides in VSAM-managed space.



l
Is the VSE library name

s
Is the VSE sublibrary name

mn
Is the VSE member name

mt
Is the VSE member type

$$PUNCH
Denotes the punch area whose contents is to be cataloged into the specified VSE sublibrary.

membername
Is the name of the member in the VSE/ICCF library file.

password
Is the password of the VSE/ICCF library member if the member is password-protected.

REPLACE
Indicates that, if a member with the specified name already exists in the VSE sublibrary, the member should be replaced by the new data.

DATA=YES
Applies only to cataloging a VSE procedure. Indicates that the procedure includes SYSIPT data.

EOD=xx
Specifies two end-of-data characters. If the operand is omitted, the end-of-data delimiter is assumed to be '/+'. Defining a delimiter other than '/+' is useful, for example, if you catalog a VSE procedure which contains '/+' as data in columns 1 and 2. Do not use the character '¬' (the not sign).

If you invoke the macro with less than two operands, the macro displays the complete invocation syntax.

Example

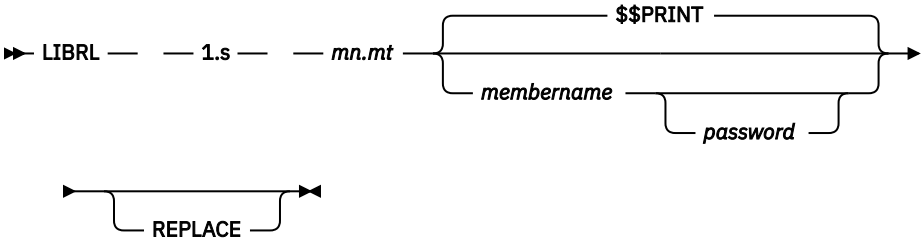
Catalog the VSE/ICCF library member DHTA1 into the VSE sublibrary PRVICCF.CICS with the member specification NAME1.TYPE1. A member with this name and type already exists in the target library.

```
librc prviccf.cics name1.type1 dhta1 replace_
 ...+....1....+....2....+....3....+....4....+....5....+....6....+. .CM
*READY
_
 ...+....1....+....2....+....3....+....4....+....5....+....6....+. .CM
*RUN REQUEST SCHEDULED FOR CLASS=A
* * * * * START OF PROCEDURE (CLIST) * * * * *
  ACCESS S=PRVICCF.CICS
L113I RETURN CODE OF ACCESS IS  0
  CATALOG NAME1.TYPE1          EOD=/+  REP=Y
L113I RETURN CODE OF CATALOG IS  0
*PARTIAL END PRINT
**** JOB TERMINATED - ICCF RC 00, EXEC RC 0000, NORMAL EOJ
*READY
```

LIBRL Macro

The LIBRL macro displays a member of a VSE sublibrary, or stores it in print-type format as a member in your VSE/ICCF library or in the print area.

The macro requires an interactive partition of a size of 384K bytes. It requires the partition to have a size of 512K bytes if the accessed VSE library resides in VSAM-managed space.



l
Is the VSE library name

s
Is the VSE sublibrary name

mn
Is the VSE member name

mt
Is the VSE member type

$$PRINT
Denotes the print area, where member mn.mt is to be stored in print-type format.

membername
Is the name of the VSE/ICCF library member into which member mn.mt is to be stored. Because the member is stored in print-type format, the two rightmost characters should be '.P' to declare the member as a print-type member.

If you do not supply a member name, $$PRINT is taken as the default.

password
Is the password of the VSE/ICCF library member if the member is password-protected.

REPLACE
Indicates that if a member with the specified name already exists in the VSE/ICCF library file, the member should be replaced by the new data.

If you invoke the macro with less than two operands, the macro displays the complete invocation syntax.

Example

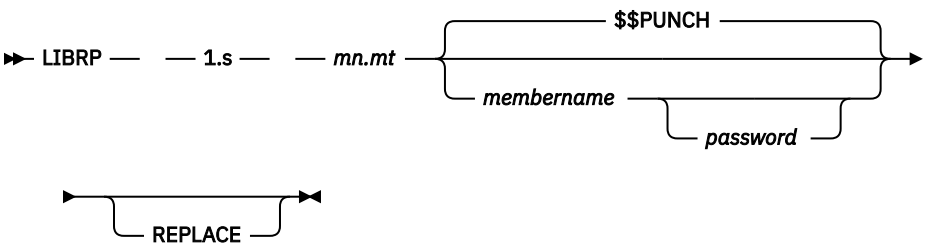
Display the member NAME1.TYPE1 of the VSE sublibrary PRVICCF.CICS.

```
librl prviccf.cics name1.type1 $$print
 ...+....1....+....2....+....3....+....4....+....5....+....6....+. .CM
*READY
_
 ...+....1....+....2....+....3....+....4....+....5....+....6....+. .CM
*RUN REQUEST SCHEDULED FOR CLASS=A
  ACCESS S=PRVICCF.CICS
L113I RETURN CODE OF ACCESS IS  0
  LIST NAME1.TYPE1
MEMBER=NAME1.TYPE1     SUBLIBRARY=PRVICCF.CICS    DATE: 84-01-17
                                                  TIME: 14:10
------------------------------------------------------------------
   .
   .      (member data)
   .
L113I RETURN CODE OF LIST IS 0
*PARTIAL END PRINT
DATA STORED INTO ICCF-MEMBER OR $$PRINT AREA
*READY
```

LIBRP Macro

The LIBRP macro punches a member from a VSE sublibrary, or stores it as a member in your VSE/ICCF library or in the punch area.

The macro requires an interactive partition of a size of 384K bytes. It requires the partition to have a size of 512K bytes if the accessed VSE library resides in VSAM-managed space.



l
Is the VSE library name

s
Is the VSE sublibrary name

mn
Is the VSE member name

mt
Is the VSE member type

$$PUNCH
Denotes the punch area, where member mn.mt is to be stored.

membername
Is the name of the VSE/ICCF library member into which member mn.mt is to be stored. If you do not supply a member name, $$PUNCH is taken as the default.

password
Is the password of the VSE/ICCF library member if the member is password-protected.

REPLACE
Indicates that if a member with the specified name already exists in the VSE/ICCF library file, the member should be replaced by the new data.

If you invoke the macro with less than two operands, the macro displays the complete invocation syntax.

Example

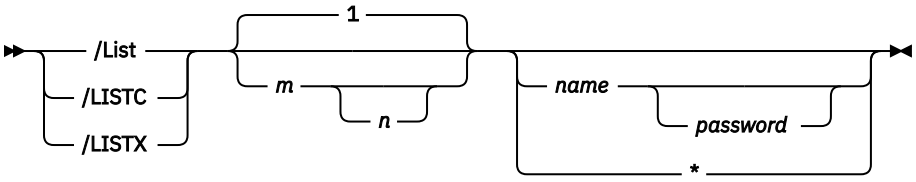
Have the member NAME1.TYPE1 of the VSE sublibrary PRVICCF.CICS punched out and stored as the VSE/ICCF library member DATA1.

```
librp prviccf.cics name1.type1 data1 replace_
 ...+....1....+....2....+....3....+....4....+....5....+....6....+. .CM
*READY
        _
         ...+....1....+....2....+....3....+....4....+....5....+....6....+. .CM
        *RUN REQUEST SCHEDULED FOR CLASS=A
          ACCESS S=PRVICCF.CICS
        L113I RETURN CODE OF ACCESS IS  0
          PUNCH NAME1.TYPE1
        L113I RETURN CODE OF PUNCH IS 0
        *PARTIAL END PRINT
        DATA STORED IN MEMBER DATA1
        *READY
```

/LIST Command

The /LIST command displays the contents of the input, punch, print or log area, or of a library member. This command, and also the /LISTC and /LISTX commands are valid in command and input mode.

/LISTC displays data in continuous output mode (as if the /CONTINU command had been entered). /LISTX displays data in combined (character and hexadecimal) format.



m
Is the number (any from 1 to 99999) of the first line to be displayed.

n
Is the number (any from 1 to 99999) of the last line to be displayed. If n is omitted, the display continues through end of file. If the specified line number (n) exceeds the number of lines in the file, the display proceeds to the end of the member.

To display a portion of the file, specify the number of both the first line and the last.

*****
Displays the last ten lines of the input area from your current position.

name
Is the name of the library member to be displayed. If no name is specified, the input area is displayed. For name, you can specify $$PUNCH, $$PRINT or $$LOG, if the contents of these areas are to be displayed. If name refers to a DBCS member or to the print area, the data to be displayed may be mixed data.

password
Need be specified only if the member is password-protected.

The library member need not be decompressed to be displayed.

A user of an IBM 3270 can press the Display key [3] to display the last ten lines in the input area.

If the listing to be displayed needs more than one screen, use the /LISTC command. With /HARDCPY you can transfer the output to a hardcopy printer. To leave list mode before the end of the display, enter the /CANCEL command. If you use an IBM 3270, press the Cancel key [4].

For a print-type member, the first two bytes of each record are interpreted as print control characters, and the member is displayed in print line format. The default PF key settings (as described in Figure 1 on page 28) are not active; you can use your own PF key settings, instead.

Note:

1. The /DISPLAY and /LIST commands are the same, except that the /DISPLAY command also prints line numbers and is valid only for non-compressed members.
2. Unless the default line size is more than 72 (see /SET command), sequence numbers in columns 73 to 80 are not displayed, whereas other data is.
3. When the contents of your log area are displayed (/LIST $$LOG), the entries appear in reverse order. That is, the most recent entries appear first.
4. The /LIST command does not use the last two lines of your screen. They are reserved for messages.
5. Print-type members are displayed on the IBM 3278 Model 5 wide screen using up to 132 display positions per line. Such members are printed on a hardcopy printer using the total line size.

   If a record of a print-type member does not contain valid print control characters in the first two columns, this record is printed as an 80-byte record.
6. If LISTX is specified for a print-type member, the member will be displayed in 80-character record format.
7. The second ENTER after having issued the /SKIP BOTTOM or /SKIP END command ends /LIST command output.

Example

```
*READY
/input
first line
second line
/list
FIRST LINE
```

[3] Defining the Display/Attention key is a VSE/ICCF tailoring option. The default is PA3. It may be different for your system.
[4] Defining the Cancel key is a VSE/ICCF tailoring option. The default is PA2. It may be different for your system.

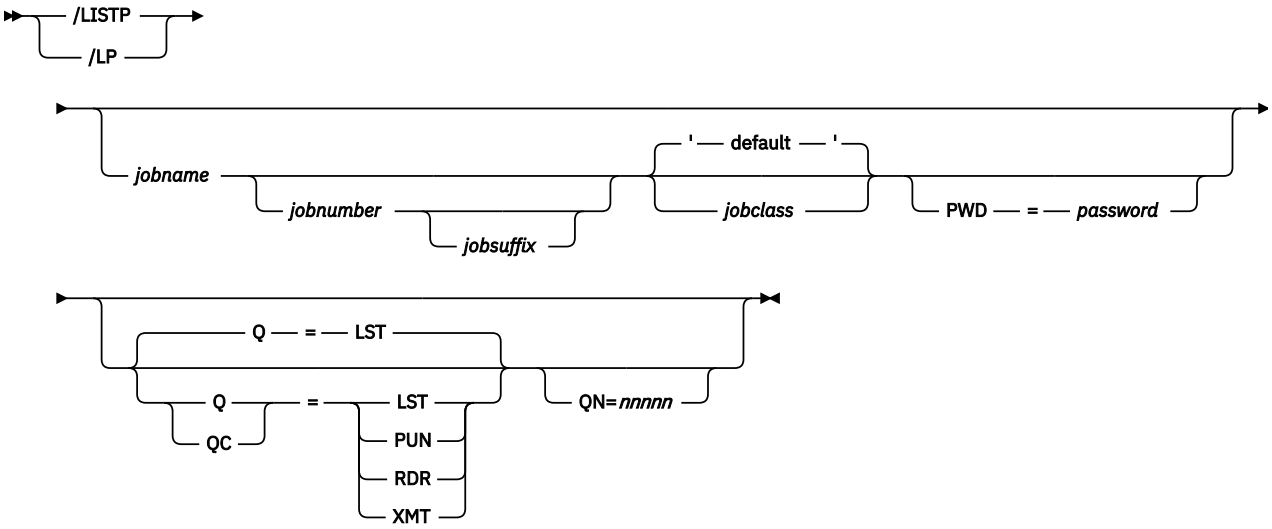
```
SECOND LINE
*END PRINT
third line
/end
*READY
/save sampl1
*SAVED
*READY
/l sampl1
FIRST LINE
SECOND LINE
THIRD LINE
*END PRINT
*READY
```

/LISTC Command

For details see the "/LIST Command" on page 72.

/LISTP Command

The command displays print output from the VSE/POWER list queue. It can be used only in command mode.



jobname
Is the name of a job previously submitted and which now has output in the VSE/POWER list queue. If this operand is omitted, jobname is assumed to be the eight-character name formed by the user ID and the terminal identification.

jobnumber
Is the number assigned to the VSE/POWER list output. Specify the number if the job name is not unique within the VSE/POWER list queue.

jobclass
Is the output class that was specified for the output of the named job. This operand need be specified only if your list output does not have the default output class (normally class Q; this may be different at your locality).

jobsuffix
Is the job suffix that designates the segment number. If this operand is omitted, segment number 1 is assumed. If the jobclass is numeric, the jobnumber **and the jobsuffix** must be specified. 0 must be specified for jobsuffix if the job does not have a jobsuffix.

PWD=*password*
Specifies the password assigned to the VSE/POWER list output.

Q= or QC=
Specifies one of the following queues:

LST
list queue (default)

PUN
punch queue

RDR
reader queue

XMT
transmission queue

Note: For queue entries in-creation, **QC=** must be specified. There is no queue type XMT for in-creation queue entries.

QN=*nnnnn*
Specifies the queue entry number. It is mandatory for browsing the XMT queue and for accessing in-creation queue entries.

The data to be displayed can be alphanumeric or mixed. However, data that has already been prepared for printing on a 3200 printer – for example with the utility PRPQ 5799 BGF – is not displayed correctly any more.

If you request the display of a job just by its name and this job name occurs more than once in the queue, the job occurring first will be displayed. Therefore, specify the VSE/POWER assigned job number in addition to the name of the job.

This command is normally used to display the print output from a job previously submitted to VSE/POWER to be run in another VSE partition. If the SUBMIT procedure was used, the job name and also the job number assigned to it by VSE/POWER are returned to you.

The execution of the job must have been completed before you can display the print output. Wait for VSE/POWER to send you the completion message or check the status of the job by issuing the /STATUSP or the /DQ command.

While displaying the print output, your terminal is in list mode until end of output is reached. If the display needs more than one screen, scroll through the file by pressing ENTER, or by using the /SKIP command. Use the /SHIFT command to 'scroll' left and right. With /LOCP you can locate a character string in the print output. With /CONTINU you can start or stop continuous display, and with /HARDCPY you can transfer the output to a hardcopy printer.

You can leave list mode at any time with the /CANCEL command or, if you use an IBM 3270, by pressing the Cancel key [5].

The display is in print line format; that is, part of the print line may be invisible (contrary to 'wrap-around' or 80-character format where two lines may be displayed for a print line of more than 80 characters). The /SHIFT command allows you to move the screen window over to the portion that is invisible. The default PF key settings (see Figure 1 on page 28) are not active; use your own PF key settings, instead. If you want to use those default PF key settings, you must first issue the command /SHIFT OFF.

You can decide yourself how to dispose of the output after display. For example, you could erase it using the /ERASEP command. Or, you could use the /ROUTEP command to route it to the local printer or to a VSE/POWER RJE printer. The /ROUTEP command changes the disposition of the file to the default (normally D for dispatchable).

Note:

1. If /LISTP is used in hardcopy mode, forms control characters are changed to 'WRITE WITH SINGLE SPACE' (X'15'), except for CICS Transaction Server controlled 328x terminal printers with the form feed feature. For these, a skip to the next print page is supported.

[5] The definition of the Cancel key is a VSE/ICCF tailoring option. The default is PA2; this may be different for your system.

2. If your specifications for jobname, jobnumber, jobsuffix, and jobclass do not identify a unique print output, the first occurrence of a partial match is taken.
3. Only the submitter or the user designated in the DEST operand of the VSE/POWER JECL statement * $$ LST may access the list output queue entry. An authorized user may specify ANY in the DEST operand (authorization is established in the VSE/ICCF user profile). This makes the list output accessible to any user.
4. The second ENTER after having issued the /SKIP BOTTOM or END command ends /LISTP output and returns the terminal to command mode.

Example

1. List the VSE/POWER output from 'MYJOB', which has the job number 0200 and the password 'SECRET' in the installation default class:

```
/listp myjob 0200 pwd=secret
```

2. Same as above but with the output class 'A':

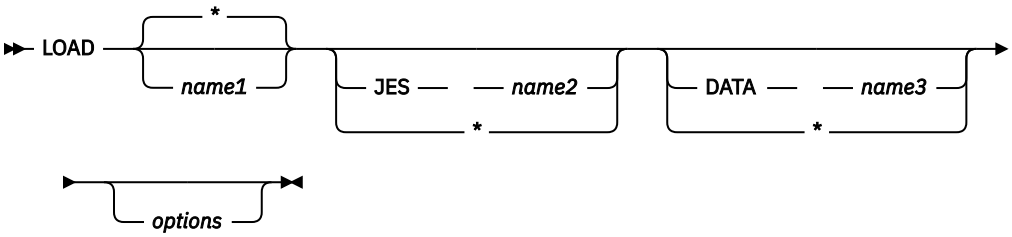
```
/listp myjob 0200 a pwd=secret
```

LISTX Command

For details see the “/LIST Command” on page 72.

LOAD Procedure

The LOAD procedure causes an object program to be loaded into storage and run.



name1
Is the name of the library member containing the object deck. If this operand is '*', the object module will be read from the punch area.

JES name2
Is the name of a member containing the file definitions and other job entry statements needed to run the program. '*' indicates that you are to be prompted for the job entry statements. To end prompting press ENTER key.

DATA name3
Is the name of the member containing the job stream data for the execution. If '*' is specified, the data will be read conversationally from the terminal as if /DATA INCON had been specified. To end data reading type '/*'.

options
Are one or more VSE/ICCF /OPTION statement options to be applied to the execution.

The object program can be in the library (name1 operand) or in the punch area ('*'). If you have pre-established file definitions and job stream data as library members, you can include these as the 'name2' and 'name3' operands. If one or both of these is specified as *, VSE/ICCF prompts you for the job entry statements or data at the proper point in time.

Examples

1. Load from the punch area and read data from the terminal:

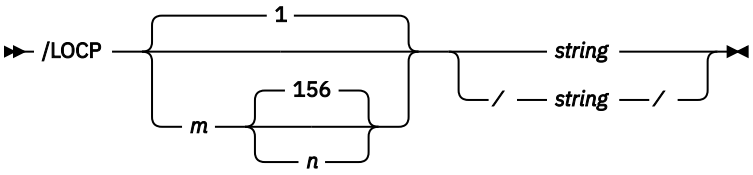
```
load * data * getvis=48K
```

2. Load from a library member with JES from another member:

```
load myjob jes fildef
```

/LOCP Command

The LOCP command locates a character string in VSE/POWER lists. It can be given in list mode after the /LISTP command.



m
Is a number between 1 and 156 specifying the column where the search is to start.

n
Is a number between 1 and 156 specifying the column where the search is to end. The string must be entirely contained in the zone from 'm' to 'n' if the locate is to be successful. If you specify, for example:

```
/locp 1 100 AAA
```

and, in a given record, the string AAA occurs in columns 99 through 101, the string is not located. If 'n' is not specified and the print line is shorter than 156 characters, the end of the print line is taken as the default.

string
Specifies the character string to be located.

/string/
Specifies the character string, including imbedded blanks, that is to be located. / is the string-delimiter character.

The search for a match of the given string starts from the line following the current position. If found, the line with the matching string is displayed at the top of the screen, below the scale line. The line is shifted so that the located data can be seen on the screen. If no line with a matching string is found, VSE/ICCF repositions processing to the point where the command was entered and displays the message:

```
*CHARACTER STRING NOT FOUND, PRESS ENTER TO RESUME
```

Note:

1. To avoid overloading your library with large VSE/POWER files, first check them, using the /LISTP and /LOCP commands.
2. VSE/ICCF supplies the date in the format active when VSE/ICCF was started.
3. If a /COMPRESS is in effect for the current display, /LOCP locates the string at the "real" position, not at the position as displayed on the screen. Therefore, a /LOCP after a /COMPRESS may not be practical in all cases.

Examples

1. Scan forward from the current line + 1 for the character string **ERROR. The search should be limited to columns 61 up through the end of the line.

```
/locp 61 **ERROR
```

2. Scan forward from current line + 1 for character string /NUMBER OF STATEMENTS/

```
/locp /NUMBER OF STATEMENTS/
```

/LOGOFF Command

The command ends a session and causes all accounting statistics to be updated.

The /LOGOFF command is effective in input and command mode.

If you have more than one user ID (for example, to access more than one library or set of defaults) you can log on with the other user ID after having logged off.

The system can log you off in certain situations, for example when you have not used your terminal beyond the time-out limit set for you (see the “[/]SHOW Command” on page 118).

Note:

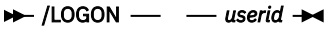
1. VSE/ICCF supplies the date in the format active when VSE/ICCF was started.
2. Make sure to log off before you leave your terminal. Leaving the terminal logged on and unattended may make sensitive data accessible to unauthorized persons. Moreover, you might get charged for terminal time that you did not actually use.

Example

```
*READY
/logoff
*DATE=09/02/1998 TIME=01/35/30
*YOU ARE NOW LOGGED OFF
*GOOD-BYE FOR NOW
```

/LOGON Command

The command starts a terminal session.



userid
Is the four-character user identification.

The command takes effect when 'iccf' (the VSE/ICCF terminal control transaction) has been entered. It is not effective in any other VSE/ICCF modes (input, command mode, etc.).

Example

```
iccf    (enter the VSE/ICCF transaction identification)
*VSE/INTERACTIVE COMPUTING AND CONTROL FACILITY
*PLEASE ENTER: /LOGON WITH YOUR USERID
/logon usrc
*ENTER PASSWORD
@@@@@@
*LOGON COMPLETE - DATE 09/02/1998 TIME 01:30
*READY
 ...
```

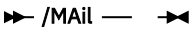
```
 ...  (user's session)
 ...
/logoff
*DATE=09/02/1998 TIME=01/50/00
*YOU ARE NOW LOGGED OFF
*GOOD-BYE FOR NOW
```

/LP Command

For details see the "/LISTP Command" on page 74.

/MAIL Command

The command displays messages from either the system operator or the VSE/ICCF administrator.



This command is effective only in command mode.

Messages can be directed to all users (general mail) or to a specific user (specific mail). Make it a habit to display your mail as soon as you have logged on.

The messages can contain double-byte characters, if the DBCS attribute has been set for the A$MAIL member.

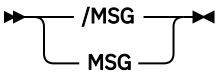
Example

```
*READY
/mail
* * * DISPLAY OF MAIL FILE * * *
*
A NEW PROCEDURE HAS BEEN ADDED TO THE SYSTEM
THIS PROCEDURE (C$SORT) PROVIDES A GENERALIZED
SORT FACILITY.
SPECIFICATIONS FOR USE ARE - - -
 ...
 ...                                (And so on)
 ...
USRA ALL STUDENTS IN INTRO. TO COBOL (CC5738-02) SHOULD
USRA DELETE SAVED MEMBERS FROM THE LIBRARY.
*END PRINT
*READY
```

[/]MSG Command

The command displays alphanumeric messages that arrived for you while your terminal is set to non-automatic message display

For details see "MSGauto" on page 105.



/MSG (with '/') is valid in command, list, and execution modes. MSG (without '/') is valid in edit mode.

Messages can be sent by all VSE/ICCF users, the system operator, and VSE/POWER.

Example

The example shows a job completion message issued by VSE/POWER when the execution of a job submitted by you is finished.

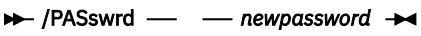
```
/msg_
 ...+....1....+....2....+....3....+....4....+....5....+ .CM
*READY
_
 ...+....1....+....2....+....3....+....4....+....5....+.... ..CM
 03/15 - 13:45 MSG FROM VSE/POWER
1Q5DI EXECUTION COMPLETED FOR JOBX 00135 ON NODE1, TIME=13:44:36
*END MSG
*READY
```

MVLIB Procedure

For details see “CPYLIB and MVLIB Procedures” on page 43.

/PASSWRD Command

The /PASSWRD command is used to specify a new logon password.



newpassword
Is a three- to six-character word which is to be the new logon password.

You can change your password if your user profile allows it.

This command is effective only in command mode. Access to VSE/ICCF must have been obtained via transaction *ICCF* (as opposed to selecting *Command Mode* in the VSE/ESA Interactive Interface).

Note:

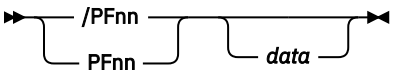
1. Write down your new logon password on a piece of paper that is not accessible to anyone else. There is no handy way of obtaining your password if you forget it.
2. The password specified as the operand of this command can be specified on the hardcopy printer (IBM 2740 and 3767) terminals. For security reasons, you may want to tear off this portion of the printout and destroy it.

Example

```
*READY
/passwrd jacpas
*PASSWORD CHANGED
*READY
```

[/] PFnn Command

The /PFnn command invokes a function associated with a PF key. It is valid in any mode.



nn
Is the number of the program function invoked (a decimal number from 1 to 24).

data
Is a character or variable to be appended to the program function setting.

This facility allows terminals which do not have PF keys to nevertheless allow program key functions. You can assign functions to certain PF keys by way of the /SET PF command.

The /PFnn command can also be useful on terminals that do have PF keys. An example is when you are entering multiple lines separated with the logical line end character; you can imbed a PF key function

within your string of commands. Thus, you do not have to press the Program Function key itself, which avoids a terminal interrupt.

Note: PF keys cannot be used in procedures.

If the program function has been set to include a parameter (see /SET PF command), the operand portion of the /PFnn command replaces this parameter. If the program function has not been set with a parameter, any operand specified with the /PFnn command is added to the end of the program function before it is performed.

Figure 2 on page 81 shows which of the defined function you get when you enter the /PFnn command. The paragraph below discusses this in more detail. You may have defined up to four different functions for one PF key:

For command mode (PFnn)
For edit mode (PFnnED)
For execution and conversational read mode (PFnnEX)
For list and spool mode (PFnnLS).

In addition, a default function is available for each PF key.

You get the function which is defined for the mode of your terminal. If the terminal is in list mode (LS), for example, /PFnn would get you the function assigned to PFnnLS. However if PFnnLS were not defined or if all settings belonging to list mode were suspended (by /SET PFLS OFF), you would get the function of the next deeper level in the hierarchy which is PFnn defined for command mode. If this is still not defined you get the default setting.

/PFnn in	Invokes the Function		
CM mode		PFnn ===>	Default PF-key setting *
ED,FS mode	PFnnED ===>	PFnn ===>	Default PF-key setting *
EX,RD mode	PFnnEX ===>	PFnn ===>	Default PF-key setting *
LS,SP mode	PFnnLS ===>	PFnn ===>	Default PF-key setting *

* For the default PF-key setting, see Figure 1 on page 28.

Figure 2. Program Function Hierarchy

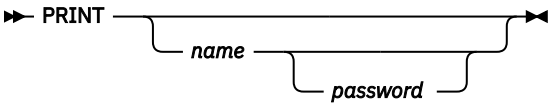
Example

```
*READY
/set pf1 /list 1 18 $$LOG
*PROGRAM FUNCTION SET
*READY
/pf1
 ...                          (Display of the first 18
 ...                          lines of your log area)
 ...
*END PRINT
*READY
/set pf2 /statusp             (A blank character must follow
*PROGRAM FUNCTION SET         /statusp)
*READY
/pf2 joba
*STATUS = LL - JOB COMPLETED
*READY
```

PRINT Macro

The macro routes the contents of a library member, or the input area, to a hardcopy printer associated with your terminal.

This macro applies only if you use an IBM 3270 terminal.



name
Is the name of the library member to be printed. If no name is given, the input area will be printed.

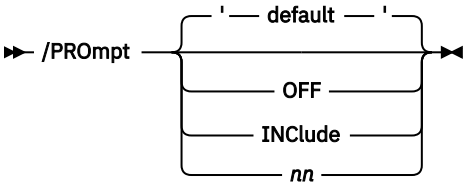
password
Is the password associated with the member, if any.

A /HARDCPY command that specifies the name of the hardcopy printer to which the data is to be sent must already have been issued during the session.

Print-type members will be printed in print format using up to 132 characters of the hardcopy printer.

/PROMPT Command

The command turns line-number or inclusion prompting on or off. It is valid in input mode only.



OFF
Prompting is turned off.

INClude
Inclusion prompting is set on.

nn
Is the decimal increment (1 to 32767) for inclusion prompting, which is also set on.

Line-number prompting, with input verification set off (see /SET VERIFY on page “VERify” on page 106), means that you are prompted for each input line with the number of the next line to be entered. The prompt number is not included in the input data. If you are using input verification, a maximum of the last ten lines is displayed and no prompt number will appear on your screen.

Inclusion prompting means that you are prompted with a five-digit number that is included in columns 1 to 5 of your input data. The prompt increment is increased by 10 for each new line, unless you vary it by specifying a value for nn.

Line-number prompting is turned on, if:

1. No operand is specified.
2. No prompting is in effect.
3. Input verification is set off.

If no operand is specified and line-number or inclusion prompting is in effect, the prompting status is not changed, but the current prompt increment will be displayed.

The /PROMPT command displays the current line number, if this has been lost while printing (/LIST or /DISPLAY) in input mode. It can also be used to change the default in your user profile.

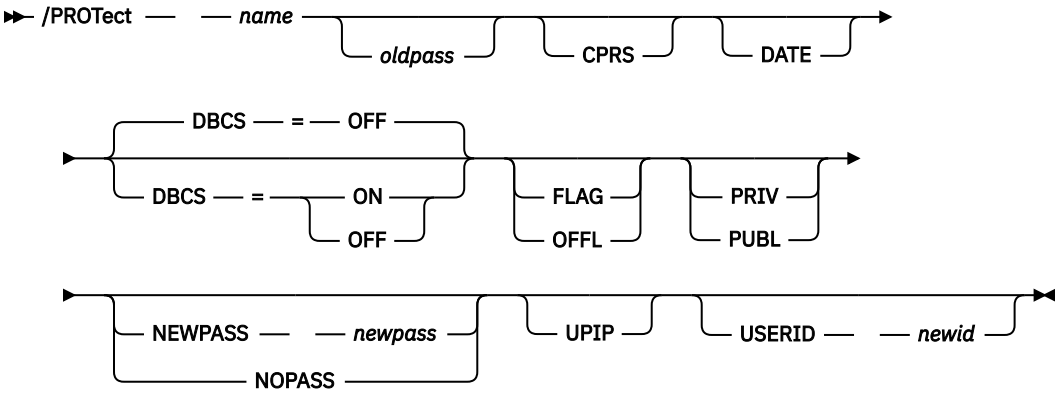
```
*READY
/set verify off
/input
/prompt
0001 first input line
0002 second input line
/prompt off
third input line
/end
```

```
*READY
/list
FIRST INPUT LINE
SECOND INPUT LINE
THIRD INPUT LINE
*END PRINT
*READY
/input
line 1
/PROMPT INC
00020 line 2
00030 line 3
/end
*READY
/list
LINE 1
00020 LINE 2
00030 LINE 3
*END PRINT
*READY
```

/PROTECT Command

The /PROTECT command mainly allows you to set certain data-protection and status options for a library member.

This command is effective only in command mode.



name
Is the name of the library member whose protection characteristics are to be altered.

oldpass
Is the existing four-character password of a password-protected member; it has to be entered if the existing member is password-protected.

CPRS
Sets the compressed member attribute on.

Specify this operand when a compressed member is being stored in the VSE/ICCF library file without VSE/ICCF knowing that the member is compressed: for example, when members are punched online from a DTSUTIL-archive or backup tape. In such a case, issue the /PROTECT command with the CPRS operand after having deleted any noncompressed records from the member (control cards may have been punched into the member automatically, for example).

DATE
Indicates that the current date is to replace the entry date in the directory record for the member. This can be useful if, at your location, old library members are purged periodically based on the date in their directory records.

DBCS=ON
DBCS=OFF
Sets or resets the DBCS attribute for the specified member. By default, a newly created member does not have the DBCS attribute.

FLAG
OFFL
Sets automatic editor change flagging on or off for the member. The OFFL operand can be used only by the VSE/ICCF administrator.

Automatic editor-change flagging causes the editor to place information in each changed record, usually in columns 73 to 80. This information indicates the type and the date of the change and also the user ID of the user making the change.

Do not use editor change flagging for RPG source members.

NEWPASS newpass
NOPASS
For newpass, specify the four-character password to be applied to the named library member. Specifying NOPASS removes password protection from the member.

PRIV
PUBL
Makes the member either PRIVATE or PUBLIC.

UPIP
Sets the update-in-progress attribute off (can be used only by the VSE/ICCF administrator).

This operand is useful for a member that may have been left in an update-in-progress state because of a system failure.

USERID newid
Is the four-character user ID to be applied to the member.

Note: After having changed the user ID for a member, you may no longer be able to access this member.

Examples

To add password protection to a member:

```
*READY
/protect mycode newpass conf
*MEMBER CHANGED
*READY
```

To change the password protection of a member:

```
*READY
/protect mycode conf newpass secr
*MEMBER CHANGED
*READY
```

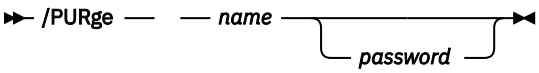
To remove the password protection from a member:

```
*READY
/prot mycode secr nopass
*MEMBER CHANGED
*READY
```

/PURGE Command

The /PURGE command removes a member from a library.

It is only effective in command mode.



name
Is the name of the member to be purged (permanently removed) from the VSE/ICCF library file.

password
Is the password associated with the member that is to be purged. The password must be specified if the member is password-protected.

If a member has been saved as private, only the user who entered the member may purge it.

A public member can be purged by any user who shares the library, unless the alternate security option is in effect.

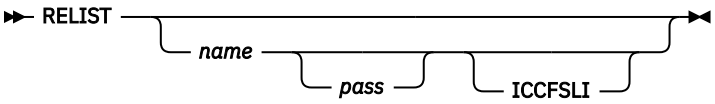
Common data library members cannot be purged by a terminal user. They can be purged only by the VSE/ICCF administrator.

Example

```
*READY
/list comment
THIS IS THE MEMBER NAMED COMMENT
*END PRINT
*READY
/purge comment
*PURGED
*READY
/list comment
*FILE NOT IN LIB
*READY
```

RELIST Macro

The RELIST macro routes the contents of the print area, of a print-type member, or of a normal library member to the printer.



name
Is the name of a library member containing the data to be printed. The member can contain both VSE/POWER JECL and VSE JCL statements. If this operand is omitted, the contents of the print area will be printed. Certain names should be avoided; refer to "Submit-to-Batch Capability" on page 326.

The member to be routed may contain data other than print data. Such a member will always be displayed in 80-character (80-80) record format.

The data to be listed may not contain the commercial-at sign (@) in the 1st column.

pass
Is a four-character password which need only be specified if the library member is password-protected.

ICCFSLI
Applies only to a library member. If ICCFSLI is specified, VSE/ICCF generates an * $$ SLI statement when the VSE/POWER print job is being built. The data will not be written to the VSE/POWER reader queue. Instead, at execution time, VSE/POWER itself will read the data from the VSE/ICCF library file. If ICCFSLI is not specified, the library member will be written into the VSE/POWER reader queue when the print job is being built.

When the display of the output of an interactive partition job is complete, you can also print this output. Use the RELIST macro to route the contents of the print area to your central printer. You can also first save the contents of the print area as a print-type library member and route it for printing later on. Specify the member name as the operand of the RELIST macro.

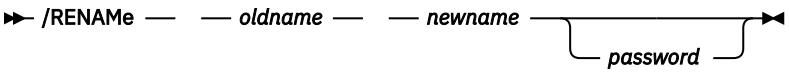
Note:

1. To set the size of the print area, see the "The BUFFER Feature" on page 106.

2. If the submission of the relist job is not successful, the first two lines of the print area will be overlaid.
3. For a print-type member (the member name ends with .P) the first two characters of each 80-character record are interpreted as control characters, and the member will be printed in print line format.

/RENAME Command

The /RENAME command changes the name of a library member.



oldname
Is the current name of the member.

newname
Is the new name that you want to apply to the member.

password
Is the password of the member to be renamed. The password must be specified if the member is password-protected. Although the name of the member is changed, the password remains the same.

This command is effective only in command mode; it can be used to rename members in libraries to which you have access. It cannot be used to change the names of common members and members of the common library, or of members of a generation member group.

Example

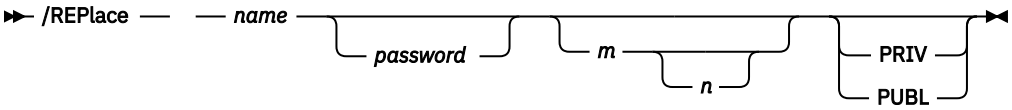
```
*READY
/input
 ...+....1....+....2
/end
*READY
/save scale publ
*SAVED
*READY
/rename scale s
*RENAMED
*READY
/list s
 ...+....1....+....2
*END PRINT
*READY
```

/RENUM Command

For details see the “/RESEQ Command” on page 87.

[/]REPLACE Command

The [/]REPLACE command replaces a library member with all or part of the input area.



name
Is the name of the library member whose contents are to be replaced by the contents of the input area. The characters '.P' at the end of the member name indicate a print-type member.

If the specified member name does not exist within the library, VSE/ICCF issues an error message. You should then enter the /SAVE command.

password
Is the password of the member to be replaced. The password must be specified if the member is password-protected.

m
Is a number from 1 to 9999 representing the first line number in the input area to replace the library member.

n
Is a number from 1 to 9999 representing the last line number in the input area to replace the library member. If 'n' exceeds the number of lines in the input area, the replace operation proceeds through the end of the input area.

PRIV
PUBL
PRIV indicates that you want the contents of the member to be replaced and the new member to be given the PRIVATE attribute. PUBL indicates that you want the contents of the member to be replaced and the new member to be given the PUBLIC attribute.

If neither PRIV nor PUBL is specified, the previous privacy attribute of the member is retained.

With the slash (/), the command is effective in command or input mode; without the slash, it is valid only when you are editing in the input area.

To replace the existing library member with only a portion of the input area, specify a beginning and, optionally, an ending line number (operands m and n). The records not participating in the replace will remain in the input area while the replacing records are deleted from that area.

If only one number is specified, it is assumed to be m. In this case, all lines from line m through the end of the file are affected.

If the '-0' (current) member of a generation member group is being replaced, this current -0 member becomes the '-1' member, the '-1' becomes the '-2', and so on.

When the command is issued in input or edit mode, VSE/ICCF performs the replace function and then places your terminal in command mode.

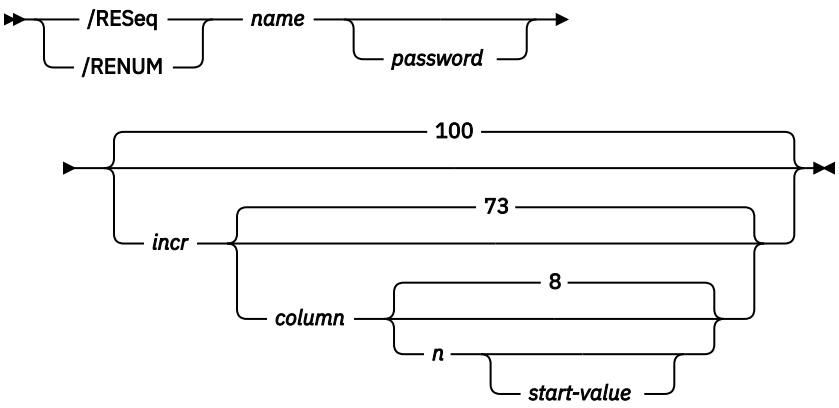
Example

```
*READY
/input
09/02/1996 900902 90002 SEPT. 02, 1990
/end
*READY
/save daterec publ
*SAVED
*READY
/input
09/02/1996 900902 90002 SEPT. 02, 1990
/end
*READY
/replace daterec
*REPLACED
*READY
```

/RESEQ Command

The /RESEQ (or /RENUM) command resequences, or initially applies sequence numbers to, the card-image records of a member in the library.

The command is valid in command mode only.



name
Is the name of the library member that is to be resequenced.

password
Is the password of the member to be resequenced. The password must be specified if the member is password-protected.

incr
Is the resequencing increment. The maximum increment is 9999.

column
Is the first column where sequence numbers are to be inserted. The maximum value that you can specify is 80-n + 1. For DBCS members, only 1 and 73 are valid specifications.

n
Is the number of columns for the sequence number field. The maximum length of the field is 16 columns. If you resequence a DBCS member and if column has a value of 73, n must be 8.

If you specify n, you must also specify the operands incr and column.

start-value
Is the sequence number to start with. If this operand is omitted, VSE/ICCF will start sequencing with the specified (or default) increment.

If you specify start-value, you must also specify the preceding operands incr, column, and n. The maximum start-value is 9999. Refer to the last of the **Notes** below which shows a method of establishing a starting sequence number larger than 9999.

If you provide no sequence specification, sequence numbers are placed in columns 73-80. The increment value will be 100 and the initial sequence number will be equal to the increment.

Note:

1. The /RESEQ command causes new sequence numbers to be inserted into each of the 80-byte records of the named member. It makes no difference whether the member is print-type or not. That is, print data is treated the same way as 80-character records.
2. A statement will not be resequenced if the sequence number begins at a column below 73 and either:

   The first column contains a slash (/), or
   The first five columns contain the string '* $$ '.
3. To establish a starting sequence number that is higher than 9999, you should issue the /RESEQ (/RENUM) twice in succession. This is demonstrated in the following example:

   Assume you want to have a starting number of 860000. First enter the command

   ```
   /RESEQ memname 1 73 4 8600
   ```

   The resulting sequence numbers are:

```
                                                    8600
                                                    8601
                                                    ...
                                                    ...
```

Now enter the command

```
/RESEQ memname 1 75 4 0001
```

The resulting sequence numbers are:

```
                                                    860001
                                                    860002
                                                    ...
                                                    ...
```

Examples

1. Resequence a password-protected member in the standard sequence location:

```
    *READY
    /reseq utilprg jaca
    *RESEQUENCED
    *READY
```

2. Resequence a COBOL program in the standard COBOL sequence location (colums 1-6):

```
    *READY
    /res cobprog 100 1 6
    *RESEQUENCED
    *READY
```

/RETRIEV Command

The /RETRIEV command places a previously entered command in the terminal input area.

There, the retrieved command may be modified and then reissued by pressing ENTER.

OFF
ends the retrieve function. Previously entered commands can no longer be retrieved.

The /RETRIEV command is valid in the following modes:

CM
Command mode

EX
Execution mode

LS
List mode

RD
Conversational-read mode

SP
Spool-data display mode

It is recommended to define a PF key for the /RETRIEV command as shown by the example at the end of this discussion.

Before any command can be retrieved, the retrieve function must be set on once after logon (and also after you issued /RETRIEV OFF). You do this by entering /RETRIEV without an operand. With this first /RETRIEV you do not yet retrieve a command.

When the retrieve function is on, the /RETRIEV command retrieves the command that was entered last. If you issue /RETRIEV again without entering any other command in between, you get the command that was entered second to the last, and so on.

The number of commands that can be accumulated depends on the length of the commands. For an average command size of 20 characters, 12 commands can be accumulated. The maximum size for a command to be retrieved is 255 characters. If the internal stack is full, each new command replaces one or more of the oldest commands in the stack.

Only commands entered via the ENTER key can be retrieved. Multiline input is split up into single commands. Therefore, each command of multiline input must be retrieved separately.

Example

```
/set pf12 /ret   (Defines PF key 12 as /RETRIEV)
|PF12|           (Press PF key 12 to set on the retrieve function)
/lib
submit ...
/list ...
|PF12|
/LIST            (Command retrieved at the terminal)
|PF12|
SUBMIT ...       (Command retrieved at the terminal and modified)
|ENTER|
|PF12|
SUBMIT ...       (Command retrieved at the terminal)
|PF12|
/LIST            (Command retrieved at the terminal)
```

/RETURN Command

The /RETURN command is only valid if you entered VSE/ICCF through the Interactive Interface of z/VSE. The command brings you back into that interface.

The command is effective in command mode only.

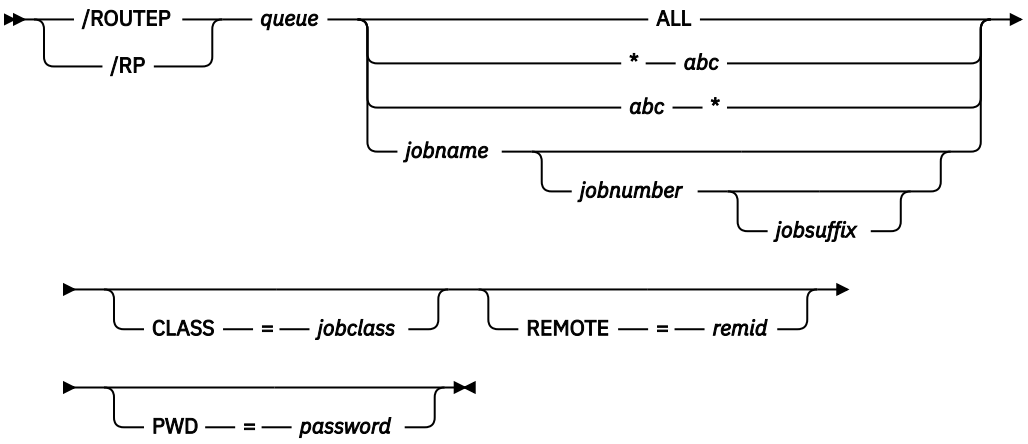
Typically, you would leave the interactive interface of z/VSE to enter a VSE/ICCF command (macro, procedure) that is not supported in that interface. The /RETURN command provides a fast return to z/VSE.

If you changed the environment as used by the Interactive Interface of z/VSE (for example, you changed the PF key settings), return to that interface via the /LOGOFF command. By doing this, you maintain your system's integrity.

/ROUTEP Command

The /ROUTEP (or /RP) command routes a VSE/POWER queue element to either the installation printer or punch, or to a remote VSE/POWER RJE workstation.

It is valid in command mode only.



queue
Specifies the VSE/POWER queue from which the job is to be routed:

LST for list queue
PUN for punch queue

ALL
Causes all output jobs of the specified queue to be routed.

***abc**
abc*
Causes all jobs whose names begin with the characters that you specify to be routed. For 'abc' you can specify up to eight alphameric characters, including the characters dollar ($), commercial at (@), hyphen (-), period (.), and slash (/).

jobname
Causes a specific output job of the specified queue to be routed. For 'jobname', give the name (2 to 8 characters long) of the output job.

jobnumber
Specifies the job number assigned to the job by VSE/POWER. Specify this number if the job name is not unique within the specified queue.

jobsuffix
Is the job suffix which designates the segment number. If this operand is omitted, the entire output of the named job will be routed. Do not specify a suffix if the output is not segmented.

CLASS=jobclass
Is the class to be assigned to the output of the named job. Your administrator normally instructs you which class to use. For 'jobclass', VSE/ICCF accepts any letter from A to Z. Normally, A is the default, but a different default class may have been set up at your locality.

REMOTE=remid
Is the number of the VSE/POWER RJE workstation to which the output of the named job is to be routed. If you omit this operand, the output is routed to the system printer.

PWD=password
Specifies the password that was assigned to the VSE/POWER job when it was submitted.

The command causes the following action:

1. It assigns the specified class to the output of the named VSE/POWER job.
2. It assigns a disposition of D (delete) to the output.
3. It routes the output to the specified RJE workstation.

If a local or remote writer task is active for the target output class, the output will be printed or punched and removed from the VSE/POWER queue. Mixed data should be routed to a 3200 printer only after that data has been prepared for printing (for example, with Utility PRPQ 5799-BGF).

Note:

1. The operands of the /ROUTEP command are identical to those of the VSE/POWER PALTER command, except for *jobname* which must be at least 2 characters long. Contrary to the PALTER command, the /ROUTEP command does not allow *class* as positional operand.

   Only a subset of all possible operands is shown here. For a description of the remaining operands of the PALTER command, refer to VSE/ Power Administration and Operation.
2. The displayed output associated with this command can also include the VSE/POWER messages. These messages are identified by the characters '1Q', '1R' and '1V' followed by a message number. For an explanation of these messages, refer to z/VSE Messages and Codes 1.

Example

Route the print output of 'MYJOB' from class Q to (default) class A [6] to get it printed.

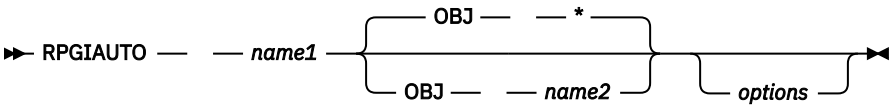
```
    /routep lst myjob
    *OK
    *READY
```

/RP Command

For details see the "/ROUTEP Command" on page 90.

RPGIAUTO Procedure

The procedure causes an RPG II source program to be preprocessed by the AUTO REPORT feature and then to be compiled by the RPG II compiler.



name1
Is the name of a member in the library containing the RPG II source program to be compiled.

OBJ name2
OBJ *
For name2, specify the name by which the object module resulting from the compilation is to be stored in your library. This name can be up to eight characters long, and its first character must be alphabetic. If the object module is to be submitted for processing under control of VSE/POWER, certain names should not be used. For details, see the restriction "6" on page 331.

The named member need not initially exist. If it exists already, the generated object module will overlay that member.

Specify OBJ * or omit the operand if the object module is to be placed into the punch area.

options
Are one or more /OPTION job entry statement options that alter the way the compiler processes its input and handles its output.

You can specify the operand as one or more options such as LIST or RESET. If, for example, the NODECK option is specified, no object module is produced, even if the OBJ operand is present. Specify the NORESET option to have multiple object decks stacked in the punch area.

The LOAD procedure can be used to load and run the object module produced by this procedure.

[6] A different default may have been defined for your location.

Examples

1. Compile the RPG II source program contained in the library member 'AUTOPGM' and place the resulting object module in the punch area:

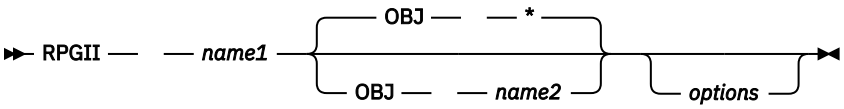
```
rpgiauto autopgm
```

2. Compile the RPG II source program contained in library member 'PROGR', place the resulting object module in library member 'OBJMEMB' and list the input source program on SYSLST:

```
rpgiauto progr obj objmemb list
```

RPGII Procedure

The RPG II procedure causes a library member to be processed by the RPG II compiler.



name1
Is the name of a member in the library containing the RPG II language source program that is to be compiled.

OBJ name2
OBJ *
For name2, specify the name by which the object module resulting from the compilation is to be stored in your library. This name can be up to eight characters long, and its first character must be alphabetic. If the object module is to be submitted for processing under control of VSE/POWER, certain names should not be used. For details, see the restriction “6” on page 331.

The named member need not initially exist. If it exists already, the generated object module will overlay that member.

Specify OBJ * or omit the operand if the object module is to be placed into the punch area.

options
Are one or more /OPTION job entry statement options that alter the way the compiler processes its input and handles its output. The operand can be specified as one or more options such as LIST or XREF. If, for example, the NODECK option is specified, no object module is produced, even if the OBJ operand is present. Specify the NORESET option if multiple object decks are to be stacked in the punch area.

The LOAD procedure can be used to load and run the object module produced by this procedure.

Examples

1. To compile source code stored as member RPGTEST and have the object module written into the punch area:

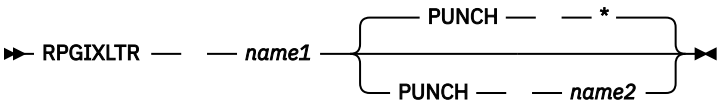
```
rpgii rpgtest
```

2. To compile source code stored as member XXPROG, have the object module stored as member OBJMEM and request a compiler listing on the printer assigned to SYSLST:

```
rpgii xxprog obj objmem list
```

RPGIXLTR Procedure

The RPGIXLTR procedure prepares RPG II source program that will call the DL/I Translator for later processing by the RPG II compiler.



name1
Is the name of a member in the library containing the RPG II source program to be translated.

PUNCH name2
PUNCH *
For name2, specify the library member name into which the translated source module is to be placed. This name can be one to eight characters long, and its first (or only) character must be alphabetic.

The named library member need not initially exist. If it exists already, the source module will overlay that member.

Specify PUNCH * or omit the operand if the source module resulting from the DL/I translation is to be placed into your PUNCH area.

If your source program calls the DL/I Translator, it must first be processed by the translation procedure before it can be processed by the RPGII (or the RPGIAUTO) procedure.

Examples

1. Translate the RPG II source program contained in library member 'RPGDLI' and place the translated source module into the punch area:

```
rpgixltr rpgdli
```

2. Translate the RPG II source program contained in library member 'PROGR' and place the translated source module into the library by the name 'DECK':

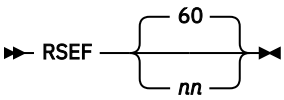
```
rpgixltr progr punch deck
```

3. Translate the RPG II source program contained in library member 'PROGR' and place the translated source module into your PUNCH area:

```
rpgixltr progr punch *
```

RSEF Procedure

The RSEF procedure calls the RPG II source entry facility.



nn
Is the GETVIS space in multiples of 1K bytes needed for this facility to become active.

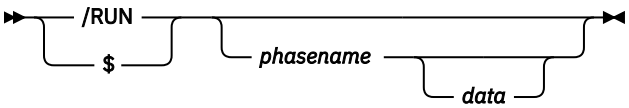
After having entered RSEF, you are prompted for the name of the file and for the password.

The *Programmer's Guide* for RPG II describes how to use this source entry facility.

/RUN or $ Command

The command causes the specified phase to be loaded from a VSE library and run.

With no operand, it causes the job or job stream in the input area to be run. It is only effective in command mode.



phasename
Is the one- to eight-character name of a phase to be loaded from a VSE library and run.

data
Is from 1 to 72 columns of data to be passed to the phase being loaded. It is the first and only 80-character input record that is read from the job stream. Column 1 of the data operand begins one space after the name of the phase.

If the '$' form of the command is used, no space is needed between the '$' and the phase name. Thus, $XYZ will cause the phase named XYZ to be loaded and run.

The command causes a job stream to be built in the input area and run. Thus, any prior contents of the input area will be destroyed. For example, the commands:

```
/RUN DTSUTIL DISPLAY USERS
```

or

```
$DTSUTIL DISPLAY USERS
```

are functionally equivalent to the following set of VSE/ICCF commands:

```
/INPUT
/LOAD DTSUTIL
 DISPLAY USERS
/ENDRUN
```

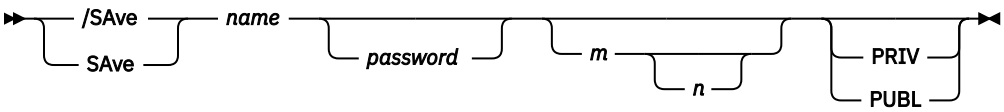
Once the /LOAD and 80-character input record have been placed into the input area, the job can be rerun simply by entering /RUN or $ with no operands; in this form the command will no longer destroy the contents of the input area.

Examples

```
*READY
$SSERV   DSPLY A.SORSBK
*RUN REQUEST SCHEDULED
 ...
 ...                 (Output from SSERV)
 ...
*READY
/input
/load vfortran
;write (3,10)        (The ; is the logical tab character)
10;format (' this means the program ran')
;end
/end
*READY
/run
*RUN REQUEST SCHEDULED
 ...
 ...                 (Compiler output)
 ...
TOTAL MEMORY REQUIREMENTS 000154 BYTES
HIGHEST SEVERITY LEVEL OF ERRORS FOR THIS MODULE WAS 0
****** BEGIN LINKNGO
  12,235 BYTES REQUIRED FOR PROGRAM STORAGE
  PGM LOADED AT 09B100
  XFER ADDRESS 09B100
******
THIS MEANS THE PROGRAM RAN
**** JOB TERMINATED - RETURN CODE 00 NORMAL EOJ
*READY
/run                 (Issuing /RUN again causes the job to be rerun)
```

[/]SAVE Command

The command saves all or part of the contents of the input area as a member in your primary library.



name
Is the one- to eight-character name to be applied to the member once it is saved in the library. This name must begin with an alphabetic character. The characters '.P' at the end of the name indicate a print-type member.

If the object module is to be submitted for processing under control of VSE/POWER, certain names should not be used. For details, see the restriction “6” on page 331.

If the specified name exists already in your library, then VSE/ICCF issues an error message, and your input area will not be saved. To replace a library member with the contents of the input area, use the [/]REPLACE command.

If the specified name refers to the -0 (current) member of a generation member group, this -0 member becomes the -1 member, a -1 member becomes a -2 member, and so on. The oldest member of the group is purged.

password
Is a four-character password that you can specify if you want the library member to be password-protected. The password you specify cannot be PRIV or PUBL.

m
Is a number from 1 to 9999; it represents the first line number in the input area to be saved. If 'm' is not specified, the first line saved is line 1. If only one number is specified, it is assumed to be 'm'.

n
Is a number from 1 to 9999; it represents the last line number in the input area to be saved. If 'n' is not specified, or if the number specified exceeds the number of lines in the input area, the last line saved is the last line of the input area.

PRIV
PUBL
PRIV indicates that the member is to be PRIVATE; PUBL indicates that the member is to be PUBLIC. If you specify neither PRIV nor PUBL, the data is saved as defined in your user profile.

The /SAVE command (with /) is effective in command or input modes. Without the slash (/), SAVE is valid only if you are editing in the input area. The command has no meaning for a library member, since the member is already in the library and commands processed against it have already taken effect.

All input lines not saved remain in the input area, and the lines saved are no longer in the input area when the SAVE operation is complete.

If the /SAVE command (with /) is used in input mode, the data is saved and command mode is entered. When the SAVE command (without /) is entered, the editor is terminated and command mode is entered.

If your library or directory is full when the command is entered, VSE/ICCF issues an error message. Use the /PURGE command to remove unnecessary library members thus freeing library and directory space. Then reenter the [/]SAVE command.

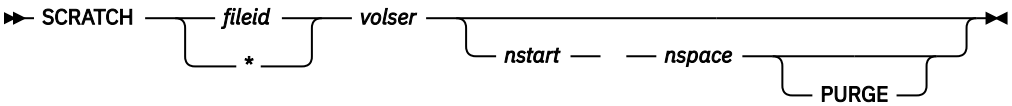
Examples

```
*READY
/input
/load vfortran
      write (3,10)
10    format (' this means the program worked')
      end
/save testprog
```

```
*SAVED
*READY
/exec testprog
*RUN REQUEST SCHEDULED
 ...
 ...                 (Compiler and LINKNGO output)
 ...
THIS MEANS THE PROGRAM WORKED
**** JOB TERMINATED - RETURN CODE 00 NORMAL EOJ
*READY
   /input
/insert $$PUNCH      (Place the resulting object deck
                     into your input area)
*END INSERT
/end
*READY
/save testobj
*SAVED
*READY
/exec testobj        (Rerun the program from the object deck)
*RUN REQUEST SCHEDULED
 ...
 ...                 (LINKNGO output)
 ...
THIS MEANS THE PROGRAM WORKED
**** JOB TERMINATED - RETURN CODE 00 NORMAL EOJ
*READY
```

SCRATCH Procedure

The procedure removes DISP=KEEP files (specified in the /FILE statement) from the dynamic space area.



fileid
*****

For fileid, specify the file identification as it appears in the disk VTOC of the file to be scratched.

If no file identifier was specified in the /FILE statement for file creation, VSE/ICCF supplies a name in the form

```
filename.userid.termid
```

where:

filename:
The name given in the NAMe operand of the /FILE statement.

userid:
The identifier by which the owner of the file is known to VSE/ICCF.

termid:
The identifier (as defined to CICS Transaction Server) of the terminal used when the file was created.

Specify '*' if you include the PURGE option.

volser
Is the volume serial number of the volume on which the file resides.

nstart
Is the starting track/block number of the file; it need be specified only if DASD file protection is active (DASDFP=YES in the IPL command SYS) or if PURGE is specified.

nspace
Is the number of tracks (for CKD) or the number of blocks (for FBA) in the file; it need be specified only if 'nstart' is specified.

The command can be used by the VSE/ICCF administrator to scratch areas of the dynamic space even when no files are associated with the area (PURGE option). This may become necessary if a permanent area was allocated but the job was canceled before the file could be opened.

Examples

1. To scratch the file MYFILE on volume ICFVOL:

```
scratch myfile icfvol
```

2. To scratch the file PAYFILE on a system with DASD file protection:

```
scratch payfile 111111 36 12
```

3. To scratch an area:

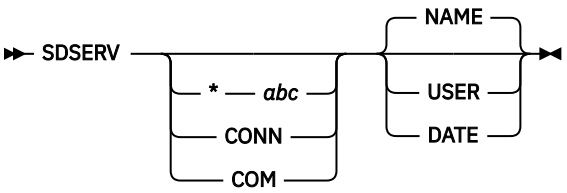
```
scratch * 111111 240 36 purge
```

4. If no file identifier was specified (for fileident) in the /FILE for file creation:

```
scratch hisfile.a001.rj01
```

SDSERV Procedure

The SDSERV procedure displays sorted directory information from your primary or connected library, or from the common library.



***abc**
CONN
COM
*abc causes directory information to be displayed for members whose names start with abc, where 'abc' can be a string from 1 to 7 characters.

CONN causes directory information to be displayed from your connected library.

COM causes directory information to be displayed from the common library.

NAME
USER
DATE
NAME orders the display by member names.

USER orders the display by user identifiers.

DATE orders the display by date of entry or of change.

If no operands are specified, you get a display of your primary library directory sorted by member name.

Examples

1. To get a display of the primary library directory, sorted by member names:

```
sdserv
```

2. To get a display of the directory of your connected library, sorted by dates:

```
sdserv conn date
```

3. To get a display of the primary library directory, sorted by dates:

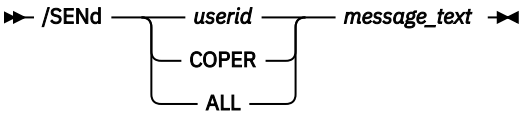
```
sdserv date
```

4. To get a display of the primary library directory, sorted by member names beginning with R710:

```
sdserv *r710
```

/SEND Command

The /SEND command sends a message to the system operator or to another terminal user.



userid
Specifies the terminal user who is to receive the message.

If this user is not logged on, the message is saved and will be displayed automatically when this user logs on next.

If the user is logged on, the message is displayed automatically, provided automatic message display is in effect. If automatic message display is not in effect, the message is saved and will be displayed when this user logs on next. The user can request a display of the message with the [/]MSG command.

COPER
Specifies that the message is to be sent to the system operator.

ALL
Specifies that the message is to be sent to every logged-on terminal user (VSE/ICCF administrator only).

message text
Is the message to be sent. It can contain only alphanumeric characters. The maximum length of the message is, in number of characters:

246
If the ID of a terminal user is specified

247
If the 'ALL' operand is used

80
If the message is directed to the system operator.

The /SEND command is valid in command, list, and execution mode.

Send messages to the system operator only if absolutely necessary. They could "lock up" VSE/ICCF if they are not handled promptly.

The /SEND command is not effective if used within a procedure.

Examples

```
*READY
/send coper this is a message to the system operator
*MESSAGE SENT
*READY
```

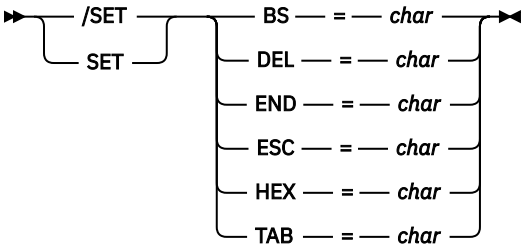
[/]SET Command

The [/]SET (or [/]CTL) command has various formats, which can be used to set control values.

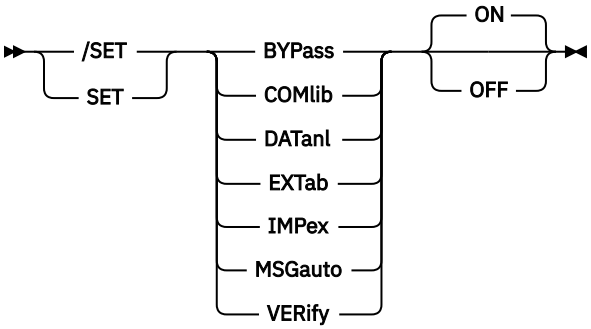
The command is valid in all modes except in message mode. Its formats are shown below for [/] SET; they apply also to [/]CTL. The command has no effect if issued from a program running in an interactive partition, or from a procedure.

The formats of the command are:

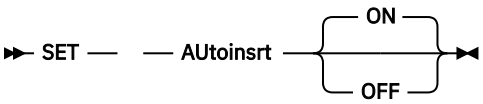
To Set Control Characters



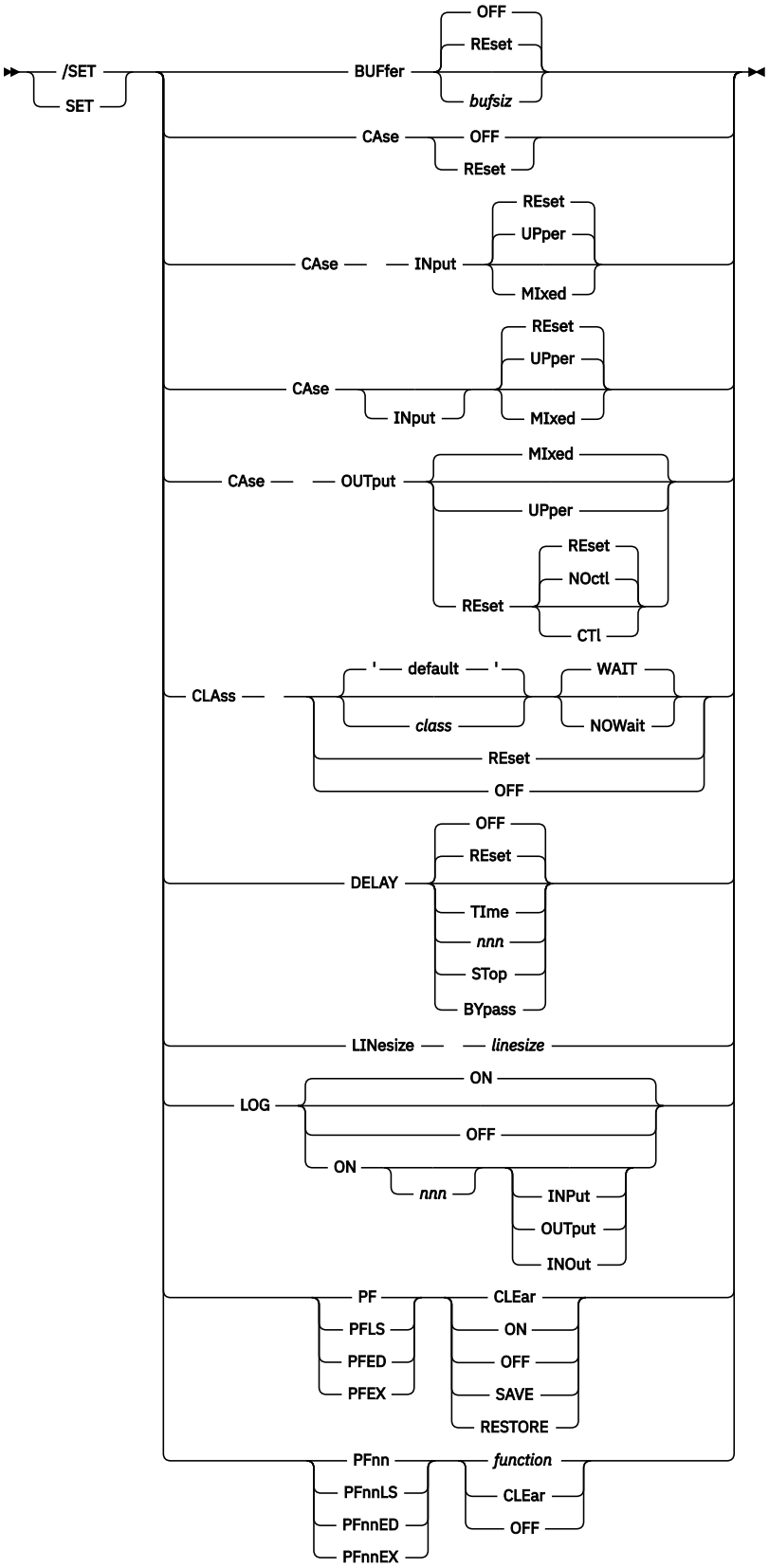
To Set VSE/ICCF Features On or Off



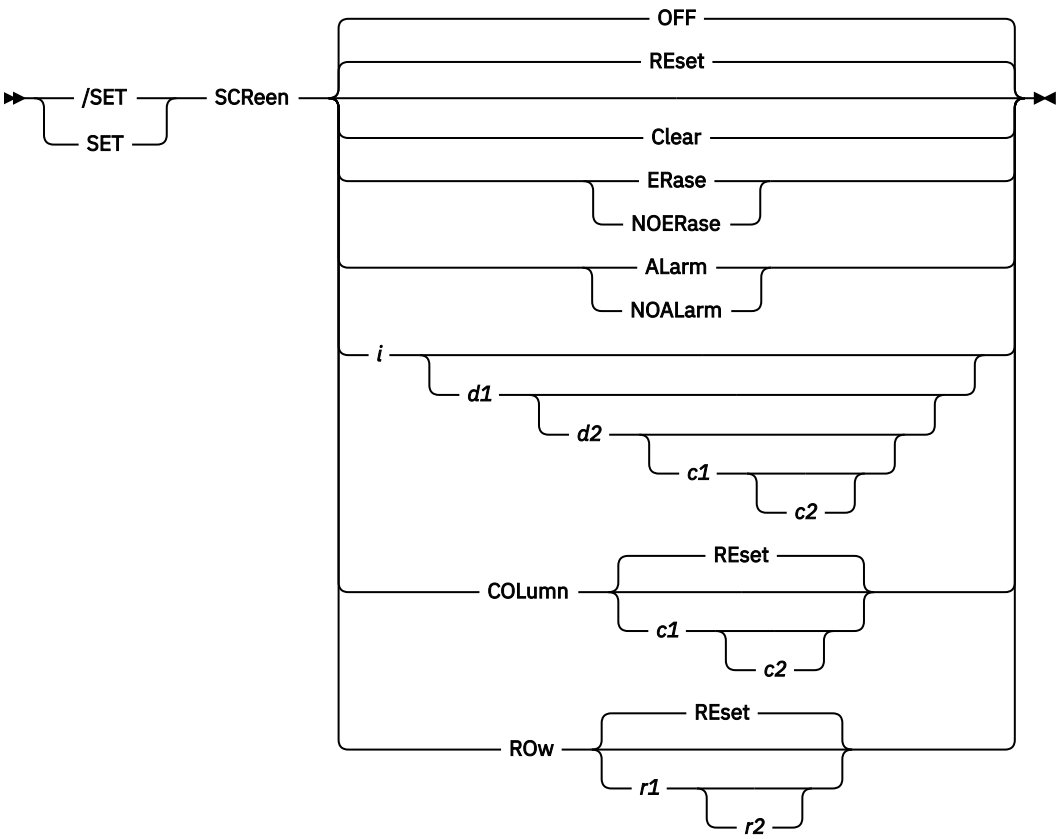
To Set the Editor Autoinsert Feature



To Set System Control Features

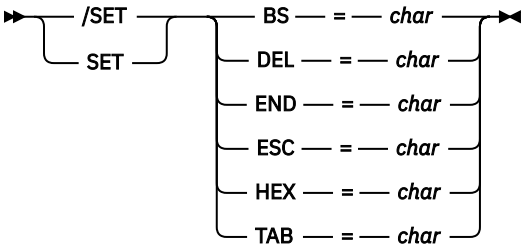


To Set IBM 3270 Screen Features



You can use the [/]SHOW command to display the current settings of most [/]SET command functions.

To Set Control Characters



This form of the [/]SET command is used to set any of the control characters that affect the entry of data or commands from your terminal. In the possible operands:

char
Is either a single character, which may not be alphabetic or numeric or already set as a control character, or it is OFF.

Do not use any of the following characters because they have a special significance within VSE/ICCF:

Symbol
Description

$
Dollar sign

@
Commercial at sign

*****
Asterisk

/
Slash

(
Open parenthesis

)
Close parenthesis

=
Equal sign

,
Comma

'
Single quote

blank

If you use print-type members, do not use the character period (.) because it is part of the print-type member name.

If 'char' is specified as OFF, the effect of a previous [/]SET is removed.

Following is a discussion of the individual operands. Examples are given for the use of some of them.

BS=char
Is used to alter the logical backspace character. It should not be set to '&'.

For hardcopy terminals (IBM 2740, 2741, and 3767), the backspace character is usually set to the Backspace key. However, you can set any other available character as the backspace character. Usually, the backspace character is not required for display terminals.

Example:

```
*READY
/set bs=#
*CONTROL SET
*READY
/input
abc##de#f
/list
ADF
```

DEL=char
Defines a line delete character which, when used as the last character of an input line, causes the input line to be ignored. The line delete character feature is useful on hardcopy type terminals (IBM 2740, 2741, and 3767).

END=char
Defines a logical line end character, which allows you to type in several commands and/or lines of data without having to press the ENTER or EOB key after each line. It causes the lines to be sent to the system and processed one by one until the last logical line has been processed. If you are using an IBM 3270, read also the discussion of the “The DELAY Feature” on page 109.

Example:

```
*READY
/set del="
*CONTROL SET
*READY
/set end=#
*CONTROL SET
*READY
/input#aaa#bbb#ccc#/end#/list
*READY
AAA
BBB
CCC
```

```
*END PRINT
*READY
```

ESC=char
Defines an escape character, which indicates that the following character is not to be treated as a control character even though it is set to a control character function. See also the example for HEX=char below.

HEX=char
Defines a hexadecimal-entry character which, when it occurs in terminal input, causes the two following characters to be treated as hexadecimal digits forming a single EBCDIC character. If either of the characters following the 'hex' control character is not a hexadecimal digit, the 'hex' character will be treated as an ordinary data character. To use unprintable characters (in a member name for security reasons, for example), the characters following the hex character must be greater than hex '40'.

Example:

```
*READY
/set esc=#
*CONTROL SET
*READY
/set hex=?
*CONTROL SET
*READY
/input
xyz#?abc?45?2c
/end
*READY
/listx
XYZ?ABC..
EEE6CCC42
789F1235C
*END PRINT
```

TAB=char
Sets a logical tab character. When logical tabs were set by the /TABSET command, use of the logical tab character in an input line causes this line to be separated into fields according to where the tab positions were set. If no logical tab character is set, the Tab key on the terminal (Field Mark key on the IBM 3270) is the default for logical tabbing. The characters '¬' (the not sign) and '&' must not be set as tab characters when the LIBRC, LIBRL or LIBRP macro is being invoked.

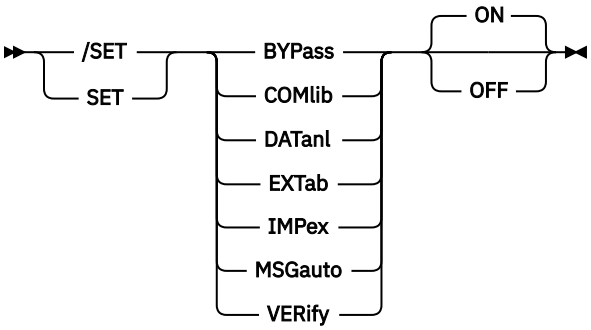
Tab characters in DBCS members are interpreted as such only if the following two conditions hold:

- The tab characters do not appear in a string that has a double-byte representation.
- The tab characters are included in a new line, that is, a line that has just been written or inserted in edit mode.

Example:

```
*READY
/input
/tabset  10  16  36
/set tab=;
main;start;0
;balr;5,0
;using;*,5
;l;1,0(1)
;eoj
;end
/end
*READY
/list
MAIN   START 0
       BALR  5,0
       USING *,5
       L     1,0(1)
       EOJ
       END
*END PRINT
*READY
```

To Set VSE/ICCF Features



This form of the command is used to set various VSE/ICCF features ON or OFF. Following is a discussion of the possible operands.

BYPass
When ON, this feature causes all VSE/ICCF control character interpretation on input to be bypassed. If a tab, backspace or another control character occurs on an input line, the control character function is not performed. Instead, the control character is treated as input data. This feature is initially set off, that is, it must be explicitly set on.

COMlib
Is normally ON. When you set this feature OFF, directory look-ups will not include the common library. Only your primary and connected library will be searched.

DATanl
Applies only to IBM 3270 terminals with the data analysis or text features. The feature is normally OFF. Set it ON before you start an input or output operation that includes special APL- or TEXT-mode characters.

ON causes VSE/ICCF to scan input and output lines for the IBM 3270 data-analysis character set. Characters preceded by 1D are translated to unused positions within the EBCDIC character set. On output, these translated characters are reconverted to 1D followed by the actual terminal character.

OFF causes the APL-alternate characters to enter VSE/ICCF as some character preceded by a 1D control character.

Note: The DATANL feature is not supported on IBM 3274 control units.

EXTab
Allows you to set execution tabbing on. Normally, when a job runs in an interactive partition and this job issues a conversational read, the input data will not be scanned for tab characters. Setting this feature ON forces tabbing to be enabled for conversational reads. This feature is automatically set OFF again at the end of interactive partition execution.

IMPex
Is normally set ON. It allows a procedural CLIST to be invoked as if it were a command. When the (implied execute) feature is set ON, the command

```
/EXEC PROCA CLIST PARMA PARMB
```

can be entered as

```
PROCA PARMA PARMB
```

Setting this feature OFF prevents an automatic library search when an input command appears to be invalid.

MSGauto
Lets you control the display of messages received for your user ID. The default setting for this feature is defined in your user profile.

When the feature is set ON, messages are displayed automatically before any other terminal operation is performed. The screen is saved, and messages are displayed until you acknowledge the receipt by pressing ENTER; then the screen is restored and normal processing is resumed. Use this setting sparingly if you work at a remote terminal. This setting places an extra burden on the traffic on the line.

When the feature is set off, messages are not displayed when they are received; they are just kept. Such messages can be displayed with the [/]MSG command. However, common messages issued by the system operator are always displayed automatically.

Messages are deleted after they have been displayed and cannot be looked at again, unless they were routed to an IBM 328x hardcopy printer. This is possible if they are displayed via the /MSG command.

Regardless of the MSGAUTO feature setting, messages are always displayed automatically when you log on to VSE/ICCF.

VERify

Causes the last 10 lines of input to be displayed on your IBM 3270 terminals each time a new input line is entered. The VERIFY feature is set ON automatically for local IBM 3270 terminals. For non-3270 terminals, setting the VERIFY feature ON causes the last input line to be written back to your terminal. This can be useful for verifying that tabbing or other control character functions are working properly. The VERIFY feature is normally OFF for remote 3270 and non-3270 terminals.

Examples:

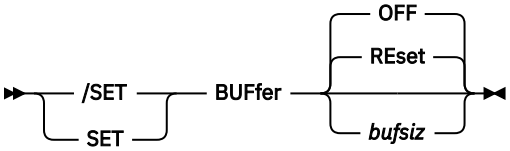
```
/set impex OFF
*CONTROL SET
*READY
/set verify
*CONTROL SET
*READY
/set tab=;
*CONTROL SET
*READY
/tabset 5 10 15
*TABS SET
*READY
/input
a;b;c
A    B    C            (Because VERIFY is on)
```

To Set the Editor Autoinsert Feature

When the feature is set ON, any data processed by the editor that is not an editor command is processed as if it were part of an INSERT command. Such data is inserted into the file just as if an INSERT command had preceded this data. This feature is normally OFF; if you want to use it for an editing session, you must set it ON.

To Set System Control Features

The BUFFER Feature



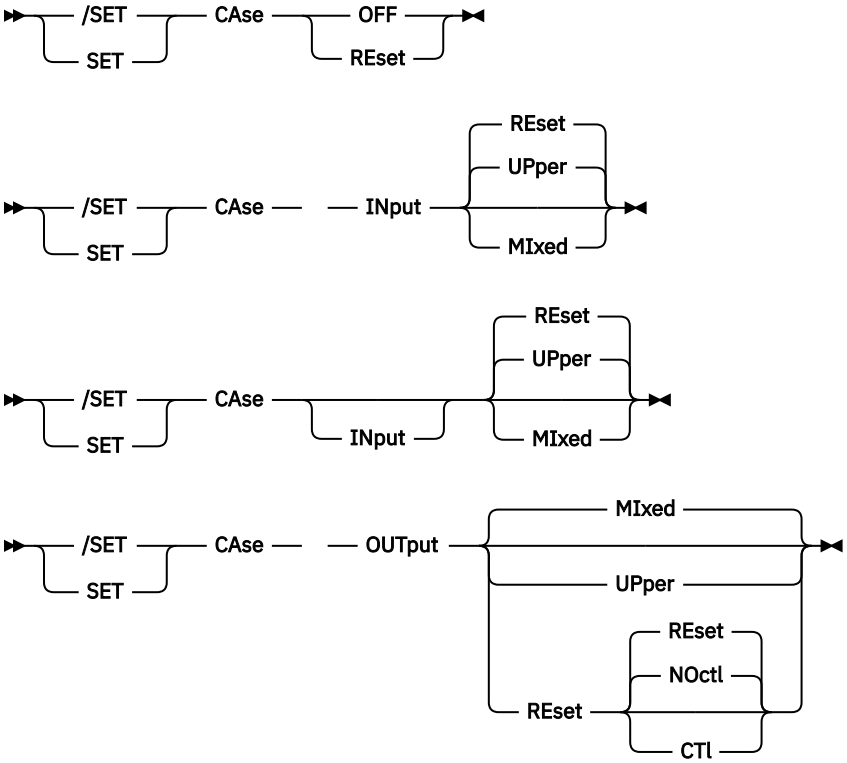
The feature lets you control the size of your print area.

bufsiz
You can specify a value as small as 20 lines for a hardcopy terminal and up to 60 lines for an IBM 3270 terminal. The upper limit depends on the setting of a VSE/ICCF tailoring option or on the value in your user profile. Use the /USER PROFILE command to check this limit.

If your print area is too small, print output from interactive partition jobs will be displayed more often. This can mean that print output that you want to save will be incomplete. A larger print area allows you to store all of a printed report for display. This report can then be saved as a member of your library. You can also transfer the contents of your print area to VSE/POWER for list output (RELIST macro) or to a local hardcopy terminal (/HARDCPY command).

To reset your print area to the default size, specify RESET or nothing.

The CASE Feature



The feature allows you to vary the character translation performed during terminal input or output.

The SET CASE editor command (without a slash) sets overall system character translation. To vary character translation for the current session only, use the CASE and IMAGE editor commands. The CASE operand is ignored in full-screen edit mode.

The defaults are set so that lowercase input data is translated to uppercase. The default for output data is set such that no uppercase output translation occurs; however, terminal control characters (such as backspace or carriage return) are removed from the output. To change the default setting, specify:

INPUT MIXED
To allow lowercase input data.

INPUT UPPER
INPUT RESET
To set input translation back to the default uppercase mode. If the DBCS attribute is set for a member, this command is ignored, and the member is treated as if case were set to mixed.

OUTPUT UPPER
To cause output data to be displayed in uppercase. If the DBCS attribute is set for a member, this command is ignored, and the member is treated as if case were set to mixed.

OUTPUT RESET
To set output translation back to the initial default.

OUTPUT MIXED
The same as CASE OUTPUT RESET. In addition, it causes control characters imbedded in the data (new line, backspace, carriage return, tab, line feed) to be displayed untranslated, thus allowing you some control over hardcopy terminal output formatting.

OUTPUT UPPER CTL
The same as CASE OUTPUT MIXED.

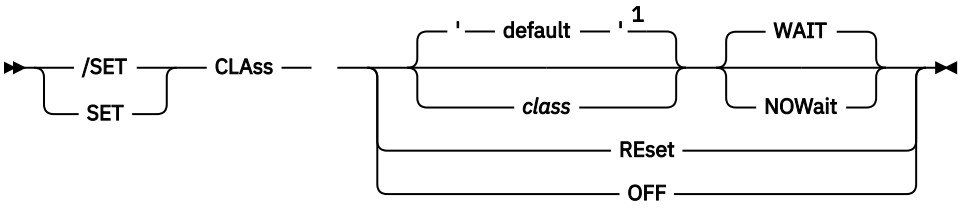
OUTPUT CTL
To cause control character and output character translation.

OUTPUT UPPER NOCTL
The same as CASE OUTPUT RESET.

RESET
OFF
To return both input and output translation to the defaults described above.

When setting CASE, remember that not all IBM 3270s can display lowercase characters. Also your installation may have set up your terminal to accept only uppercase data.

The CLASS Feature



Notes:

[1] If no class is specified, VSE/ICCF uses the class currently set.

The feature sets your interactive partition scheduling class.

Each interactive partition is associated with one or more classes. When you run a job (with the /RUN or /EXEC command), it is executed in an interactive partition whose assigned class matches your class. By changing the class specification with the /SET CLASS command, you can direct your job to an interactive partition with characteristics that you need. For example, you may need a larger partition or one with preallocated work files. Your VSE/ICCF administrator can tell you the interactive partition classes to which you have access.

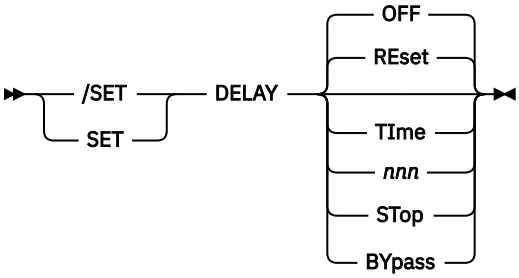
When you request to run a program, an interactive partition that matches your class may currently not be available. Through /SET CLASS, you can indicate ahead of time whether you want to wait or continue in command mode:

- The WAIT operand, the default, tells VSE/ICCF that you want to wait (in execution mode) until an interactive partition becomes available.
- The NOWAIT operand indicates that you want to take back the run request and do other work in command mode.

If you issue SET CLASS without specifying a class, VSE/ICCF uses the class currently set for you.

The operands RESET and OFF return you to the default class as specified in your user profile and set this class to WAIT.

The DELAY Feature



The feature lets you delay the output to the terminal of commands or lines of input. Normally, all commands or lines in a macro, or in multiple line input, are processed one after the other and with no pause in between. The SET DELAY command has no effect in full-screen edit mode. Specify:

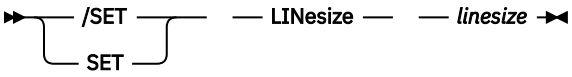
TIME
nnn
To delay the output to the terminal from each command for three seconds; this lets you see the output before it is overlayed. To vary this interval, specify a value between 0 and 255 instead of the TIME operand.

STOP
To halt processing between each line. Press ENTER to display the results of the next command in the macro or input sequence.

BYPASS
To have the output from only the last command or line to be transmitted to your terminal from either multiple line input or multiple commands of a macro.

RESET
OFF
To turn off the DELAY setting so that no delay occurs. This is the default.

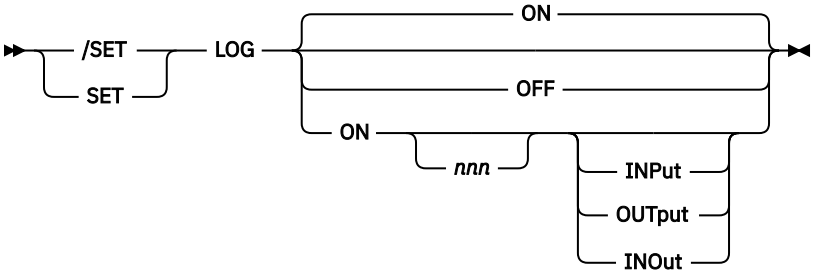
The LINESIZE Feature



The feature varies the line width used for list mode display functions, including the /DISPLAY and /LIST commands and the input mode VERIFY function. If columns 73 through 80 are blank or numeric, the default linesize is 72, otherwise it is 80 (or whatever is specified in the user profile).

For linesize, specify a value from 1 to 80.

The LOG Feature



The feature lets you control terminal logging, which is useful if you have an IBM 3270 terminal and do not get a hardcopy record of the terminal activities. It can be specified only once and only before logging is

set on for a session. The editor command form (without the slash '/') is only valid when working with the **context editor**. Specify

ON
To set the basic logging (you can omit ON if no additional operands follow). All VSE/ICCF commands that you enter or which are issued from a macro are logged.

OFF
To set all forms of logging off, which is the usual setting.

nnn
Is a value between 20 and 300 which sets the number of lines in the log area before logging wraparound occurs. If not specified, or if ON is specified, the default size is 100. It can be specified only once and only before logging is set on for a session.

ON INPUT
To have input data you enter in input or edit mode to be logged along with the basic logging.

ON OUTPUT
To have only command responses logged.

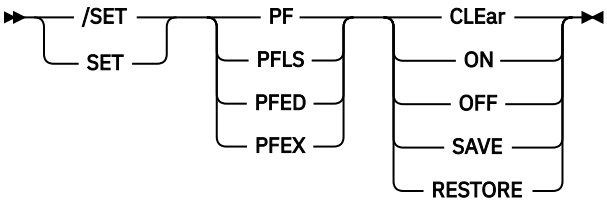
ON INOUT
To have both input and output logged.

The log area contains:

- An identifier (I or O) indicating whether the line was input or output.
- The time when the logging occurred.
- The mode of your terminal during logging, and the first 64 characters of the command, data or output line.

To view your log area, issue the /LIST $$LOG command.

The PF[ED|EX|LS] Feature



The feature allows you to manipulate the PF key settings for:

Command mode (PF)
Edit mode (PFED)
Execution and conversational-read mode (PFEX)
Spool mode (PFLS)

Specify:

CLEar
To have the named PF key settings cleared.

OFF
To have the named PF key settings suspended.

ON
To have the named PF key settings reinstated if they were suspended.

SAVE
To have the named PF key settings saved in the PF-key save area. The currently valid PF-key set remains unchanged, and the previously saved values are overwritten.

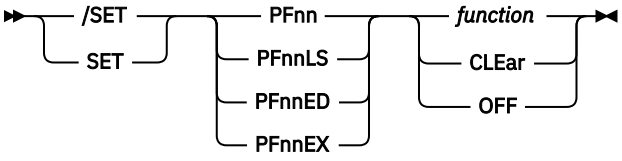
RESTORE
To have the named PF key settings replaced by the content of the PF-key save area and to have these settings activated with the restored values. The content of the PF-key save area remains unchanged. However, the use of the HELP macro changes the contents of the PF-key save area.

If PF key settings are cleared or suspended, the next in the hierarchy become available as shown below.

Mode	Corresponding PF Key Setting Hierarchy
CM	PF
ED,FS	PFED --> PF
EX,RD	PFEX --> PF
LS,SP	PFLS --> PF --> default setting *

* For the default PF-key setting, see Figure 1 on page 28.

The PFnn[LS|ED|EX] Feature



The feature allows you to set a PF key to a function or to clear this setting.

nn
Must be a decimal number from 1 to 24.

LS
ED
EX
Defines the mode for which the setting is to be defined (or changed).

function
Can be any single command or line of data. Setting a PF key in this way and pressing it has the same effect as entering the command or data line itself.

Only in full-screen editor mode can you set a function to handle multiple lines separated with end-of-line characters. You can, however, set a program function to a macro that consists of more than one line or command.

The SET command translates all characters specified in the 'function' field to uppercase. However, in edit mode you can enter both upper- and lowercase characters using the SETLC PFnn command; specify the PF as uppercase, and have the CASE feature set to MIXED. (The SETLC command is not described any further in this documentation.)

CLEAR
OFF
Terminates the previous setting, after which the next lower PF key setting in the hierarchy becomes active, provided it is defined and not set off.

If your terminal does not have PF keys, you can still set PF functions and invoke them via the [/]PFnn command. These functions are also valid in edit mode.

If you use a double ampersand ('&&') when setting a PF key, data entered before the key is pressed will replace the '&&'. This allows you to set more general program functions.

If you specify data when setting a PF key and no '&&' appears within the PF key setting, the previously entered data will be added to the end of the PF key setting before the command or data line is processed. However, if you type a valid command in the command area in full-screen edit mode, this command will not replace the '&&'. It will be processed as an editor command. Similarly, if you set on the AUTOINSRT

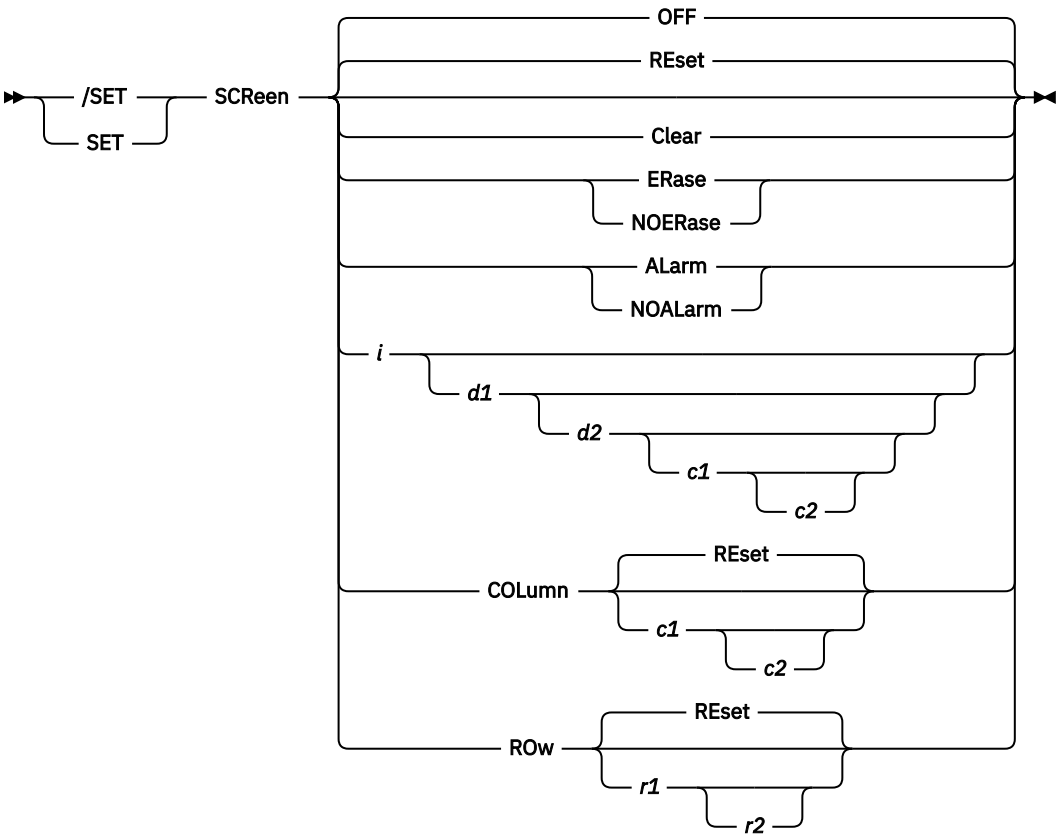
feature, data of the command area are inserted after the current line pointer and again will not replace '&&'.

A PF key cannot be set and accessed within a procedure; it can, however, be set and accessed within a macro.

Examples

```
*READY
/set case input mixed
*CONTROL SET
*READY
/input
aaa
bbb
/end
*READY
/list
aaa
bbb
*END PRINT
*READY
/set case output upper
*CONTROL SET
*READY
/list
AAA
BBB
*END PRINT
*READY
/set pf1 /list 1 5 $$log
*CONTROL SET
*READY
/pf1  (Or press PF1)
I 09/19 CM  /list 1 5 $$LOG
I 09/18 CM  /list
I 09/18 CM  /list
I 09/17 IN  /end
I 09/17 CM  /input
```

To Set the IBM 3270 Screen Features



This form of the [/]SET command is used to set either the screen input or output area, or the screen-related features ON or OFF. The /SET SCREEN command applies only to IBM 3270 terminals. The full-screen editor formats the screen independently. The editor command form (without the slash '/') is only valid when working with the context editor.

The RESET operand resets the values to the screen default values of your system.

With the /SET SCREEN command you can divide your screen into separate areas. This allows you, for example, to display information in one part of the screen, return to command mode, and then transfer the active part of the screen to another screen area. The kind of screen definitions required for split-screen operation is shown by the “Example” on page 116. Your terminal should have PF keys to enable you to move easily from one split screen to another.

The minimum and maximum values that can be set for output areas depend on the model of your IBM 3270:

IBM 3270 Model	First Row Min.	First Row Max.	Last Row Min.	Last Row Max.	First Line Min.	First Line Max.	No. of Lines Min.	No. of Lines Max.
3278 Model 5	1	21	5	25	3	23	5	25
3278 Model 4	1	37	5	41	3	39	5	41
3278 Model 3	1	26	5	30	3	28	5	30
All others	1	18	5	22	3	20	5	22

Note: For list-type commands (such as /LIBRARY, /LIST), the last two rows of a screen are reserved for messages; they are not available for display purposes.

By specifying these values, you override any screen width specifications in the editor VERIFY and PRINT commands. The setting of the screen values in the context editor are effective until logoff or an explicit change of this setting by a new SET command.

VSE/ICCF uses the wide screen of the IBM 3278 Model 5 only for:

Print output from jobs in interactive partitions
Print output from jobs submitted to batch partitions
Display of the line command area to the right of the data line

Your administrator must have requested an alternate screen size in the CICS Transaction Server Terminal Control Table DFHTCT and Program Control Table DFHPCT.

Print output from an interactive partition is placed in the print area and then formatted for the wide screen when it is displayed. The output can also be saved in your library and then displayed by a /LIST or /DISPLAY command, or by the /LISTP command for output from a batch partition.

The wide screen is also used if you retrieve output with the GETL procedure and place it in your print area, from where it is displayed automatically. Another possibility yet: save the output in a library member, using the GETL procedure with the PRINT operand, and later display this output with the /LIST or /DISPLAY command.

Following is a description of the operands:

OFF
Sets the features off.

REset
Sets all of the features to the various defaults (as described for each operand).

ERASE
NOERASE
The operands allow you to set the VSE/ICCF full-screen erase feature on or off. Normally, the screen of your IBM 3270 terminal is erased each time any new data is written to the screen. Specifying the NOERASE operand causes only the lines receiving new data to be erased before data is written to the screen. Specifying the ERASE operand returns the terminal to the standard screen erase setting.

If you work at an IBM 5550 with 3270 Emulation and either of the following conditions exists, all data on the screen is erased also when the NOERASE operand was specified:

- The screen window has been reduced to less than 80 columns by a /SET SCREEN command.
- Data (such as print or SYSLOG) from an interactive partition execution other than full-screen writes (DTSWRTRD) is displayed on the screen.

When switching between two screen output areas, you should first establish a NOERASE environment so that data written to one screen area will not be erased when data is being written to another screen area. See also the “Example” on page 116.

ALARM
NOALARM
The operands allow you to set the audible alarm for your terminal ON and OFF. If ALARM is specified, and if the terminal has the audible alarm facility, a 'beep' will occur each time something is written to the screen, except when hardcopy mode is set.

CLEAR
The operand causes the **active** output area of the screen (and no more) to be erased. This feature is useful when NOERASE has been specified. See also the “Example” on page 116.

i d1 d2 c1 c2
These (numeric) operands allow you to vary:

- The size of the screen input area (i).
- The screen lines that make up the output area (d1 and d2).
- The screen columns that make up the output area (c1 and c2).

If any one of these values is specified, all preceding values in the series must be specified. A value that is not specified retains its present setting, unless this setting is inconsistent with a specified value.

i

Gives the number of lines you want to have available for input following the scale line. The normal screen consists of one 80 character input line, followed by a scale line followed by a display (or output) area of up to 4 output lines, depending on the screen size.

You may want to increase the size of the input area so that you can enter more command or data lines in a single terminal transmission. For example, the length of a typical command or data line (using tabbing) is 15 to 20 characters. Having an input area of four lines would allow you to enter up to 18 command or data lines (separated by logical line end characters) in a single terminal transmission.

VSE/ICCF automatically adjusts the size and the location of the screen output area if:

1. The value i is the only one specified in the command.
2. You specified a value higher than the current setting.
3. The output area as currently defined follows immediately the input area.

For a value of i less than the current setting, VSE/ICCF does not adjust the screen output area.

d1 d2

These values control the vertical size and location of the output area. The normal output area on a 22-line screen, assuming a one line input area followed by a scale line, is from 3 to 22 (the defaults). To vary this area of the screen, specify the number of the first line of the area as d1 and the number of lines in the area as d2. If the number of the area's first line would cause the input area or scale line to be overlapped, VSE/ICCF adjusts the d1 value to the line following the scale line.

c1 c2

These values control:

- The horizontal location of the screen output area (c1 specifies the first column of this area).
- The width of the output area (c2 specifies the number of columns of the area).

The normal screen width is 80 columns. However, the output area can be specified to begin at any given screen column location and can continue for any given number of columns. For example, the output area could be set to occupy columns 41 through 80 of the screen. All output which normally would start at column 1 would now start at column 41. Lines exceeding 40 characters would be truncated accordingly.

For wide screen support, c1 and c2 have a range from 1 to 132. The default settings are 1 for c1 and 132 for c2.

COLumn c1
COLumn c1 c2
COLumn REset

The operands allow you to set the c1 and c2 values without having to specify the three preceding values (i, d1, and d2). If only one value is specified, this is taken as c1, and c2 is assumed to be 80. If c1 and c2 are omitted or if RESET is specified, the horizontal output area is set to column 1 through column 80.

Note: Do not set your screen width to 1, because this will not accommodate meaningful output.

ROw r1
ROw r1 r2
ROw REset

The operands allow you to specify the vertical location of the screen relative to the end of the input area of the screen:

- For r1, specify the number of the first row of your output area.
- For r2, specify the number of the last row of your output area.

Since the specified row numbers are relative to the end of the input area, you need not be concerned about the present number of lines in the input area.

If you specify RESET or omit the r1 and r2 values, VSE/ICCF sets the vertical length of your output area to the largest possible value based on the size of your input area. If only one value is specified, this is assumed to be r1; r2 is then set based on the specification of r1. If r1 and r2 are specified, the value for r2 must be 4 higher than the value for r1.

Example

To define two active portions of the screen so that you can use one area while retaining information in the other:

1. Set three PF keys as follows:

```
/set  pf1  /set  screen  row  1   10
/set  pf2  /set  screen  row  11  20
/set  pf3  /set  screen  clear
```

   These settings allow you to vary your output area between the top and the bottom half of the screen by pressing PF keys 1 or 2. To clear the screen output area, press PF3.

2. Set the NOERASE screen feature so that a write to one area of the screen will not erase the entire screen:

```
/set  screen  noerase
```

Now you can use two sections of the screen: one to retain information while you are actively using the other.

Suppose you want to purge unneeded members from your library. You could, for example:

1. Place a library display in one area of the screen and retain it.
2. Use the other area for the purges and lists of library members.

Pressing the PF2 key sets you to the second screen area. Now enter the /LIB command. If the display will not fit entirely into the lower screen area, the scale line will indicate that you are in list mode (LS). Entering the /CANCEL command (or pressing the Cancel key) will return you to command mode.

Pressing the PF1 key causes the output area to be set to the top portion of the screen. You can now do /LIST, /PURGE or other functions while retaining the library display in the lower portion of the screen.

You can also page the display in the top portion of the screen forward by pressing the PA2 key.

/SETIME Command

The /SETIME command alters any of the three time factors that can affect terminal activity or interactive partition execution.

It allows you to exceed your user profile specified maximum execution units for any one job. The command is effective in command and execution mode.

EXEC

The operand overrides the limits set for execution units per job and for the time a job may run in an interactive partition:

m

Is a number from 0 to 32767. It represents the maximum number of interactive-partition execution units that the next or current job can accumulate without being canceled by VSE/ICCF. An execution unit is about one second during which the interactive partition is eligible for execution cycles.

n
Is a number from 0 to 65535. It represents the maximum time in seconds that a given job can run in an interactive partition. Expiration of this time causes the job to be canceled. If you do not specify a value for n, VSE/ICCF assumes this to be four times the value of m or your default, whichever is larger.

After the current or next execution has completed, the limits for the number of execution units, and for interactive partition time, will be reset to your default values.

Do not set unnecessarily large limits because this would give you and the system little protection. Time limits protect you and the system if the job enters a loop. In that case, the job is canceled and the storage is freed when the time limit had been reached.

The execution units and total interactive partition time limits can also be set for a given job by a `TIME=mm[nn]` specification in the /OPTION statement for the job.

TIMEOUT seconds
For seconds, specify a number from 60 to 3600. It represents the number of seconds your terminal is allowed to be idle. When this time expires, logoff is forced, unless your user profile indicates that you should not be forcibly logged off due to a timeout condition.

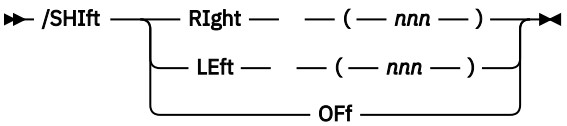
A timeout-forced logoff does not apply when a user is in execution mode.

Examples

```
*READY
/setime 300
*LIMIT HAS BEEN SET
*READY
/setime 500 5000
*LIMIT HAS BEEN SET
*READY
/setime timeout 1800
*LIMIT HAS BEEN SET
*READY
```

/SHIFT Command

The SHIFT command causes the display on a screen to move left or right.



RIGHT (nnn)
Moves the screen window to the right by nnn columns. This enables you to view the right-hand part of a record which is currently not visible on the screen. The data being displayed moves to the left.

LEFT (nnn)
Moves the screen window to the left by nnn columns. The data being displayed moves to the right.

The maximum value that can be specified for 'nnn' is 100. However, the left margin of the screen (the window) cannot be moved to a position higher than column 100 of the data, and not lower than column 1. The scale line moves by the same amount as the data.

If you enter the /SHIFT command more than once, the nnn values are accumulated. For example, entering /SHIFT RIGHT (30) twice moves the left margin of the screen to column 60 of the data.

OFF
Causes a shift left to column 1 and a display in wraparound format (also known as "80-character format"). When the display has this format, pressing ENTER causes the next screen to be displayed in the same format. The default PF key settings in list and spool mode (see Figure 1 on page 28) are reestablished.

The /SHIFT command is valid in LS (list), SP (spool print), and RD (conversational read) mode. It controls shifting of print-type data: VSE/POWER list output, print-type members (members whose names end with .P), and the print area $$PRINT. It causes a display in print line format (as opposed to wraparound format), that is, part of the line may be invisible. The command suppresses all the default PF key settings for list and spool mode.

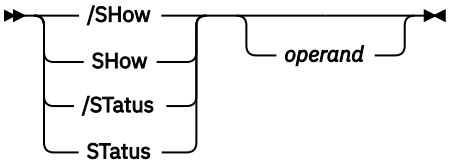
While your display is in print line format, pressing ENTER causes the next screen to be displayed in print line format with the same shift amount as the previous screen. For a display in print line format, SYSLOG messages, too, have a portion of the message line truncated.

When issued for normal VSE/ICCF library members or for print-type members that are compressed or displayed in hexadecimal format, the /SHIFT command is ignored and treated as if ENTER had been pressed.

[/]SHOW Command

This command (and also the [/]STATUS command) displays the settings of options, features, and control values affecting your terminal.

The format with a slash – /SHOW (or /STATUS) – is effective in any mode except edit; the format without the slash is effective only in edit mode.



The command displays the current status of the specified option. If no option is given, the options highlighted in the list below apply (except in edit mode).

Operand:
Displays the Setting/Value Of

BS
Backspace character

BUFfer
Size of the print area

BYPass
Control-character bypass

CASe
Input/output character translation

CHar
Control-character assignments

CLAss
Your execution class

COMlib
Common library search

CTL
Control-character assignments

DATAnl
Data analysis feature

DATe
Date and time

DEL
Delete character

DELAy
Multi-command input delay

END
Logical line-end character

ESC
Escape character

EXec
Status of your job in an interactive partition

HEX
Hex control character

IMPex
Implied execute

LIMit
Various time limits

LIBrary
Your primary and connected libraries

LINesize
Line length

LOG
Type of terminal logging

MOde
Terminal mode

MSG
Message mode

PF[ED][EX][LS]
Set PF keys in CM, ED, EX, and LS modes

PFnn[ED][EX][LS]
Set PFnn keys in CM, ED, EX, and LS modes

PF SAVE
Content of PF-key save area

PSize
Default partition size

SCreen
Screen attributes, size, and location

TABChr
Tab character

TABs
Tab location

TErmid
Your terminal identifier

TIme
Various time limits

USer
Your user identifier

VERify
Input verify

XLate
Input/output character translation

If you specify an invalid operand, the default display will consist of the settings for IMPEX, LOG, PF, VERIFY, LINESIZE, BYPASS, DATANL, and COMLIB.

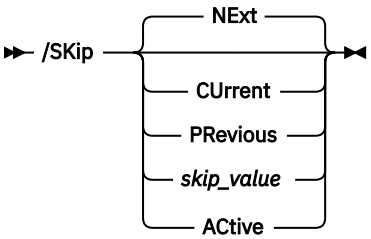
Examples

```
*READY
/status
DATE=09/02/1998 TIME=01/30/00 EXECUTION UNITS=0030
USER ID CODE  = USRA
TERMINAL CODE = L2D3
LIBS: MAIN = 0001 CONN = 0005 COMMON = 0002
SYSTEM MODE = Command
DEFAULT PARTITION SIZE=0192K
NUMBER OF TIME SHARING USERS LOGGED ON = 0003
*READY

/show tabs
TABS = 10 16 36 72
*READY

/show pf
01 /SET SCREEN ROW 1 10
02 /SET SCREEN ROW 11 20
03 /SET SCREEN CLEAR
*PROGRAM FUNCTIONS SET OFF
*END PF DISPLAY
*READY

/show char                 (All control character assignments
D T B E E H                are shown in vertical arrangement)
E A S N S E
L B   D C X

¬ ; < | % ≠
5 5 4 4 6 7
F E C F C B
*READY
```

/SKIP Command

The /SKIP command moves the display forward or backward in the data being viewed.

The command is effective in SP (spool print) and in LS (list) mode during a display of print output.



CUrrent
Causes a (re)display of the current page.

This can be handy if you temporarily leave the current display. You might, for example:

- Want to see if any messages have arrived. The /MSG command causes the current display to be left.
- Want to set a PF key. The /SET PFnn command causes the current display to be left.

NExt
PRevious
Causes a scroll forward or backward by one screen. The unit of scrolling is the size of one logical screen, which is defined through the /SET SCREEN command. These operands are easier to use than S+n or S–n; they are valid for the display of print-type and normal VSE/ICCF library members. Consider setting PF keys to `/SKIP NEXT` and `/SKIP PREVIOUS`.

skip-value
For skip-value, specify one of the following:

m (or) –m
This is a number from 0 to 99999 representing the number of logical lines (from the last line displayed) to be skipped over during a display. –m causes backward skipping.

S+n (or) S–n
n is a number from 0 to 99999 representing the number of logical lines. to be skipped over during a display to an IBM 3270 terminal. By specifying S+n (or S–n), you request VSE/ICCF to skip over n lines of printout beginning at the first line of the last screen of displayed print data. –n causes backward skipping.

This form of the skip value is useful if, for example, you want your display to be scrolled forward or backward by n lines. For example, PF keys might be set to `/SKIP S–5` and `/SKIP S+5`. Then each time one of these PF keys is pressed, the display moves back or forward by 5 lines.

P+k (or) P–k
k is a number from 0 to 99999 representing the number of execution mode printer pages (skips to carriage channel 1 or top of form) to be skipped over during a display of print output. –k causes backward skipping.

You can use this form of the skip value when you are displaying the output of an interactive partition job.

If you use the P operand and the program does not issue any advances to the top of the form, the skip proceeds either to the beginning or to the end of the print area.

This form of the skip value is not effective when you are displaying VSE/POWER-spooled output.

TOP
Displays the top of the file, which can be a VSE/POWER list file for /LISTP processing or a VSE/ICCF member for /LIST processing.

BOTtom (or) END
Displays the end of the file present when you use the /LIST or /LISTP command.

VSE/ICCF advances the display to the end of the data or 99999 lines, whichever is the smaller range of advance. After having displayed the end of the data, VSE/ICCF issues a message. When you press ENTER, your terminal returns to command mode.

ACtive
The result of this parameter depends upon when /SKIP ACtive is issued:

Browsing Active Queue Entries
Issuing a /SKIP ACTIVE command while browsing an active queue entry will result in an restart-to-active request to VSE/POWER asking for the currently active line/record. VSE/ICCF will display the active line followed by message *ACTIVE OR LATEST (similar to the /SKIP BOTTOM command). In addition,

- The whole queue entry can be browsed using all the /SKIP and /LOCP commands provided by VSE/ICCF.
- Issuing a /SKIP ACTIVE command again means refreshing.
- Issuing a /SKIP ACTIVE command to a queue entry which in the meantime has completed or was complete from the beginning results in message:

```
*QUEUE ENTRY NEITHER ACTIVE NOR IN-CREATION, PRESS ENTER TO RESUME
```

Browsing In-Creation Queue Entries
Issuing a /SKIP ACTIVE command while browsing an in-creation queue entry will result in a QUIT request to VSE/POWER followed by GET BROWSE OPEN. VSE/ICCF will display the latest line/record followed by message *ACTIVE OR LATEST (similar to the /SKIP BOTTOM command). In addition,

- The queue entry from top to the latest line/record can be browsed using all the /SKIP and /LOCP commands provided by VSE/ICCF.
- Issuing a /SKIP ACTIVE command again means refreshing.
- Issuing a /SKIP ACTIVE command to a queue entry which in the meantime has completed results in message:

```
*QUEUE ENTRY COMPLETE IN xxx QUEUE   (where xxx=LST/PUN/RDR/XMT)
```

Browsing Queue Entries Using PF-Keys
PF keys for browsing VSE/POWER queue entries are assigned by the Interactive Interface. This also applies for /SKIP ACTIVE.

Your terminal is placed into list mode when:

- You enter commands that display library members or work areas, for example with /LIST and /DISPLAY.
- During editing, you enter the PRINT or PRINTF editing command.
- You enter the /LISTP command to display output held in a VSE/POWER spool file.
- An interactive partition execution displays print output.

The default is m=0.

Note:

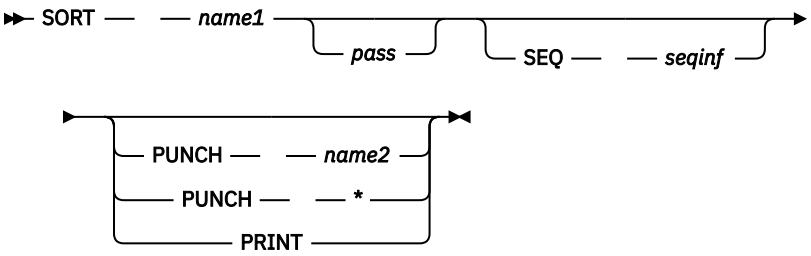
1. When a /LIST command is issued for a compressed member, only forward skips are allowed; it must be a SKIP to a line that is not yet displayed on the screen. For example, if six logical lines are displayed on the screen, VSE/ICCF rejects the command /SKIP S+5.
2. For print-type members specified in one of the /LIST commands, the value for 'm' or 'n', as well as the line number printed in front of each line by the /DISPLAY command, relate to lines in print format and not to the records of the member. Thus two records of a member can account for one line of print output. This is true also if /SKIP has been entered in list mode and the output is displayed in print format and if /SKIP is issued in execution mode.

Examples

```
*READY
/display counts
0001 CARD 1
0002 CARD 2
0003 CARD 3
0004 CARD 4

 (end of print area)

/sk 200
0205 CARD 205
0206 CARD 206

 (and so on)
```

SORT Procedure

The procedure sorts a library member according to the EBCDIC sequence of hex values and stores or displays the sorted member in accordance with your specification.



name1
Is the library member to be sorted.

pass
Is a 1 to 4 character password which must be specified for password-protected members.

SEQ seqinf
Is the information indicating where the sort sequence fields are located. The format is `xmmnnxmmnn...`, where:

x
Is A or D indicating ascending or descending sequence.

mm
Is the number of the first column in the sequence field.

nn
Is the number of columns in the sequence field.

Up to four sequence fields may be specified. If the SEQ operand is omitted, the sort will be on columns 1 through 15.

PUNCH name2
PUNCH *
PRINT
Defines where the sorted output is to be saved. If you specify:

PUNCH name2
VSE/ICCF stores the sorted output into your library with the name specified for name2.

PUNCH *
VSE/ICCF places the sorted output into the punch area $$PUNCH.

PRINT
VSE/ICCF displays the sorted output at your terminal.

Note: Because /* signals end of data to the sort program, members must not have /* as data in columns one and two.

Examples

1. Sort a member into ascending order and display the result at the terminal:

```
sort dtamem seq a0105d6704 print
```

2. Sort a member into default sequence and have it replace itself:

```
sort memba
```

/SP Command

For details see the "/STATUSP Command" on page 125.

$SPACE Procedure

The $SPACE procedure causes the DTSSPACE program to be loaded into an interactive partition and to run.

This program prints each of the dynamic disk-space allocation areas of your VSE/ICCF. The information printed includes the volume and location of the space area, individual allocations within it, and the percentage of the area that is free.

Examples

```
*READY
$space
*RUN REQUEST SCHEDULED FOR CLASS=A
-------------------------------------------------------------------------
        DYNAMIC SPACE ALLOCATION MAP -- VOLUME # 01
BEG=1110  FOR 60      DEV=3380  SYS001 SER=SYSWRK COLD AREA COLD STARTED
-------------------------------------------------------------------------
                  FILE ID                   FROM      FOR    OWNER  DISP
DYNAMIC-SPACE-CONTROL-INFORMATION              1110       1  SYS  KEEP
IJSYS02.AAAA.T264                              1151       1  ***  ****
 .
***TOTAL*** ALLOCATED=0002 FREE=0058 % FREE=96%
 .
-------------------------------------------------------------------------
        DYNAMIC SPACE ALLOCATION MAP -- VOLUME # 02
BEG=1170  FOR 30      DEV=3380  SYS001 SER=SYSWRK WARM AREA WARM STARTED
-------------------------------------------------------------------------
                  FILE ID                   FROM      FOR    OWNER  DISP
DYNAMIC-SPACE-CONTROL-INFORMATION              1170       1  SYS  KEEP
 .
***TOTAL*** ALLOCATED=0001 FREE=0029 % FREE=96%
 .
*PARTIAL END PRINT
```

/SQUEEZE Command

The /SQUEEZE command converts a library member from display to compressed format.

It is valid only in command mode.

name
Is the name of a library member in display format, which is to be compressed.

password
Is required only if the library member to be compressed is password-protected.

SAVEIN
Causes the display format version of the member to be saved in your input area (destroying any previous contents of the input area). The operand cannot be used if you are in asynchronous execution mode (after having issued the /ASYNCH command).

If the operand is used in a /SQUEEZE command of a procedure, ensure that a /SAVE command is issued in the procedure following the /SQUEEZE; else the contents of the input area will be lost.

LOWER
Is specified when the member to be compressed contains more lowercase than uppercase alphabetic data.

Library members are normally retained in display format so that certain VSE/ICCF operations (such as edit) can be performed on them. However, once a library member has reached a final state, it can be compressed to save disk space.

A compressed member usually takes up 55 to 70 % less disk space than a member in display mode. The actual savings depends on the number of blanks in the data and the data itself. If the data consists mainly of characters from the 64 character EBCDIC graphic subset and is not very dense, the saving will be greatest.

The compressed member replaces the display copy of the member in the library. To compress a library member and yet retain the display version of it, enter the SAVEIN operand. Then, to save the display format version of the member, issue the /SAVE command after the /SQUEEZE command.

A compressed library member can be converted back to display format by entering input mode (/INPUT command) and issuing the /INSERT command for the member. You can use the /LIST command to list a compressed library member, and you can use the /SKIP command to skip forward (but not backward) while displaying the member.

Print-type members that have been processed by the /SQUEEZE command are displayed in 80 character record format, not in print format.

Examples

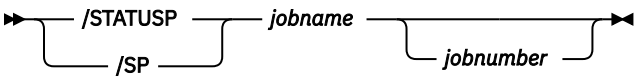
```
*READY
/squeeze payprog savein
*INPUT RECS = 1230 OUTPUT RECS = 0467 SAVINGS = 62 PERCENT
*END COMPRESS
*READY
/save payproga   (save display format version)
*SAVED
*READY
```

[/]STATUS

For details see the “[/]SHOW Command” on page 118.

/STATUSP Command

The /STATUSP (and /SP) command displays the status of a job submitted to VSE/POWER either for being run in a VSE batch partition or for being transmitted to another node. The command can be used only in command mode.



jobname
Is the name of the VSE/POWER job whose status is to be displayed.

jobnumber
Specifies the job number assigned to the job by VSE/POWER. Specify this operand if the job name is not unique. If you do not supply this number, you get a display of the job located first in the VSE/POWER queues.

The status is displayed as:

```
*STATUS = xy - job status
```

Where x and y indicate the following:

x =
One of

L:
The job is in the list queue.

N:
The job cannot be located.

P:
The job is in the punch queue.

R:
The job is in the reader queue.

X:
The job is in the transmission queue.

y =
The one-character disposition code as returned by VSE/POWER. If this is an asterisk (*), the job is running.

For job status, the display can be one of the following:

`NOT FOUND` – The named job is not known to VSE/POWER.
`COMPLETED` – The named job has completed execution. You can display list output of the job by issuing the /LISTP command. You can retrieve the job's list or punch output via the GETL or GETP procedure.
`EXECUTING` – The job is running.
`AWAITING EXEC` – The job is in a queue awaiting execution in a VSE batch partition.

Note:

1. The output displayed by the /STATUSP command may include VSE/POWER messages, which are identified by the characters '1Q', '1R' and '1V', followed by a message number. These messages are explained in z/VSE Messages and Codes 1.
2. The /STATUSP command cannot be used within a procedure.
3. If you are using an IBM 3270 you can set a PF key to part of a command, for example /STATUSP (don't forget the blank after /STATUSP), which allows you to type in the job name and job number and then press the PF key to have the status of the named job displayed.

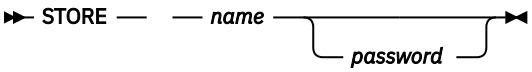
Examples

Display the status of MYJOB, whose jobnumber is 00024:

```
/sp  myjob  00024
*STATUS = LL - JOB COMPLETED
```

STORE Macro

The STORE macro stores the contents of the punch area in your library.



name
Is the library member name under which the contents of the punch area will be saved.

password
Is a four-character password which needs to be specified if the member is to be password-protected.

The macro copies the punch area to the input area and executes a /SAVE command.

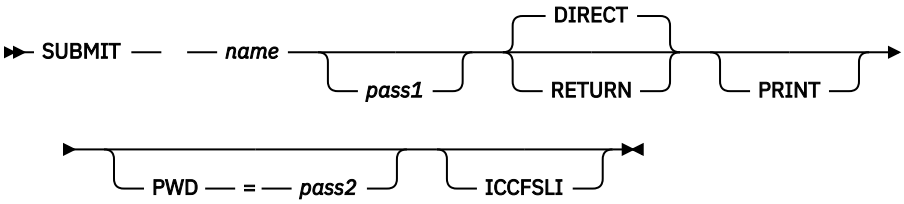
Issue the macro immediately after the execution that produced the punch output. Do not allow any other execution to take place, even if it does not write data to the punch area.

Since the editor stack area and the execution punch area are the same area, you can use the macro to store data that was placed in the stack during an edit operation.

If the message `NAME IN LIBRARY` appears, you must first issue the /SAVE or /REPLACE command for the data that is now in the input area.

SUBMIT Procedure

The procedure submits VSE/ICCF or VSE job streams for processing in another VSE partition.



name
Is the name of a VSE/ICCF library member that contains a job stream and is to be submitted for batch execution via VSE/POWER in another VSE partition. The member must include the required VSE job control or VSE/ICCF job entry statements. Certain names should be avoided; refer to the restriction "6" on page 331.

pass1
Is the password that you must specify if the library member is password-protected.

DIRECT
RETURN
Specify DIRECT or omit the operand to have the output from the job directed to the printer or punch unit.

Specify RETURN to have the output placed into the VSE/POWER output queues. If you do this, you can:

- View this output by using the /LISTP command.
- Retrieve this output by using the GETL or GETP procedures.
- Finally dispose of the output by using the /ERASEP command. However, only the submitter or the user designated in the DEST parameter of the VSE/POWER JECL statement * $$ LST may access an output queue entry.

  An authorized user may specify ANY in the DEST parameter (authorization is established in the VSE/ICCF user profile). This makes the output accessible to any user.

PRINT
The operand causes a display of all VSE job control statements that are passed to VSE/POWER.

PWD=pass2
Is a one- to eight-character password that is to be inserted in the VSE/POWER $$ JOB statement. The specified password will protect the output. If you specify a password for your output, you must specify this password also in the retrieve procedures (GETL, GETP, and GETR).

Note: Do not use CANCEL as a password.

ICCFSLI
Specify this operand to have an * $$ SLI statement generated. The data is not written to the VSE/POWER queue. Instead, VSE/POWER itself reads the data from the VSE/ICCF library file when the job is run.

The * $$ JOB and * $$ EOJ statements in the library member are ignored. Instead, the * $$ JOB statement that SUBMIT produces is in effect. In addition, an * $$ LST or * $$ PUN statement causes VSE/POWER to discard the original job number and to assign a new one.

Note: VSE/ICCF adds a /* and a /& statement to the end of the member. If the retrieved member includes these statements already, the system interprets the added statements as an incomplete job without a name.

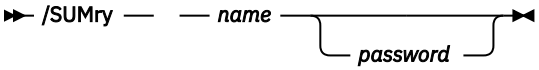
If **ICCFSLI is not specified**, the VSE/ICCF member to be submitted to the VSE/POWER reader queue is read from the VSE/ICCF library file and written to the VSE/POWER queue.

For the retrieval of the member, the library-search chain (primary – connected – common) valid at the time of submission is used. This applies to either of the above cases.

/SUMRY Command

The command produces a summary listing of a library member.

The listing includes any line beginning with a slash (/) plus a count of all lines which do not begin with a slash. The command is effective only in command mode and is valid only for noncompressed members.



name
Is the name of the library member for which a summary listing is requested.

password
Is the password that you must specify if the library member is password-protected.

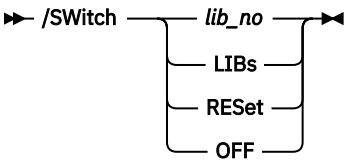
Examples

```
*READY
/sumry fortprog
0001 /LOAD VFORTRAN
0002 /OPTION TRUNC
CARD IMAGE COUNT = 0009
0012 /DATA
CARD IMAGE COUNT = 0005
*END PRINT
*READY
```

/SWITCH Command

The command is used to switch from one library to another.

It is valid only in command mode.



lib-no
Is the number of the library which you want to become your primary library. This can be any library which you are authorized to use: your default library, one of the alternate libraries defined in your user profile, or a public library. The /USER LIBRARY command will tell you which private libraries you can access. The /LIBRARY and /LIBRARY CONN commands will display the library identifications of the current primary and connected libraries.

All later requests to access, update, or save a library member will cause the directory of the new library to be scanned.

You can switch only to a public library, or to a private library that you are authorized to use (this does not apply to the VSE/ICCF administrator). If the /SWITCH command is used in a procedure, your library access will be switched only until the processing for this procedure is complete.

LIBS
Causes the connected library to become the primary and the primary the secondary library. This can be useful if you have used the /CONNECT command to connect a secondary library to the primary library and you want to reverse the search order of the two libraries.

RESET
OFF
Causes the library configuration to be as it is defined in your user profile.

In some situations the /SWITCH command may have no effect. For example, if the specified library number does not exist, or if you are not authorized to use it. /SWITCH also has no effect if the library is already in use or if it is a connected library.

If the RESET operand is issued, your main library as indicated in your user profile becomes your primary library. If any other library is connected when the command is issued, this logical connection is terminated, and RESET restores the library configuration to user profile status.

Examples

```
*READY
/switch 12
*SWITCHED
*READY
/input
aaaaa
/end
*READY
/save tmemb
*SAVED
*READY
/switch 14
*SWITCHED
*READY
/list tmemb
*FILE NOT IN LIB
*READY
```

/SYNCH Command

The /SYNCH command is used to re-synchronize interactive partition execution with your terminal.

The use of this command assumes that you had previously started a job in an interactive partition and then disconnected the execution from your terminal by using the /ASYNCH command. The command is effective only in command mode.

While in asynchronous execution mode you can perform any normal VSE/ICCF command function (such as editing) other than starting another job execution. When you want to see the result of your execution, issue the /SYNCH command to place your teminal back into execution mode. Use the /SHOW EXEC command to display the status of your job, and to tell whether the /SYNCH command is required.

Examples

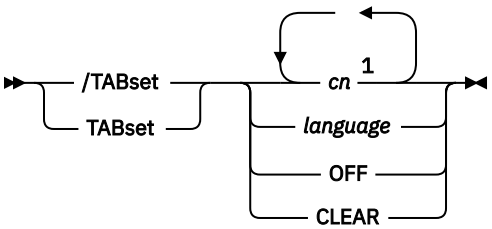
```
*READY
/exec payprog
*RUN REQUEST SCHEDULED
/asynch
*ASYNCHRONOUS EXECUTION MODE ENTERED
/edit artest
 ...
 ...                          (Edit commands)
 ...
quit
END EDIT
*READY
/synch
 ...
 ...                          (Job output)
 ...
```

[/]TABSET Command

The [/]TABSET comand lets you set up your own logical tab settings for a line.

These setings determine the number of spaces to be inserted in the line when either the logical tab character or the Tab key is used.

The /TABSET form of this command is effective in command and input mode; the TABSET form is effective only in edit mode. Tab settings defined by this command remain in effect until another [/]TABSET command is issued.



Notes:

[1] Up to 11 numbers can be specified.

cn
Are column positions for logical tab settings.

You can specify from one to eleven numbers from 1 to 80 in ascending order. If you specify more than eleven numbers, VSE/ICCF uses only the first eleven.

language
Is the programming language or other character identification used to represent preset tab values.

OFF
CLEAR
Causes all logical tab settings to be cleared so that tabbing is not possible.

Each [/]TABSET request overrides all previous /TABSET commands. Tab settings can be displayed with the /SHOW TABS command.

You can set certain predefined tab values. Below is a table of the keywords that are allowed, and the tab positions that they set:

Keyword Operand	Tab Settings
ASsembler	10,16,36,72,73
BASic	10,20,30,40,50,60
COBol	8,12,16,20,24,28,32,36,40,44,73
FORTran	7,73
PL1	5,10,15,20,25,30,35,40,45,50
PLI	5,10,15,20,25,30,35,40,45,50
RPG	6,20,30,40,50,60
TENS	10,20,30,40,50,60,70

Only the capitalized portion of the keyword need be specified as the operand.

/TIME Command

The /TIME command displays the date and time and the number of background execution units.

This command is valid in command and execution mode.

If your terminal is in command mode, the number of execution units displayed is the number accumulated by the last background job that was run. If the /TIME command is entered while the background job is running, the execution units displayed are those that the job has accumulated up to the time when the command was entered.

The date appears in the format defined for the system when VSE/ICCF was started.

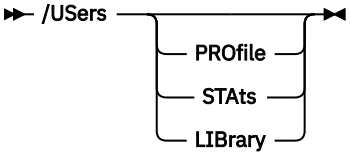
Examples

```
*READY
/time
DATE=09/02/1998 TIME=01/30/55 EXEC UNITS=0024 PARTN TIME=00120
*READY
```

/USERS Command

The /USERS command displays various fields in your user profile and the libraries to which you have access.

Without an operand, the command displays the total number of terminal users currently logged on. The command is effective in any mode except the edit and message mode.



PROfile
Displays various defaults in your user profile, such as maximum number of statements per input area and your execution time limit.

STAts
Displays statistics in your user profile such as total number of logons and of run requests (/RUN or /EXEC). It also displays the execution units used, your total logon time, and the disk space. The statistics cover the time since the last accounting run done by your administrator.

LIBrary
Displays the libraries to which you have access. This includes your main library, the common library, and any alternate libraries to which you can switch.

Examples

1. Display the number of users logged on:

```
/users
NUMBER OF TIME SHARING USERS LOGGED ON = 0003
*READY
```

2. Display the defaults in my user profile:

```
/users profile
USER LIB MAXST MAXPU MAXPR TIMLM TIMOT PPTIM LNSIZ DEFLTS PROC
bbbb  4   350   2000 2000   900   600   3600  80   +#<>!% start
*READY
```

Following is an explanation of the values displayed in Example 2:

Column Heading	Explanation
USER	Your user identification
LIB	Your default main library
MAXST	Size of input area (statements)
MAXPU	Size of punch area (statements)
MAXPR	Size of print area (statements)
TIMLM	Default time limit per execution
TIMOT	Time-out value
PPTIM	Maximum time in interactive partition before the job is canceled.
LNSIZ	Line size for /LIST and /DISPLAY commands

```
DEFLTS     Delete, tab, backspace, line-end, escape and
           hex characters
PROC       Logon procedure
```

Chapter 4. General Information about the Editor

VSE/ICCF has two editors:

- A context editor, which was designed for typewriter terminals. This editor must be used in procedures and macros and in the DTSBATCH utility.

  The editor works on one line at a time, the line that is determined by locating a given string. The context editor is described in Appendix B, “Context Editor,” on page 341.
- A full-screen editor for IBM 3270 and similar display terminals.

  This editor utilizes the entire screen. It allows you to modify data simply by positioning the cursor at a point on the screen and typing the new data. Then, by entering a command or pressing a key, the modification is applied to the file.

This section describes how to use the full-screen editor (in VSE/ICCF publications mostly referred to as the editor). It consists of two major parts:

1. A discussion of some important editor applications.
2. A reference section giving detailed descriptions of all editor commands and macros in alphabetical order. The section includes examples that guide you into the usage of the editor commands.

The editor allows you to create and modify files in the VSE/ICCF library from your display terminal, utilizing the facilities of the IBM 3270 Information Display System. A file being edited is one of the following:

- A member of the VSE/ICCF library.
- The input area (which you can save as a library member).
- The punch area.
- The print area.

Editing your print area is useful when, for example, you are looking for errors in a compiler listing. Having both the print area and your source program on one screen together may aid in finding an error.

Several files can be created, edited, saved in the library, and edited further without leaving edit mode. Split screen control allows displaying and editing multiple files on one screen – you can display and edit on one screen different areas of the same file.

The editor lets you move and copy data (contiguous or non-contiguous) from one or more places in a file to different places of the same or another file.

Note: The editor does immediate updating. A change is written to disk when you press ENTER or a PF key. Therefore, prior to applying "dangerous" changes to a file, consider using the COPYFILE macro to create a copy of the file under a new name (see also “Generation Member Group” on page 5 and “Maintaining Multiple Change Levels of a Member” on page 162).

Capabilities and Characteristics

The editor's major capabilities are:

- Creating a new library member while in edit mode.
- Split screen control, allowing multiple displays.
- Concurrent editing of several files in a single session.
- Concurrent editing of different areas within a given file.
- Formatting of the displayed data.
- Display or modification of data in hexadecimal or EBCDIC format.
- Moving and copying of data within one file or between two files.

- Entering of frequently used commands or data via PF keys.

Chapter 5, “Working with the Editor,” on page 141 and the descriptions of the available editor commands describe these capabilities in more detail.

Screen Layout and Formatting

To understand the information presented in this section, you should be familiar with the basic screen layout, the associated terminology, and the concept of logical screens and format areas.

Basic Screen Layout

The basic screen layout, the default, is displayed after you have invoked the editor by the ED macro. This screen layout is shown in Figure 3 on page 134. The file being used for this example contains only blank lines. You can deviate from the default layout by entering the VERIFY command with suitable operands.

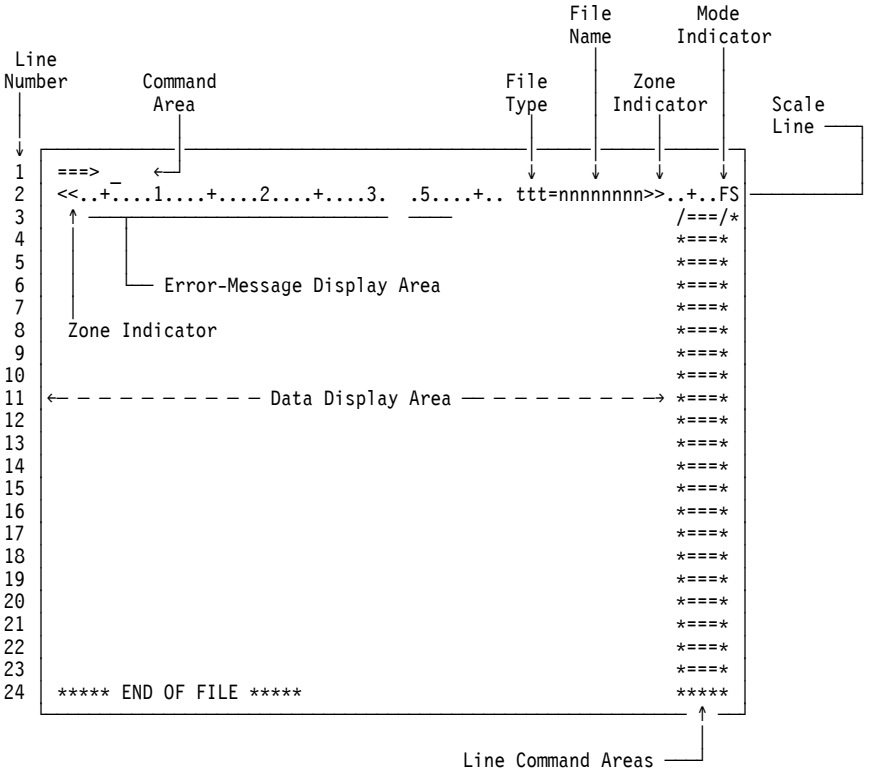


Figure 3. Basic Screen Layout

The various screen areas as shown in Figure 3 on page 134 are discussed below.

Command Area

Line 1 constitutes the command area. This area, identified by the characters ===>, normally consists of one line. Through the VERIFY command, you may expand this area to up to four lines.

Any combination of editor commands separated by logical line end characters can be entered in the command area.

Scale Line

The scale line immediately follows the command area. It contains column indicators as an aid to working with fixed format data.

It displays the following information:

- A pair of two arrow heads designating the editing zone.

- Error message, if any.

  A message issued by the editor temporarily overlay the first 52 columns of the scale line. If no error message is displayed and a display offset is in effect (see the “LEFT Command” on page 196 and “RIGHT Command” on page 212), the offset amount is displayed in the message display area.
- File type.

  The file type ('ttt' in Figure 3 on page 134) can be any of the following:

  INP
  The input area.

  MEM
  To indicate one of the following:

  A library member
  The print area $$PRINT
  The punch area $$PUNCH.

  NEW
  A file opened via the ENTER command without operand, that is, a file being newly created.
- File name.
- Mode indicator.

  Throughout the edit session, this indicator appears as FS for full-screen edit mode. If mixed data are edited, the FS indicator is preceded by DB.

Data Display Area

This is the area where the individual records are displayed. Data displayed in this area of the screen can be modified simply by positioning the cursor and typing in the new data. This way, you can alter several lines before pressing ENTER.

The **current line** record is identified by highlighting and, if the NUMBERS option is set off, the string /===/* in the line command area (see the next paragraph).

Line Command Area

Following each record displayed on the screen is a line command area. In this area, you can enter line commands to manipulate individual lines or groups of lines on the screen. Line commands are used to add, delete, duplicate, shift, move, or copy lines.

Line command areas are indicated by the strings *===*, /===/*, and /***/. You enter line commands by overlaying these characters in the line command area. Section “Editor Line Commands” on page 230 presents more details.

Logical Screens

To edit multiple files concurrently, the screen may be divided into up to eight logical screens.

Each logical screen has its own command area and its own scale line. Each logical screen displays a different file. Figure 4 on page 136 shows a screen which has been divided into two logical screens.

```
===>
<<..+....1....+....2....+....3. .5....+.. MEM=MEMBER1 >>..+..FS
RECORD  1  OF MEMBER1                                    /===/*
RECORD  2  OF MEMBER1                                    *****
RECORD  3  OF MEMBER1                                    *****
RECORD  4  OF MEMBER1                                    *****
RECORD  5  OF MEMBER1                                    *****
RECORD  6  OF MEMBER1                                    *****
RECORD  7  OF MEMBER1                                    *****
RECORD  8  OF MEMBER1                                    *****
RECORD  9  OF MEMBER1                                    *****
RECORD 10  OF MEMBER1                                    *****
===>
<<..+....1....+....2....+....3. .5....+.. MEM=MEMBER2 >>..+..FS
***** TOP OF FILE *****                                  /***/
RECORD NUMBER  1 OF MEMBER2                              *****
RECORD NUMBER  2 OF MEMBER2                              *****
RECORD NUMBER  3 OF MEMBER2                              *****
RECORD NUMBER  4 OF MEMBER2                              *****
RECORD NUMBER  5 OF MEMBER2                              *****
RECORD NUMBER  6 OF MEMBER2                              *****
RECORD NUMBER  7 OF MEMBER2                              *****
RECORD NUMBER  8 OF MEMBER2                              *****
RECORD NUMBER  9 OF MEMBER2                              *****
```

Figure 4. Screen Layout with Two Logical Screens

Details about creating and using multiple logical screens will be provided in "Editing Two Logically Related Files" on page 152.

Format Areas

For the purpose of editing different portions of one file concurrently, a logical screen may be divided into up to eight format areas.

Each format area has its own command area, but only the first format area in a logical screen has a scale line. Each format area can display a different portion of the same file. Figure 5 on page 136 shows a screen which has been divided into two format areas.

Details about creating and using multiple format areas will be provided in "Editing Two Areas of One File Simultaneously" on page 151.

```
===>
<<..+....1....+....2....+....3. .5....+.. MEM=SAMPASMB ...+..F>
MAINPGM  CSECT                                           /===/*
         PRINT GEN                                       *****
         BALR  5,0                                       *****
         USING *,5                                       *****
         OPEN FILOUT                                     *****
LOOPA    EXCP  RDCCB                                     *****
         WAIT  RDCCB                                     *****
         CLC   R(3),=C'/* '                              *****
         BE    ENDCARD                                   *****
         MVC   0(80,2),R                                 *****
===>
SAVEAREA DS    18F                                       /===/*
P        DC    CL120' '                                  *****
R        DC    CL80' '                                   *****
RDCCB    CCB   SYSIPT,RDCCW                              *****
PRCCB    CCB   SYSLST,PRCCW                              *****
RDCCW    CCW   2,R,0,80                                  *****
PRCCW    CCW   9,P,0,120                                 *****
IOA      DS    CL408                                     *****
IOB      DS    CL408                                     *****
FILIN    DTFSD IOAREA1=IOA,IOAREA2=IOB,IOREG=(2),      X *****
               TYPEFLE=INPUT,BLKSIZE=400,RECSIZE=80,   X *****
```

Figure 5. Screen Layout with Two Format Areas

Types of Editor Commands

Editor Commands

You enter editor commands in the command area. These commands either control the editor operation (screen setup or processing options, for example), or they manipulate (locate, change, add, delete, copy, move etc.) data, starting at the current line.

Editor Line Commands

Line commands offer a convenient way of applying multiple changes (adding, deleting, copying, moving of shifting of lines) with one interaction of the processing unit. These commands are entered in the line command area of the first or only affected line.

Line commands are, strictly speaking, editor commands, too. But they are special in several ways (line command area, order of processing, and so on) so that they will be discussed as a group by themselves. The line commands are described in detail in section “Editor Line Commands” on page 230.

Order of Input Processing

When you press ENTER or any PF key, the screen input is processed in the following sequence:

1. The data area is scanned for changed lines, which are recorded in the file as they are found.
2. The line command area is scanned for line commands, which are processed except for I (insert) commands.
3. If any I commands were encountered, they are processed.
4. The command area is inspected for commands.
5. If a PF key is pressed, any commands associated with this key are processed. The commands are processed as part of that format area in which the cursor is positioned when the PF key is pressed. Modifications to the cursor position via the CURSOR command are also processed in this step.

Operating Modes

In **full-screen edit** (FS) mode, all editor commands and line commands are accepted and also a subset of the system commands – if entered without the slash (/).

Certain commands that produce more than one line of output may cause the editor to temporarily leave full-screen edit mode. In such a case, the mode indicator changes to ED (context edit) or LS (list) mode. While the terminal remains in either of these modes, context editor commands or system commands are not recognized; press ENTER to return to full-screen edit mode. See also the information under “Temporarily Leaving Full-Screen Display” on page 163.

If you enter the INPUT or the INSERT command, **editor-input** mode is set. This does not change the mode indicator (FS), but the editor provides an empty data area for entering input data. In the line command areas, the string `*INPUT` is displayed.

In editor-input mode, you may still use editor and system commands as in full-screen edit mode. However, as soon as you press ENTER or any PF key, the editor leaves input mode and adds the entered data to the file. Then all commands found in the command areas or entered via PF key are executed in the normal sequence.

Character Translation

Normally, character data is entered from the terminal as lowercase data. By default, VSE/ICCF translates this data to uppercase. If, however, you enter the 'CASE M' command, mixed-case data is stored as it is entered.

Note: The characters X'FF' and X'00 through X'3F' may represent screen control characters. VSE/ICCF converts them to an internal edit control character before it sends them to the screen. Currently, this control character is X'E1'; this character may change if there is a need.

Characters in the range from X'40' to X'FE' are not converted; VSE/ICCF sends them to the screen unchanged. This applies also to those of these characters which are unprintable (because they are undefined according to the I/O interface code defined for the involved terminal).

Normally, object data or other unprintable data stored in a library remains unchanged. *However*, unprintable data in a library may be destroyed nevertheless. Some control units (the IBM 3272, for example, or the IBM 3274 without Configuration Support C, D, or T) modify unprintable characters if these characters are undefined I/O interface codes.

If you do not change any of the data displayed for a library line, this data remains in the library as is. To ensure that no data is destroyed if you have to make a change, do this change in hexadecimal format only (for example by OVERLAYX).

String Arguments

Several of the editor commands require arguments called string arguments. A string argument is either matched against strings or replaces a string in the text.

A string argument may or may not begin with a delimiter character (usually a slash (/) unless changed by the DELIM command). The only commands that require the delimiter character are the CHANGE command and the string version of the DELETE command.

A string that does not begin and end with the delimiter character is assumed to begin with the first character. This includes space characters, but not the one that separates the command from the string. The string is considered to end with the last non-space character on the line.

If a string begins or ends (but not both) with the delimiter character, the character is assumed to be part of the string.

Using Special Control Keys for Editing

The keyboard of an IBM 3270 has special keys that cause certain signals to be passed to the editor. The following keys have a special function when used within the editor:

CLEAR
By pressing this key, you cause the screen to be reformatted. The screen then presents the same display as when ENTER was pressed last, except that the cursor will be positioned at the beginning of the command area. Any commands entered on the screen or changes made to lines following the last ENTER will be lost.

ERASE INPUT
Pressing this key has the same effect as pressing CLEAR. The only exception is that a new display does not appear unless you press ENTER.

ENTER
Pressing ENTER requests the editor to read the data entered on the screen and to process any changed lines and any commands read from the screen.

FIELD MARK
This key inserts a logical tab character into the data stream (unless the logical tab character has been set to some other character via the SET TAB= command).

PA1, PA2, PA3
The functions assigned to these keys depend on how your system is tailored. In the VSE/ICCF system that is delivered to you, they have the following meaning:

PA1
No function in VSE/ICCF but may be used as print key in CICS Transaction Server.

PA2
This key forces an editor CANCEL command to be executed. All files being edited are handled as if a QUIT command had been issued. The editor terminates. Automatic saves on newly created files will not be done; changes applied on the screen will not be processed.

PA3
Same as the CLEAR key.

PF1 through 12
(for some keyboards 24). The PF keys can be set to any command, data line or multiple commands separated with logical line end characters. When a PF key is pressed, all the ENTER key functions are performed first. That is, the editor reads from the screen and processes all changed lines and commands. Then the functions associated with the PF keys are executed. For more information on PF keys and how they are used, see the descriptions of the following commands:

- "The PF[ED|EX|LS] Feature" on page 110
- "[/] PFnn Command" on page 80

Entering Multiple Commands

Multiple editor commands can be typed into the command area. They must be separated by the logical line end character as set by `/SET END=char`, see page "END=char" on page 103 (examples in this publication use the semicolon as logical line end character). You can enter as many commands as fit into the area.

Editor commands that result in leaving the current logical screen, for example: SCREEN, FILE, and QUIT, should not be followed by other commands. When leaving the full-screen editor, any command chained to such commands will be ignored.

Repeating Commands

A command (or a series of commands separated with logical line end characters) can be repeated any number of times simply by preceding the command with an ampersand ('&'). For example, if '&FORWARD' is entered in the command area, the FORWARD command will remain on the screen allowing you to 'scroll' through the file by pressing ENTER.

Chapter 5. Working with the Editor

Invoking the Editor

You invoke the editor through one of three macros:

ED [member [passwd]]
To edit a VSE/ICCF library member or the input area.

EDPRT
To edit the print area, $$PRINT.

EDPUN
To edit the punch area, $$PUNCH.

Consider assigning frequently used commands to PF keys; you can set and change PF keys anytime. However, the most practical way is to include such settings in a logon routine (normally a macro) which is run automatically at logon time. To achieve this, do the following:

1. If not done already, ask your administrator to enter the name of your logon macro into your user profile (via the LOGONRTN operand of the DTSUTIL command ADD USER or ALTER USER).
2. Create your logon macro as follows:

```
ED
INPUT
@MACRO
...
...                          (Other commands and required)
...
/SET PF1ED  command1         (These PF key settings will be
/SET PF2ED  command2         effective in edit mode only)
...
/SET PF12ED  command12
                             (A null line to end input mode)
FILE name                    (Use the same name as in user profile)
```

Creating a New File

A new file is created by entering the data into the input area and then saving the contents of this area as a VSE/ICCF library member. After having invoked the editor (to edit the input area, just enter ED without a member name), you will see the screen as shown in the upper part of Figure 6 on page 142.

```
===> input_
<<..+....1....+....2....+....3. .5....+.. INP=*INPARA*>>..+..FS
***** TOP OF FILE *****                                  /***/
***** END OF FILE *****                                  *****
```

```
===> file newdata_
<<..+....1....+....2....+....3. .5....+.. INP=*INPARA*>>..+..FS
***** TOP OF FILE *****                                  /***/
data record 1                                           *INPUT
data record 2                                           *INPUT
data record 3                                           *INPUT
data record 4                                           *INPUT
data record 5                                           *INPUT
data record 6                                           *INPUT
data record 7                                           *INPUT
data record 8                                           *INPUT
 ...                                                    *INPUT
 ...                                                    *INPUT
 ...                                                    *INPUT
last data record                                        *INPUT
                                                        *INPUT
                                                        *INPUT
```

Figure 6. Creating a New File

The INPUT command formats the screen for full-screen data input as shown in the second half of Figure 6 on page 142. When entering the data records, use the NEW LINE key to start a new record. After the last record, type the FILE command as shown in the figure, and press ENTER. The editor will save the data as member NEWDATA and terminate.

All data lines are initially null lines (X'00'). When you press ENTER to terminate input mode, any remaining null lines are discarded. If you want to enter a blank line, you must type at least one blank (via the space bar) and press ENTER.

Intermediate Saving of a File

Make sure to save your data input, especially when you enter large amounts of data. Proceed as follows:

1. Issue a 'SAVE name' command after having entered at least one record. This will save your data as a library member. For the rest of the session, you will be adding data to that member, not to the input area. This is a precaution against loss of your data in case of a system breakdown.
2. After the initial SAVE, use &INPUT instead of INPUT. Thus, if you press ENTER after typing one screen full of data, your data is written to the file, and the editor returns to editor-input mode immediately (the data area is cleared to receive the next batch of data records).
3. When having entered all data, purge the &INPUT command from the command area, and press ENTER.
4. If your data has been saved before, terminate the session via the QUIT command. Otherwise, use FILE as described in the previous section.

Displaying and Changing a File

Most of the data manipulation operations can be carried out in two ways: through editor commands or through line commands. Although line commands can handle up to 999 records at a time (only 99 in case of copy or move operations), their main use lies in handling smaller portions of data which start and end on one screen.

In contrast to line commands, the editor commands start working from the current line, which often requires additional commands for exact line positioning. On the other hand, many editor commands and macros accept character string operands to denote the last record to be handled. Thus, you can avoid counting the lines down to the end of the area to be processed by a command.

Positioning the Line Pointer

Associated with each format area is a **line pointer**. Its setting determines which line is displayed as the current line (the first line below the scale line, unless the VERIFY command specified otherwise).

The editor maintains a 'pseudo line' at the top of the file:

```
***** TOP OF FILE *****
```

This line is displayed as the current line when the editor is invoked or after you have entered the TOP command. With the line pointer at this position, you can insert data before the first record of your file. With other commands (for example DELETE), the TOP OF FILE line is ignored.

The current line is the starting point for editor commands that modify, copy or move data.

Search-type commands and also editor commands that insert data lines, start from the line after the current line. An exception is LOCUP, which starts at the line before the current line.

If a search-type operation finds the desired string, the line containing it becomes the current line; if the string is not found, the line pointer is set at the last record in the file.

On input type operations, the last line entered becomes the current line. On line deletion, the line after the last deleted line becomes the current line.

To position the line pointer for viewing and editing a certain portion of a file, use the editor commands listed in Figure 7 on page 143.

Editor Commands	Function
Bottom	Positions the line pointer past the last line in the file.
Top	Positions the line pointer to the null line in front of the first line in the file.
FOrward	Scrolls the display forward by the specified number of pages.
BAckward	Scrolls the display back by the specified number of pages.
Next	Advances the line pointer a given number of lines.
Up	Repositions the line pointer a given number of lines before the current line.
Find	Searches the file for a string, to be found at a specific column.
Locate	Searches the file for a string, to be found at any column (unless a column is specified).
LOCUp	Searches the file backward for a string, to be found at any column (unless a column is specified).
LOCNot	Searches the file for the first non-occurrence of a given string.
ENTER key	Pressing ENTER is equivalent to entering NEXT 1.
Line Commands	
/	Positions the line pointer to the line in which the / was given.

Figure 7. Vertical Line Positioning Commands

Limiting Editor Operations to Certain Columns

Many applications require that certain change, shift or align operations are carried out only within a certain column range, the *editing zone*. The primary means to set this column range is the ZONE

command, which defines a start and an end column for subsequent editing operations. Data outside of that column range will be ignored by search-type commands and will not be affected by any change operation. Also, full-screen changes (any new data that you typed in and any deletion of characters) outside of the editing zone are ignored once you press ENTER. They will not result in a library member being changed.

In addition, a so-called *Cnn suffix* can be attached to many commands. The suffix defines column nn as the starting (left) zone column. This definition is valid only for one execution of the command. The Cnn suffix may be attached to any valid command abbreviation.

The examples in the following section will show the effects of zone setting and the Cnn suffix in the context of data manipulation operations. Figure 8 on page 144 gives an example of how to use the Cnn suffix. In the example, the suffix puts the missing continuation character into column 72 of the second line of the DTFSD macro call.

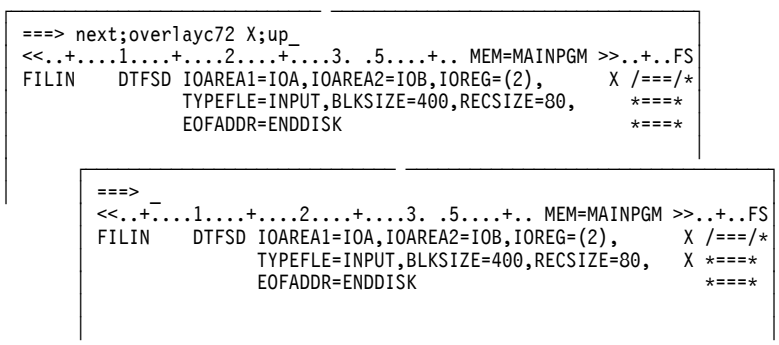


Figure 8. Using the Cnn Suffix

Global Changes

You can make global, or repetitive changes, with the CHANGE and ALTER commands. Operands of these commands allow you to specify:

- The number of lines to be searched for a character or character string. An asterisk (*) indicates that all lines, from the current line to the end of the file, are to be searched.
- Whether only the first occurrence or all occurrences on each line are to be modified. The letter 'G' indicates all occurrences. If you do not specify the letter 'G', only the first occurrence on any line is changed.

For example if you are creating a file that uses the special character [(X'AD') and you do not want to use the ALTER command each time you need to enter this character, you can use the character $ as a substitute each time you need to enter [. When you are finished entering input, move the line pointer to the top of the file, and issue the global ALTER command as shown:

```
alter $ ad * G
```

This causes all occurrences of the character $ to be changed to X'AD' and the line pointer to be positioned at the end of the file.

When you use a global CHANGE command, you must be sure to use the final delimiter on the command line. Example:

```
change /hannible/hannibal/ 5
```

This command changes the first occurrence of the string 'HANNIBLE' on the current line and the next four lines. You can also make global changes with the OVERLAY command, by issuing a REPEAT command just

prior to the OVERLAY command. Use the REPEAT command to indicate how many lines you want to be affected. For example, if you are editing a file containing the three lines

```
A
B
C
```

with the line pointer at line A, issuing the commands:

```
repeat 3
overlay   I     I     I
```

results in

```
A   I     I     I
B   I     I     I
C   I     I     I
```

The line pointer is now at the line beginning with the character C.

Figure 9 on page 145 summarizes the commands that are used for making global changes.

Editor Commands	Function
Change	Changes one string of characters to another.
Alter	Changes one character to another, or references a character by its hexadecimal value.
REPEAT	Causes the following OVERLAY or BLANK command to execute a given number of times (also used with the same effect on the commands ALIGN, CENTER, JUSTIFY, and SHIFT; see the following figure).
Overlay	Overlays a string of characters in the current line by the specified string.
BLank	Blanks out characters in the current line.
RENum	Generates new sequence numbers in the file being edited.

Figure 9. Global Change Commands

Shifting Data Within Lines

Figure 10 on page 146 summarizes the commands that are used for shifting data within a line. Figure 11 on page 146 shows how the REPEAT and CENTER commands are used together.

Editor Commands	Function
REPEAT	Causes the following ALIGN, CENTER, JUSTIFY, or SHIFT command to execute a given number of times (also used with the same effect on the commands OVERLAY and BLANK; see the preceding figure).
ALIgn	Aligns (justifies) text to both the left and the right margin.
CENter	Centers text as determined by the current zone setting or by the column suffix.
JUStify	Aligns (justifies) text to either the left or the right margin.
SHIft	Shifts the data within the current zone left or right the specified number of columns.

Line Commands	Function
TA	Aligns (justifies) text to both the left and the right margin, in one or more consecutive lines.
TC	Centers text, in one or more consecutive lines.
TL	Aligns (justifies) text to the left margin, in one or more consecutive lines.
TR	Aligns (justifies) text to the right margin, in one or more consecutive lines.
>	Shifts text to the right, in one or more consecutive lines.
<	Shifts text to the left, in one or more consecutive lines.

Figure 10. Shift Commands

```
===> zone 1 30;repeat 2;center;top_
<<..+....1....+....2....+....3. .5....+.. INP=*INPARA*>>..+..FS
***** TOP OF FILE *****                                  /***/
VSE/ICCF                                                *****
TERMINAL USER'S GUIDE                                   *****
***** END OF FILE *****                                  *****
```

```
===> _
<<..+....1....+....2....+...>>. .5....+.. INP=*INPARA* ...+..FS
***** TOP OF FILE *****                                   /***/
           VSE/ICCF                                      *===*
    TERMINAL USER'S GUIDE                                *===*
***** END OF FILE *****                                   *****
```

Figure 11. Use of REPEAT and CENTER Commands

Adding or Deleting Lines

Figure 12 on page 147 gives a summary of the commands that you can use to add or delete one or more lines.

Editor Commands	Function
LAdd	Adds blank lines to the file.
Insert	Inserts a string as a new line in the file.
INPut	Places the terminal into editor input mode.
DELete	Deletes lines from the file.
Line Commands	**Function**
A	Adds blank lines to the file.
D	Deletes lines from the file.

Figure 12. Add and Delete Commands

Copying or Moving Lines

Figure 13 on page 147 summarizes the commands that are used for copying and moving one or more lines.

Editor Commands	Function
DUP	Duplicates the current line the specified number of times.
@COPY	Copies lines within a file.
@MOVE	Moves lines within a file.
GETfile	Inserts data from a library member, or a work area, into the file being edited.
Line Commands	**Function**
"	Duplicates the line the specified number of times.
C	Copies lines into the editor stack. The data already in the stack is erased before the new data is stored.
K	Copies lines into the editor stack. The lines are placed behind the data that is already there.
M	Moves lines into the editor stack, deleting the lines in the file. The data already in the stack is erased before the new data is stored.
I	Inserts the lines indicated by either the 'C' command or the 'K' command after the line where 'I' is entered.

Figure 13. Copy and Move Commands/Macros

Using the Line Command Area

The line command area lets you easily move and copy small amounts of data from one place to another in the same file, or in different files. You can also collect lines from one or more screens (which may contain data from different files) for insertion somewhere else in the same, or in different, files.

The line commands that you need for these operations are:

M
for move

C
for copy

K
for collect or stack

I
for insert

Moving and Copying on the Same Screen

To move or copy data from one place to another on the same screen, flag each line with an M or C command in the line command area. If multiple contiguous lines are to be moved or copied, you need not flag each line. Instead, flag only the first and follow the 'M' or 'C' command with the number of lines to be moved or copied.

The point at which the moved or copied lines are to be inserted is indicated by the I (insert) command in the line command area on the line after which the data is to be inserted.

You can move or copy data upward or downward on the screen. Multiple I commands on the screen cause the lines to be moved or copied to be inserted at each of the specified line positions.

A move command deletes the data indicated by M after the data has been inserted.

Moving or Copying Data From One Screen to Another

To move or copy data from one screen to another, follow the instructions in the preceding paragraphs. However, instead of placing the 'I' command in a line command area on the current screen, simply find the area (perhaps the same file or a different file) in which the data is to be inserted such that the point of insertion appears on the screen. Then just enter the 'I' command on the line after which the data is to be inserted.

Collecting Data From Several Screens for Insertion

It is also possible to collect data from several screens for insertion at a place on some other screen. To accomplish this, do the following:

1. Use the C (or M) command on the first screen containing data to be copied. This causes the stack area to be opened (for more information about the stack area, see the section “Using the Editor Stack” on page 149).
2. Use the K command on the other screens containing data to be copied. The command causes the new data to be added behind the existing data in the stack area.
3. When the point in the file is reached where the collected data is to be inserted, use the 'I' command to force the insertion.

Inserting Data via GETFILE

The GETFILE command allows you to copy all or a portion of a library member into any part of the file you are editing. For example:

```
getfile single
```

inserts all the lines of member SINGLE into your file immediately following the line pointer. When this copy operation is complete, the line pointer is at the last line that was copied into your file.

As another example, you could specify

```
getfile double 10 25
```

to copy 25 lines, beginning with the tenth line, from the file named DOUBLE.

GETFILE can be used also to copy data from the stack, or from another part of the file you are editing.

Another way of copying small amounts of data is to use the STACK command which is explained in the next section.

Using the Editor Stack

When you are in full-screen edit mode, the punch area ($$PUNCH) associated with your terminal is used for certain editor functions. In this context, the punch area is referred to as the stack. Also, the name $$STACK is synonymous with $$PUNCH.

The stack serves to temporarily store data. You store data in the stack by various means:

- The STACK command.

  This command stores, in the stack, any of the following:

  – Records from the member being edited.

  – Editor settings such as CASE or ZONE

    The STACK EDIT command is used together with the RESTORE command. The STACK EDIT command saves editor settings such as message and verification display, line-number and control-character settings, or zone setting. When you are editing a file and you want to temporarily change some of these settings, enter the STACK EDIT command to save their current status. Then, when you have finished editing, issue the RESTORE command to restore the original settings.

    Note: Make sure that a STACK EDIT command does not interfere with a previous move/copy operation, or vice versa. For example, if you had copied data into the stack and then issued STACK EDIT, the data could be erased before you had retrieved it.

  – Any commands or data specified in the STACK command.

    A series of commands which you wish to repeat could be placed in the stack as a temporary macro. By specifying @$$STACK as a command name, you cause the commands in the stack to be processed. If the commands in the stack include parameters, these parameters are filled in from the operands of the @$$STACK macro call.

    The commands in the stack may also be system commands.

    To copying data, proceed as follows:

    1. Set the line pointer in the source file to the first line to be copied.
    2. Open the stack by entering 'STACK OPEN'.
    3. Issue the STACK nn command; up to 99 lines can be placed in the stack.
    4. After all lines to be copied have been stacked, close the stack with STACK CLOSE.
    5. Set the line pointer in the receiving file to the line after which the copied lines are to be inserted.
    6. Enter GETFILE $$STACK to copy the data to the target location.

- The CHANGE or the ALTER command.

  During global change operations, these commands allow the changed lines to be stored in the stack. When the global changes are done, you can issue the LIST $$STACK command to view the changes.

- Most of the commands listed in Figure 13 on page 147.

  The data to be copied or moved within a file or between files is temporarily kept in the stack area. For example, the first M or C command opens the stack and stores the data to be moved or copied in the stack. All later M or C commands add their lines to the stack. The next I command causes the stack to be closed and the data in the stack to be inserted behind the line indicated by the line pointer.

  You cannot stack more than 99 records with one C or M command. However, each additional K command will extend the stack by another 100 records, if required. The @COPY and @MOVE macros, on the other hand, can never handle more than 99 records.

  The stacked data is preserved until new data is written into the stack. Thus, after stacking some data either via @COPY/@MOVE or one of the commands C, K, M, or STACK, you may issue any number of 'I' commands to insert that data at different points of the same or another file. You may, in fact, end the edit session and edit a different member and still refer to the previously stacked data.

Because the $$STACK area is physically the same as the $$PUNCH area, you may use the editor to view and process punch output created by the execution of a program in an interactive partition. Although you may edit (that is: change) the $$STACK area, you cannot add or delete lines in this area.

Note: Do not use the stack when working in asynchronous mode because this may interfere with the execution of your program in the interactive partition, that is, do not use any STACK or move/copy commands.

Using Tab Stops for Column-Oriented Editing

For column-oriented editing, you can use a programmed equivalent of the tabulator stops that are available on an ordinary typewriter. Using this facility is a two-step process:

1. Set the tab stops at the desired columns (as you would do on a typewriter) by using the editor command TABSET.
2. Define a logical Tab character and set two PF keys.

   The logical Tab character is used during command and data input to represent the number of blanks needed to fill the line up to the next Tab stop.

   The PF keys are set to the editor commands CURSOR TABBACK and CURSOR TABFORWARD to help you position the cursor in the desired column for full-screen update.

In Figure 14 on page 150 you see how the Tab characters found in the input data are expanded into the correct number of blanks when you press ENTER to end full-screen data input. It is assumed that the semicolon has been defined previously as line end character.

The Tab characters work the same way when entered on blank lines created by the LADD or the 'A' commands.

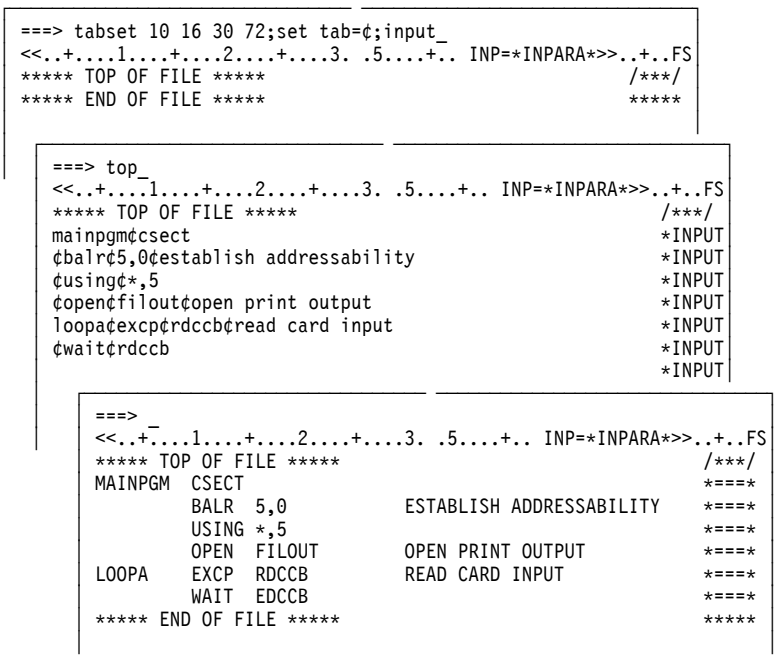


Figure 14. Using Tab Stops for Column-Oriented Input

As Figure 15 on page 151 shows, the Tab characters may also be used within an editor command to ensure proper placement of the operand data (Tab and END character settings are the same as in Figure 14 on page 150).

For the purpose of interpreting Tab characters in the command line, the second byte after the command (INSERT in this case) is counted as column 1 of the data string operand.

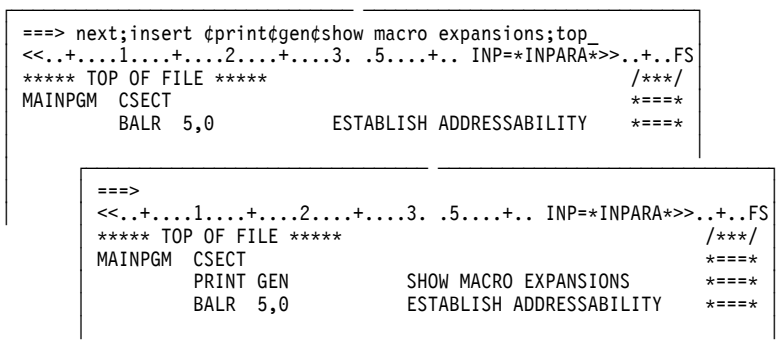


Figure 15. Using Tab Stops in Editor Commands

Editing Two Areas of One File Simultaneously

While studying or changing a program, you may want to view or edit two different portions of a file at the same time.

The FORMAT command allows you to divide a logical screen into multiple format areas as shown in Figure 16 on page 152. Thus you can independently edit a given file at different points. However, the figure cannot show how you flip the cursor from one command area to the other. This is done most easily by setting a PF key equal to the command 'CURSOR INPUT'. This command puts the cursor into the next (or only) command area on the screen.

Assume that you want to know how the field RDCCB has been defined. To do this:

1. Enter the FORMAT command as shown.
2. Press ENTER.
3. Press the PF key that was set to CURSOR INPUT.

VSE/ICCF now formats the screen as shown in the second screen of Figure 16 on page 152 and places the cursor into the second command area where you enter your `find rdccb` command. After you have pressed ENTER, the second format area will show the desired portion of your file (see the third screen of Figure 16 on page 152).

If you want to return to a screen format with only one format area, just enter `format 1`.

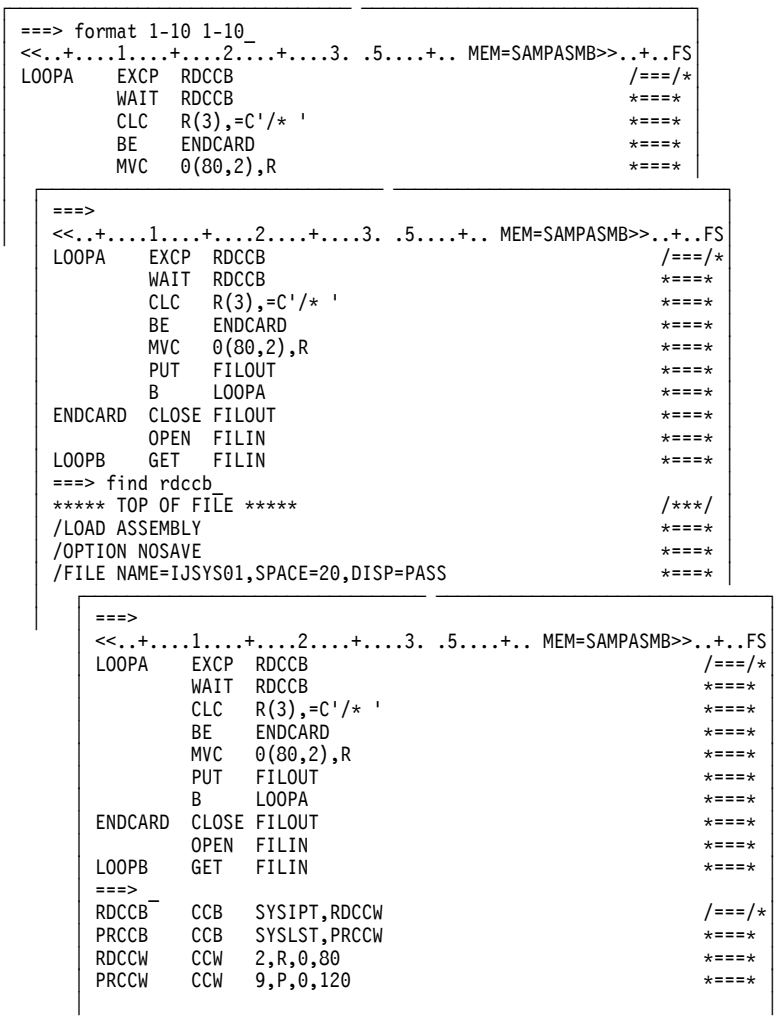


Figure 16. Using the FORMAT Command

Editing Two Logically Related Files

You may experience situations where you want to copy various pieces from one file to another. The editor offers two aids: recursive editing and split-screen editing.

Recursive Editing

Recursive editing means alternate editing of two files within one edit session. This is the preferred way of editing if large blocks of data are to be copied. Refer to the example in Figure 17 on page 153 and in Figure 18 on page 153.

While editing member NEW, you start a new edit session for member OLD by issuing the command ENTER OLD. This lets member NEW disappear from the screen, but the edit session is kept "alive", that is, the current line position and setting of edit options are maintained. With the full-screen available for the display of member OLD, it is easy to locate the data to be copied and to count the number of involved lines. After you have defined the data to be copied (`c14` in the line command area), an `enter new` command brings you back to the first member. Here you insert the data from the stack (by `i` in the line command area).

```
===> enter old
<<..+....1....+....2....+....3. .5....+.. MEM=NEW     >>..+..FS
         DC    C'IJSYS01'                                /===/*
         ORG                                             *****
         END                                             *****
```

```
===>  F *******
<<..+....1....+....2....+....3. .5....+.. MEM=OLD     >>..+..FS
***** TOP OF FILE *****                                  /***/
OLDPGM   CSECT                                           *****
         PRINT GEN           SHOW MACRO EXPANSIONS       *****
         BALR  5,0           ESTABLISH ADDRESSABILITY    *****
         USING *,5                                       *****
         OPEN  OUTPUT        OPEN DISK  OUTPUT           *****
LOOPA    EXCP  RDCCB         READ CARD INPUT             *****
         WAIT  EDCCB                                     *****
INPUT    DTFSD IOAREA1=IOA,IOAREA2=IOB,IOREG=(2),      X *****
               TYPEFLE=INPUT,BLKSIZE=400,RECSIZE=80,   X *****
               EOFADDR=ENDDISK,RECFORM=FIXBLK            *****
         ORG   FILIN+22                                  *****
         DC    C'IJSYS01'                                *****
         ORG                                             *****
OUTPUT   DTFSD IOAREA1=IOA,IOAREA2=IOB,IOREG=(2),      X *****
               TYPEFLE=OUTPUT,BLKSIZE=408,RECSIZE=80,  X *****
               RECFORM=FIXBLK                            *****
         ORG   FILOUT+22                                 *****
         DC    C'IJSYS01'                                *****
         ORG                                             *****
******************************************************** *****
*                                                      * *****
```

Figure 17. Alternate Editing of Two Files (ENTER OLD)

```
===> enter new_
<<..+....1....+....2....+....3. .5....+.. MEM=OLD     >>..+..FS
******************************************************** c14=/*
*              I N P U T  R E C O R D  L A Y O U T    * *****
******************************************************** *****
INPUT    DSECT                                           *****
KEY      DS    C                                         *****
PNR      DS    ZL6                                       *****
NAME     DS    CL20                                      *****
ADDR     DS    CL20                                      *****
WAGES    DS    ZL6                                       *****
HRS      DS    ZL4                                       *****
DEDUCT   DS    ZL4                                       *****
PAY      DS    ZL6                                       *****
```

```
===>
<<..+....1....+....2....+....3. .5....+.. MEM=NEW     >>..+..FS
         DC    C'IJSYS01'                                /===/*
         ORG                                             i_==*
         END                                             *****
```

```
===> _
*MSG=*END INSERT    ....+....3. .5....+.. MEM=NEW     >>..+..FS
         DC    C'IJSYS01'                                /===/*
         ORG                                             *****
******************************************************** *****
*               I N P U T  R E C O R D  L A Y O U T    * *****
******************************************************** *****
INPUT    DSECT                                           *****
KEY      DS    C                                         *****
PNR      DS    ZL6                                       *****
NAME     DS    CL20                                      *****
ADDR     DS    CL20                                      *****
WAGES    DS    ZL6                                       *****
HRS      DS    ZL4                                       *****
DEDUCT   DS    ZL4                                       *****
PAY      DS    ZL6                                       *****
         END                                             *****
```

Figure 18. Alternate Editing of Two Files (Return to NEW)

Creating a New File through Recursive Editing

If the name specified in the ENTER command is not a member of your library, VSE/ICCF assumes that you want to create a new file.

When you create a new file, you can add lines to it in a number of ways; for example with the INPUT, LADD and GETFILE commands. At this point the file is not a library member.

The data that you enter is in a simulated input area; it will be lost if you enter a QUIT or CANCEL command.

If you decide to retain this data in the library as a member, issue the SAVE or the FILE command. In this way you can build several new library members without leaving edit mode.

Split Screen Editing

Use this way of editing if simultaneous viewing of two files is important. The following example, although complicated, shows a typical application as it may occur in the process of debugging a program.

Assume you have entered an assembler program and your first assembly run shows assembly errors (with `/option list`, you get the list of errors displayed on your screen at the end of the run). However, you cannot remember all of the errors while browsing through the list display. So what you need is a means of looking up the diagnostics in the listing one by one while editing the program source and correcting the errors.

Figure 19 on page 155 shows how the screen is first divided into two logical screens (one for each file) and how the second logical screen is divided into two format areas. Because the number of data lines for the second format area is not specified, the editor takes whatever is available on that logical screen.

The two format areas are needed to read the error messages in the 'DIAGNOSTICS AND STATISTICS' section and to inspect the failing instructions themselves. The second part of Figure 19 on page 155 shows the various steps of the process. On the last screen, the failing instruction has been located in the program source member, and the instruction may now be corrected.

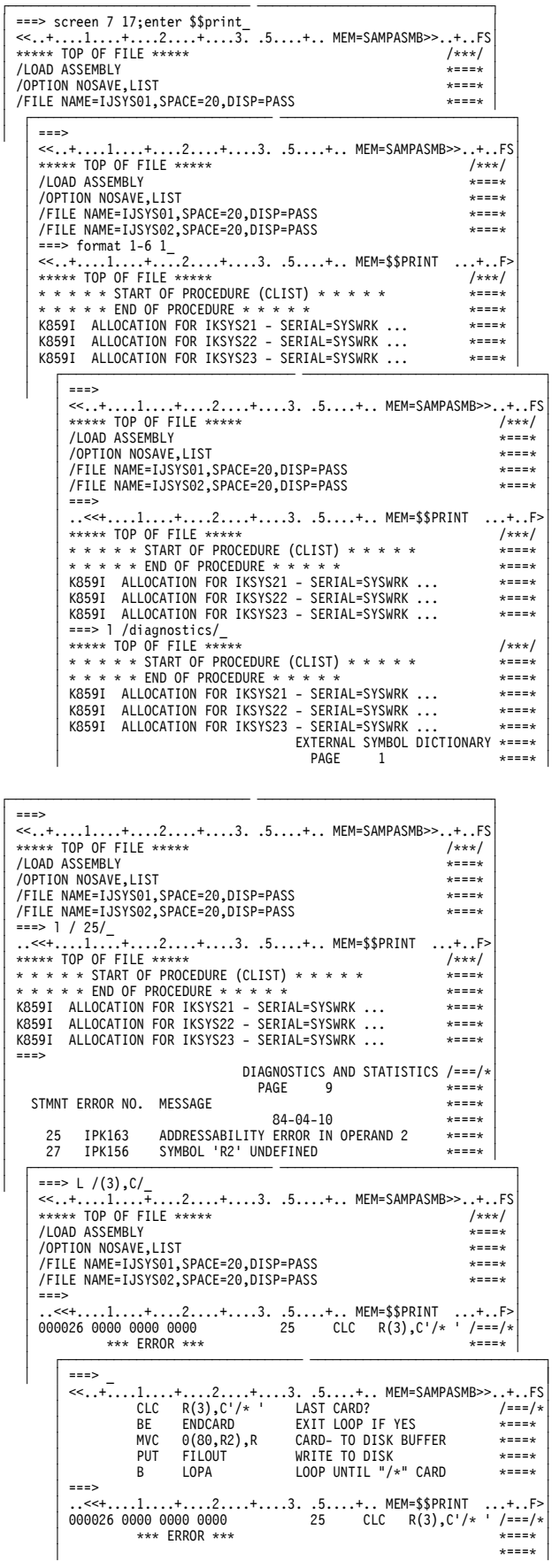


Figure 19. Simultaneous Editing of Two Files Using Three Format Areas

Saving Your Data and Ending the Session

You can save your data and end an edit session in various ways:

- SAVE or REPLACE command

  These commands store the new or changed data into a VSE/ICCF library member. REPLACE places the data into an existing member; SAVE creates a new member.

  The editor remains at the current edit level.

- FILE command

  This command performs a SAVE followed by a QUIT.

- QUIT or END command

  These commands return your screen to the next higher edit level. Data created as a NEW member is lost.

- Press PA2 or issue the CANCEL command

  Either action ends the edit session regardless of the number of editing levels being active. Data created as a NEW member (via the ENTER command) is lost.

  Note: A PA key other than PA2 may have been selected as the Cancel key by your VSE/ICCF administrator.

Special Applications and Techniques

Difficult Global Changes

Figure 20 on page 156 shows how the ZONE command can help to perform difficult global changes.

The file in Figure 20 on page 156 contains records of similar format. However, the district code, a three-part number at the beginning of each record, has leading zeros in some records and no leading zeros in others.

The objective is to have the leading zeros removed such that all the records appear properly aligned. Because pairs of zeros are also found at other places in the file, you must use the ZONE command to limit the change to the first two columns.

A simple change command does not achieve the desired result because the zone setting prevents the rest of the record from being shifted left. This is shown in Figure 20 on page 156.

Figure 21 on page 157 shows how you can bypass that problem. The first change converts the zeros to some other unique characters (?? in the example). After that, you do not need the zone restriction any more; you use a second change command to eliminate those characters.

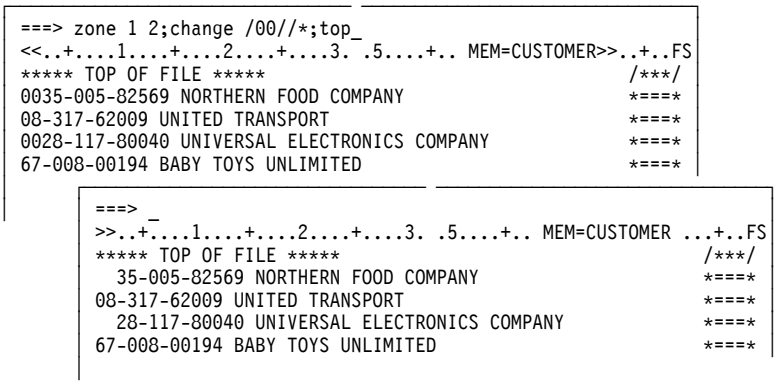


Figure 20. Incorrect Application of the ZONE Command

```
===> zone 1 2;change /00/??/*;top;zone 1 *;change /??//*;top_
<<..+....1....+....2....+....3. .5....+.. MEM=CUSTOMER>>..+..FS
***** TOP OF FILE *****                                  /***/
0035-005-82569 NORTHERN FOOD COMPANY                     *****
08-317-62009 UNITED TRANSPORT                            *****
0028-117-80040 UNIVERSAL ELECTRONICS COMPANY             *****
67-008-00194 BABY TOYS UNLIMITED                         *****
***** END OF FILE *****                                  *****

    ===> _
    <<..+....1....+....2....+....3. .5....+.. MEM=CUSTOMER>>..+..FS
    ***** TOP OF FILE *****                                  /***/
    35-005-82569 NORTHERN FOOD COMPANY                       *****
    08-317-62009 UNITED TRANSPORT                            *****
    28-117-80040 UNIVERSAL ELECTRONICS COMPANY               *****
    67-008-00194 BABY TOYS UNLIMITED                         *****
    ***** END OF FILE *****                                  *****
```

Figure 21. Use of the ZONE Command for Complex Global Changes

Line-Number Editing

After a file has grown to a certain size, locating a desired record may become difficult; similar records may exist elsewhere in the file, or you just cannot remember whether the desired record is up or down relative to the current line. Because LOCATE operations are relatively slow on large members, use line number editing wherever possible.

Line-number editing means that sequence numbers in certain columns of your file are used to locate the desired record. You enter the line number of the record and the editor will search the file (in forward direction) for that line number. Line-number editing can be of advantage when you know the number of the line you are looking for (for example, when you have a listing with line numbers available).

If the desired line number is smaller than the current one, the search will begin at the top of the file. However, if your file is very large and the line to be searched for is only a few lines above the current line, it may be quicker to use the UP, BACKWARD, or LOCUP command to go backward in the file.

Figure 22 on page 158 shows you how to get started with line number editing. If you issue the LINEMODE command as suggested in the figure, the editor automatically generates line numbers during data input. The PROMPT command may be used to set the line number increment to any value. Later, when applying changes to the member, the LINEMODE command tells the editor in which columns to find the line number if a line is to be located by number.

For the most frequently used settings, you need not specify the exact column numbers: LINEMODE RIGHT automatically selects columns 73 through 80; LINEMODE LEFT selects columns 1 through 5.

Note: When entering data with 'LINEMODE LEFT', you must use tab setting and logical tab characters (see “Using Tab Stops for Column-Oriented Editing” on page 150) to get your data moved to column 6; otherwise, the generated line numbers will overlay the first five columns of your data. As another help, consider setting two PF keys equal to CURSOR TABFORWARD and CURSOR TABBACK, respectively.

If you want to add line numbers to an existing member, issue the LINEMODE command, followed by a RENUM command. The RENUM command creates or updates the line numbers within the columns defined by the LINEMODE command.

When you work with 'LINEMODE RIGHT', the line command area would hide your line numbers. To avoid this inconvenience, issue SET NUMBERS ON, and you get columns 75 through 80 of the line numbers displayed in the line command area.

Note: With 'NUMBERS ON', you can still use line commands; however, they must be entered starting in column 1 of the line command area, and they must be followed by a blank.

Setting NUMBERS ON can be utilized with COBOL programs, too. The commands:

```
linemode 1 6             (Establish line number editing)
view 7 80                (Display columns 7-80 of each record
                          beginning at column 1 of the screen)
```

```
     set numbers on           (Displays the line numbers in the
                               line-command columns of the screen)
```

cause the body of the COBOL statement to be displayed at the left of the screen. You can then modify lines without having to move the cursor across the sequence number. SET NUMBERS ON causes the sequence number from the line number columns (rather than from columns 73-80 of each record) to be displayed in the line command area.

Figure 23 on page 159 illustrates the use of the editor command 'nn'. On the first screen, line number 70 is to be located. The second screen shows how the 'nn' command is used to add a line. The figure also shows how you can use the PROMPT command: numbers are assigned to the two inserted lines (`i print ' '`) with increments of 3.

```
===> tabset 1 6;set tab=¢;linemode left;input_
<<..+....1....+....2....+....3. .5....+.. INP=*INPARA*>>..+..FS
***** TOP OF FILE *****                                  /***/
***** END OF FILE *****                                  *****
```

```
===> top_
....+<<..1....+....2....+....3. .5....+.. INP=*INPARA* ...+..F>
***** TOP OF FILE *****                                   /***/
¢print 'enter three numbers separated with commas'        *INPUT
¢print 'enter /* to terminate'                            *INPUT
¢l = i + j + k                                           *INPUT
¢m = I * &sqr2                                           *INPUT
¢n = j * &pi                                             *INPUT
¢print i,j,k                                             *INPUT
¢print l,m,n                                             *INPUT
¢print 'end of calculation'                              *INPUT
¢go to 10                                                *INPUT
¢end                                                     *INPUT
```

```
===> _
....+<<..1....+....2....+....3. .5....+.. INP=*INPARA* ...+..F>
***** TOP OF FILE *****                                   /***/
00010PRINT 'ENTER THREE NUMBERS SEPARATED WITH COMMAS'    *****
00020PRINT 'ENTER /* TO TERMINATE'                        *****
00030L = I + J + K                                       *****
00040M = I * &SQR2                                       *****
00050N = J * &PI                                         *****
00060PRINT I,J,K                                         *****
00070PRINT L,M,N                                         *****
00080PRINT 'END OF CALCULATION'                          *****
00090GO TO 10                                            *****
00100END                                                 *****
***** END OF FILE *****                                  *****
```

Figure 22. Line-Number Editing (Input)

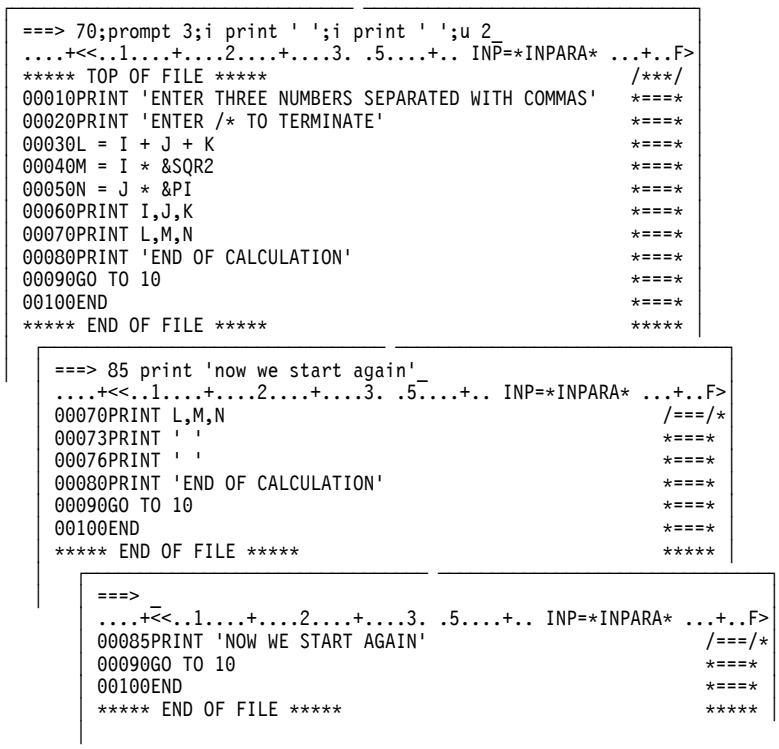


Figure 23. Line-Number Editing (Extension)

Flagging Changes

Via the FLAG editor command or the /PROTECT system command, you can request that all editor functions which change the data within a member will cause the changes to be noted ('flagged') on the changed lines themselves. For example, you may want changes to be noted in columns 73-80. The change flag contains the following information:

Position
Content

0
Type of change

1-3
Date of change

4-7
User ID of the user requesting the change

Through the VSE/ICCF tailoring option EDFLAG, your VSE/ICCF administrator can choose which columns are to contain the change flag.

Editing All 80 Columns

Normally, columns 73 through 80 of the file being edited are "hidden" behind the line command area. If you set the editing zone (by a `zone 1 80` command), the editor offers the following ways to use these columns for data:

- Using the LEFT and RIGHT commands to shift the display of your data in the indicated direction.

  This method works fine if you assign the LEFT and RIGHT commands to PF keys. However, you never see all 80 columns at the same time.

- Using the VIEW command to display only the desired column range(s).

  This method offers an advantage over the first one if somewhere in the middle of the records of your file you have a range of at least eight columns that need not be displayed or changed.

- Using the FORMAT command to assign two physical screen lines to each data record. The effect of this method is shown in Figure 24 on page 160.

  The method allows simultaneous viewing and editing of all 80 columns; however, the double-line display may appear somewhat confusing.
- Using the FORMAT command to eliminate the line command areas (see Figure 25 on page 160).

  This method produces the clearest display, but you cannot use line commands any more and column 80 stays invisible (in many cases, the file can be reorganized to leave column 80 unused).

```
===> format 2_
<<..+....1....+....2....+....3. .5....+.. MEM=INVENTRY ...+..F>
***** TOP OF FILE *****                                    /***/
ITEM1             22.75   PUMPS & PIPES LTD., HUSTON     *****
ITEM2              3.40   JAME & JONES, NEW YORK         *****
ITEM3            127.00   MODERN PLASTICS, DALLAS        *****
ITEM4             60.05   CANADIAN PLYWOOD FACTORY, TORO *****
ITEM5              2.87   CHEMICAL INSTRUMENTS, ALABAMA  *****
ITEM6           3940.00   MASSATRONICS LTD., MASSACHUSET *****
```

```
===> _
<<..+....1....+....2....+....3. .5....+.. MEM=INVENTRY ...+..F>
***** TOP OF FILE *****
                                                          /***/
 ITEM1             22.75   PUMPS & PIPES LTD., HUSTON        PUP
I                                                         *****
 ITEM2              3.40   JAME & JONES, NEW YORK            JAJ
O                                                         *****
 ITEM3            127.00   MODERN PLASTICS, DALLAS           MOP
L                                                         *****
 ITEM4             60.05   CANADIAN PLYWOOD FACTORY, TORONTO CAP
F                                                         *****
 ITEM5              2.87   CHEMICAL INSTRUMENTS, ALABAMA     CHI
N                                                         *****
 ITEM6           3940.00   MASSATRONICS LTD., MASSACHUSETTS  MAT
R                                                         *****
 ***** END OF FILE *****
                                                          *****
```

Figure 24. Screen Format Using Two Lines for Each Record

```
===> format *1_
<<..+....1....+....2....+....3. .5....+.. MEM=INVENTRY ...+..F>
***** TOP OF FILE *****                                    /***/
ITEM1             22.75   PUMPS & PIPES LTD., HUSTON     *****
ITEM2              3.40   JAME & JONES, NEW YORK         *****
ITEM3            127.00   MODERN PLASTICS, DALLAS        *****
ITEM4             60.05   CANADIAN PLYWOOD FACTORY, TORO *****
ITEM5              2.87   CHEMICAL INSTRUMENTS, ALABAMA  *****
ITEM6           3940.00   MASSATRONICS LTD., MASSACHUSET *****
```

```
===> _
<<..+....1....+....2....+....3. .5....+.. MEM=INVENTRY ...+..F>
***** TOP OF FILE *****
ITEM1             22.75   PUMPS & PIPES LTD., HUSTON        PUP
ITEM2              3.40   JAME & JONES, NEW YORK            JAJ
ITEM3            127.00   MODERN PLASTICS, DALLAS           MOP
ITEM4             60.05   CANADIAN PLYWOOD FACTORY, TORONTO CAP
ITEM5              2.87   CHEMICAL INSTRUMENTS, ALABAMA     CHI
ITEM6           3940.00   MASSATRONICS LTD., MASSACHUSETTS  MAT
***** END OF FILE *****
```

Figure 25. Screen Format for Viewing 79 Columns

File-Type Dependent Setting of Editor Options

If you work with different data formats such as assembler programs, PL/I programs, text files etc., you may find it convenient to have certain edit controls set automatically to fit the respective data format. This is done most easily by creating copies of the ED macro under new names and adding the required commands at the end of these copies.

The following example illustrates how to create a macro that makes the editing of PL/I programs easier:

You enter (as administrator) the following commands:

```
(1) /sw 2
(2) ed
(3) getfile ed
(4) input
(5) set tab=#
(6) tabset pli
(7) set pf10ed cursor tabback
(7) set pf11ed cursor tabforward
(8) echo tab character = number sign
        (null line to end input mode)
(9) file edp
```

(1)
Switches to common library, assuming that library 2 is your common library.

(2)
Invokes the editor for editing in the input area.

(3)
Sets the line pointer to end of file.

(4)
Causes subsequent statements to be added.

(5)
Sets a tab character; select a character that is not used in PL/I.

(6)
Sets tabs suitable for PL/I.

(7)
Recommended for ease of cursor positioning during full-screen changes.

(8)
Reminds the user of setting tab characters.

(9)
Creates the member EDP.

From now on, you may use EDP `membername` for editing PL/I programs, and you need no extra work to prepare for tab setting and cursor positioning.

Indexed Editing for Large Members

The editor offers two aids to make editing of large members easier and quicker:

- Line-number editing. For advantages and techniques of this way of editing, see the section “Line-Number Editing” on page 157.
- Indexed editing. This is discussed below.

In indexed editing, you use the INDEX command to create an internal index. The entries of this index point directly to every n-th record in the member ('n' being the index interval specified with the INDEX command). If you then issue the POINT command to set the line pointer at a certain record, that command uses the index information to locate the record more rapidly.

If you issue the LINEMODE command before the INDEX command, the line numbers of the selected lines are stored in the index, too. Thus, also the 'nn' command (add, locate, replace by line number) use the index.

When you do record additions or deletions, the index is not updated. As a result, the 'POINT nn' command that should point to the nn-th record in a member may become somewhat imprecise. Therefore, after a certain number of additions or deletions, reissue the INDEX command.

Indexed editing is recommended with members larger than 500 records. The index interval should be 1/30 of the member size, but must lie between 40 and 400. You may use the /COUNT command to find out the number of records in your member.

An index, to be useful, can cover up to 12,000 records. The performance gain through indexing decreases when you edit a member beyond that limit.

Special Views of Data

When editing column-oriented data (for example an RPG program), you may use the VIEW command to rearrange the way in which the columns are displayed. Up to 12 fields (column ranges) may be selected from the file and displayed in any order and any position on the screen.

Hexadecimal Editing

In certain situations, you may want to edit non-printable hexadecimal patterns (for example to apply a quick patch to an object deck). Again, the VIEW command allows you to display any field in your records in two-digit hexadecimal format. Changes must be entered as hexadecimal digits. For a detailed example, refer to the section “VIEW Command” on page 223.

Editing Mixed Data

If you work at an IBM 5550 with 3270 Emulation, you can edit mixed data other than DBCS print-type members. In Table 2 on page 165 the editor commands which support mixed data editing are marked with a Y in the 'DBCS SUPPORT' column. The commands marked with N in this column do not apply to or do not support the editing of mixed data.

Maintaining Multiple Change Levels of a Member

When developing a program or a document, you may feel the need to retain older versions of it. VSE/ICCF supports this in a convenient way through **generation member groups**.

A generation member group comprises from 2 to 10 versions of a member. The names of the members in a group consist of a root part of up to six characters plus a suffix of the form -n, where n is a number from 0 to 9 denoting the change level. The member with the -0 suffix is the most recent update.

VSE/ICCF maintains, in the directory entries of these members, flags that identify the members as part of a generation member group. Whenever a new update of the member is added to the group with a suffix of '-0', the suffixes of all other members in the group are stepped up by one. If all positions in the group are occupied, the oldest version of the member is purged. A new update level is added when you save the contents of the input area as '-0' member of the group (see examples below). Only level '-0' of a generation member group may be edited, older levels cannot be changed unless you 'ungroup' the members.

The following example for creation and usage of a generation member group is based on the assumption that only selected versions of the member are put into the group. Therefore, a working copy is maintained which replaces the level '-0' member of the group when certain checkpoints in the development process have been reached:

```
(1) /group create myprog 5
(2) ed myprog
     ...
    quit
(3) /input
(3) /insert myprog
(3) /save myprog-0
```

(1)
Creates dummy members: MYPROG-0, MYPROG-1, ... MYPROG-4.

(2)
This optional work copy is not part of the group; it is used to accumulate changes up to a certain checkpoint.

(3)
After having applied (and tested) all changes, you may want to put the member into the group. These commands cause the current change level of MYPROG to be stored as member MYPROG-0.

The following example works without a work copy; each change is put into the group.

```
(1) /group create myprog 5
    ed
(2) get myprog-0
     ...
(3) file myprog-0
```

(1)
Creates dummy members: MYPROG-0, MYPROG-1, ... MYPROG-4.

(2)
All changes will be based on the latest version in the generation member group.

(3)
Files the new version of MYPROG as member MYPROG-0.

Logging

The VSE/ICCF logging option (set via 'SET LOG ON') is valid for full-screen editing operations. However, what you see in the log file are sometimes the equivalents to the data manipulation functions that you perform on the screen. For example, if lines on the screen are changed, the changed lines are internally passed as replace commands (hence the RC01 in the log area). Similarly, other screen and line command functions will appear as equivalent editor commands.

Most editor commands are logged as they are processed; a few editor command functions are not logged at all.

Temporarily Leaving Full-Screen Display

Certain editor commands (such as PRINT, LIBRARY, SHOW, and HARDCPY) cause your terminal to temporarily leave full-screen edit mode. The output from these commands is not displayed in full-screen editing format. To reinstate the full-screen edit display after a SHOW or HARDCPY command, press ENTER or CLEAR.

If the LIBRARY, PRINT, or PRINTFWD command displays a single screen or less and returns to edit mode (indicated by 'ED' on the scale line), you can return to the full-screen editor formatted display by pressing ENTER or CLEAR.

If the LIBRARY, PRINT, or PRINTFWD commands cause more than a single screen to be displayed, your terminal will be in list mode (as indicated by 'LS' on the scale line) until the last page is reached. To get a display of a succeeding page, simply press ENTER. While in list mode, system commands (such as /SKIP, /CONTINU, /CANCEL, etc.) will be effective.

To end list mode before the last page is reached, enter the /CANCEL command or press the Cancel key. At this point, the *END PRINT message should appear, and your terminal will return to full-screen edit mode. Now press CLEAR or ENTER to get the full-screen editor formatted display again.

Because the PRINT, PRINTFWD, LIBRARY, SHOW, and HARDCPY commands cause the terminal to temporarily leave the full-screen editor, an editor command entered on the same screen following one of these commands will be ignored. Thus, if a PRINT command is entered followed by a logical line end character and one or more editor commands, the commands following the PRINT command will not be executed. This applies also if there are multiple command areas on the screen. If a PRINT or SHOW command, for example, is entered in one command area, any commands in command areas farther down the screen will be ignored.

Chapter 6. Editor Commands and Macros

Table 2 on page 165 gives a summary of the available editor commands and macros. Following Table 2 on page 165, you find a more detailed description of these commands and macros in alphabetic order of the command and macro names.

In the **DBCS Support** column of Table 2 on page 165,

N =
No mixed-data support or not affected by DBCS support.

Y =
Mixed-data support.

In the **Type** column of Table 2 on page 165,

C =
Editor command.

M =
Editor macro.

Table 2. Summary of Editor Commands and Macros

Operation	Function	DBCS Support	Type
Add	Adds data beyond the end of data within the edit zone.	N	C
ALIgn	Aligns data in the zone to right and left margins.	N	C
ALter	Changes a single character to another character in one or more lines.	N	C
BAckward	Pages backward through the file.	Y	C
BLank	Blanks out characters in the current line.	N	C
Bottom	Moves the line pointer past end of file.	Y	C
CANcel	Ends the editor session.	Y	C
CAse	Controls upper- /lowercase input data translation.	N	C
CENter	Centers the data within the current zone.	N	C
Change	Changes one character string to another in one or more lines.	Y	C
@COPY	Copies lines from one part of a file to another.	Y	M
CTL	See section "[/]SET Command" on page 100.	Y	C
CURsor	Positions the cursor on the screen.	Y	C
DELete	Deletes one or more lines.	Y	C
DELIM	Defines the delimiter character.	Y	C
DOwn	The same as the Next command.	Y	C
DUP	Duplicates the current line a specified number of times.	Y	C
ECHO	See section "[/]ECHO Command" on page 48.	Y	C
END	The same as the Quit command.	Y	C
ENTer	Starts editing an additional file.	Y	C
FILe	Saves the input area file or a newly created file by storing the file in the library; ends editing of this file.	Y	C

Table 2. Summary of Editor Commands and Macros (continued)

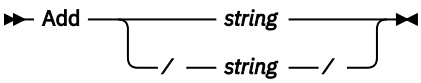
Operation	Function	DBCS Support	Type
Find	Performs a column-dependent search for specified data.	Y	C
FLag	Sets change flagging on or off.	Y	C
FORMat	Defines the number and configuration of format areas within a logical screen.	Y	C
FOrward	Pages forward through the file.	Y	C
@FSEDPF	Sets PF keys for use in full-screen edit mode.	Y	M
GETfile	Copies data from another member or work area into the file that is being edited.	Y	C
HARdcpy	See the section "[/]HARDCPY Command" on page 61.	N	C
INDex	Builds an index of the current file for rapid editing.	Y	C
INPut	Places a format area into editor-input mode.	Y	C
Insert	Inserts a new line into the file.	Y	C
JUStify	Justifies data right or left within the specified zone.	N	C
LAdd	Adds blank lines to the file.	Y	C
LEft	Shifts the data within the logical screen to the left (the view is shifted to the right).	Y	C
LIBRARY	See the section "[/]LIBRARY Command" on page 67.	Y	C
LINemode	Sets line number editing on or off.	Y	C
LN	See the LOCNot command.	Y	C
Locate	Locates a string of characters within a file.	Y	C
LOCNot	Locates the first nonoccurrence of a string.	Y	C
LOCUp	Locates a string of characters within a file in a search upward (or backward) from the current line.	Y	C
LUp	See the LOCUp command.	Y	C
MSG	See the section "[/]MSG Command" on page 79.	Y	C
@MOVE	Moves lines from one part of a file to another (see also the @COPY macro).	Y	M
Next	Moves the line pointer forward a given number of lines.	Y	C
Overlay	Overlays the current line with the characters specified in the operand field of the command.	N	C
OVERLAYX	Overlays the current line with hexadecimal format data.	N	C
OX	The same as the OVERLAYX command.	N	C
PF	The same as the Print command.	Y	C
PFnn	Simulates the function of a PF key.	Y	C
POint	Positions the line pointer based on an established index.	Y	C
Print	Displays a given number of lines.	Y	C
PRINTFwd	See Print command.	Y	C

Table 2. Summary of Editor Commands and Macros (continued)

Operation	Function	DBCS Support	Type
PROmpt	Sets or displays prompt increment for line number editing.	Y	C
Quit	Ends editing for a file.	Y	C
RENum	Puts new sequence numbers into the file being edited.	Y	C
REPEAT	Executes a subsequent OVERLAY, BLANK, ALIGN, CENTER, JUSTIFY or SHIFT command a given number of times.	N	C
REPlace	Replaces the contents of a library member.	Y	C
REStore	Restores various editor settings from the previous stack edit.	Y	C
Rewrite	Replaces the current line with a specified string (primarily used for context editing). See section “REWRITE Command” on page 346.	N	C
RIght	Shifts the data within the logical screen to the right (the view is shifted left).	Y	C
RPT	The same as the REPEAT command.	N	C
SAve	Saves the contents of the input area as a member of the library.	Y	C
SCReen	Defines and displays the number and sizes of logical screens.	Y	C
Search	Searches the file from the top for a given string of characters.	Y	C
SET	Sets various functions on or off. See also the section “[/]SET Command” on page 100.	Y	C
SHIft	Shifts data within the defined zone left or right a given number of columns.	Y	C
SHow	Displays the names of the files being edited. See also the section “[/]SHOW Command” on page 118.	Y	C
SPlit	Splits the current line into two lines.	N	C
STACk	Places data or commands into the editor stack.	Y	C
STATus	See the section “[/]SHOW Command” on page 118.	Y	C
TABset	See the section “[/]TABSET Command” on page 130.	Y	C
Top	Positions the line pointer to the null line in front of the first line of the file.	Y	C
TYpe	The same as the Print command.	Y	C
Up	Moves the line pointer upward a given number of lines.	Y	C
Verify	Alters screen options.	Y	C
VIEW	Indicates how data is to be displayed on the screen.	Y	C
Zone	Sets the current zone for editing, or scanning operations.	Y	C
nn	Replaces or locates lines by sequence number.	Y	C

ADD Command

The ADD command is used to add a string of characters behind the last nonblank character in the edit zone of the current line.



string
/string/
Is the string of characters to be added to the line. If the string contains any commas, parentheses or blanks, delimiter characters must be used.

The string is added behind the last nonblank character of the current line (see the example below) . If the string does not fit into the edit zone, it gets truncated. If there is no space left, message `*MSG=FUNCTION REQUESTED BEYOND ZONE` is displayed.

The line pointer remains unchanged, the cursor is set to the command area.

Note: You can display the delimiter character with the SHOW command. You can change it with the DELIM command.

Example

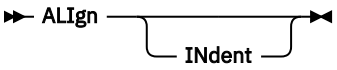
Add the string ',XREF' to an existing /OPTION statement.

```
===> add ,xref_
<<..+....1....+....2....+....3. .5....+.. MEM=ASSEMJOB>>..+..FS
// OPTION DECK                                            /===/*
// EXEC ASSEMBLY                                          *====*
```

```
===>
<<..+....1....+....2....+....3. .5....+.. MEM=ASSEMJOB>>..+..FS
// OPTION DECK,XREF                                       /===/*
// EXEC ASSEMBLY                                          *====*
```

ALIGN Command

The ALIGN command is used in text editing to align both the left and right hand margins of lines. The right and left margins are determined by the current zone setting or by a column suffix.



INdent
Bypasses left justification, for example where you want an indented margin at the beginning of a paragraph.

The column suffix Cnn can be used with this command.

Alignment is performed by left justifying the line within the zone and then inserting additional blanks between words where existing blanks occur until the line is also justified at the right hand margin. If more than one line is to be aligned, issue the REPEAT command before you issue the ALIGN command to specify the number of lines to be aligned.

The line pointer remains unchanged; the cursor is set to the command area.

Example

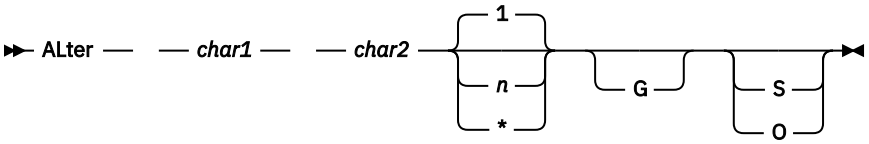
Align two lines of text (as indicated by the REPEAT command) within the zone as set by the ZONE command (from column 1 to 22 in the example). Assume that the semicolon (;) is the line-end character.

```
===> zone 1 22;repeat 2;align;top_
<<..+....1....+....2....+....3....+.. ..5....+.. MEM=member  >>..+..FS
HOW ARE YOU TODAY?                                           /===/*
I AM FINE THANK YOU.                                         *====*
***** END OF FILE *****                                      *****
```

```
===> _
<<..+....1....+....2>>..+....3....+.. ..5....+.. MEM=member  >>..+..FS
***** TOP OF FILE *****                                       /***/
HOW   ARE  YOU  TODAY?                                       *****
I AM  FINE  THANK YOU.                                       *****
***** END OF FILE *****                                      *****
```

ALTER Command

The ALTER command lets you change one character to another, and reference a character by its hexadecimal value. Any of the 256 EBCDIC combinations may be referenced.



The column suffix Cnn can be used with this command.

char1
Is the character to be altered. It can be either a single character or a pair of hexadecimal digits (00 through FF).

char2
Is the character to which char1 is to be altered. It can be either a single character or a pair of hexadecimal digits.

n
*****

Indicates the number of lines to be searched for char1.

'*' indicates that all lines in the file beginning with the current line are to be searched.

If this operand is omitted, only the current line is searched. The maximum number of lines that can be searched is 99999.

G

Indicates a global change. Every occurrence of char1 in the edit zones of the searched lines is changed. If G is omitted, only the first occurrence of char1 in each searched line is altered.

S
O

S indicates that changes are to be recorded in the stack successively.

O indicates that changes are to be recorded in the stack starting at the beginning of the stack.

VSE/ICCF searches for the character specified for char1 within the edit zone of the specified lines. The search starts with the current line. If the line pointer is positioned at end of file, the search starts with the first line. If char1 is found, it is replaced by char2.

The line pointer is positioned at the last line that was searched. The cursor is set to the command area.

Global changes, that is changes of a character throughout a member may incorporate many single changes. They can be best controlled by recording each change in the stack. The stack must have been previously allocated either via the STACK OPEN command, or earlier through the use of the punch area. You can use the command PRINT $$STACK to print the stack for viewing of the changes recorded by VSE/ICCF.

Example

Add a REP statement to an object deck. A REP statements must begin with X'02' in column 1. Because you cannot enter this character directly from a terminal keyboard, insert `zrep` first and then alter z to X'02'.

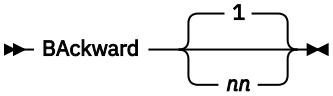
Another possibility to do such a change would be to view the data in hexadecimal format and then enter '02' directly (see “VIEW Command” on page 223).

```
===> l /rld /;insert zrep  0000e8 00147f0;alter z 02_
<<..+....1....+....2....+....3. .5....+.. MEM=TPROGOBJ>>..+..FS
***** TOP OF FILE *****                                  /***/
CATALOG TPROG.OBJ         EOD=/+          REPLACE=YES    *****
 ESD            TPROG           IJ2L0010                *****
 TXT              K  Ys K    h&        0s         s  1   *****
 TXT              30        K     .     0 ¢       0      *****
 TXT                                                     *****
 TXT                                      &              *****
 TXT                                                     *****
 TXT   y        HERE IS THE TEST PROGRAM                 *****
 RLD                                              D      *****
 END                                                     *****
/+                                                       *****
```

```
===> _
<<..+....1....+....2....+....3. .5....+.. MEM=TPROGOBJ>>..+..FS
 REP  0000E8 00147F0                                    /===/*
 END                                                    *****
/+                                                      *****
/*                                                      *****
***** END OF FILE *****                                 *****
```

BACKWARD Command

The BACKWARD command allows 'backward' scrolling through a file.



nn

Is a decimal number greater than zero that represents the number of logical pages to be paged backward. A page is defined as the number of lines contained in the format area of the screen.

If the line pointer is less than nn pages away from the top of the file, it is set to the first line in the file and the 'INVALID-RANGE' message is issued.

The cursor is set to the command area.

To scroll backward through a file one page at a time, precede the BA 1 command with an ampersand (&). This causes the command to be retained in the command area. Thus, each time you press ENTER, your display moves back one page.

Examples

Page backward five logical screens:

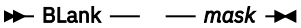
```
BA 5
```

Page backward one page and retain the command:

```
&BA 1
```

BLANK Command

This command places blanks into the current line wherever nonblank characters occur in the mask. Unless IMAGE OFF is set, the logical tab character can be used to generate blanks in the mask operand.



mask
Is any valid input string. The mask is separated from the command by only one blank. Any consecutive blank (after the first one) is considered a part of the mask.

The column suffix Cnn can be used with this command.

You can enter the REPEAT command before the BLANK command to extend the effect to more than one line.

Example

Place blanks in columns 73-80 thus removing the sequence numbers:

```
===> repeat *;blankc73 --------;top_
<<..+....1....+....2....+....3. .5....+.. MEM=COMMANDS ...+..F>
***** TOP OF FILE *****                                  /***/
ASSUME THIS MEMBER CONTAINS SOME TEXT DESCRIBING         000100
THE VARIOUS COMMANDS OF THE VSE/ICCF EDITOR.  BY         000200
MISTAKE, IT HAS BEEN SERIALIZED, AND WE SHALL USE        000300
THE "BLANK" COMMAND TO ERASE THE SERIAL NUMBERS          000400
TEXT TEXT ................................ TEXT          000500
TEXT TEXT ................................ TEXT          000800
```

```
===> _
<<..+....1....+....2....+....3. .5....+.. MEM=COMMANDS ...+..F>
***** TOP OF FILE *****                                  /***/
ASSUME THIS MEMBER CONTAINS SOME TEXT DESCRIBING
THE VARIOUS COMMANDS OF THE VSE/ICCF EDITOR.  BY
MISTAKE, IT HAS BEEN SERIALIZED, AND WE SHALL USE
THE "BLANK" COMMAND TO ERASE THE SERIAL NUMBERS
TEXT TEXT ................................ TEXT
TEXT TEXT ................................ TEXT
```

BOTTOM Command

The BOTTOM command positions the line pointer past the last line of the file.

Use the command followed by the INPUT or INSERT command to add new lines at the end of the file.

Example

Position the line pointer to the bottom of the file. Inserts (or input mode data) can now be entered and will be added behind the currently last line of the file.

```
===> b_
<<..+....1....+....2....+....3. .5....+.. MEM=EXAMPLE >>..+..FS
***** TOP OF FILE *****                                 /***/
FIRST LINE                                             *****
LAST LINE                                              *****
***** END OF FILE *****                                 *****
```

```
===> _
<<..+....1....+....2....+....3. .5....+.. MEM=EXAMPLE >>..+..FS
LAST LINE                                              *====*
***** END OF FILE *****                                 *****
```

CANCEL Command

The CANCEL command ends the editor session immediately and returns your terminal to command mode.

The CANCEL command has the same effect as the QUIT command on each active file. The CANCEL command is forced you press the PA2 key of your IBM 3270 terminal.

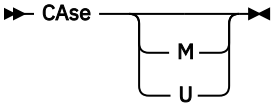
Note: PA2 is the Cancel key in the distributed VSE/ICCF. Your VSE/ICCF administrator may have chosen PA1 or PA3 as the Cancel key.

Example

```
===> cancel_
<<..+....1....+....2....+....3. .5....+.. MEM=EXAMPLE >>..+..FS
***** TOP OF FILE *****                                  /***/
FIRST LINE                                              *****
LAST LINE                                               *****
*****
       _
        ...+....1....+....2....+....3. .5....+....6....+....7....+..FS
       *FULL SCREEN EDITOR TERMINATED
       *READY
```

CASE Command

The CASE command is used to control how upper- and lowercase translation is handled during editing.



M
Causes the editor to accept input of both upper- and lowercase data.

U
Causes the editor to return to uppercase translation.

If the DBCS attribute is set for a member, the command with this specification will be rejected, and the member will be treated as if case were set to mixed.

VSE/ICCF normally translates all input data to uppercase. The CASE command alters this translation so that lowercase data can be entered.

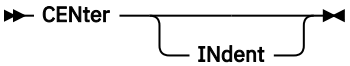
The command applies only to the current edit session. At the end of the session, case translation reverts to what you were using when you started the edit session (see the description of /SET, section “The CASE Feature” on page 107). In other words, if mixed case translation was in effect when the editor was entered and you changed it by a CASE U command, case translation will revert back to mixed case at the end of your edit session.

If the CASE command is entered with no operand, the current CASE setting is displayed. A display of CASE=S indicates that no CASE setting has been specified and that the system CASE setting is in effect; that is, the case translation you were using when the editor was entered is still in effect.

If your display unit shows only uppercase, you can still enter lowercase data if CASE=M is in effect. However, this data will not be displayed as lowercase, and this might cause problems: for example, you might try to locate a string of characters that only **appear** to be in uppercase but are, in fact, in mixed case.

CENTER Command

The CENTER command centers data within the edit zone of the current line.



The column suffix Cnn can be used with this command.

INdent
Can be used to cause any leading blanks within the zone to be considered as part of the data for centering.

If more than one line is to be centered, enter the REPEAT command prior to the CENTER command to specify the number of lines to be centered.

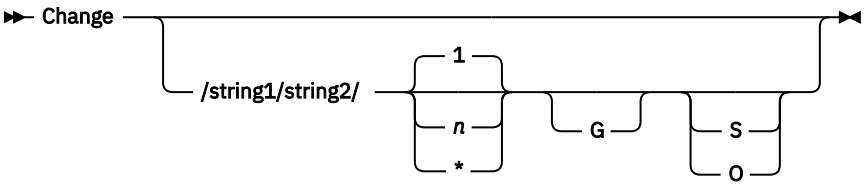
Example

```
===> zone 1 30;repeat 2;center;top_
<<..+....1....+....2....+....3. .5....+.. INP=*INPARA*>>..+..FS
***** TOP OF FILE *****                                  /***/
VSE/ICCF                                                *****
TERMINAL USER'S GUIDE                                   *****
                                                        *****
```

```
===> _
<<..+....1....+....2....+...>>. .5....+.. INP=*INPARA* ...+..FS
***** TOP OF FILE *****                                  /***/
          VSE/ICCF                                      *****
    TERMINAL USER'S GUIDE                               *****
```

CHANGE Command

The CHANGE command replaces one string of characters with another.



The column suffix Cnn can be used with this command. If you issue the command without an operand, VSE/ICCF sets the cursor to the current line.

/
Is any unique delimiting character (not alphabetic or numeric) that does not appear in string1 or string2. The ending delimiter on string2 need be specified only if additional operands follow or if string2 includes trailing blank characters.

string1
Is the character string to be replaced (old data). VSE/ICCF searches for this string within the edit zone of the specified lines starting with the current line. If the line pointer is positioned at end of file, VSE/ICCF starts the search with the first line. If VSE/ICCF cannot find the string, no data is changed.

string2
Is the character string to replace string1 (the new data). The length of this string may be different from the length of the specified old data. However, the result may be truncated so that it fits into the edit zone. For string2, you may specify a null string.

n
*****
For n, specify the number of lines to be searched for string1.

Specify * if all lines of your file beginning with the current line are to be searched.

If you specify neither, VSE/ICCF searches only the current line. The maximum number of lines that VSE/ICCF can search is 99999.

G
Indicates global change. Every occurrence of string1 in the edit zones of the lines being searched is changed. If G is omitted, only the first occurrence in each of the searched lines is changed.

S
O
Indicates that changes are to be recorded in the stack successively ('S') or starting at the beginning of the stack ('O').

The CHANGE command also processes mixed data. Characters in string1 and string2 may be combined in the following way:

String 1	String 2
One-byte characters	One-byte characters
	Mixed data
	Double-byte characters
Mixed data	Mixed data
	Double-byte characters
Double-byte characters	Double-byte characters

The CHANGE operation is rejected if:

- A subfield of double-byte characters specified in string2 would exceed the zone.
- The difference between the number of bytes of string1 and the number of bytes of string2 is not even and the double-byte characters specified in string1 are part of a subfield that extends beyond the zone column. (When you determine the number of bytes that make up string1 or string2, do not count an SO or SI control character, if it occupies the leftmost or rightmost position in the string.)

If the change yields a subfield of double-byte characters that has a length of zero, the adjacent SO and SI control characters are not removed.

The line pointer is positioned at the last line that was searched. The cursor is set to the command area, except for a CHANGE command without operands.

Global changes may incorporate many single changes. They can be best controlled by recording each change in the stack. However, the stack must have been previously allocated either via the STACK OPEN command or by the prior use of the punch area. You can use the command PRINT $$STACK for viewing of the changes recorded by VSE/ICCF.

Note:

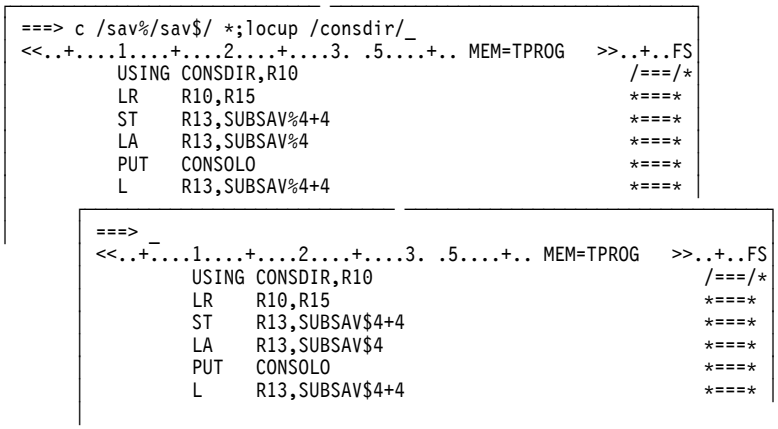
1. The CHANGE command can be used to display, without changing, the first 100 lines (or more than 100 if the PUNCH/STACK area was greater than 100 lines when the editor was entered) that contain the information specified in string1. Enter:

   ```
   change /string1/string1/ * 0
   print $$stack
   ```

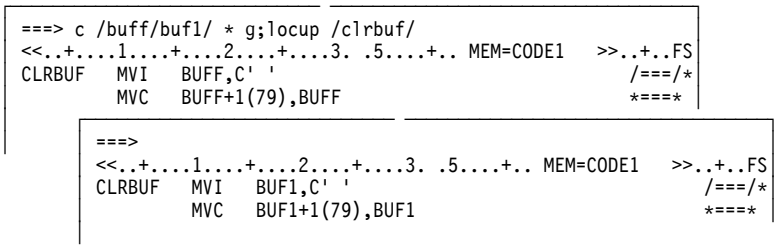
2. Use the ALTER command or OVERLAYX command to change a single character to some special character (one that is not available on your keyboard). The CHANGE command can be used to alter them to displayable characters.
3. No tab character should be in string1 or string2.

Examples

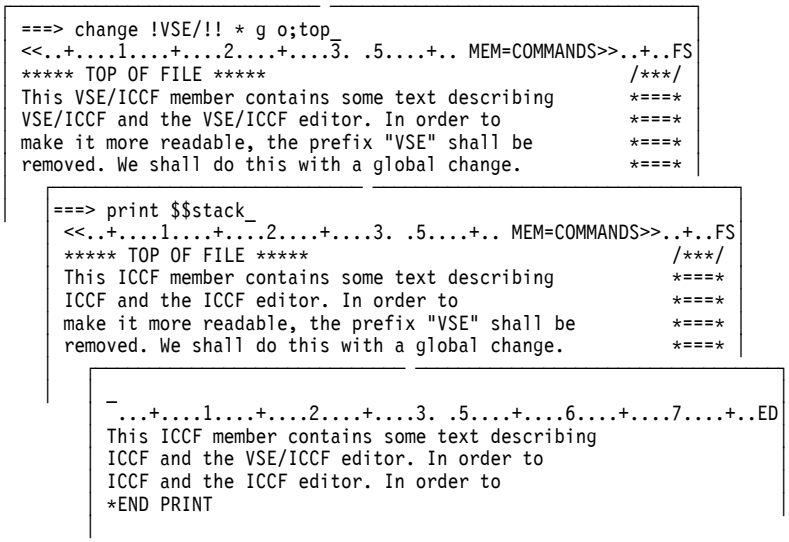
1. Change (globally) all occurrences of 'SAV%' to 'SAV$':



2. Another global change: g is specified to make the change effective for all occurrences on a line.



3. Globally eliminate 'VSE/' from the file and open the stack to record all changes. The exclamation mark (!) is used as delimiter character; a slash (/) would not work because it is part of the change argument.



4. Within the specified zone, replace all occurrences of the two double-byte characters specified in string1 by the four double-byte characters specified in string2. The characters specified in string1 are located on the first line and modified. Because string2 would exceed the zone column, it is truncated.

```
*************** Sample 1-Part 1 ********************

===> ZONE 10 26;C /港区/みなとく/* G;TOP
<<-+----1----+----2----+----3----+----4----+----5----+-- MEM=MEMBER1 >>DB+-FS
***** TOP OF FILE *****                                          /***/
ADDRESS1 = G'東京都港区六本木３丁目'; *---*
 ADDRESS2 = G'東京都新宿区西新宿２丁目'; *---*
  ADDRESS3 = G'広島県広島市稲荷町４丁目'; *---*
   ADDRESS4 = G'福岡県北九州市小倉北区紺屋町'; *---*
    ADDRESS5 = G'東京都新宿区新宿７丁目'; *---*
     ADDRESS6 = G'千葉県松戸市新松戸７丁目'; *---*
      ADDRESS7 = G'東京都千代田区永田町１丁目'; *---*
       ADDRESS8 = G'神奈川県藤沢市桐原町１'; *---*
***** END OF FILE *****                                          *****
```

```
*************** Sample 1-Part 2 ********************

===>
----+----<<---+----2----+-->>---3----+----4----+----5----+-- MEM=MEMBER1  .DB+-.FS
***** TOP OF FILE *****                                          /***/
ADDRESS1 = G'東京都みな六本木３丁目'; *---*
 ADDRESS2 = G'東京都新宿区西新宿２丁目'; *---*
  ADDRESS3 = G'広島県広島市稲荷町４丁目'; *---*
   ADDRESS4 = G'福岡県北九州市小倉北区紺屋町'; *---*
    ADDRESS5 = G'東京都新宿区新宿７丁目'; *---*
     ADDRESS6 = G'千葉県松戸市新松戸７丁目'; *---*
      ADDRESS7 = G'東京都千代田区永田町１丁目'; *---*
       ADDRESS8 = G'神奈川県藤沢市桐原町１'; *---*
***** END OF FILE *****                                          *****
```

5. Within the specified zone, replace all occurrences of the two double-byte characters specified in string1 by the one-byte and the double-byte character specified in string2. No data is changed because:

   a. The difference between the number of bytes of string1 (8 bytes).

   b. The number of bytes of string2 (7 bytes) is not even.

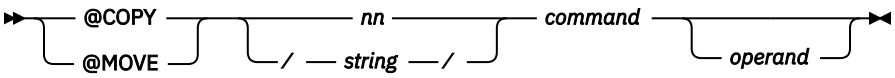
   c. The string specified for string1 is part of a subfield that extends beyond the zone column.

```
*************** Sample 2-Part 1 ********************

===> ZONE 10 30;C /=G'/=G'/* G;TOP
<<-+----1----+----2----+----3----+----4----+----5----+-- MEM=MEMBER1 >>DB+-FS
***** TOP OF FILE *****                                          /***/
ADDRESS1 = G'東京都港区六本木３丁目'; *---*
 ADDRESS2 = G'東京都新宿区西新宿２丁目'; *---*
  ADDRESS3 = G'広島県広島市稲荷町４丁目'; *---*
   ADDRESS4 = G'福岡県北九州市小倉北区紺屋町'; *---*
    ADDRESS5 = G'東京都新宿区新宿７丁目'; *---*
     ADDRESS6 = G'千葉県松戸市新松戸７丁目'; *---*
      ADDRESS7 = G'東京都千代田区永田町１丁目'; *---*
       ADDRESS8 = G'神奈川県藤沢市桐原町１'; *---*
***** END OF FILE *****                                          *****
```

```
*************** Sample 2-Part 2 ********************

===>
----+----<<---+----2----+-->>---+----4----+----5----+-- MEM=MEMBER1 >>DB+-.FS
***** TOP OF FILE *****                                          /***/
ADDRESS1 = G'東京都港区六本木３丁目'; *---*
 ADDRESS2 = G'東京都新宿区西新宿２丁目'; *---*
  ADDRESS3 = G'広島県広島市稲荷町４丁目'; *---*
   ADDRESS4 = G'福岡県北九州市小倉北区紺屋町'; *---*
    ADDRESS5 = G'東京都新宿区新宿７丁目'; *---*
     ADDRESS6 = G'千葉県松戸市新松戸７丁目'; *---*
      ADDRESS7 = G'東京都千代田区永田町１丁目'; *---*
       ADDRESS8 = G'神奈川県藤沢市桐原町１'; *---*
***** END OF FILE *****                                          *****
```

@COPY and @MOVE Editor Macros

The @COPY and @MOVE editing macros copy or move up to 99 lines from one part of the file to another. The two macros are similar, except that the @MOVE deletes the specified lines from their former location whereas @COPY does not.



nn
/string/
For nn, specify the number of lines to be copied or moved. This can be any number from 1 to 99.

/string/ is a character string defining the end of the lines to be copied or moved. All lines from the line pointer down to (but not including) the line with the specified string will be copied or moved. The range of lines indicated by this string cannot be more than 99 lines. The character string must be surrounded by slashes as shown; it may not include any blanks.

command
Is an editing command that positions the line pointer at the line after which the copied or moved data is to be inserted. Valid commands are: DOWN, FORWARD, LOCATE, LOCNOT, LOCUP, NEXT, POINT, TOP, UP, nn.

operand
Is associated with the pointer movement command. This can be a decimal number (as in UP 3) or a string (as in LOC /XREF/). If this operand is a string, it must be bounded by the delimiter character and may not include any blanks.

The macro operands *command* and *operand* together form an editing command that sets the line pointer to the place where the data is to be inserted. Before the target-setting command is processed, the position of the line pointer is as follows:

- For @COPY: your current line.
- For @MOVE: the next line which is not going to be moved (because the lines to be moved are first deleted).

The last line copied or moved becomes the new current line.

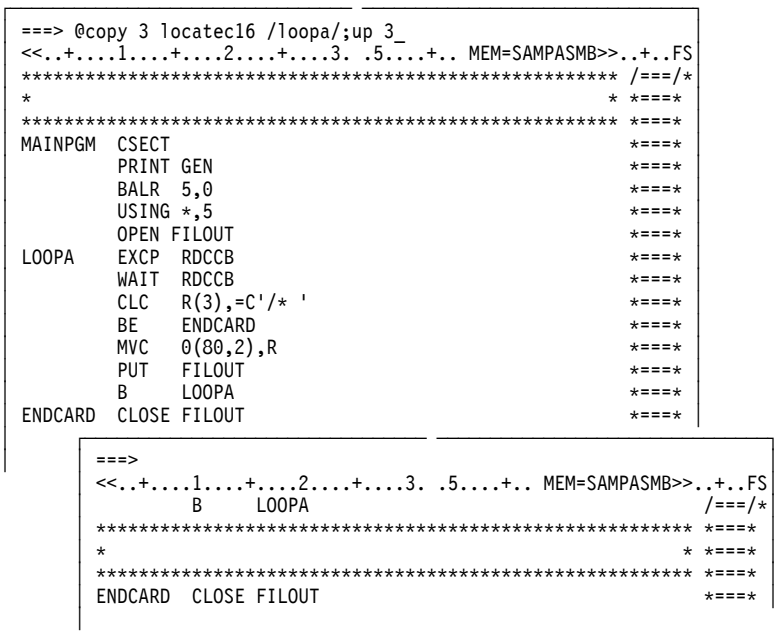
Note:

1. The @COPY and @MOVE macros use the stack area; therefore the previous contents of the stack area are lost.
2. The @COPY or @MOVE macro should not be executed from the stack.
3. If the destination (command operand) is incorrect, @MOVE can cause lines to be deleted from your file. You can recall them by issuing the GETFILE $$STACK command.

Examples

The first example is shown with screen images. The remaining examples focus on the command specification only;

1. Three separator lines are to be copied behind the first line that contains, in columns 16-72, the string 'LOOPA'.



2. Copy three lines beginning at the line pointer and place these lines 7 lines further down in the file.

```
@copy 3 down 7
```

3. Copy five lines beginning at the line pointer and insert these lines after the first line containing string XREF.

```
@copy 5 loc /XREF/
```

4. Make a copy of all the lines down to (but not including) the first occurrence of the string PHASE2, do an upward search for the first line that includes the string INSRTX, and put this copy behind that line.

```
@copy /PHASE2/ lup /INSRTX/
```

5. Copy 10 lines to the top of the file.

```
@copy 10 top
```

6. Make a copy of all lines down to the string STOP and put this copy after the line with sequence number 127100 (assuming line number editing is in effect).

```
@copy /STOP/ 127100
```

7. Move three lines beginning at the line pointer downward in the file seven lines.

```
@move 3 down 7
```

8. Move five lines beginning at the line pointer behind the first line that includes the string XREF.

```
@move 5 loc /XREF/
```

9. Move all the lines down to (but not including) the first occurrence of the string PHASE2 behind the first line found during an upward search to contain the string INSRTX.

```
@move /PHASE2/ lup /INSRTX/
```

10. Move ten lines to the top of the file.

```
@move 10 top
```

11. Move all lines down to the string STOP behind the line with sequence number 127100 (assuming line number editing is in effect).

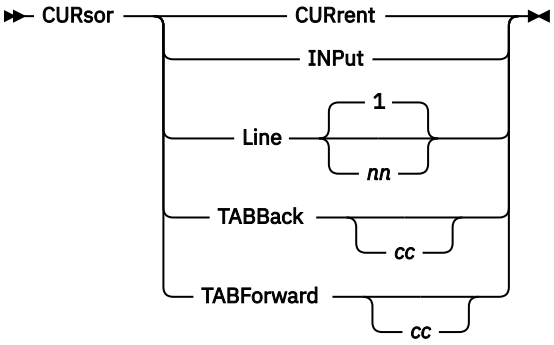
```
@move /STOP/ 127100
```

CTL Command

See the section "[/]SET Command" on page 100.

CURSOR Command

The CURSOR command is used to set the cursor to a certain position on the screen.



CURrent
Moves the cursor to position 1 of the current line. If multiple 'current lines' are on the screen (because of multiple logical screens) or if multiple format areas are in effect, the cursor will be moved to the current line of the logical screen or format area with the most recent command activity (as indicated by the cursor position at the time the ENTER or PF key is pressed). For information about defining a logical screen or a format area, see the sections "SCREEN Command" on page 214 and "FORMAT Command" on page 188, respectively.

INPut
Moves the cursor to the next (or only) command area on the screen. If the CURSOR INPUT command is equated to a PF key, pressing this key will set the cursor to this command line. If there is no command line on the screen following the present cursor position, the cursor will be set to the first (or only) command line.

LINe
Advances the cursor nn lines and sets it to position 1 of the line. If the line advanced to is not a line of data, the cursor will be set to the next line. If the nn value causes the end of the logical screen to be exceeded, a wraparound to the top of the physical screen will occur.

You cannot specify a number larger than the number of lines of your physical screen.

TABBack
TABForward
Moves the cursor backward or forward cc columns from its present position. If cc is omitted and if tab positions are set, the cursor will be moved backward or forward to the last or next tab position (see the "[/]TABSET Command" on page 130 for the setting of tab positions).

Because most of the CURSOR command options are relative to the current cursor position, the command options are most useful when equated to PF keys. The '@FSEDPF' macro is available for setting PF keys to commands for use in full-screen edit mode. You may use a logon macro instead (as described in "Invoking the Editor" on page 141).

If a PF key is set to 'CURSOR TABF 0', it can be used instead of the ENTER key when you want to retain the cursor at its present location.

Note: Do not use the CURSOR command on a multi-command line. This may not result in the expected cursor positioning.

Examples

Set PF keys to some useful CURSOR command options:

1. To move the cursor to the next command line.

```
set pf4ed cur inp
```

2. To move the cursor forward five lines.

```
set pf5ed cur line 5
```

3. To move the cursor to position 1 of the current line.

```
set pf6ed cur current
```

4. To move the cursor backward 20 columns.

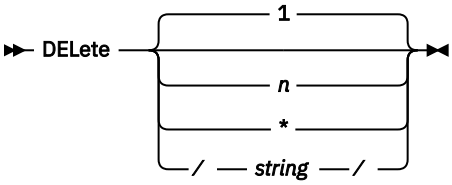
```
set pf7ed cur tabb 20
```

5. To move the cursor forward 20 columns.

```
set pf8ed cur tabf 20
```

DELETE Command

The DELETE command is used to delete a line from the file.



The column suffix Cnn is effective with the 'string' version of this command.

n
Specifies the number of lines (any from 1 to 9999) to be deleted. The command removes the specified number of lines from the file, starting with the current line. Upon completion of the delete request, the line pointer is positioned at the first line following the last deleted line.

*****
Specifies that all lines in the file from and including the current line through the end of the file, or to a maximum of 9999 lines, are to be deleted. However, an entire library member cannot be deleted in this manner.

/string/
Specifies the string which, when matched, ends the delete operation. The string delimiter must be specified in this form of the command.

If /string/ is specified, all lines, starting with the current line and up to (but not including) the first line in which /string/ is matched, are deleted.

If you issue the command without an operand, the editor deletes only the current line.

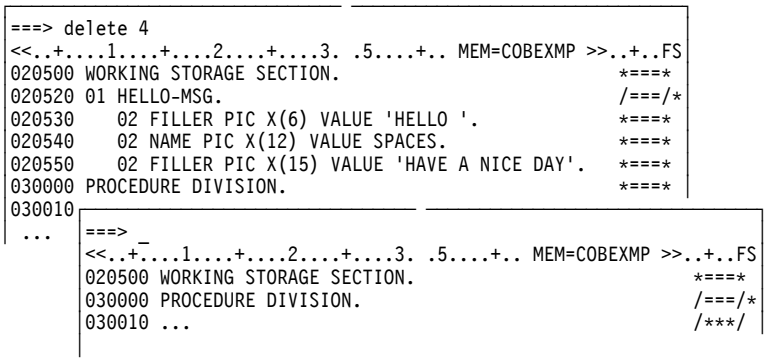
The response *EOF indicates that the end of file was reached during the processing of the command. To position the pointer at the top of the file, a Top command must be issued.

Note:

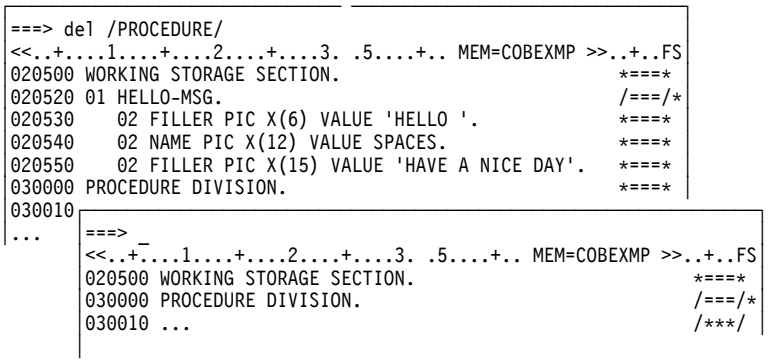
1. You can also delete lines using the edit line command D (see “Delete Line(s)” on page 232).
2. If the logical screen is divided into more than one format area, use of the '*' or '/string/' operands sets all other format areas to the top of the file.

Examples

1. Delete the current line plus the next three lines of the file being edited. This positions the line pointer at the line following the last deleted line.



2. Delete all lines starting with the current line up to (but not including) the line which contains the string PROCEDURE within the defined editing zone.



DELIM Command

The DELIM command is used to set the string delimiter to a character other than a slash (/). It can also be used to return to the slash as delimiter character.

character
Is any non-alphabetic, non-numeric character other than space, comma, right parenthesis, left parenthesis or one of the SET command control characters.

Use the SHOW command with no operand to display the current delimiter setting.

The DELIM command is needed for the DELETE and STACK /string/ commands when the string itself contains a slash. In that case, the string operand cannot be delimited with a slash. The new delimiter remains in effect until the end of your editing session or until you issue another DELIM command.

The delimiter character has a special use with the OVERLAY command. It can be used as a substitute blank character. When used in an OVERLAY line, a delimiter character appearing on the line causes a blank to overlay the corresponding position in the record being edited.

Example

```
delim *
```

Following this command, you could use, for example

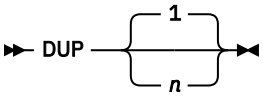
```
delete *// JOB*
```

DOWN Command

See the section “NEXT Command” on page 202.

DUP Command

The DUP command duplicates the current line 'n' times.



n

Specifies the number of times (from 1 to 1000) the current line is to be duplicated. If no value is specified, 1 is assumed.

After the duplication is completed, the line pointer is positioned at the last of the newly created lines.

You can duplicate lines also with the editor line command ' as described in “Duplicate Line” on page 233.

Example

```
===> dup 2
<<..+....1....+....2....+....3. .5....+.. MEM=COBEXMP >>..+..FS
020500 WORKING STORAGE SECTION.                          *****
020520 01 HELLO-MSG.                                      /===/*
020530    02 FILLER PIC X(6) VALUE 'HELLO '.             *****
020540    02 NAME PIC X(12) VALUE SPACES.                 *****
020550    02 FILLER PIC X(15) VALUE 'HAVE A NICE DAY'.    *****
030000 PROCEDURE DIVISION.                                *****
030010
 ...
      ===> _
      <<..+....1....+....2....+....3. .5....+.. MEM=COBEXMP >>..+..FS
      020500 WORKING STORAGE SECTION.                          *****
      020520 01 HELLO-MSG.                                      *****
      020520 01 HELLO-MSG.                                      *****
      020520 01 HELLO-MSG.                                      /===/*
      020530    02 FILLER PIC X(6) VALUE 'HELLO '.             *****
      020540    02 NAME PIC X(12) VALUE SPACES.                 *****
      030000 ...                                               /***/
```

ECHO Command

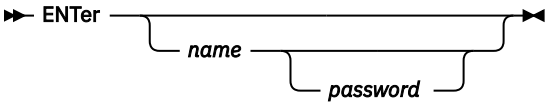
See the section “[/]ECHO Command” on page 48.

END Command

See the section “QUIT Command” on page 208.

ENTER Command

The ENTER command allows you to access (edit), or create, a file in addition to the one (or more than one) that you are presently editing. Use the command only when you want to start editing another file without ending the editing session for the currently edited file.



name
Is the one- to eight-character name of the file to be edited in addition to the currently edited file. This can be the name of an existing library member or of a new file to be created; it can be $$PRINT or $$PUNCH. The characters .P at the end of the name indicate a print-type member.

If the specified name has not been used previously during the session in an ENTER command, VSE/ICCF searches your library directories for that name. VSE/ICCF assumes that you are creating a new file if the specified name is not in the searched directories. Editing then proceeds with an empty file with the specified name.

If you have already edited a file with the specified name in the current session, editing of this file resumes where it left off. The line pointer, editing options, tabs, and so forth will be as they were.

If you do not specify a name, it is assumed that you want to resume editing the input area. However, this is possible only if you first edited the input area via ED (you can nevertheless save the input area as a library member without leaving edit mode, by means of the SAVE or FILE command).

If you specify $$PRINT or $$PUNCH, you should not have a job running asynchronously in an interactive partition. Any changes that you make in these areas would be overlaid by your interactive partition job. Remember, too, that the punch- and editor stack areas are physically the same area. If you use the stack area, explicitly or implicitly (as with the @MOVE macro or the line command M), the previous contents of the punch area is overlaid.

password
Need be specified only when you want to access (or edit) a password-protected library member for the first time during the session.

“Recursive Editing” on page 152 explains how to use the ENTER command.

Any number of files can be edited concurrently, but only up to eight can appear on the physical screen at any given time.

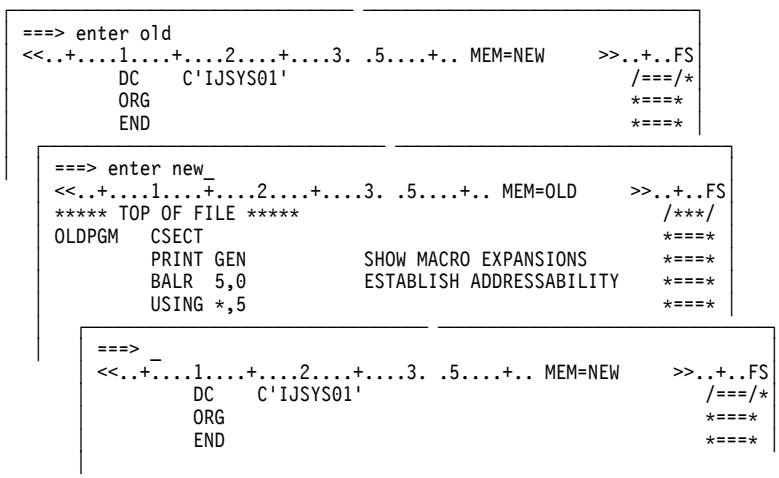
When you issue the command, VSE/ICCF assigns a logical screen to the entered file (how to define a screen is described in the section “SCREEN Command” on page 214). If a free logical screen is available, the file being entered will be displayed on that screen. If all logical screens are in use, the file being entered will use the logical screen associated with the command line on which the ENTER command is typed. The previous file (owner of the logical screen) will disappear from the physical screen. You can bring it back simply by issuing an ENTER command for that file. In other words, the editing session for a file that has disappeared from your physical screen for lack of a logical screen has not ended. By issuing an ENTER command for that file, you make a logical screen available to the file and can carry on editing it at the point where you left off.

Note:

1. You can use the SHOW NAMES command to get a display of the names of the files that are being edited, that is, the names specified in previous ENTER commands.
2. When the print area is entered, VIEW and ZONE are set to begin at column 3.
3. Since the punch and stack areas are physically the same, changes or replacements can be made to data already in the stack by entering the punch area via the ENTER command and making the changes.

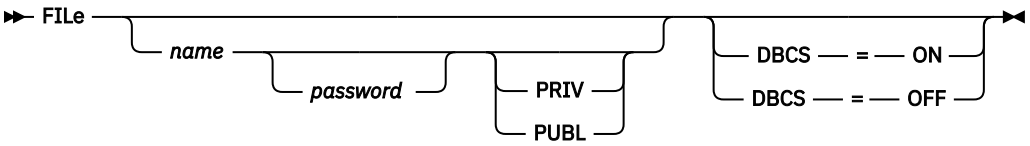
Example

While editing member NEW, you want to temporarily look at member OLD.



FILE Command

The FILE command saves the file that you are editing by storing this file in your library. FILE is the equivalent of a SAVE command followed by QUIT.



name
Is the one- to eight-character name, beginning with an alphabetic character, of the file to be saved in your library.

password
Is a four-character password which you can specify when initially saving a file in the library.

PRIV
PUBL
Allows you to specify that the new member is to be saved in the library as private or public. If neither operand is specified, the private or public status will be set according to the default in your user profile.

DBCS=ON
DBCS=OFF
Allows you to save the new member with the DBCS attribute. If the DBCS operand is not specified, the member is saved with the attribute that was set during the editor session by a SET DBCS command. If a SET DBCS command was not given either, DBCS=OFF is assumed for the new member.

If no other file is being edited, your editing session ends and your terminal changes to command mode. If other files are being edited (via ENTER commands), the logical screen associated with the file being filed is released for use by another file.

If the FILE command is used for the input area, a name must be specified and the contents of your input area will be saved in your library by this name.

If the FILE command is used for a file newly created following an ENTER command, you need not supply a name. The file will be saved in the library using the name that you specified on the ENTER command. If a 'name' operand is specified, the operand will override the name specified in the ENTER command.

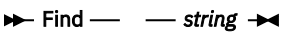
If the FILE command is used for a file which is a member of your library, you need not supply a name. For a library member that has been created previously, a FILE command is equivalent to a QUIT.

Example

```
===> file
<<..+....1....+....2....+....3. .5....+.. MEM=COBEXMP >>..+..FS
020500 WORKING STORAGE SECTION.                          ****
020520 01 HELLO-MSG.                                      /===/*
020530    02 FILLER PIC X(6) VALUE 'HELLO '.             ****
020540    02 NAME PIC X(12) VALUE SPACES.                 ****
020550    02 FILLER PIC X(15) VALUE 'HAVE A NICE DAY'.    ****
030000 PROCEDURE DIVISION.                                ****
030010
 ...
       _
       ...+....1....+....2....+....3. .5....+....6....+....7....+..CM
       *FULL SCREEN EDITOR TERMINATED.
       *END PRINT
```

FIND Command

The FIND command causes a column-dependent comparison of the nonblank characters in 'string' with each line in the file.



The column suffix Cnn can be used with this command.

string
Is any valid input line. It can include blanks, double-byte characters, and the logical tab character.

The compare begins on the current line and continues down the file until a match occurs or until the end of file is reached. If an end-of-file condition immediately preceded the FIND command, an automatic TOP command is performed before FIND begins. If a match occurs, the line pointer is positioned to the record that contains the specified string, and this record is displayed. If no match occurs, the line pointer remains after the last line of the file, and the *E0F message is displayed. To position the pointer at the top of the file again, issue a TOP command.

The specified string is matched against the leftmost columns of the zone only; that is, the search does not look for the occurrence of the string in **any** columns of the line. Also, only columns that correspond to nonblank characters in the specified string take part in the comparison. For example, if 'string' is A C, the search will be for an A in the first column of the zone and for a C in the third column of the zone.

The specified string must be separated from the command by only one blank. All other blanks are considered part of the string. If the string includes double-byte characters and if the SO and/or SI control character(s) occupy the leftmost and/or rightmost position in it, they are not considered as part of the string.

FIND can be used to search for a specific line identifier in columns 73-80 as long as the zone has been set to cover this area. One technique is to issue FIND with a C73 suffix.

Noncolumn-dependent scans are handled by:

- The LOCATE command if you want to scan from a position within the file through end of file (see “LOCATE, LOCNOT, and LOCUP Commands” on page 199);
- The SEARCH command if you want to scan through an entire file (see “SEARCH Command” on page 215).

Examples

1. Find a string starting in column 1.

```
===> f 030000_
<<..+....1....+....2....+....3. .5....+.. MEM=member  >>..+..FS
***** TOP OF FILE *****                                /***/
 CBL APOST                                            *****
000000 IDENTIFICATION DIVISION.                       *****
000010 PROGRAM-ID. 'HELLO'.                           *****
010000 ENVIRONMENT DIVISION.                          *****
020000
       ===> _
       <<..+....1....+....2....+....3. .5....+.. MEM=member  >>..+..FS
       030000 PROCEDURE DIVISION.                              /===/*
       030010 START-OF-PROGRAM.                                *****
       030020     DISPLAY 'ENTER YOUR FIRST NAME' UPON CONSOLE. *****
       030030     ...                                          *****
```

2. Use a logical tab character with the FIND command (for general information on the use of the logical tab character, see “Using Tab Stops for Column-Oriented Editing” on page 150).

```
===> f $PROCEDURE
<<..+....1....+....2....+....3. .5....+.. MEM=member  >>..+..FS
***** TOP OF FILE *****                                /***/
 CBL APOST                                            *****
000000 IDENTIFICATION DIVISION.                       *****
000010 PROGRAM-ID. 'HELLO'.                           *****
010000 ENVIRONMENT DIVISION.                          *****
020000
       ===> _
       <<..+....1....+....2....+....3. .5....+.. MEM=member  >>..+..FS
       030000 PROCEDURE DIVISION.                              /===/*
       030010 START-OF-PROGRAM.                                *****
       030020     DISPLAY 'ENTER YOUR FIRST NAME' UPON CONSOLE. *****
       030030     ...                                          *****
```

   Assuming that the logical tab settings are set to columns 8, 12, 16,... and the logical tab character is '$' then the request searches for the string PROCEDURE starting in column 8.

3. Use the column suffix (c08 in the example) with the FIND command. The request searches for the string PROCEDURE starting in column 8.

```
===> fc08 PROCEDURE
<<..+....1....+....2....+....3. .5....+.. MEM=member  >>..+..FS
***** TOP OF FILE *****                                /***/
 CBL APOST                                            *****
000000 IDENTIFICATION DIVISION.                       *****
000010 PROGRAM-ID. 'HELLO'.                           *****
010000 ENVIRONMENT DIVISION.                          *****
020000
       ===> _
       <<..+....1....+....2....+....3. .5....+.. MEM=member  >>..+..FS
       030000 PROCEDURE DIVISION.                              /===/*
       030010 START-OF-PROGRAM.                                *****
       030020     DISPLAY 'ENTER YOUR FIRST NAME' UPON CONSOLE. *****
       030030     ...                                          *****
```

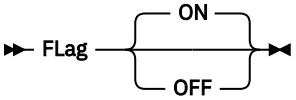
4. Find the specified double-byte character starting in column 15.

```
*************** Sample 3-Part 1 ********************

====> FC15 SOGSI
<<...+....1....+....2....+....3....+....4....+....5....+.. MEM=MEMBER1 >>DB+..FS
***** TOP OF FILE *****                                    '/***/
ADDRESS1 = SOG ' 東京都港区六本木３丁目 ' SI;                    *****
 ADDRESS2 = SOG ' 東京都新宿区西新宿２丁目 ' SI;                 *****
  ADDRESS3 = SOG ' 広島県広島市稲荷町４丁目 ' SI;                *****
   ADDRESS4 = SOG ' 福岡県北九州市小倉北区紺屋町 ' SI;           *****
    ADDRESS5 = SOG ' 東京都新宿区新宿７丁目 ' SI;               *****
     ADDRESS6 = SOG ' 千葉県松戸市新松戸７丁目 ' SI;            *****
      ADDRESS7 = SOG ' 東京都千代田区永田町１丁目 ' SI;         *****
       ADDRESS8 = SOG ' 神奈川県藤沢市桐原町１ ' SI;           *****
***** END OF FILE *****                                    *****
```

```
*************** Sample 3-Part 2 ********************

====>
<<...+....1....+....2....+....3....+....4....+....5....+.. MEM=MEMBER1 >>DB+..FS
  ADDRESS3 = SOG ' 広島県広島市稲荷町４丁目 ' SI;                 /****/*
   ADDRESS4 = SOG ' 福岡県北九州市小倉北区紺屋町 ' SI;           *****
    ADDRESS5 = SOG ' 東京都新宿区新宿７丁目 ' SI;               *****
     ADDRESS6 = SOG ' 千葉県松戸市新松戸７丁目 ' SI;            *****
      ADDRESS7 = SOG ' 東京都千代田区永田町１丁目 ' SI;         *****
       ADDRESS8 = SOG ' 神奈川県藤沢市桐原町１ ' SI;           *****
***** END OF FILE *****                                    *****
```

FLAG Command

The FLAG command is used to record changes to a file.



ON
OFF

ON causes all later changes to the file being edited to be flagged in columns 73-80 (unless some other flagging column has been defined by the VSE/ICCF administrator in the tailoring macro DTSOPTNS). It causes each changed record to be coded with an 8-character change indication:

1. The first character can be one of the following:

   A – An Addition to the file
   C – Only part of the record was changed
   N – The record was replaced via line number editing
   R – The record was replaced

2. The second through fourth characters:

   They indicate the date of the alteration in the format mdd, where m (month) is 1 through 9 for January through September, O for October, N for November, and D for December; dd is the day of the month.

3. The fifth through eighth characters:

   They identify the user who made the change.

OFF turns this feature off. The OFF operand can be used only by the VSE/ICCF administrator.

Certain library members may have flagging automatically set on each time the editor is entered. This can be caused by the /PROTECT command (as described in section "/PROTECT Command" on page 83).

If you are using both line number editing and flagging, and if the flag would occupy the same columns as the sequence number, the sequence number takes precedence. In other words, flagging does not take place.

The 8-character change indicator can be modified only by the VSE/ICCF administrator. Editor change flagging should not be used for RPG source members.

Note:

1. Flagging might overlay every column in the record, depending on your system's default for editor change flagging. To protect certain members (such as procedures, macros, RPG programs, and DTSUTIL jobs) from flagging, contact your VSE/ICCF administrator.
2. The FLAG command is always directed to one particular file. Therefore, if the ENTER command is used to edit several different files concurrently, flagging is active only for the file for which the FLAG command was given.

Example

Assume user USR1 had entered the command

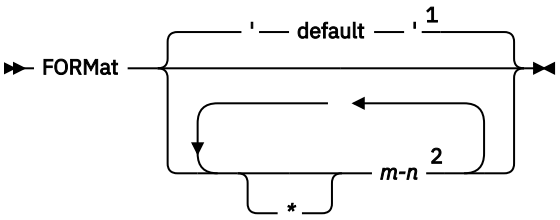
```
flag on
```

Then a change operation would cause the record to be flagged as shown:

```
===> change /ar/xr/
<<..+OFFSET=008 ...2....+....3. .5....+.. MEM=FLAGMEM >>..+..FS
***** TOP OF FILE *****                                  /***/
 MVC   FLDAR,FLDBR                                      *****

      ===>
      <<..+OFFSET=008 ...2....+....3. .5....+.. MEM=FLAGMEM >>..+..FS
       MVC   FLDXR,FLDBR                              CN30USR1 /***/
```

FORMAT Command

The FORMAT command lets you divide a logical screen into several format areas in order to edit or view a file on several places at the same time. It also displays the current setting of format areas.



Notes:

[1] The FORMAT command, if entered without operands, displays the current format settings.
[2] Up to eight format areas may be defined. -n can be omitted on the last specification.

***m-n**
Defines one format area. Up to eight format areas may be defined.

*****
Indicates that the corresponding format area will not include line command areas. This increases the number of data columns that can be displayed per line from 72 to 79. However, you can no longer use the editor line commands. For hints on how to make the 72 data columns suffice for displaying your data, see “Editing All 80 Columns” on page 159.

m
Is a digit (1, 2, or 3) indicating the number of lines to be used for displaying each logical record.

More than one line may be necessary if the VIEW specification makes the displayed record exceed 79 characters. For example, to display all 80 characters in hexadecimal format would require three physical lines per record.

n
Is a decimal number no lower than 2 indicating the number of logical lines to be displayed within the format area. This value can be omitted on the last operand, in which case the editor calculates how many lines will fit into the remaining area of the logical screen and assigns this value as the second operand of the pair.

If the FORMAT command is entered with no operands, the current format settings will be displayed in the first command area for the logical screen.

With the SCREEN command you can divide the physical 3270 screen into logical screens which allows you to view and edit more than one file concurrently. With the FORMAT command you can split a logical screen into up to eight subsections called format areas. This allows you to edit and display different parts within a file at the same time.

Each operand of the command (*m-n specification) represents one format area, and each format area has its own line pointer and its own command area. The number of physical lines within this command area (usually 1 or 2) is determined by the VERIFY command.

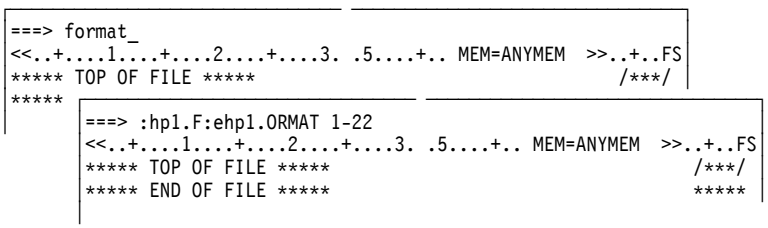
The first format area within a logical screen also contains the scale/header line for that logical screen.

Examples

For an additional use of the FORMAT command, see the second example of the VIEW command on page “2” on page 225.

1. Display the currently active format settings.

   Upon entry to the editor, there is one logical screen of 24 lines (for an IBM 3270 Model 2) on the physical screen and one format area within the logical screen. The number of lines in the command area is set in the ED macro to 1 (this may have been changed for your use of the editor). In addition, there is a scale/header line. Thus, the initial format specification is:



   As a result, one logical record is displayed per line, and there are 22 data lines (plus one scale line plus one line for the command area) on the screen.

2. Divide the logical screen into two format areas. The first format area can display ten records, while the second one can display as many as remain on the screen. If you enter the FIND command in the first format area (as shown), VSE/ICCF displays the first occurrence of SUBPGM in the second format area.

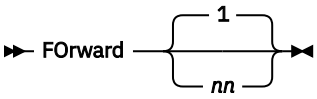
```
===> format 1-10 1
<<..+....1....+....2....+....3. .5....+.. MEM=SAMPASMB>>..+..FS
LOOPB    GET   FILIN                                      /===/*
         LA    13,SAVEAREA                                *****
         LR    1,2                                        *****
         CALL  SUBPGM                                     *****
```

```
===>
<<..+....1....+....2....+....3. .5....+.. MEM=SAMPASMB>>..+..FS
LOOPB    GET   FILIN                                      /===/*
         LA    13,SAVEAREA                                *****
         LR    1,2                                        *****
         CALL  SUBPGM                                     *****
         LTR   1,1                                        *****
         BZ    CARDOK                                     *****
         MVC   P+85(12),=C'INVALID CARD'                  *****
CARDOK   MVC   P(80),0(2)                                 *****
         EXCP  PRCCB                                      *****
         WAIT  PRCCB                                      *****
===> find subpgm_
***** TOP OF FILE *****                                   /***/
/LOAD ASSEMBLY                                           *****
/OPTION NOSAVE,LIST                                      *****
```

```
===>
<<..+....1....+....2....+....3. .5....+.. MEM=SAMPASMB>>..+..FS
LOOPB    GET   FILIN                                      /===/*
         LA    13,SAVEAREA                                *****
         LR    1,2                                        *****
         CALL  SUBPGM                                     *****
         LTR   1,1                                        *****
         BZ    CARDOK                                     *****
         MVC   P+85(12),=C'INVALID CARD'                  *****
CARDOK   MVC   P(80),0(2)                                 *****
         EXCP  PRCCB                                      *****
         WAIT  PRCCB                                      *****
===> _
SUBPGM   CSECT                                            /===/*
         SAVE  (14,12)                                    *****
         LR    5,15                                       *****
         USING SUBPGM,5                                   *****
         LA    4,5                                        *****
         LR    3,1                                        *****
LOOP     CLI   0(3),C'0'                                  *****
```

FORWARD Command

The FORWARD command allows you to scroll (or page) through the file that you are editing.



nn
is a decimal number greater than zero representing the number of logical pages to be scrolled forward.

When the command is entered, the line pointer is moved forward (toward the bottom of the file) by nn pages. A logical 'page' consists of the lines displayed on the logical screen.

Note:

1. If the line pointer is less than nn pages away from the bottom of the file, VSE/ICCF sets the pointer to the last line in the file and issues the `INVALID RANGE` message.
2. To scroll forward through a file one page at a time, precede the FORWARD 1 command with the repeat prefix &;

Examples

1. Scroll forward 5 logical screens:

```
forward 5
```

2. Scroll forward 1 page and retain the command:

```
&fo 1
```

@FSEDPF Editor Macro

The @FSEDPF macro sets the editor set of PF keys for use with the full-screen editor. It is valid only in edit mode.

This macro sets PF keys to functions that are useful in full-screen edit mode. You can use the SET command to change the PF key settings.

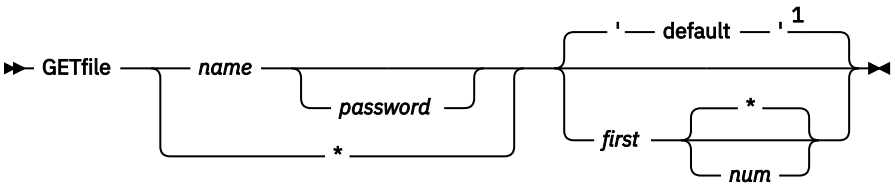
The following commands are issued within the macro as shipped by IBM.

Command	Requested Function

```
SET PF1ED BACK 1             Page backward one page.
SET PF2ED NEXT 10            Page forward ten lines.
SET PF3ED FORW 1             Page forward one page.
SET PF4ED CURSOR INP         Set the cursor to the next command line.
SET PF5ED CURSOR LINE 16     Set the cursor forward 16 lines.

SET PF6ED CURSOR LINE 5      Set the cursor forward 5 lines.
SET PF7ED CURSOR CUR         Set the cursor to the current line.
SET PF8ED CURSOR TABB 20     Set the cursor backward 20 columns.
SET PF9ED CURSOR TABF 20     Set the cursor forward 20 columns.
SET PF10ED CURSOR TABF 0     Leave the cursor unchanged.
```

GETFILE Command

The GETFILE command copies all or a portion of a library member or work area into the file that you are editing.



Notes:

[1] 1 through end-of-file.

name
name password
*****

For *name*, specify the name of the library member that is to be copied.

If the data to be copied is alphanumeric, you can also specify the name of one your work areas: $$STACK for the editor stack, $$PUNCH for the punch area, $$PRINT for the print area, or $$LOG for your log area. For example, you can place data from one or more locations within a program into the stack using the STACK command. You can then re-insert this data into the program by specifying GETFILE $$STACK.

If mixed data is to be copied, 'name' can refer only to a member for which the DBCS attribute has been set and which is not a print-type member, or to the temporary work areas $$LOG, $$PUNCH, and $$STACK. To display the double-byte characters correctly, you first have to save the new member and set the DBCS attribute for it.

For *password*, specify the password, if any, associated with the file that is being copied.

Specify *an asterisk (*)* if data is to be copied from the member being edited and in accordance with the line values specified explicitly or by default. The copy operation stops one line before the current line. This avoids copying lines more than once. If you issue a GETFILE * with the line pointer positioned at the top of your file, the editor copies the entire file.

first
Is the number of the first record to be copied from the referenced member. The default is the whole file.

num
*****

For num, specify the number of lines to be copied. You can specify any number from 1 to 99999.

Specify * or omit the operand to have the editor copy all lines from the line specified as first to end of file.

The last line copied always becomes the current line, unless the GETFILE command was issued after a BOTTOM command. In that case, the line pointer would remain past end of file.

The *END INSERT message indicates when a GETFILE operation is complete. The editor displays the last line copied into the file.

Example

```
===> getfile member2 10 3;top_
<<..+....1....+....2....+....3. .5....+.. MEM=MEMBER1 >>..+..FS
RECORD  1  OF MEMBER1                                    /===/*
RECORD  2  OF MEMBER1                                    *****
RECORD  3  OF MEMBER1                                    *****
RECORD  4  OF MEMBER1                                    *****
```

```
===> _
<<..+....1....+....2....+....3. .5....+.. MEM=MEMBER1 >>..+..FS
***** TOP OF FILE *****                                  /***/
RECORD  1  OF MEMBER1                                    *****
RECORD NUMBER 10 OF MEMBER2                              *****
RECORD NUMBER 11 OF MEMBER2                              *****
RECORD NUMBER 12 OF MEMBER2                              *****
RECORD  2  OF MEMBER1                                    *****
RECORD  3  OF MEMBER1                                    *****
RECORD  4  OF MEMBER1                                    *****
```

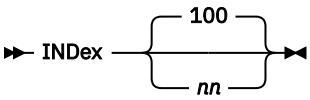
HARDCPY Command

See the section “[/]HARDCPY Command” on page 61.

INDEX Command

The INDEX command is useful when your are editing a large member. You use it to move the line pointer more easily from one area of your file to another.

The command cannot be used in a procedure.



nn
Is a decimal number between 40 and 400 indicating the number of lines between entries in the index.

The command causes an index table to be built with a maximum of 31 entries. Each entry contains information that allows you to use the POINT command to quickly move to the indicated area of the file.

Indexed editing is useful in a member containing more than 500 lines, especially if your editing within the member is fairly random. For members of 3000 lines or more, it is most efficient to specify the operand of the INDEX command as approximately 1/30th of the size of the file.

If you specify a value less than 1/31st of the size of a large file, it may take a bit more time to position the line pointer to a record toward the end of the file.

If you use line number editing, make sure to issue the LINEMODE command before the INDEX command. In this way, the sequence numbers can be included in the index table. Moreover, when the editor builds the INDEX, the sequence numbers associated with index table pointers are also retained. This gives you a handy way of rapidly moving the line pointer. To quickly move the line pointer to some distant area of the file, simply enter the sequence number of the line you want (see “nn Command” on page 229).

Example

Assume you are editing a member that has sequence numbers in columns 73-78.

```
linemode 73 6
index
```

INPUT Command

The INPUT command allows you to insert new lines of data below the current line.

After you have entered the command, the editor sets all lines in the format area below the current line to unused blank or null lines. These lines are not a part of the file until they are filled-in and ENTER has been pressed. The cursor will be set to the beginning of the first input line.

You can use the logical tabbing facility for the entry of new data or you can move the cursor to the correct point on the line and enter the data. Unless the NULLS option is set OFF (the default is ON), forward cursor movement within this area should be done with the space bar and not with the forward cursor (right arrow) key (see “SET Command” on page 216).

Input lines are indicated on the screen by the characters `*INPUT` in the line command area. Line commands can be entered in the line command area of input lines. If the format area does not have line command areas, the characters `*I*` will appear at the beginning of each input line. These characters will be overtyped as input data is entered.

After all new data has been typed in, press ENTER. This causes all new lines to be added to the file and the line pointer to be set to the last new line entered.

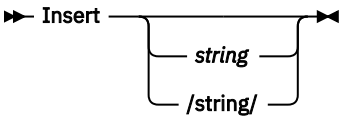
All input lines are treated as beginning in column 1 and ending in column 80; that is, the specifications ZONE, LEFT, and VIEW do not apply. However, once the lines have been included in the file and are redisplayed on the screen, they are again subject to these specifications.

Note:

1. The cursor movement commands (CURSOR, TABF, and TABB) should not be used while entering new data in the input area. Use logical tabbing instead.
2. If you want to enter more new lines than can be entered on one screen, enter the command with a preceding ampersand (`&INPUT`). Then each time you press ENTER, the editor will add the input lines to the file and once again reformat the screen for more input.

INSERT Command

The INSERT command inserts a single new line of data into the file.



string
Is the string of characters to be inserted into the file. It can contain blanks, double-byte characters, and the logical tab character.

If you specify a string without delimiters, only one blank separates the string from the command; all other blanks are considered part of the string. If you specify a string within delimiters, the first character of the string will become the first character of the inserted line.

The column suffix Cnn can be used with this command.

The string is inserted after the current line, and the editor positions the line pointer (but not the cursor) to this inserted string (unless a BOTTOM command preceded the INSERT, in which case the line pointer remains at the bottom).

Without an operand, the INSERT command has the same effect as the INPUT command. The only exception is that, after the data has been entered, the line pointer remains at its original position (that is, it will not be set to the last line entered).

Examples

1. Insert a string of characters that is specified without surrounding delimiters:

```
===> i dddd dddd
<<..+....1....+....2....+....3. .5....+.. MEM=MEMBER2 >>..+..FS
***** TOP OF FILE *****                                  /***/
11111111111                                             /===/*
11111111111                                             *====*
11111111111                                             *====*
```

```
===> _
<<..+....1....+....2....+....3. .5....+.. MEM=MEMBER2 >>..+..FS
DDDD DDDD                                                  /===/*
11111111111                                                *====*
11111111111                                                *====*
11111111111                                                *====*
```

2. Insert a string of characters that is specified with surrounding delimiters:

```
===> i     /dddd dddd/
<<..+....1....+....2....+....3. .5....+.. MEM=MEMBER2 >>..+..FS
***** TOP OF FILE *****                                  /***/
11111111111                                             /===/*
11111111111                                             *====*
11111111111                                             *====*
```

```
===> _
<<..+....1....+....2....+....3. .5....+.. MEM=MEMBER2 >>..+..FS
DDDD DDDD                                                  /===/*
11111111111                                                *====*
11111111111                                                *====*
11111111111                                                *====*
```

3. Insert a blank line into the file by using delimiter characters with the command:

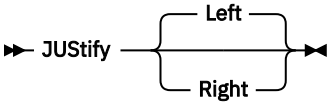
```
===> i  / /
<<..+....1....+....2....+....3. .5....+.. MEM=MEMBER2 >>..+..FS
***** TOP OF FILE *****                                  /***/
11111111111                                             /===/*
11111111111                                             *****
11111111111                                             *****
```

```
===> _
<<..+....1....+....2....+....3. .5....+.. MEM=MEMBER2 >>..+..FS
                                                        /===/*
11111111111                                             *****
11111111111                                             *****
11111111111                                             *****
```

Another example of the INSERT command is shown in Figure 23 on page 159.

JUSTIFY Command

The JUSTIFY command justifies data either left or right.



Left
Justification is to the left margin.

Right
Justification is to the right margin.

The column suffix Cnn can be used with this command.

The right and left margins are determined by the current zone setting or by a column suffix if present. Data within the zone can be shifted to either the left or right edge of the zone, thus squeezing out any intervening blanks.

If more than one line is to be margin justified, you can issue a REPEAT command before the JUSTIFY command to specify the number of lines to be justified.

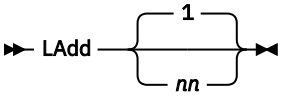
Example:

```
===> zone 20 25;repeat *;justify right;top
<<..+....1....+....2....+....3. .5....+.. MEM=INVENTRY>>..+..FS
***** TOP OF FILE *****                                 /***/
ITEM1              5                                   *****
ITEM2             28                                   *****
ITEM3              2                                   *****
ITEM4           2378                                   *****
ITEM5          28355                                   *****
ITEM6            289                                   *****
```

```
===>
....+....1....+....<<..>>....3. .5....+.. MEM=INVENTRY ...+..FS
***** TOP OF FILE *****                                 /***/
ITEM1                5                                 *****
ITEM2               28                                 *****
ITEM3                2                                 *****
ITEM4             2378                                 *****
ITEM5            28355                                 *****
ITEM6              289                                 *****
```

LADD Command

The LADD command adds blank lines to the file that is being edited.



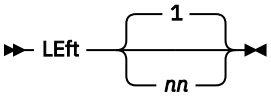
nn
For nn, specify a number from 1 to 999 to indicate the number of blank lines to be added to the file. The editor adds the specified number of lines to the file after the current line and sets the cursor to the beginning of the first of these new lines. You can then type new data into these blank lines.

Logical tabbing can be used within the new lines. If logical tab characters appear within the data on the screen, it is assumed that the line represents a complete 1 to 80 replacement of the blank line. If no tab characters appear within the newly entered data, the input will be interpreted according to your present VIEW, ZONE, LEFT, and RIGHT specifications.

If you add more blank lines than are needed, make sure to delete the ones not required.

LEFT Command

The LEFT command shifts displayed data to the left within a logical screen.



nn
For nn, specify the number of columns by which the data is to be shifted to the left. You can specify a number from 1 to 149.

The number of columns shifted to the left from the normal or base position is called the offset. If a non-zero offset exists, it will be displayed in the scale/header line.

It is often necessary to view columns of data which are not displayed on the screen. There are two ways to accomplish this. One is by using the VIEW command to specify that only certain columns are to be displayed (see the VIEW command). The second method is by shifting the displayed data left (or right) using the LEFT (or RIGHT) command.

When you issue the command, the editor shifts the data within the logical screen to the left by the number of columns specified explicitly or by default. No data loss occurs. To return the display to the original (base) position, issue a RIGHT command for as many columns as are indicated in the offset.

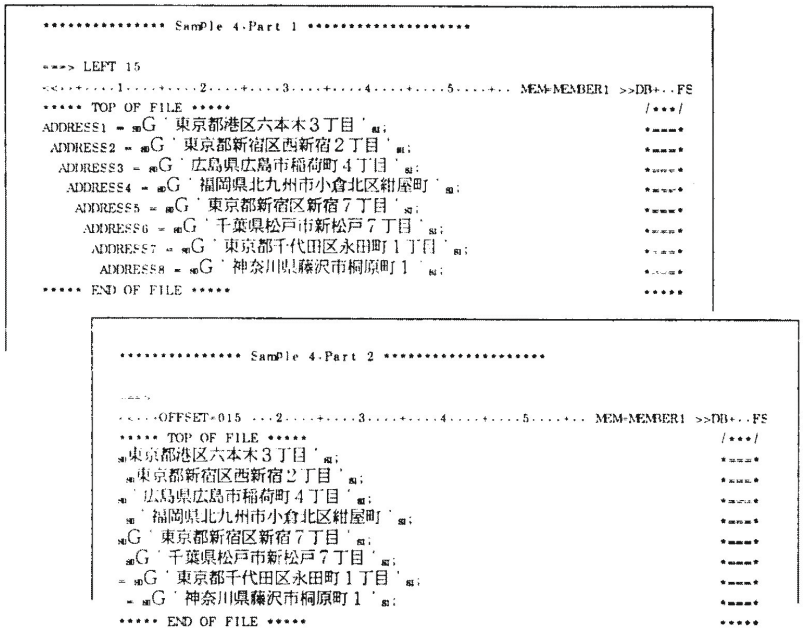
When you shift mixed data and when a double-byte character would occupy the leftmost position on the screen after the shift operation, this double-byte character will not be displayed. It is replaced by one or two control characters, which guarantee that the remainder of the subfield is displayed correctly. The data in the VSE/ICCF library is not affected by this operation, as long as you do not:

- Replace the added control characters
- Change double-byte characters to one-byte characters.

When you shift mixed data and an SI control character would occupy the leftmost position on the screen after the shift operation, the control character will be displayed as an unprintable character.

Example

Shift mixed data 15 columns to the left.

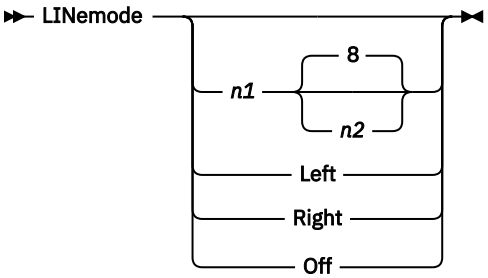


LIBRARY Command

See the section "[/]LIBRARY Command" on page 67.

LINEMODE Command

The LINEMODE command is used to start or terminate line number editing.



n1
Is a decimal number from 1 to 80 indicating the column number where the sequence number for line number editing begins.

n2
Is a number from 1 to 8 indicating the number of columns occupied by the sequence number. The editor will use the default value if n1 is specified but n2 is omitted.

Left
Right
These operands can be specified instead of the exact sequence number location and length. Specify:

Left
To have the sequence number start at column 1 with a length of five columns.

Right
To have the sequence number start in column 73 with a length of eight columns.

Off
Ends line number editing.

With line number editing on, you can use the nn command to add, replace, or locate lines based on this imbedded sequence number. The editor sets the default prompt increment for input mode to 10. Use the PROMPT command (see “PROMPT Command” on page 207) to change this value.

Note:

1. When you are prompted on a typewriter terminal, enter your input line on the same line as the prompted line number. In right-hand line number editing, the sequence numbers are not displayed in columns 73 to 80 (unless you use the VERIFY command to increase the verification setting).

   In line number editing on a display terminal, the prompting numbers in input mode appear on line 2 of the display output area. Enter your input lines in your input area.
2. If line number editing is set so that the sequence number field does not begin in column 1 or in column 73 or beyond, you cannot use the INSERT or REPLACE commands to insert a single line; use the nn command instead.
3. When you initialize line number editing for a file that already exists, the editor assumes that the records are in proper format and numbered in ascending order.
4. The LINEMODE command may alter the zone setting if the sequence number field is at the beginning or the end of the input area and if the zone would overlap this area.

Examples

1. Type in a new member and set it up for line number editing (Example 2 shows how you actually use line numbers to your advantage).

```
===> tabset 1 6;set tab=¢;linemode left;input_
<<..+....1....+....2....+....3. .5....+.. INP=*INPARA*>>..+..FS
***** TOP OF FILE *****                                  /***/
***** END OF FILE *****                                  *****
```

```
===> top_
....+<<..1....+....2....+....3. .5....+.. INP=*INPARA* ...+..F>
***** TOP OF FILE *****                                  /***/
¢print 'enter three numbers separated with commas'       *INPUT
¢print 'enter /* to terminate'                           *INPUT
¢l = i + j + k                                          *INPUT
¢m = I * &sqr2                                          *INPUT
¢n = j * &pi                                            *INPUT
¢print i,j,k                                            *INPUT
¢print l,m,n                                            *INPUT
¢print 'end of calculation'                             *INPUT
¢go to 10                                               *INPUT
¢end                                                    *INPUT
```

```
===> _
....+<<..1....+....2....+....3. .5....+.. INP=*INPARA* ...+..F>
***** TOP OF FILE *****                                  /***/
00010PRINT 'ENTER THREE NUMBERS SEPARATED WITH COMMAS'   *****
00020PRINT 'ENTER /* TO TERMINATE'                       *****
00030L = I + J + K                                      *****
00040M = I * &SQR2                                      *****
00050N = J * &PI                                        *****
00060PRINT I,J,K                                        *****
00070PRINT L,M,N                                        *****
00080PRINT 'END OF CALCULATION'                         *****
00090GO TO 10                                           *****
00100END                                                *****
***** END OF FILE *****                                  *****
```

2. This example does not show the LINEMODE command as such. It demonstrates how you can use line number editing once you have established line numbers as part of the member records (by using the LINEMODE command as shown in the preceding example). The example:

   a. Adds two lines behind line 70 by using the INSERT command.

   b. Uses the nn command to add a line behind line 80.

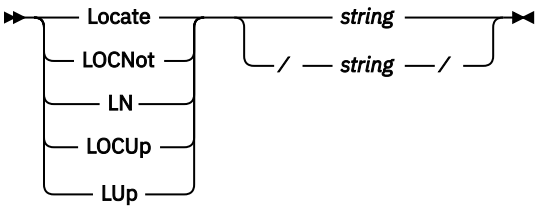
```
===> 70;prompt 3;i print ' ';i print ' ';u 2_
....+<<..1....+....2....+....3. .5....+.. INP=*INPARA* ...+..F>
***** TOP OF FILE *****                                    /***/
00010PRINT 'ENTER THREE NUMBERS SEPARATED WITH COMMAS'     *****
00020PRINT 'ENTER /* TO TERMINATE'                         *****
00030L = I + J + K                                         *****
00040M = I * &SQR2                                         *****
00050N = J * &PI                                           *****
00060PRINT I,J,K                                           *****
00070PRINT L,M,N                                           *****
00080PRINT 'END OF CALCULATION'                            *****
00090GO TO 10                                              *****
00100END                                                   *****
***** END OF FILE *****                                    *****
```

```
===> 85 print 'now we start again'_
00070PRINT L,M,N                                        /===/*
00073PRINT ' '                                          *****
00076PRINT ' '                                          *****
00080PRINT 'END OF CALCULATION'                         *****
00090GO TO 10                                           *****
00100END                                                *****
***** END OF FILE *****                                 *****
```

```
===>
00085PRINT 'NOW WE START AGAIN'                      /===/*
00090GO TO 10                                        *****
00100END                                             *****
***** END OF FILE *****                              *****
```

LOCATE, LOCNOT, and LOCUP Commands

Each of the commands scans the characters of each record (as defined by the column suffix or the ZONE setting) for the specified string.



The column suffix Cnn can be used with this command.

string
Is any group of characters to be searched for in the file. The slash (/) is the string delimiter character; its specification is required if blanks must be included to the right. If 'string' includes double-byte characters and if the SO and/or SI control character(s) occupy the leftmost or rightmost position in it, they are not considered as part of the string.

The locate operation of the editor varies slightly depending on the command that you use:

1. The LOCATE command

   The command scans the records of your file from the line following the current line through end of file. If an end of file condition immediately preceded the LOCATE request, an automatic TOP request is performed before the locate operation starts. If the specified string is located, the pointer is positioned at the line that contains the string. If the string cannot be located, the pointer is positioned at the last line of the file.

   The command is not column-dependent. All characters are scanned, as specified by the ZONE command or the column suffix.

   You can use the command to scan for a line identifier in columns 73-80 if the zone has been set to include these columns.

2. The LOCNOT (locate not equal) command

This version of the command proceeds forward through the file as does the LOCATE. However, the LOCNOT request is satisfied on the first non-matching rather than equal condition found within the zone; that is, with the first line on which the string does not occur (see also Example “4” on page 201).

3. The LOCUP (locate up) command

   This version of the command causes a search backward (or upward) from the current line. If the line pointer happens to be at the top of the file, the LOCUP command has no effect.

When it finds the specified locate condition, the editor displays the line containing the matching (or non-matching) string. If it cannot find this locate condition, the editor displays the *E0F message instead.

For column-dependent scans, use the FIND command; for scans through the entire file, use the SEARCH command.

Examples

1. Locate the string LOOPA, searching through all columns of each line of the file.

```
===> locate loopa_
<<..+....1....+....2....+....3. .5....+.. MEM=SAMPASMB>>..+..FS
***** TOP OF FILE *****                                  /***/
MAINPGM  CSECT                                          *****
         PRINT GEN                                      *****
         BALR  5,0                                      *****
         USING *,5                                      *****
         OPEN FILOUT                                    *****
LOOPA    EXCP  RDCCB                                    *****
         WAIT  RDCCB                                    *****
         CLC   R(3),=C'/* '                             *****
         BE    ENDCARD                                  *****
         MVC   0(80,2),R                                *****
         PUT   FILOUT                                   *****
         B     LOOPA                                    *****
```

```
===> _
<<..+....1....+....2....+....3. .5....+.. MEM=SAMPASMB>>..+..FS
LOOPA    EXCP  RDCCB                                    /===/*
         WAIT  RDCCB                                    *****
         CLC   R(3),=C'/* '                             *****
         BE    ENDCARD                                  *****
         MVC   0(80,2),R                                *****
         PUT   FILOUT                                   *****
         B     LOOPA                                    *****
ENDCARD  CLOSE FILOUT                                   *****
```

2. Locate the string LOOPA, searching through all columns starting with column 16.

```
===> locatec16 loopa_
<<..+....1....+....2....+....3. .5....+.. MEM=SAMPASMB>>..+..FS
***** TOP OF FILE *****                                  /***/
MAINPGM  CSECT                                          *****
         PRINT GEN                                      *****
         BALR  5,0                                      *****
         USING *,5                                      *****
         OPEN FILOUT                                    *****
LOOPA    EXCP  RDCCB                                    *****
         WAIT  RDCCB                                    *****
         CLC   R(3),=C'/* '                             *****
         BE    ENDCARD                                  *****
         MVC   0(80,2),R                                *****
         PUT   FILOUT                                   *****
         B     LOOPA                                    *****
ENDCARD  CLOSE FILOUT                                   *****
```

```
===> _
<<..+....1....+....2....+....3. .5....+.. MEM=SAMPASMB>>..+..FS
         B     LOOPA                                    /===/*
ENDCARD  CLOSE FILOUT                                   *****
         OPEN  FILIN                                    *****
LOOPB    GET   FILIN                                    *****
```

3. Locate the string ' 59', searching upward from the current line toward the top of the file.

```
===> locup /  59/_
..<<+....1....+....2....+....3. .5....+.. MEM=$$PRINT  ...+..F>
  STMNT ERROR NO.  MESSAGE                                /===/*
                                 84-03-31                 *====*
    59   IPK156    SYMBOL 'PROCSAV4' UNDEFINED            *====*
                                                          *====*
```

```
===> _
..<<+....1....+....2....+....3. .5....+.. MEM=$$PRINT  ...+..F>
0000A0 0000 0000          59     ST    R13,PROCSAV4+4    /===/*
                        PUN07570                          *====*
         *** ERROR ***                                    *====*
0000A4 41D0 A1C0  0025A   60     LA    R13,PROSAV$4      *====*
                        PUN07580                          *====*
                           61     PUT   CONSOLO           *====*
                        PUN07590                          *====*
```

4. Find the first occurrence of a nonblank character in position 15.

```
===> zone 15 15;locnot / /_
<<..+....1....+....2....+....3. .5....+.. MEM=SAMPASMB>>..+..FS
MAINPGM  CSECT                                           /===/*
         PRINT GEN                                       *====*
         BALR  5,0                                       *====*
         USING *,5                                       *====*
         OPEN  FILOUT                                    *====*
```

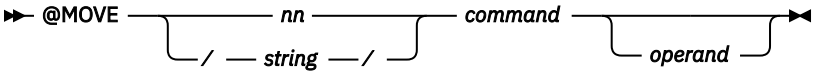
```
===> _
....+....1...>><...2....+....3. .5....+.. MEM=SAMPASMB ...+..FS
               EOFADDR=ENDDISK                            /===/*
         ORG   FILIN+22                                   *====*
         DC    C'IJSYS01'                                 *====*
         ORG                                              *====*
FILOUT   DTFSD IOAREA1=IOA,IOAREA2=IOB,IOREG=(2),       X *====*
               TYPEFLE=OUTPUT,BLKSIZE=408,RECSIZE=80,    X *====*
               RECFORM=FIXBLK                             *====*
```

5. Find the first occurrence of the two double-byte characters that match the specified string.

```
*************** Sample 5-Part 1 ********************

===> L /㎏小倉㎏/
<<..+....1....+....2....+....3....+....4....+....5....+.. MEM=MEMBER1 >>DB+..FS
***** TOP OF FILE *****                                   /***/
ADDRESS1 = ㎏G'東京都港区六本木３丁目'㎏;                  *====*
 ADDRESS2 = ㎏G'東京都新宿区西新宿２丁目'㎏;               *====*
  ADDRESS3 = ㎏G'広島県広島市稲荷町４丁目'㎏;              *====*
   ADDRESS4 = ㎏G'福岡県北九州市小倉北区紺屋町'㎏;         *====*
    ADDRESS5 = ㎏G'東京都新宿区新宿７丁目'㎏;             *====*
     ADDRESS6 = ㎏G'千葉県松戸市新松戸７丁目'㎏;          *====*
      ADDRESS7 = ㎏G'東京都千代田区永田町１丁目'㎏;       *====*
       ADDRESS8 = ㎏G'神奈川県藤沢市桐原町１'㎏;         *====*
***** END OF FILE *****                                   *****
```

```
*************** Sample 5-Part 2 ********************

===>
<<..+....1....+....2....+....3....+....4....+....5....+.. MEM=MEMBER1 >>DB+..FS
   ADDRESS4 = ㎏G'福岡県北九州市小倉北区紺屋町'㎏;         /===/*
    ADDRESS5 = ㎏G'東京都新宿区新宿７丁目'㎏;            *====*
     ADDRESS6 = ㎏G'千葉県松戸市新松戸７丁目'㎏;         *====*
      ADDRESS7 = ㎏G'東京都千代田区永田町１丁目'㎏;      *====*
       ADDRESS8 = ㎏G'神奈川県藤沢市桐原町１'㎏;        *====*
***** END OF FILE *****                                   *****
```

@MOVE Editor Macro



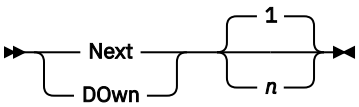
See the section "@COPY and @MOVE Editor Macros" on page 177.

MSG Command

See the section "[/]MSG Command" on page 79.

NEXT Command

The NEXT command advances the line pointer in the file by 'n' lines.



n
Indicates the number of lines by which the pointer is to be advanced. You can specify a number from 1 to 99999. If end of file is reached before the line pointer is advanced n lines, the pointer is positioned at the last line, and the INVALID RANGE message appears.

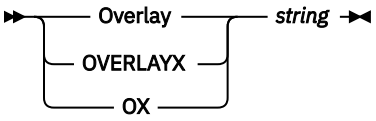
When verification is on, the new current line is displayed. To restore the line pointer to the top of the file, issue the TOP command.

Note:

1. If an index was built using the INDEX command and the number of lines to be advanced is greater than 100 or your specified index value, it will take less time to advance the pointer by using the POINT command. If you were using line number editing when the index was built, the pointer can be rapidly repositioned by specifying the desired sequence number.
2. Pressing ENTER (in edit mode) with no data input is assumed to be a request to move the line pointer to the next line. In other words, it has the same effect as `N 1`.

OVERLAY, OVERLAYX, and OX Commands

The OVERLAY (OVERLAYX, OX) command is used to overlay the current line with the characters specified in the operand field of the command.



The column suffix Cnn can be used with this command.

string
Is a string of characters that replaces parts of the current line.

The OVERLAYX and OX (overlay hex) versions of this command let you insert hexadecimal character combinations into the source line being edited. Instead of typing one overlay character per column to be replaced, type in two hexadecimal digits for each column to be replaced. If invalid or unpaired hexadecimal digits are encountered, the digits are assumed to be valid character data so that both character and hex data can be inserted by using the same command.

Blank characters make no change to the characters in the corresponding positions of the current line. If there is more than one space after the command, these spaces are considered to be part of the specified string.

If you want to blank out a character in the record being edited, use the delimiter character (usually /) in the string. It will overlay the corresponding position with a blank character.

Enter the REPEAT command before the OVERLAY command to extend the effect of the command to more than a single line.

If the *EOF message is displayed, end of file was reached while the editor processed the request. For OVERLAY to reach the end of file, the REPEAT request had to be entered before the OVERLAY.

Examples

1. Overlay the blanks in columns 71 and 72 by two asterisks (*).

```
===> repeat 9;overlayc71 **;top_
<<..+....1....+....2....+....3. .5....+.. INP=*INPARA*>>..+..FS
***** TOP OF FILE *****                                      /***/
****************************** **********************  *****
** THIS                                                     *****
**      PROGRAM                                             *****
**              PROLOG                                      *****
**                      SHOULD                              *****
**
**    ===>
**    <<..+....1....+....2....+....3. .5....+.. INP=*INPARA*>>..+..FS
*****  ***** TOP OF FILE *****                                      /***/
       ****************************** ********************** *****
       ** THIS                                                 ** *****
       **      PROGRAM                                         ** *****
       **              PROLOG                                  ** *****
       **                      SHOULD                          ** *****
       **                             HAVE                     ** *****
       **                                  A NEAT              ** *****
       **                                         FRAME        ** *****
       ****************************** ********************** *****
       ***** END OF FILE *****                                   *****
```

2. Insert a REP statement in an object deck. The REP statement needs an X'02' in position 1. The 'OVERLAYX 02' command replaces the '$', which was placed in position 1 via the INSERT command.

```
===> l /rld /;insert $rep  0000e8 00147f0;overlayx 02_
<<..+....1....+....2....+....3. .5....+.. MEM=TPROGOBJ>>..+..FS
***** TOP OF FILE *****                                      /***/
CATALOG TPROG.OBJ         EOD=/+          REPLACE=YES       *****
 ESD             TPROG           IJ2L0010                   *****
 TXT               K  Ys K    h&        0s         s  1     *****
 TXT               30        K     .     0 ¢       0        *****
 TXT                                                        *****
 TXT                                      &                 *****
 TXT                                                        *****
 TXT   y         HERE IS THE TEST PROGRAM                   *****
 RLD                                                 D      *****
 END                                                        *****
/+                                                          *****
      ===> _
      <<..+....1....+....2....+....3. .5....+.. MEM=TPROGOBJ>>..+..FS
       REP  0000E8 00147F0                                      /===/*
       END                                                      *****
      /+                                                        *****
      /*                                                        *****
      ***** END OF FILE *****                                   *****
```

PF Command

See the section “PRINT, PF, PRINTFWD, and TYPE Commands” on page 205.

PFnn Command

The PFnn command simulates the function of a Program Function key. You can use the command when no PF keys are available.

In the command, nn is a number from 1 to 24 indicating the requested PF key function.

Entering a PFnn command has the same effect as pressing PF key 'nn'. You can place the PFnn command in a list of commands separated from one another by logical line-end characters. The command to which the PFnn command is equated will be processed after all the other commands in the list.

If multiple PFnn commands are entered in a list (or in separate command areas) only the function associated with the last PFnn command on the physical screen will be performed. If a PFnn command is entered on the screen and an actual PF key is pressed, the PFnn command will override the PF key.

A PF key (or PFnn command) within the full-screen editor can be equated to multiple commands separated by logical line end characters. When the PF key is pressed, or the PFnn command is entered, the commands associated with the PF key are performed after all other command activity on the screen.

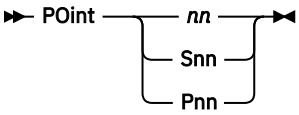
The invoked function is selected from the editor PF key set 'PFED' if it is defined and active. Otherwise, the function is selected from the command mode PF key set 'PF'.

The /PFnn system command can also be used. This command is restricted to its normal use outside the full-screen editor; thus, its effect differs from that of the PFnn command. Specifically, the /PFnn command can invoke only a single command or data line; if it is encountered in a command list, it is processed as it occurs in the list rather than at the end like the PFnn command.

POINT Command

The POINT command positions the line pointer to a given area of the file based on an index built by the editor when you issued the INDEX command.

For details see “INDEX Command” on page 192).



nn
Is a number from 1 to 9999, which represents the line number (relative to line 1) within the file to which the line pointer is to be set.

The editor sets the line pointer to approximately the nnth line within the file. The approximation becomes less exact if many additions and/or deletions have taken place in this area of the file after the INDEX command.

Snn
Is a decimal number consisting of up to 8 digits preceded by an S; it represents the sequence number of the line to which the line pointer is to be set.

The editor sets the line pointer to the line on which the specified sequence number occurs. If it cannot find the specified sequence number, the editor will set the line pointer to the line with the next lower sequence number. This form of the POINT command requires that line number editing be in effect from the time when the INDEX command is entered. See the “nn Command” on page 229 for an easier way of using the POINT Snn function.

Pnn
Is a number from 1 to 32 preceded by a P; it indicates that the line pointer is to be set to the first line of an index table page. The number of lines in a page is defined by the INDEX command.

The editor positions the line pointer to the line associated with the nnth entry within the index. For example, if the index was built with an operand of 100, P1 would point you to the first line in the file, P2 to the 100th, and so on. This form of the command can be used to initially determine where the index pointers are within the file being edited.

If a line representing one of the index points within the file is deleted, the corresponding entry is also deleted from the index thus causing, perhaps, slower responses when pointing to lines immediately beyond the affected point in the file. In this situation, reenter the INDEX command to build a new index.

Using the POINT command to move the line pointer long distances can slow response time. For a large file that has heavy editing activity, consider building an index (as described in the “INDEX Command” on page 192). This can speed up the movement of the line pointer considerably.

Examples

1. `point 1200`

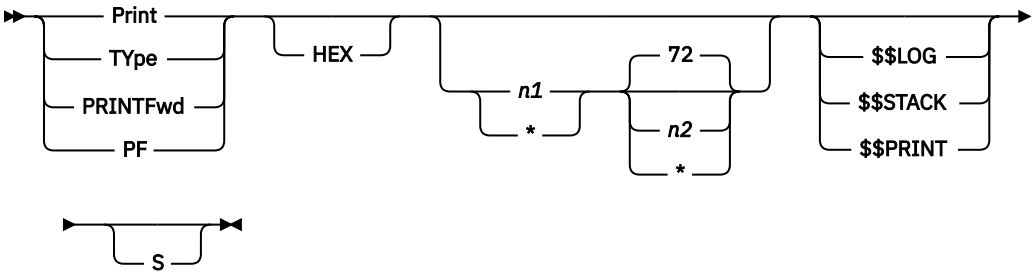
   Causes the line pointer to be set to approximately the 1200th line in the file.

2. `point S198500`

   Causes the line pointer to be set to the line containing sequence number 198500.

PRINT, PF, PRINTFWD, and TYPE Commands

Each of the commands displays lines from the member currently being edited or from the specified area. The data to be displayed can be alphanumeric or mixed data.



HEX
Specifies that lines will be displayed in both character and hexadecimal format.

n1
Is the number of lines to be displayed. You can specify any number from 1 to 99999. If '*' is specified, the remainder of the file up to a maximum of 99999 lines is displayed. The default is one line for a typewriter, and one screen for a display terminal.

n2
Specifies the last column of each line to be printed. The default is the line size value at the time the editor is entered (usually 72). For n2, you can specify any value between 1 and 80. If a width has been set by the VERIFY command, the editor will use this width as the default for n2. If '*' is specified, all 80 columns will be displayed. 'n1' is required if 'n2' is to be specified.

$$LOG
Displays your log area.

$$STACK
Displays the editor stack, which is identical to the punch area.

$$PRINT
Displays the last print output (in the print area) from the last execution.

S
Causes a scale line to be printed, which is useful if you are entering fixed format data.

The command displays n1 lines, starting with the current line.

The position of the line pointer is not changed by the PRINT command. If you use PRINTFWD or PF, however, the editor advances the line pointer as the display proceeds. This is useful for scrolling through files.

If more lines are requested than fit on one screen, pressing ENTER gives you the next screen. As an alternative, you may place the terminal in continuous output mode with the /CONTINU command. This will display the remaining output automatically.

You can use the /SKIP command to forward or backward space the display if the total lines exceed one screen.

The /CANCEL command (or PA2 on the IBM 3270) can be used to terminate a display in progress. Your terminal then returns from list to edit mode.

Note:

1. The PRINT command displays data in 80-character record format.
2. The PRINT command does not use the last line of your screen if the default values of n1 and n2 are used. In this case, the last line is reserved for messages.

Examples

1. Display the contents of an object deck in both character and hexadecimal formats.

```
===> print hex_
<<..+....1....+....2....+....3. .5....+.. MEM=TPROGOBJ>>..+..FS
***** TOP OF FILE *****                                    /***/
 ESD            TPROG           IJ2M0011                  *****
 TXT              K  Ys K    h&         &  D      s  1    *****
 TXT                   Dq       $ s    u      s g   n    *****
 TXT   Y                    s                             *****
 TXT                   &         &                        *****
 TXT              30     00 k      00 ¢         j     0   *****
```

```
 ...+....1....+....2....+....3. .5....+....6....+....7....+..LS
 ESD            TPROG           IJ2M0011
0CEC444444034400EDDDC4440007400 CDFDFFFF0000400344444444
2524000000000001379670000008002 912400110000000800000000
 TXT                K  Ys K    h&         &  D      s  1
0EEE4007440344000AD1AEA2D3A0A85 009ED01A5DAC4DAC51A25F01
2373000800080001502708 2E2710180 AE0A0C8F00141010802A8100
 TXT                   Dq       $ s    u      s g   n
0EEE400B440344004E005DAC9ED00F9 50A042AA111250A082A29420
2373000000080001 5F0C80148A0C7E0 B02A100482A0802E700E5000
 TXT   Y                    s
0EEE400E4403440048A212420051A05 008000000002000000000000
2373000800080001 7002B1120180268 000000040010000000003000
 TXT                   &
0EEE400244014400000B00050000000 444444444444444444444444
2373001000020001901200000000000 000000000000000000000000
 TXT               30     00 k      00 ¢         j     0
0EEE400044034400003FF03004FF3901 124FF24212421100981041F3
2373000000080002A230A200700C290 08700EA008000EA01002700A
 TXT                K     k     0  00;    .
0EEE400344034400000FD0111290110 F44FF5421242100F44444444
2373000800000002A77E210E082A08A 06700E8008B0007E00000000
 TXT                              &         &
```

2. Display the $$LOG area, a typical use of the PRINT command. This example shows a somewhat unusual use of the COPY macro. Assume you modified the COPY macro as indicated by the arrows below:

```
    ===> _
    <<..+....1....+....2....+....3. .5....+.. MEM=COPY    >>..+..FS
    ***** TOP OF FILE *****                                  /***/
    @MACRO    COPY (# LINES,DIRECTION,DISPLACEMENT)         *****
    @NOPR     E.G. @COPY 3 DOWN 5  OR  @COPY 2 LOC  XYZ     *****
    STACK OPEN                                              *****
==> SET LOG ON INOUT                                        *****
==> SET LOG ON INOUT ------------------------               *****
    STACK &&PARAM1                                          *****
    STACK CLOSE                                             *****
    &&PARAM2 &&PARAM3                                       *****
==> SET LOG OFF                                             *****
    GETFILE $$STACK                                         *****
==> PRINT $$LOG                                             *****
    ***** END OF FILE *****                                  *****
```

Now open an editing session for member TEST2. You want to copy the portion from top of file down to the first line that contains 'ENDFROM' to the place where 'BEGINTO' occurs first.

The PRINT $$LOG command that you inserted in the COPY macro causes a display of the $$LOG area as shown in the second screen below (it must be read from bottom to top). The arrows to the left of that screen point to important messages:

- The STACK IS FULL message indicates that the copy operation did not complete successfully.
- The string ENDFROM was not found within the first 99 lines. Also, the target location was not found. This is indicated by the message *EOF.

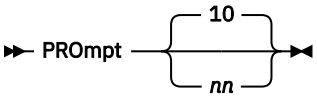
Press PA2 (cancel) and ENTER to return your terminal to edit mode.

```
===> @copy /endfrom/ locate /beginto/
<<..+....1....+....2....+....3. .5....+.. MEM=TEST2   >>..+..FS
***** TOP OF FILE *****                                  /***/
RECORD    00001                                         *****
RECORD    00002                                         *****
RECORD    00003                                         *****
RECORD    00004                                         *****
RECORD    00005                                         *****
```

```
_
...+....1....+....2....+....3. .5....+....6....+....7....+..LS
I 15/28 ED   SET LOG OFF
O 15/28      *EOF
I 15/28 ED   LOCATE   /BEGINTO/
I 15/28 ED   STACK CLOSE
O 15/28      STACK IS FULL
I 15/28 ED   STACK /ENDFROM/
O 15/28      *CONTROL SET
I 15/28 ED   SET LOG ON INOUT ------------------------
O 15/28      *CONTROL SET
I 15/20 ED   SET LOG OFF
```

PROMPT Command

The PROMPT command is used to set the current prompt increment and to change this increment from its default to a different value.



nn

For nn, specify a number from 1 to 32767 indicating the desired prompt increment. The specified increment is used by the editor when, during line number editing, it assigns line numbers to lines entered via an INPUT command.

A PROMPT command with no operands causes the current prompt increment to be displayed.

Example

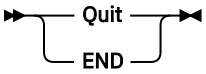
Insert two lines behind line 70 with a prompt increment of 3.

```
===> 70;prompt 3;I print ' ';i print ' ';u 2_
....+<<..1....+....2....+....3. .5....+.. INP=*INPARA* ...+..F>
***** TOP OF FILE *****                                  /***/
00010PRINT 'ENTER THREE NUMBERS SEPARATED WITH COMMAS'   *****
00020PRINT 'ENTER /* TO TERMINATE'                       *****
00030L = I + J + K                                       *****
00040M = I * &SQR2                                       *****
00050N = J * &PI                                         *****
00060PRINT I,J,K                                         *****
00070PRINT L,M,N                                         *****
00080PRINT 'END OF CALCULATION'                          *****
00090GO TO 10                                            *****
00100END                                                 *****
***** END OF FILE *****                                  *****
```

```
===>
00070PRINT L,M,N                                        /===/*
00073PRINT ' '                                          *****
00076PRINT ' '                                          *****
00080PRINT 'END OF CALCULATION'                         *****
00090GO TO 10                                           *****
00100END                                                *****
***** END OF FILE *****                                 *****
```

QUIT Command

The QUIT (or END) command ends editing for the file associated with the logical screen on whose command line it is entered.



All storage areas associated with the file being edited are released, including the logical screen associated with the file. This logical screen will then be available to another file.

If there is only one file being edited (that is, all previous files accessed via the ENTER command have been QUIT), the command terminates the full-screen editor and returns your terminal to command mode.

If the QUIT command is used for a library member, the changes made to the member will not be affected, since these have been applied directly to the member.

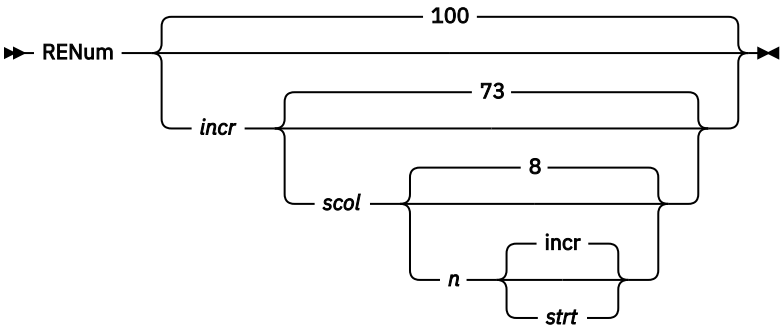
If you enter the QUIT command for a newly created file, the file will be lost, unless you had previously saved it with the SAVE command or, preferably, with the FILE command.

If the QUIT command is used for the file associated with the input area, the contents of the input area are not saved. However, these contents are still available immediately after the full-screen editor session has ended. To retrieve these contents again (for editing or saving, for example), simply enter the “ED Macro” on page 48.

Note: If a QUIT or a FILE command is followed by other editor commands separated with logical line-end characters, these other commands will be applied to a different file.

RENUM Command

The RENUM command is used to resequence or renumber the records of the file being edited. The command controls the location and size of the sequence number and the increment.



incr
Is the increment by which the file is resequenced. The default is 100, unless you are using line number editing. In this case, the default is ten times the prompt, or 100, whichever is greater. The maximum increment is 9999.

scol
Is the starting column number for the sequence number field. The default is column 73, unless line number editing is in effect. In this case, the editor uses the location of the sequence number as defined by the LINEMODE command. When you are editing mixed data, only 1 and 73 are valid starting columns.

n
Is the number of columns in the sequence number field. The default is 8, unless you are using line number editing. In this case, the editor uses the location and size of the line number field as defaults. The maximum number of columns is 16. If you resequence a DBCS member and 'scol' has a value of 73, n must be 8.

strt
Is the starting sequence number. If omitted, the starting sequence number is assumed to be the same as the sequencing increment. The maximum value that you can specify is 9999. Refer to the last of the **Notes** under the "/RESEQ Command" on page 87 for a method of how to establish a starting sequence number that is higher than 9999.

The line pointer is not altered by the renumbering operation.

If an index is in effect when the RENUM command is issued, its effect will be negated, and the INDEX command must be reissued. For more information about this command, see the section "INDEX Command" on page 192.

Note: A statement will not be resequenced if the sequence number begins at a column below 73 and either:

The first column contains a slash (/), or
The first five columns contain the string '* $$ '.

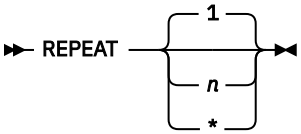
Example

```
===> linemode left;renum 10;top_
<<..+....1....+....2....+....3. .5....+.. MEM=CALC2   >>..+..FS
***** TOP OF FILE *****                                  /***/
00010PRINT 'ENTER THREE NUMBERS SEPARATED WITH COMMAS'   *****
00020PRINT 'ENTER /* TO TERMINATE'                       *****
00030L = I + J + K                                       *****
00040M = I * &SQR2                                       *****
00043N = J * &PI                                         *****
00046PRINT I,J,K                                         *****
00050PRINT L,M,N                                         *****
00060PRINT 'END OF CALCULATION'                          *****
00070GO TO 10                                            *****
00080END                                                 *****
```

```
===>
....+<<..1....+....2....+....3. .5....+.. MEM=CALC2    ...+..F>
***** TOP OF FILE *****                                  /***/
00010PRINT 'ENTER THREE NUMBERS SEPARATED WITH COMMAS'   *****
00020PRINT 'ENTER /* TO TERMINATE'                       *****
00030L = I + J + K                                       *****
00040M = I * &SQR2                                       *****
00050N = J * &PI                                         *****
00060PRINT I,J,K                                         *****
00070PRINT L,M,N                                         *****
00080PRINT 'END OF CALCULATION'                          *****
00090GO TO 10                                            *****
00100END                                                 *****
```

REPEAT Command

The REPEAT (or RPT) command causes the next command to be processed the specified number of times if that command is one of the following: ALIGN, BLANK, CENTER, JUSTIFY, OVERLAY, and SHIFT. The operands of the RPT command are the same as those shown for the REPEAT command below.



n

Is a number from 1 to 99999, indicating how often the next command is to be repeated. If n is greater than the number of lines between the current line and end of file, REPEAT remains in effect until the end of the file.

*****

Indicates that the operation requested by the next command is to proceed from the line pointer to end of file, or to the maximum number of repetitions (9999).

REPEAT remains in effect until the edit session is ended, or until you enter an ALIGN, BLANK, CENTER, JUSTIFY, OVERLAY, or SHIFT command. The repeat value is then reset to 1.

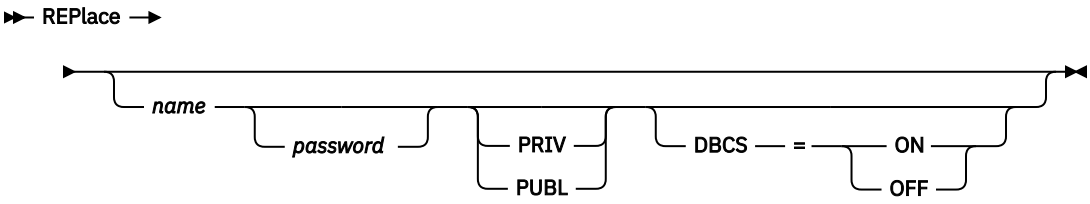
Example

```
===> set num on;zone 1 80;repeat *;blankc73 --------;top_
<<..+....1....+....2....+....3. .5....+.. INP=*INPARA*>>..+..FS
***** TOP OF FILE *****                                 /***/
Unfortunately, this text member has been entered        000100
with 'LINEMODE RIGHT'; of course, the sequence          000200
numbers in column 73-80 must be removed. This is        000300
a good application for the REPEAT command.              000400
```

```
===> _
<<..+....1....+....2....+....3. .5....+.. INP=*INPARA* ...+..F>
***** TOP OF FILE *****                                 /***/
Unfortunately, this text member has been entered
with 'LINEMODE RIGHT'; of course, the sequence
numbers in column 73-80 must be removed. This is
a good application for the REPEAT command.
```

REPLACE Command

The REPLACE command replaces an existing library member with the contents of the input area or with a newly created file.



name
Is the name of the library member to be replaced.

If the command is entered for a newly created file (see the section “ENTER Command” on page 182), this name will override the name specified in the ENTER command. If you do not specify a name, the editor will take the name that you specified with the ENTER command.

password
Needs to be specified if the member being replaced is password-protected.

PRIV
PUBL
Can be specified if you want to assign the private or public attribute to the replacement member. If neither PRIV nor PUBL is specified, the previous privacy attribute of the member is retained.

DBCS=ON
DBCS=OFF
Allows you to set or reset the DBCS attribute for the new member. If the DBCS operand is not specified, the member will have the attribute that was set during the editor session by a SET DBCS command. If a SET DBCS command has not been specified either, the attribute of the old member is retained.

Example

Edit the input area, put some records into this area, and replace member MEMBRA with the contents of the input area.

```
ed
get membrb  1 50
replace membra
```

RESTORE Command

The RESTORE command restores the editor settings that were in effect when the previous STACK EDIT command was issued.

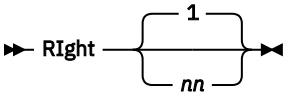
Several levels of STACK EDIT and RESTORE can be nested so that settings can be saved and changed, saved and changed, and so on, and then restored, restored, and so on.

The settings for the following editor commands are restored:

```
CASE            SET (for control characters only)
FLAG            TABSET
IMAGE           VERIFY
LINEMODE        ZONE
DELIM
```

RIGHT Command

The RIGHT command shifts the displayed data to the right within a logical screen.



nn
nn is a number from 1 to 149, indicating by how many columns the displayed data is to be shifted to the right.

It is often necessary to view columns of data which are not displayed on the screen. There are two ways to do this:

- By using the VIEW command to display only certain columns (see the VIEW command).
- By shifting the existing data right (or left) using the RIGHT (or LEFT) command.

No loss of data occurs as a result of this command; just enter a LEFT command specifying the same number for nn. This causes the data to be returned to its previous position.

The number of columns shifted to the right from the normal or base position is called the offset. If a nonzero offset exists, it will be displayed in the scale/header line as a reminder. To return the display to the original position, issue a LEFT command for as many columns as are indicated in the offset.

When you shift mixed data and when a double-byte character would occupy the rightmost position on the screen after the shift operation, this double-byte character will not be displayed. It is replaced by one or two control characters, which guarantee that the remainder of the subfield is displayed correctly. The data in the VSE/ICCF library is not affected by this operation, as long as you do not:

- Replace the added control characters
- Change double-byte characters to one-byte characters.

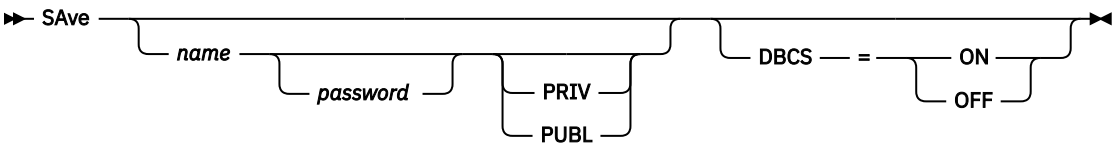
When you shift mixed data, and an SO control character would occupy the rightmost position on the screen after the shift operation, the control character will be displayed as an unprintable character.

RPT Command

See the section “REPEAT Command” on page 210.

SAVE Command

The SAVE command is used to save a newly created file, or to save the input area as a new library member.



name
Is a string of 1 to 8 characters beginning with a letter of the alphabet. This string becomes the name of a newly created file. It becomes the name of the member created by VSE/ICCF if you save the contents of your input area. To save that contents, your SAVE request must include a name.

For a newly created file, you need not specify a name. In this case the file is saved in the library by the name that you specified in the ENTER command (see "ENTER Command" on page 182). However if you do specify a name, this name overrides the one that you supplied with the ENTER command.

The characters .P at the end of the member name indicates a print-type member. These characters are part of the member name.

If the member may be submitted to VSE/POWER, do not use any of the names listed in restriction "6" on page 331.

password
Is a 4-character word which you can specify to password-protect a newly created file that is to be saved in the library.

PRIV
PUBL
Allows you to specify that a newly created member is to be saved in the library as private or public. If neither operand is specified, the private or public status will be set according to the default in your user profile.

DBCS=ON
DBCS=OFF
Allows you to save the new member with the DBCS attribute. If the DBCS operand is not specified, the new member is saved with the attribute that was set during the editor session by a SET DBCS command. If a SET DBCS command has not been specified either, DBCS=OFF is assumed for the new member.

After the editor has processed the SAVE command, you can continue editing the file; however, instead of editing a newly created file or the input area, you will be editing a library member.

If the SAVE command is used for the input area, the 'name' operand must be specified and the contents of the input area will be saved in your library by this name.

If the SAVE command is used for a file which is already in the library, the 'name' operand must not be specified. Also, the editor will not perform a save operation because all changes that you have made to the member have been applied directly to the member. However, the SAVE command will cause any buffers retained in storage by VSE/ICCF to be written to disk, thus protecting you against a loss of data should a system failure occur.

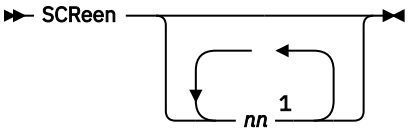
Example

Edit the input area, put some records into this area, save the contents of this area in the library, and continue editing that saved contents as a member named NEWFIL.

```
ed
get membrb 1 50
save newfil
top
```

SCREEN Command

The SCREEN command is used to specify the number and size of logical screens within the physical screen. It also displays the current setting of logical screens.



Notes:

[1] You can define up to eight logical screens.

nn
Is a number representing the number of lines for a logical screen. The first nn for the first logical screen, the second nn for the second logical screen, and so on. You can specify up to eight numbers, representing up to eight logical screens.

If the SCREEN command is entered with no operands, the current setting of the SCREEN command will be displayed in the first command area associated with the file being edited.

If the total number of lines specified in all logical screens is less than the actual physical screen size, the remaining lines will be assigned to the final logical screen. This number of remaining lines must meet a minimum requirement.

The minimum number of lines in a given logical screen depends upon the number of physical lines associated with the command area. A minimum logical screen must contain two lines, one scale/header line, and one command area. The command area can be from 1 to 4 physical lines as determined by the VERIFY command (for more information about this command, see the section "VERIFY Command" on page 223).

If the number of logical screens defined by the SCREEN command is greater than or equal to the previous number of logical screens, the files displayed on the screen will remain on the screen. However, if the screen sizes themselves are smaller than the previous screen sizes, the format specifications (see FORMAT command) can be adjusted so that the format areas fit within the logical screen.

Format areas will automatically be reduced to fit within a smaller logical screen but will not automatically be expanded to encompass a larger logical screen.

If the number of logical screens defined by the SCREEN command is greater than the previous number of logical screens, the extra new screens will be assigned when the ENTER command is issued for files not displayed on the screen.

If the number of logical screens defined by the SCREEN command is less than the previous number of logical screens, the first available screen will be assigned to the file associated with the logical screen where the SCREEN command was entered. Any subsequent screens will be assigned to files not on the screen, beginning with the oldest file.

Note:

1. Because commands are processed from top to bottom on the screen, any editor commands following the SCREEN command itself will be based on the new setting of the SCREEN control values. Thus, when you change your SCREEN specifications, the SCREEN command should be the only command that you typed before pressing ENTER.
2. The full-screen editor controls its own screen formatting. Thus, all [/]SET SCREEN command specifications are ignored while the full-screen editor is in control.

Example

Split the screen into two logical screens. Use the lower screen to edit MEMBER2. After having completed editing MEMBER2, go back to a display of one logical screen only (by issuing `screen 24`).

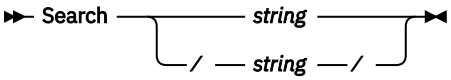
```
===> screen 12 12;enter member2_
<<..+....1....+....2....+....3. .5....+.. MEM=MEMBER1 >>..+..FS
***** TOP OF FILE *****                                    /***/
RECORD  1  OF MEMBER1                                     *****
RECORD  2  OF MEMBER1                                     *****
RECORD  3  OF MEMBER1                                     *****
RECORD  4  OF MEMBER1                                     *****
RECORD  5  OF MEMBER1                                     *****
RECORD  6  OF MEMBER1                                     *****
RECORD  7  OF MEMBER1                                     *****
RECORD  8  OF MEMBER1                                     *****
RECORD  9  OF MEMBER1                                     *****
```

```
===>
<<..+....1....+....2....+....3. .5....+.. MEM=MEMBER1 >>..+..FS
***** TOP OF FILE *****                                    /***/
RECORD  1  OF MEMBER1                                     *****
RECORD  2  OF MEMBER1                                     *****
RECORD  3  OF MEMBER1                                     *****
RECORD  4  OF MEMBER1                                     *****
RECORD  5  OF MEMBER1                                     *****
RECORD  6  OF MEMBER1                                     *****
RECORD  7  OF MEMBER1                                     *****
RECORD  8  OF MEMBER1                                     *****
RECORD  9  OF MEMBER1                                     *****
===> quit;screen 24_
<<..+....1....+....2....+....3. .5....+.. MEM=MEMBER2 >>..+..FS
***** TOP OF FILE *****                                    /***/
RECORD NUMBER  1 OF MEMBER2                               *****
RECORD NUMBER  2 OF MEMBER2                               *****
RECORD NUMBER  3 OF MEMBER2                               *****
```

```
===> _
<<..+....1....+....2....+....3. .5....+.. MEM=MEMBER1 >>..+..FS
***** TOP OF FILE *****                                    /***/
RECORD  1  OF MEMBER1                                     *****
RECORD  2  OF MEMBER1                                     *****
RECORD  3  OF MEMBER1                                     *****
RECORD  4  OF MEMBER1                                     *****
RECORD  5  OF MEMBER1                                     *****
RECORD  6  OF MEMBER1                                     *****
RECORD  7  OF MEMBER1                                     *****
RECORD  8  OF MEMBER1                                     *****
RECORD  9  OF MEMBER1                                     *****
RECORD 10  OF MEMBER1                                     *****
RECORD 11  OF MEMBER1                                     *****
```

SEARCH Command

The SEARCH command scans the characters of each line (as defined by the column suffix or ZONE setting) from the beginning of the file through the end of file for the specified character string.



string
/string/

Is any group of characters to be searched for in the file. If 'string' includes double-byte characters and if the SI and/or SI control character(s) occupy the leftmost and/or rightmost position in it, they are not considered part of 'string'.

If one or more blanks are part of the string at its right end, you must use the slash (/) as a string delimiter.

The column suffix can be used with this command.

This command acts like a TOP command followed by a LOCATE command. In other words, the SEARCH command always starts at the top of the file and thus searches the entire file.

If a match for the specified string is located, the editor positions the line pointer at the line that contains the string. If a match is not located, the editor positions the line pointer at the last line of the file and displays the message *E0F.

The request is not column-dependent because all characters are scanned as specified by the ZONE request or the column suffix. For column-dependent scans, see the section "FIND Command" on page 185; for current line to end of file scans, use the LOCATE command (see "LOCATE, LOCNOT, and LOCUP Commands" on page 199).

Examples

1. Find the string 'FIRST' with the search beginning at the top of the file.

```
===> search first_
<<..+....1....+....2....+....3. .5....+.. INP=*INPARA*>>..+..FS
THIRD RECORD                                              /===/*
FOURTH RECORD                                             *====*
```

```
===> _
<<..+....1....+....2....+....3. .5....+.. INP=*INPARA*>>..+..FS
FIRST RECORD                                              /===/*
SECOND RECORD                                             *====*
THIRD RECORD                                              *====*
FOURTH RECORD                                             *====*
***** END OF FILE *****                                   *****
```

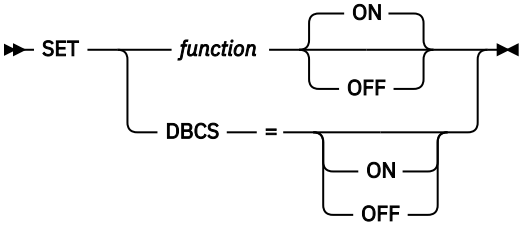
2. Locate the three double-byte characters which are specified in the search argument; start the search at the top of the file.

```
*************** Sample 6-Part 1 ********************

===> SEARCH ■1番目■
<<..+....1....+....2....+....3....+....4....+....5....+.. MEM=MEMBER1 >>DB+..FS
■3番目のレコード■                                                    /===/*
■4番目のレコード■                                                    *====*
***** END OF FILE *****                                            *****
```

```
*************** Sample 6-Part 2 ********************

===>
<<..+....1....+....2....+....3....+....4....+....5....+.. MEM=MEMBER1 >>DB+..FS
■1番目のレコード■                                                    /===/*
■2番目のレコード■                                                    *====*
■3番目のレコード■                                                    *====*
■4番目のレコード■                                                    *====*
***** END OF FILE *****                                            *****
```

SET Command

The SET command is used to set various VSE/ICCF functions on or off. The majority of these options is described under the /SET system command; see "[/]SET Command" on page 100. The following options are available only under the full-screen editor.



function

For function, you can specify one of the options discussed below. The specified option applies to the file being edited so that you can have different option settings for different files. The options are:

DBCS

Allows you to set or reset the DBCS attribute for the member that is being edited.

NULls

Specifies whether or not trailing blanks within a record are replaced with 'null' characters before the record is displayed on the screen. The option is automatically set ON when the full-screen editor is first entered, or when a file is entered for the first time.

The option is useful because it allows data to be added to a record on the screen by using the insert mode of the IBM 3270 display terminal. Also, on remote terminals, the option can reduce transmission time.

When the NULLS option is ON, be sure to use the space bar rather than the right-arrow key for all forward spacing of the cursor (null characters will not be transmitted back to the processor, whereas space characters will).

When the NULLS option is OFF, trailing blanks will be displayed on the screen. Thus, it would be impossible to use the insert mode of the IBM 3270 without first converting trailing blanks to nulls within the record. You do this by using the ERASE EOF key or by using the character-deletion key. The NULLs OFF option is useful when you want to use the CURSOR command for changes beyond the last nonblank data within the record.

NUMbers

When the option is ON, sequence numbers are displayed within the line command area, instead of the normal '*===*' identifier. This option is useful when editing files that have sequence numbers in columns 73 through 80. For these files you need not sacrifice your line command area to see your sequence numbers.

If line number editing is in effect, the sequence number displayed in the line command area will be the rightmost six digits of the line number. If line number editing is not in effect, the data from columns 75 through 80 will be displayed if column 80 is numeric or, if not numeric, the data from columns 74 through 79 will be displayed if column 79 is numeric. Otherwise, the data from columns 73 through 78 will be displayed in the line command area.

When the NUMBERS option is in effect, line commands must begin in column 1 of the line command area and must be followed by a blank (space).

The NUMBERS option normally is off when the full-screen editor is entered, or when a file is later entered for the first time via the ENTER command.

PFC

The option controls the disposition of commands or lines of data associated with PF keys. The option is ON, except when set off explicitly. When the option is ON, the data associated with a PF key is assumed to represent a command. It is processed as if the command had been entered in the command area associated with the current cursor position. The cursor need not be positioned in the command area when you press the PF key.

When the option is OFF, the data associated with the PF key (except for CURSOR commands) is interpreted according to the position of the cursor: if the cursor is in the command area, the processing of the PF key is the same as if the option were on; if the cursor is placed within a line, the data associated with the PF key replaces that line.

REPoption

The option controls the interpretation of modified lines from the screen. The option is OFF, except when set ON explicitly.

When the REPOPTION option is ON, the checking for tab characters is bypassed and all data is treated alike as modifications to the records on the screen. This does not mean, however, that the tab characters themselves will not be interpreted.

When the REPOPTION is OFF and a line on the screen is modified, and if the line contains no logical tab characters (or logical tabbing is not in effect), the line is interpreted as a modification to an existing line in accordance with the VIEW, ZONE, and LEFT/RIGHT specifications. In other words, if data within a line is modified, only that data is modified in the file, unless the modified data is outside the current zone. On the other hand, if the line is found to contain logical tab characters and tabbing is in effect (tab positions are set), the input is interpreted as a complete replacement line in a column 1 through 80 format regardless of the VIEW or LEFT/RIGHT

specifications. This helps you enter new lines to the file via the LADD editor command or the editor-line command A, regardless of VIEW or LEFT/RIGHT settings.

ON
OFF
Sets the function on or off, respectively.

The following editor command versions of the [/]SET system command (see "[/]SET Command" on page 100) will be frequently used in the full-screen editor environment:

SET PFnnED
Set the meaning of a PF key in the edit set.

SET END=char
Set the logical line end character.

SET ESC=char
Set the escape character.

SET TAB=char
Set the logical tab character.

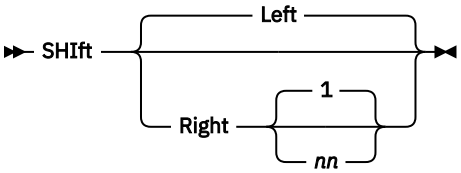
SET BYPASS
Set control-character interpretation ON or OFF.

SET DATANL
Set the APL or Text keyboard feature ON or OFF.

SET LOG
Set logging ON or OFF.

SHIFT Command

The SHIFT command shifts the data within the current zone to the left or right a given number of columns.



Left
Right
Indicates the direction of the shift.

nn
Indicates by how many columns the data is to be shifted.

Only the data within the zone defined by the ZONE command or the column suffix, if present, participates in the shift operation. Any data shifted out of the zone is lost. Space characters (blanks) are inserted to replace data shifted out of the zone. If the shift amount is greater than or equal to the size of the zone, the entire zone will end up as spaces.

The column suffix can be used with this command.

If the data within several lines is to be shifted, you can enter the REPEAT command prior to the SHIFT command to indicate the number of lines that are to participate in the operation.

Example

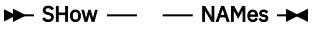
Shift three lines to the left by 10 character positions. Note how the REPEAT command is used to determine the number of executions of the SHIFT command.

```
===> next;repeat 3;shift left 10;top_
<<..+....1....+....2....+....3. .5....+.. MEM=SHIFTTXT>>..+..FS
This meaningless text example contains a few lines       /===/*
                  which were indented much too           *****
                  far such that they disturb the         *****
                  nice appearance of the document.       *****
We shall move them left using the 'SHIFT' command.       *****
```

```
===> _
<<..+....1....+....2....+....3. .5....+.. MEM=SHIFTTXT>>..+..FS
***** TOP OF FILE *****                                  /***/
This meaningless text example contains a few lines       *****
         which were indented much too                    *****
         far such that they disturb the                  *****
         nice appearance of the document.                *****
We shall move them left using the 'SHIFT' command.       *****
```

SHOW Command

The SHOW command displays all settings that can be set via the /SET system command. Under the full-screen editor, the command gives you in addition a display of the names of the files that are being edited.



The display:

1. Gives the name of each file that is being edited; that is, the name of any file that was specified via the /EDIT or ENTER commands and has not yet been quit or filed.
2. Indicates whether the file is:
   - A member of the library (LIB)
   - A newly created file (NEW)
   - The contents of the input area (INP).

   In addition, the display indicates whether the file is presently associated with a logical screen (YES in the third column of the display line).

After having received the response to the SHOW command, press ENTER to restore the screen as formatted by the full-screen editor.

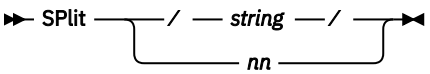
Example

```
===> show names_
<<..+....1....+....2....+....3. .5....+.. MEM=MEMBER2 >>..+..FS
***** TOP OF FILE *****                                  /***/
RECORD NUMBER  1 OF MEMBER2                              *****
RECORD NUMBER  2 OF MEMBER2                              *****
RECORD NUMBER  3 OF MEMBER2                              *****
```

```
_
...+....1....+....2....+....3. .5....+....6....+....7....+..ED
*
NAME     TYPE SCREEN
*
MEMBER1   LIB
MEMBER2   LIB   YES
NEWPROG   NEW
```

SPLIT Command

The SPLIT command is used during text editing to split the current line into two lines.



/string/
Is the character string in front of which the line split is to occur. The first occurrence of the specified string in a line becomes the start of the next line.

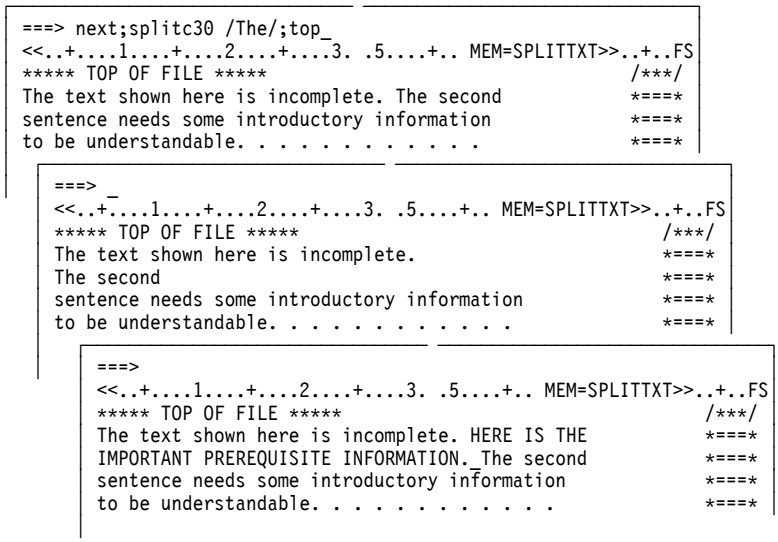
nn
Is the column number in front of which the line split is to occur. For nn, you can specify any number from 1 to 80. The data at the specified column becomes the start of the second line.

The column suffix can be used with this command.

All data up to the split point is retained as the first line. The data in the new (split) line is aligned in the current zone.

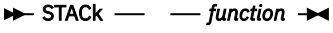
Example

Split the current line at the first occurrence, in columns 30 and above, of the string 'The'.



STACK Command

The STACK command places commands or data into the editor stack. The command can be used also to control the placement of data within the stack and to preserve editor settings.



function
For function, you can specify one of the following:

OPEN
Allocates and logically opens the stack, if it does not exist already. Sets the line pointer of the stack to the top, so that the next line of input to the stack will be the first line.

CLOSE
Logically closes the stack by marking the end with a terminator. If more commands are written to the stack, the terminator is overwritten.

BACK
BACK mm
Moves the line pointer of the stack back mm lines (the default for mm is 1). This operand can be used to delete entries from the stack. For example, if you write an incorrect line to the stack, STACK BACK would cause the following line(s) to overlay the incorrect one.

EDIT
Writes to the stack a line containing editor settings, control characters, and tab positions. Command settings that can be stored are listed below:

```
CASE              TABSET
DELIM             VERIFY
FLAG              ZONE
IMAGE             SET (control characters only)
LINEMODE
```

Use the RESTORE command to restore stacked settings.

/string/
A character string indicating that all lines down to (but not including the line containing `string`) are to be placed into the stack. You can place up to 100 lines into the stack. Unlike the other STACK options, /string/ does not extend the stack if a 'stack full' condition occurs. Thus, an overrun condition does not occur if /string/ is entered incorrectly. The specified character string may include double-byte characters.

nn
0
Causes nn lines (1 to 99) to be placed in the stack, beginning with the current line. Specify 0 to end input mode when you are stacking commands for later execution.

data
Lets you write data and/or commands to the stack. These commands can be executed by specifying the stack as if it were a macro command, for example @$$STACK. 'data' can contain double-byte characters.

The stack is a work area for use during editing. Typically, it is used as a temporary save area when you are moving or copying data (M and C commands) within a file. The stack is also a temporary holding area for editor macros such as @MOVE and @COPY.

The stack is identical with the punch area that is used during executions. Thus, it will still contain the output from the previous interactive partition execution when you enter edit mode. Lines stacked during editing remain in the stack after edit mode is terminated.

Because the stack and the punch area are the same, do not use the stack while editing in asynchronous execution mode (see "/ASYNCH Command" on page 37). Job output could overwrite your stack input.

Example

Store several statements in the stack which are later to be included in an assembler source program. Let's assume that you entered the following sequence of STACK commands:

```
stack open
stack ins          eject
stack ins *-------------------------------------*
stack ins *         &&PARAM1 routine            *
stack ins *-------------------------------------*
stack ins          space 3
stack close
```

Issuing the EDPUN macro gives the following display of the stack area:

```
===> _
<<..+....1....+....2....+....3. .5....+.. MEM=$$PUNCH >>..+..FS
***** TOP OF FILE *****                                  /***/
INS          EJECT                                      *****
INS *-------------------------------------*              *****
INS *          &&PARAM1 ROUTINE           *              *****
INS *-------------------------------------*              *****
INS          SPACE 3                                    *****
```

Now edit member SAMPASMB and, with the @$$STACK macro, retrieve the statements from the stack. Notice how the '/' line command is used to select the 5th line as the insert point. The specification 'readcard' in the @$$STACK macro will be the value of &&PARAM1

```
===> @$$stack readcard;up 5_
<<..+....1....+....2....+....3. .5....+.. MEM=SAMPASMB>>..+..FS
MAINPGM  CSECT                                          *****
         PRINT GEN                                      *****
         BALR  5,0                                      *****
         USING *,5                                      *****
         OPEN  FILOUT                                   /****
LOOPA    EXCP  RDCCB                                    *****
```

```
===> _
<<..+....1....+....2....+....3. .5....+.. MEM=SAMPASMB>>..+..FS
         OPEN  FILOUT                                  /===/*
          EJECT                                        *****
*--------------------------------------*                *****
*          READCARD ROUTINE            *                *****
*--------------------------------------*                *****
          SPACE 3                                       *****
LOOPA    EXCP  RDCCB                                    *****
```

STATUS Command

See the section "[/]SHOW Command" on page 118.

TABSET Command

See the section "[/]TABSET Command" on page 130.

TOP Command

The TOP command positions the line pointer to the top of the file (to the null line in front of the first line in the file).

An automatic TOP is performed by the FIND, LOCATE, and CHANGE requests if an end-of-file condition immediately preceded the request. An automatic TOP is also performed by the SEARCH command.

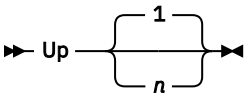
You can use the INPUT or INSERT commands immediately after the TOP command to insert new lines before the first line in the file.

TYPE Command

See the section "PRINT, PF, PRINTFWD, and TYPE Commands" on page 205.

UP Command

The UP command repositions the line pointer 'n' lines before the current line.



n

Indicates the number of lines by which the pointer is to be moved back. You can specify a number from 1 to 99999.

If top of file is reached before the line pointer has moved up n lines, the pointer is positioned at the first line and the `INVALID RANGE` message appears.

After the command has been executed successfully, the new current line is displayed.

To move the pointer to the top of the file, issue the TOP command.

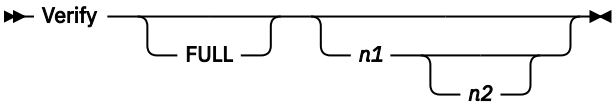
Example

To reposition the line pointer nine lines above the current line, issue:

```
u 9
```

VERIFY Command

The VERIFY command is used primarily to vary the settings of two options: (1) the number of lines displayed prior to the current line, and (2) the number of physical lines associated with the command area.



FULL
This operand is optional. Its purpose is to retain format compatibility with the VERIFY command issued from the context editor.

n1
Can be any number from 1 to 16. The number gives the number of lines (logical records) to be displayed prior to and including the current line within a format area. Specifying '1' would indicate that no record prior to the current line is to be displayed.

Assume a format area containing 21 records. This means that the current line and the next 20 lines are displayed. However, if you specify n1 as 9, the editor displays eight records before and 12 records after the current line.

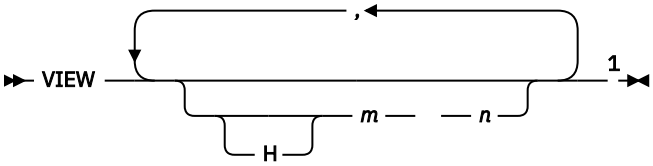
n2
Is a number from 1 to 4, indicating the number of physical lines to be associated with the command area for the file being edited.

The VERIFY command displays the complete contents of the physical screen regardless of the specified operands.

Note: It is possible to specify a value for n1 such that the current line does not appear on the screen. For example, if a format area contains space for 10 records and the n1 value is specified as 12, the current line is actually two lines beyond the end of the format area and thus not on the screen.

VIEW Command

The VIEW command is used to rearrange, format, and view the 80-character record on the screen.



Notes:

[1] Up to 12 fields can be specified.

H
Indicates that the field is to be displayed in hexadecimal format. Any modifications to the field as it is displayed on the screen must also be made in hexadecimal format.

Note that a hexadecimal format field will require twice as many columns on the screen as there are columns within the actual field. If the total length of the viewed fields exceeds the length of one output line, use the FORMAT command to increase the number of output lines per displayed record.

m
Is a number from 1 to 80, specifying the first column of the field that is to be displayed. If m is 0, a filler field of n blanks is assumed.

n
Is a number from 1 to 80, specifying the ending column number of the field to be displayed. However, if m is 0, n is the number of blanks to be inserted. For example, the command

```
VIEW 1 10,0 5,11 20,0 5,21 40
```

causes three fields to be displayed separated from one another by 5 blank columns.

The VIEW command lets you:

- View only certain columns of the 80-character record on the screen. For example, if you want to view and modify only columns 41 through 80 of each record, specify `VIEW 41 80`.
- Rearrange the order in which data is displayed. For example, to view columns 41 through 80 followed by columns 1 through 40, specify `VIEW 41 80,1 40`.
- View certain fields in hexadecimal format rather than in character format.
- Place blank filler fields between fields that are to be displayed on the screen.

If the VIEW command is entered with no operands, the current VIEW settings are displayed in the command area associated with the logical screen.

The normal VIEW specification when a file is entered is `VIEW 1 80`, which tells the editor that columns 1 through 80 are to be displayed in character format. Thus, the default view contains one 80-character field.

Up to 12 fields can be specified. Fields can overlap, in which case, if you want to modify an overlapping field, you must make the modification to the rightmost occurrence of the field for the modification to be effective.

The VIEW specifications apply to the file being edited when the VIEW command is entered. Thus, different files being edited at the same time can all have different VIEW specifications.

The VIEW command displays mixed data correctly, as long as

- Only the first field requests the display of characters.
- All fields following the first field are hexadecimal or filler fields.

If a subfield of double-byte characters exceeds the size of the specified area, the leftmost or rightmost double-byte character will be replaced by one or two control characters, which guarantee that the remainder of the subfield is displayed correctly. The data in the VSE/ICCF library is not affected by this modification, as long as you do not:

- Replace the added control characters
- Change double-byte characters to one-byte characters.

Examples

1. View columns 10 through 80, thus making visible the information that is hidden behind the line command area.

```
===> view 10 80_
<<..+....1....+....2....+....3. .5....+.. MEM=INVENTRY ...+..F>
***** TOP OF FILE *****                                    /***/
ITEM1 DESCRIPTION       5   PUMPS & PIPES LTD., HUSTON   *****
ITEM2 DESCRIPTION      28   JAMES & JONES, NEW YORK      *****
ITEM3 DESCRIPTION       2   MODERN PLASTICS, DALLAS      *****
ITEM4 DESCRIPTION    2378   CANADIAN PLYWOOD FACTORY, TO *****
ITEM5 DESCRIPTION   28355   CHEMICAL INSTRUMENTS, ALABAM *****
ITEM6 DESCRIPTION     289   MASSATRONICS LTD., MASSACHUS *****
```

```
 ===> _
 <<..+....1....+....2....+....3. .5....+.. MEM=INVENTRY ...+..F>
 ***** TOP OF FILE *****                                   /***/
CRIPTION        5   PUMPS & PIPES LTD., HUSTON            *****
CRIPTION       28   JAMES & JONES, NEW YORK               *****
CRIPTION        2   MODERN PLASTICS, DALLAS               *****
CRIPTION     2378   CANADIAN PLYWOOD FACTORY, TORONTO     *****
CRIPTION    28355   CHEMICAL INSTRUMENTS, ALABAMA         *****
CRIPTION      289   MASSATRONICS LTD., MASSACHUSETTS      *****
```

2. Display part of an object module in EBCDIC and hexadecimal format.

   First the screen is formatted to reserve three screen lines for every line in the file. Then the VIEW command is issued to segment the three lines as follows:

   a. Columns 1-72 and 74-80 in EBCDIC in the first line (`view 1 72,74 80,...`). Column 73 is sacrificed in order to make the sequence number fully visible.

   b. In the second line, the storage address (`...,H6 8,...`) plus the contents in columns 17-46 (`...,H17 46,...`).

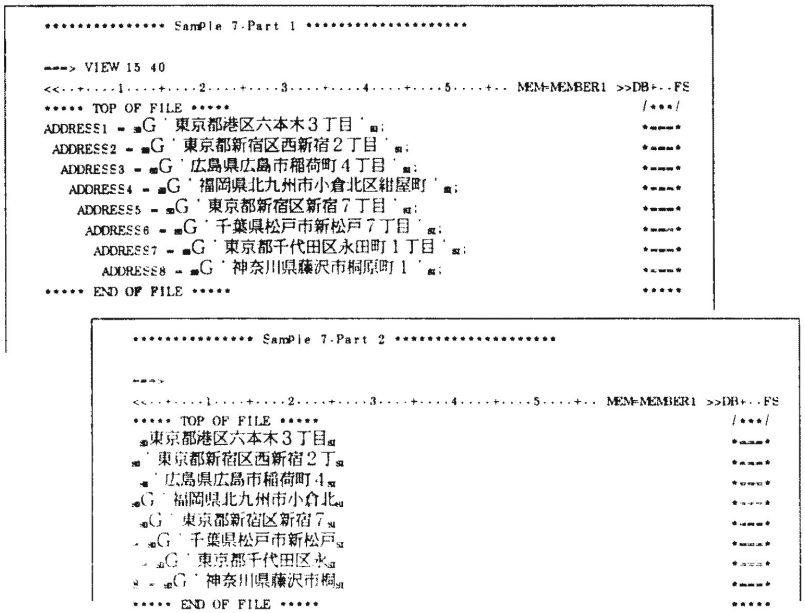
   c. In the third line the ESDID (`...,H15 16,...`) plus the contents in columns 47-72 (`...,H47 72`).

```
===> format 3_
<<..+....1....+....2....+....3. .5....+.. MEM=TPROGOBJ>>..+..FS
 ESD            TPROG           IJ2M0011                  /===/*
 TXT              K  Ys K    h&         &  D      s  1    *****
 TXT                   Dq       $ s    u      s g   n     *****
 TXT   Y                    s                              *****
```

```
 ===> VIEW 1 72,74 80,H6 8,0 4,H17 46,0 12,H15 16,0 4,H47 72
 <<..+....1....+....2....+....3. .5....+.. MEM=TPROGOBJ>>..+..FS
 ***** TOP OF FILE *****

                                                          /***/
  ESD            TPROG           IJ2M0011                   000
1
                                                          *****
  TXT              K  Ys K    h&         &  D      s  1      000
2
                                                          *****
  TXT                   Dq       $ s    u      s g   n       000
3
                                                          *****
  TXT   Y                    s  1                           000
4
```

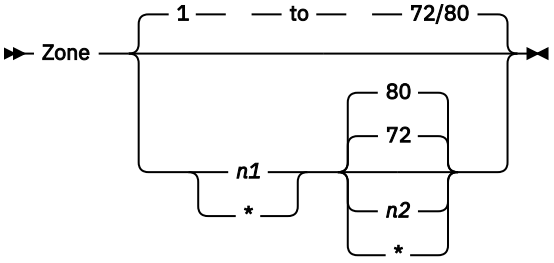
```
 ===> _
 <<..+....1....+....2....+....3. .5....+.. MEM=TPROGOBJ>>..+..FS
  ESD            TPROG           IJ2M0011                   0001
404040    E3D7D9D6C7404040000000 3F0F0F1F0000000004000
  0001    0068C9D1F2D4F0F0F1F100 0404040404040            /===/*
  TXT              K  Ys K    h&         &  D      s  1     0002
000078    05A0D217A0E8A22ED237A1 0001858F0A24605EF5820
  0001    A1DC0A0E90EAD00C18AF50 0A22A58F10010            *****
  TXT                   Dq       $ s    u      s g   n      0003
0000B0    45EF000C58D0A1C498EAD0 F50D0A1A041D0A19C5800
  0001    A1945B00A20A4120A0A418 0A02E95402000            *****
  TXT   Y                    s  1                           0004
0000E8    4780A0221B214122000158 F000C58D0A1A098EAD00C
  0001    07FE000080000000000400 0000003000000            *****
```

3. Display mixed data in columns 15 through 40.



ZONE Command

The ZONE command causes subsequent locating commands (such as FIND, LOCATE, SEARCH) and alteration commands (such as CHANGE, ALTER, OVERALAYX) to apply only to the specified zone of the lines.



n1
*****

n1 specifies the first column of the zone of each line that is to be scanned. You may specify * or omit the operand if your zone for viewing starts with column 1.

The value for n1 must be less than or equal to the value specified for n2.

n2
*****

n2 specifies the last column of the zone of each line that is to be scanned. It must be a value less than or equal to 80.

If n2 is omitted or specified as *, column 72 or 80 (depending on a tailoring option) is assumed. If n2 is specified, n1 must also be specified.

The zone values remain in effect until the next ZONE command is issued, or until the end of the edit session. These values can be temporarily overridden by the column suffix Cnn (see “Limiting Editor Operations to Certain Columns” on page 143). When editing multiple files concurrently, the zone can be set for each file being edited.

In addition to its obvious uses in searching and modifying fixed-format 80-character line files, the ZONE command is useful in source program editing. For example, it can be used to add comment fields, continuation characters, and for searching the serialization columns 73-80.

If editor change flagging is on, only the VSE/ICCF administrator can extend a search into the flag field.

When you are editing mixed data, the modification of data on the screen after a ZONE command can lead to unwanted results, as in the following cases:

- If before or after the modification the zone column cuts across a double-byte character. This is illustrated by Examples 2, 3, and 4.
- If alphanumeric data is replaced by a string of double-byte characters. In this case the alphanumeric data outside the zone is retained in their original form, but the replacement string of double-byte characters is restructured to fit inside the zone. This case is illustrated by Example 5.
- If double-byte characters are replaced by a string of alphanumeric characters. In this case, only those double-byte characters are retained that can be fit inside a pair of SO and SI characters. Example 6 illustrates this case.

Examples

The first example is somewhat complex: the editor is instructed to change the string LOOP to LOOPA in all lines within columns 1 and 30; subsequently, the editor is to reposition the line pointer to the line with the first occurrence of the string LOOPA.

Example 1

```
===> zone 1 30;change /loop /loopa/ *;search /loopa/_
<<..+....1....+....2....+....3. .5....+.. MEM=SAMPASMB>>..+..FS
          SPACE 3                                        /===/*
LOOP      EXCP  RDCCB          READ CARD                 *****
          WAIT  RDCCB                                    *****
          CLC   R(3),=C'/* '   LAST CARD?                *****
          BE    ENDCARD        EXIT LOOP IF YES          *****
          MVC   0(80,2),R      CARD- TO DISK BUFFER      *****
          PUT   FILOUT         WRITE TO DISK             *****
          B     LOOP           LOOP UNTIL "/*" CARD      *****
ENDCARD   CLOSE FILOUT                                   *****
```

```
===> _
<<..+....1....+....2....+...>>. .5....+.. MEM=SAMPASMB ...+..FS
LOOPA     EXCP  RDCCB          READ CARD                 /===/*
          WAIT  RDCCB                                    *****
          CLC   R(3),=C'/* '   LAST CARD?                *****
          BE    ENDCARD        EXIT LOOP IF YES          *****
          MVC   0(80,2),R      CARD- TO DISK BUFFER      *****
          PUT   FILOUT         WRITE TO DISK             *****
          B     LOOPA          LOOP UNTIL "/*" CARD      *****
ENDCARD   CLOSE FILOUT                                   *****
```

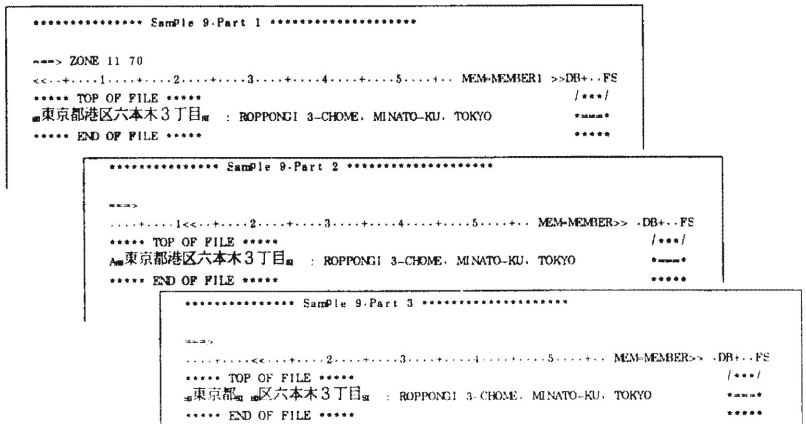
Example 2

```
*************** Sample 8-Part 1 ********************

===> ZONE 15 72
<<..+....1....+....2....+....3....+....4....+....5....+.. MEM=MEMBER1 >>DB+..FS
***** TOP OF FILE *****                                  /***/
東京都港区六本木３丁目    : ROPPONGI 3-CHOME, MINATO-KU, TOKYO          *****
***** END OF FILE *****                                 *****
```

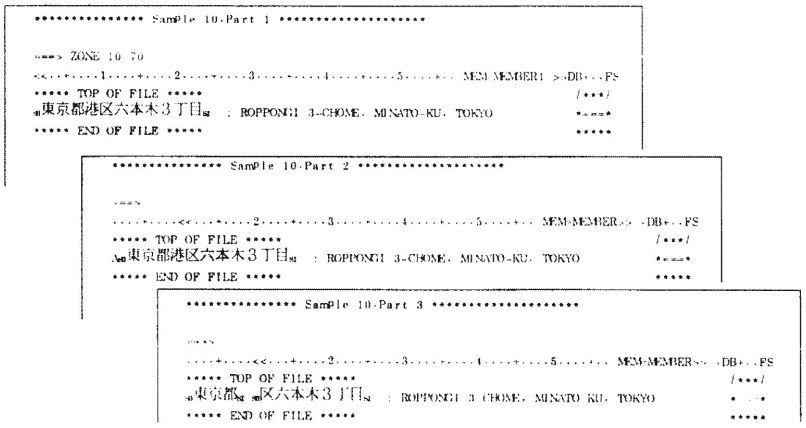
```
*************** Sample 8-Part 2 ********************

===>
....+....1....<<...2....+....3....+....4....+....5....+.. MEM=MEMBER1 >>DB+..FS
***** TOP OF FILE *****                                  /***/
東京都港区南麻布３丁目    : MINAMI-AZABU 3-CHOME, MINATO-KU, TOKYO      *****
***** END OF FILE *****                                 *****
```

```
*************** Sample 8-Part 3 ********************

===>
....+....1....<<...2....+....3....+....4....+....5....+.. MEM=MEMBER1 >>DB+..FS
***** TOP OF FILE *****                                  /***/
東京都港区六海布３丁目    : MINAMI-AZABU 3-CHOME, MINATO-KU, TOKYO      *****
***** END OF FILE *****                                 *****
```

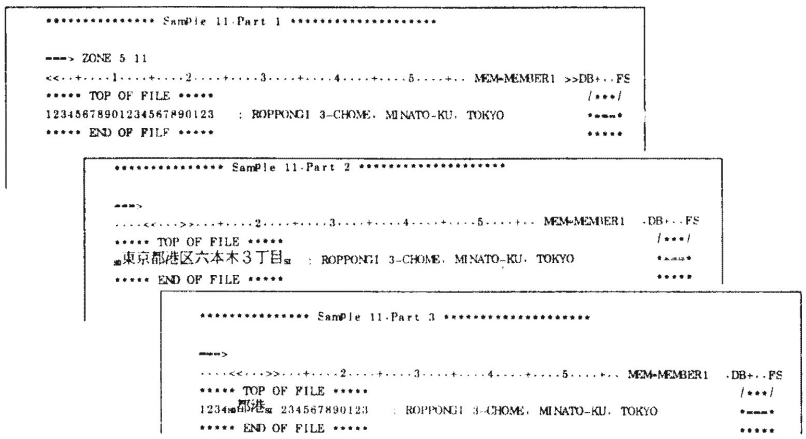
Example 3



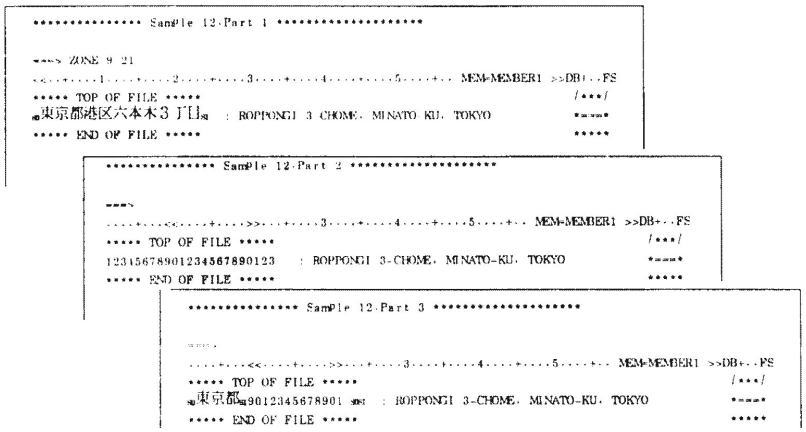
Example 4



Example 5

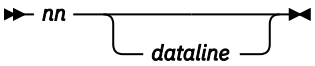


Example 6



nn Command

The nn command adds, replaces or locates lines by sequence number when line number editing is in effect.



nn

Is a number from 1 to 99999999, referencing a sequence number within the file being edited. Leading zeros can be omitted.

dataline

Specifies a line of data to be inserted into the file at the specified sequence number. If this sequence number already exists within the file, the existing line is replaced with the new line of data. If the sequence number does not exist within the file, the new line is inserted at the desired location within the file.

If the operand is omitted, the editor positions the line pointer at the specified sequence number.

After having processed the command, the editor positions the line pointer to the inserted, replaced or located line. Thus, when line number editing is in effect, this command can be used to make replacements or additions without having to position the line pointer.

If a sequence number is specified but the data line operand is omitted, the editor positions the line pointer to the line containing the specified sequence number. If this sequence number does not exist within the file, the editor will not move the line pointer and display a `LINE NOT FOUND` message instead.

Note: The `LINE NOT FOUND` message can occur also if the program that you are editing is out of sequence.

If indexed editing is in effect and line number editing was set before you issued the INDEX command (to build the index), the sequence numbers in the index will be retained. Thus, using the nn command to set the line pointer is a fast way of moving the line pointer long distances. Even though line number editing is in effect, you can still use the INPUT, INSERT and REPLACE commands where these are more convenient. They are usually more convenient when the line pointer has already been positioned to the desired location. On the other hand, the nn command will both position the line pointer and insert or replace the specified line.

Example

Replace the contents of line 60, add a line between lines 70 and 80, make line 20 the current line.

```
===> linemode left;60 print i,j,k;75 print ' ';20_
....+<<..1....+....2....+....3. .5....+.. MEM=CALC     ...+..F>
***** TOP OF FILE *****                                  /***/
00010PRINT 'ENTER THREE NUMBERS SEPARATED WITH COMMAS'   *****
00020PRINT 'ENTER /* TO TERMINATE'                       *****
00030L = I + J + K                                       *****
00040M = I * &SQR2                                       *****
00050N = J * &PI                                         *****
00060PRINT I,K,K                                         *****
00070PRINT L,M,N                                         *****
00080PRINT 'END OF CALCULATION'                          *****
```

```
===> _
....+<<..1....+....2....+....3. .5....+.. MEM=CALC     ...+..F>
00020PRINT 'ENTER /* TO TERMINATE'                        /===/*
00030L = I + J + K                                        *****
00040M = I * &SQR2                                        *****
00050N = J * &PI                                          *****
00060PRINT I,J,K                                          *****
00070PRINT L,M,N                                          *****
00075PRINT ' '                                            *****
00080PRINT 'END OF CALCULATION'                           *****
00090GO TO 10                                             *****
00100END                                                  *****
***** END OF FILE *****                                   *****
```

Editor Line Commands

Line commands perform editing functions for one line or a number of lines on the screen. They do not necessarily reference or alter the line pointer. The following commands apply to both alpahnumeric and mixed data:

A
for Add

C
for Copy

D
for Delete

I
for Insert

K
for Stack or Data Collect

M
for Move

/
for Set line pointer

"
for Duplicate

All other editor line commands process alphanumeric data only.

Line Command Area

Individual lines within a format area on the screen have line command areas associated with them, unless these command areas have been deleted by the FORMAT command. Line commands are entered into these line command areas. A command entered into the line command area applies to the record associated with the line command area and optionally to a number of records following the record for which the line command was entered.

The initial contents of the line command area will be one of the following:

1. The characters *====* or /===/* for the displayed records if the NUMBERS option (see the "SET Command" on page 216) is set off, which is the default setting. The current line displays /===/*, all others *====*.
2. A 6-digit sequence number if the SET NUMBERS ON command has been specified for the file being edited. This is the sequence number found in columns 73 through 80 or in the LINEMODE columns.
3. The characters *INPUT after an INPUT or INSERT command has been entered for lines that have been formatted to receive new input.
4. The characters /***/ for the TOP OF FILE line and the characters ***** for the END OF FILE line. The line command area for the bottom of the file is non-functional; however, the line command area for the top of the file can be used to enter the 'A' (add) or 'I' (insert) commands for adding new lines prior to the first record in the file.

Entering Line Commands

Only one command per line command area is permitted, except in the case of the '/' command which can precede another command.

If the characters in the line command area are *====*, /===/*, or *****, the line command can be entered beginning at any position within the line command area. Blanks or commas can be used as delimiters within the area although no delimiters are required, except in the case of the '>' and '<' commands with two operands, where the two numeric operands must be separated by a delimiter (blank, comma, equal sign or asterisk).

If the characters in the command area are actual columns from the file because the NUMBERS option is set on, some special rules apply:

1. The line command must begin in position 1 of the line command area.
2. Only one numeric operand is allowed and it must immediately precede or follow (no intervening delimiters) the line command code itself.
3. A blank must be entered following the command (and operand, if there is one) unless the operand precedes the command itself.

This helps avoid that the editor treats the numbers in this area as an operand (of a D (delete) command, for example, which could damage your file).

You can enter line commands on several lines before you press ENTER. If an error occurs in a line command area, a message is displayed and the audible alarm sounds. If multiple commands were entered, the only indication of which command(s) failed would be whether or not the requested function had actually been carried out.

Command Descriptions

Add Line(s)

The command can be used to add up to 999 blank lines after the line on which the command is issued. If no operand is specified, '1' is assumed. The blank lines added to the file can be used for the entry of new data to the file. When ENTER is pressed, the specified number of blanks will be added to the file and the cursor will be positioned to column 1 of the first added line. New data can then be typed into the added blank lines. If you added more blank lines than you needed, make sure to delete the unused ones.

Copy Line(s)

Beginning with the line on which it is entered, the command places nn lines into the stack area for insertion elsewhere in the file, or in a different file. For nn, you can specify any number from 1 to 99. If you omit the nn operand, the editor places one line into the stack area.

The point at which the data is to be inserted can be indicated on the same screen or on some later screen by the 'I' (insert) line command. (Functionally, a C (or M) command causes a STACK OPEN followed by placing the lines into the stack. Compare this with the K command below, which does not cause the stack to be opened).

Note: The C (copy), I (insert), K (copy to stack), and M (move) commands use the stack area, which is the same as the punch area. Therefore, do not use these commands when operating in asynchronous processing mode.

Delete Line(s)

The command can be used to delete up to 999 lines from the file, beginning with the line on which it is issued. If no operand is specified, '1' is assumed.

If multiple format areas are defined within a logical screen, deletions can be made only to one of the formats prior to pressing the ENTER key. If the current line is deleted, the editor moves the line pointer to the next line within the file.

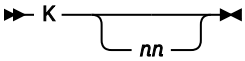
Insert Line(s)

The command is used to indicate where the moved (M command) or copied (C command) data is to be inserted. All lines previously saved by 'C', 'K', or 'M' commands will be inserted. To insert the moved or copied data at several points (also on different screens), simply specify multiple I commands. The I commands are processed after all other line commands; thus, you can move or copy data from the bottom to the top of the screen.

Functionally, the I command causes a STACK CLOSE followed by a GETFILE $$STACK.

See also the note to “Copy Line(s)” on page 231.

Stack or Data Collect



The command places nn lines into the stack area, beginning with the line on which the command is issued. For nn, you can specify any number from 1 to 99. The editor places one line into the stack area if you omit the nn operand. This command is similar to the C command, except that the first K command on a screen does not cause the stack to be opened. Thus, when collecting data from several screens to be copied to yet another location, the K command should be used instead of the C command on all screens except the first.

See also the note to “Copy Line(s)” on page 231.

Move Line(s)

The command places nn lines into the stack area, beginning with the line on which the command was issued. You can later insert these lines into the same file or another file by using the I (insert) command.

For nn, you can specify any number from 1 to 99. The editor places one line into the stack area if you omit the nn operand.

After having been stacked, the lines are deleted from the file. Thus, the M command is equivalent to a C command followed by a D command.

Functionally, a C or M command causes a STACK OPEN followed by placing the lines into the stack. Compare this with the K command above, which does not cause the stack to be opened.

See also the note to "Copy Line(s)" on page 231.

Set Line Pointer

The command positions the line pointer to the associated line. If multiple '/' commands occur within the same format area, the lowest one within the format area will take effect.

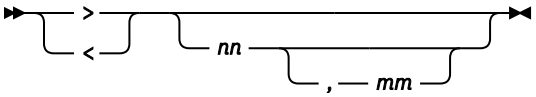
Since line commands are processed before normal editor commands, you can use the / command to set the line pointer to a line that is to be referenced by a subsequent, normal editor command entered on the same screen.

The / command can be specified in the line command area preceding another line command.

Duplicate Line

The command duplicates the line on which it is entered nn times. For nn, you can specify any number from 1 to 999. The editor duplicates the line only once if you omit the nn operand. Following the insertion, the editor sets the cursor to the first of the duplicated lines.

Shift Right, Shift Left



The commands shift the data within the zone of the line on which it is issued to the right or to the left by nn columns. Data shifted out of the zone is lost.

If the second operand (mm) is specified, the shift operation will be performed on 1 to 999 records beginning with the line on which the command is issued. The mm operand cannot be specified if the NUMBERS option is on.

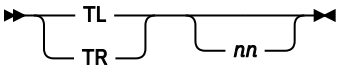
Text Align

The command justifies the data within the zone left or right nn lines, beginning with the line on which the command is issued. For more information about text alignment, see the section "ALIGN Command" on page 168.

Text Center

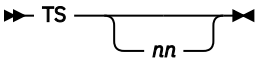
The 'TC' command centers the data that is within the zone in the 'nn' lines beginning with the line on which the command is issued. For more information about centering text, see the section “CENTER Command” on page 173.

Text Left-justified, Text Right-justified



The 'TL' or 'TR' command justifies the data within the zone left or right nn lines, beginning with the line on which it is issued. For more information about justifying text, see the section “JUSTIFY Command” on page 195.

Text Split



The 'TS' command splits the line on which the command is issued into two lines. The nn operand indicates the column number at which the split is to occur. For nn, specify a value larger than 1 and smaller than 80. For more information about splitting a line, see the section “SPLIT Command” on page 219.

Chapter 7. Job Entry Statements

Job entry statements conform to the same syntactical rules as system commands: they begin with a slash (/) followed by an operation code and optionally by one or more operands separated by blanks. However, while system commands take effect as soon as they are entered, job entry statements take effect not until the job is being run.

In the following example, /INPUT is a system command which places your terminal into input mode so that you can enter the two job entry statements (/LOAD and /INCLUDE) into the input area. /ENDRUN is a command which ends input mode and places the job in your input area into execution.

The statement /LOAD loads the FORTRAN compiler from a VSE library. The statement /INCLUDE includes some program out of member FPROG, which is going to be compiled.

```
/INPUT
/LOAD VFORTRAN
/INCLUDE FPROG
/ENDRUN
```

Job entry statements can be stored in members together with programs or with data or both. They can be held separately in an extra member or in the input area, and the related programs and data are included at the time of execution with the /INCLUDE statement (as shown by the example, above).

Length of a Job Entry Statement

A job entry statement cannot extend past column 72. Only the /FILE and /LOAD statements can be continued from one record to the next. To continue the /FILE statement, place a comma after the last operand in the first record. For the /LOAD statement, place a continuation character in column 72. Then start the continuation record with a slash and two blanks in columns 2 and 3. The continuation data can then begin in column 4.

Job Stream and Job Step

A job stream consists of one or more job steps. A job step starts with a /LOAD job entry statement and ends with the next /LOAD or with end of job. A job step consists of job entry statements and user data.

After a /LOAD statement has been processed, the first statement encountered that is not a valid job entry statement is assumed to be the start of user data for the job step.

A summary of all VSE/ICCF job entry statements is given in Table 3 on page 235. A more detailed description of each of these statements follows this summary.

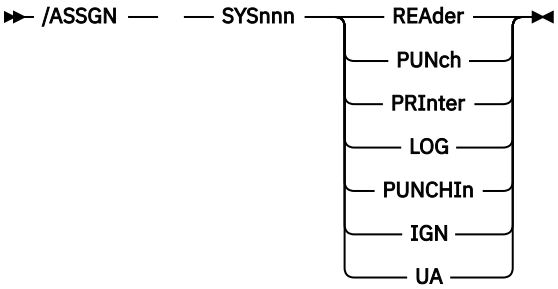
Table 3. Summary of Job Entry Statements

Statement	Function of Statement
/ASSGN	Alters the default unit record device assignments for programs in an interactive partition.
/COMMENT	Serves as a comment within a sequence of job entry statements.
/DATA	Separates the input to a language compiler from data which the compiled program will read.
/FILE	Provides information about disk files which will be processed during an interactive partition execution.
/FORCE	Causes the lines in the print area to be displayed at the terminal.

Table 3. Summary of Job Entry Statements (continued)	
Statement	**Function of Statement**
/INCLUDE	Logically groups several library members or the contents of work areas (or both) into a single source of input.
/LOAD	Specifies the name of a program that is to be loaded and run.
/OPTION	Alters the standard setting of certain job-processing options.
/PAUSE	Causes the scheduler to display a message (if a comment was specified) and then to stop processing the job stream.
/RESET	Causes any prior assignments (by /ASSGN statements) in the job stream to revert to the defined defaults.
/TYPE	Displays comments from a job stream.
/UPSI	Sets the VSE User Program Switch Indicators.

/ASSGN Job Entry Statement

The /ASSGN job entry statement is used to alter the standard assignments for unit record devices used within programs in interactive partitions.



SYSnnn REAder
SYSnnn PUNch
SYSnnn PRInter
SYSnnn LOG
SYSnnn PUNCHIn

For nnn in SYSnnn, specify a number from 000 to 240, indicating the VSE programmer logical unit to be assigned to a certain function. Following is a list of the standard defaults; your location may have defined different ones. For your convenience, you may note below the defaults defined at your location.

```
Normal              Local
Default   Device    Default   Description
-------   ------    -------   ---------------------------
SYS005    READER    ......    Job stream input device
SYS006    PRINTER   ......    Printed (terminal) output
SYS007    PUNCH     ......    Output to punch area
SYS008    PUNCHIN   ......    Read from punch area
SYS009    LOG       ......    Terminal input and output
```

If you are not sure which units to use – for conversational input (LOG or READER with INCON), for example, or for printed (terminal) output (PRINTER or LOG) – read Chapter 11, “Job Entry Considerations,” on page 309 or contact your VSE/ICCF administrator.

To run a program that does not use standard default SYS units for unit record I/O, assign your SYS unit in your job stream. For example, to run a program that reads 80-character records from SYS042, specify `/ASSGN  SYS042  READER` (unless SYS042 is the default for the job stream input).

This assignment does not change the assignment of SYS042 made during startup of VSE/ICCF. It simply tells VSE/ICCF which I/O stream is to be intercepted. Thus, for the opening of the file to be successful, your assignment must be consistent with the one set up during startup. If the assignment is to a unit record device, the open will be successful. If it is to a DASD, you must supply a dummy / FILE statement, because OPEN checks for DASD labels.

IGN
Causes VSE/ICCF to ignore the default action for the specified SYS number. For example, if SYS005 is the programmer unit for job stream input, the statement /ASSGN SYS005, IGN tells VSE/ICCF not to intercept read requests from SYS005; that is, let them pass through to the system's supervisor for it to handle them. Issue IGN only for 'SYS' numbers that correspond to the local defaults.

UA
Unassigns a previous SYS assignment. Once an assignment has been made for a SYS number, it will apply to all subsequent steps in the same job. To negate the effect of a previous SYS assignment and to make a reassignment, unassign the logical unit by specifying UA and then assign it. To negate the effect of all prior assignments, issue the /RESET job entry statement.

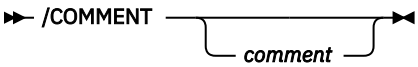
Example

Instead of using the locally defined defaults of SYS005 for control statement input and SYS006 for printer (terminal) output, the program (UTILX) to be run uses SYS004 and SYS005, respectively.

```
*READY
/input
/load utilx
/assgn sys004,reader
/assgn sys005,printer
     utilctl 75,300
/end
*READY
/run
*RUN REQUEST SCHEDULED FOR CLASS=A
```

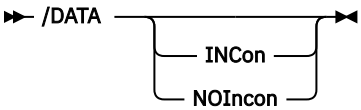
/COMMENT Job Entry Statement

The /COMMENT statement is ignored. It can be used to put comments into a sequence of job entry statements.



/DATA Job Entry Statement

The /DATA job entry statement separates the input to a language compiler from any job stream data statements (SYSIPT or SYS005 or other locally defined default) which the compiled program expects to read.



INCon
Specifies that the program which has been set up to read data from SYSIPT or SYS005 (or other local default for normal card-image input) will instead ask for the input conversationally from your terminal.

NOIncon
Can be entered only when in INCON mode to conclude conversational input and revert back to job stream processing on SYSIPT. This is useful if one program chains to another and the execution time input for the programs is to be done in conversational mode while the source program input can be done in job-stream mode.

Any data in the operand field other than INCON or NOINCON is treated as comments in the job stream.

If the program being compiled and run does not read input from the job stream VSE unit SYSIPT or the numbered SYS unit assigned to SYSIPT, there is no need to specify a /DATA statement.

A /DATA statement in a job stream invokes LINKNGO if the compiled object program has not yet been processed. That is, it has the same effect as specifying /LOAD LINKNGO. If no compilations have previously been performed, the /DATA statement forces end of file in the same way as a /* in a VSE job stream would.

When a /DATA statement implies the invocation of the LINKNGO program, it can be followed with job entry statements such as /FILE or /OPTION which are to apply to the load-and-go step.

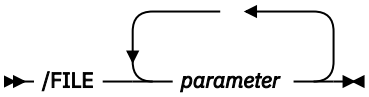
Example

```
*READY
/input
7.4
5.9
/end
*READY
/save fortdta
*SAVED
*READY
/input
/load vfortran
```

```
5    read (1,10,end=99) a
10   format (f8.3)
     x=a**2
     write (3,15) a,x
15   format (' x=',f8.3,' x**2= ',f20.3)
     go to 5
99   stop
     end
/data
/include fortdta
/end
*READY
/run
*RUN REQUEST SCHEDULED
X=       7.400 X**2=          54.760          |  FORTRAN compiler and
X=       5.900 X**2=          34.810          |  LINKNGO output
****JOB TERMINATED - ICCF RC 00, EXEC RC 0000,
*READY
/ed
*EDIT MODE
find /d
/DATA
add  incon
P
/DATA INCON
next
/INCLUDE FORTDTA
*EOF
save fortprog
*SAVED
*READY
/exec fortprog
*RUN REQUEST SCHEDULED
                      Compiler and LINKNGO output
?                     Indicates a conversational read request
4.0
X=       4.000 X**2=          16.000
?
3.5
X=       3.500 X**2=          12.250
?
/*                    Causes end of file on FORTRAN unit 1
****JOB TERMINATED - ICCF RC 00, EXEC RC 0000,
*READY
```

/FILE Job Entry Statement

The /FILE job entry statement specifies information about disk files that are to be processed in an interactive partition.



parameter
Are operands consisting of keywords followed by an equal sign followed by a specified value. The operands give information about the file that is to be processed.

You can use the statement to define different types of files as follows:

Files in dynamic space
Normal VSE files or VSE/VSAM files
VSE/ICCF-member files.

These different types of files are discussed following the description of the operands that you can specify for parameter.

Operands of the File Statement

BLKsize=n
Corresponds to the BLKSIZE operand in the VSE DLBL job control statement. For n, specify the block size that is to be used instead of the one defined in the DTFSD macro for the file. The operand is optional; if it is omitted, the block size specified in the DTFSD macro is used.

BUFfer=bufsize
Corresponds to the BUFSP operand of the DLBL job control statement and applies only to VSAM files. The operand allows you to dynamically specify the number of bytes of virtual storage to be allocated as I/O areas for the specified file.

CATalog=catname
This operand applies to VSE/VSAM files. For catname, specify the name of the VSAM catalog that is to be searched for the catalog entry for the file.

CISize=ci-size
This operand corresponds to the CISIZE operand of the VSE DLBL job control statement. For ci-size, specify the control-interval size for a SAM file on an FBA device or in VSAM-managed space. The operand is optional and can be specified only together with TYPE=SEQ.

CYL=YES
Requests that disk space dynamically allocated in accordance with your specification in the SPACE operand will start and end on cylinder boundary. The value of ntrks in SPACE=ntrks is rounded to the next higher number of cylinders for a CKD device; it is ignored for an FBA device.

DATe=expinf
For normal VSE output files and dynamically allocated DISP=KEEP files you can specify the expiration information. If this operand is not specified, the file will not be date protected. You can provide expiration information as:

- A number of days (such as DATE=14)
- A date in one of the following formats:

```
  yy/ddd
19yy/ddd
20yy/ddd
```

where yy = year and ddd = day of year.

DISp=Delete
DISp=Pass
DISp=Keep
The disposition operand applies only to dynamically allocated files. If the operand is omitted, DELETE is assumed. This means that the file area is released after the current job step has been completed.

PASS indicates that the file should be retained (passed) from step to step and released at the end of the job in which it was allocated.

KEEP indicates that the file is to be permanent. It will be kept as long as the dynamic space area is 'warm' started or until you scratch it. If the dynamic space areas are 'cold' started, the file will be lost the next time VSE/ICCF is initiated. The file must be referred to as a normal VSE file in all jobs after the job which allocated it (that is, it will require the LOC=operand).

IDent='fileident'
This operand can apply to dynamic space files or to normal VSE files. It specifies the file identifier that is stored in the file label on disk. For normal VSE files, the default is the name you specify in the NAME operand.

The IDENT operand must be specified if the file identifier is (for input) or will be (for output) different from the file name.

For dynamic space files, the default for this operand is filename.userid.termid. In this identifier, user ID is the four-character user identification and termid is the four-character terminal identification. filename is the name you specified in the NAME operand, unless name is of the form IJSYS0n. If so, the J will be replaced by a K and the 0 will be replaced with the interactive partition identification character.

The beginning and ending apostrophes can be omitted if the specified identifier does not include delimiter characters.

You can use also the tags &USR, &TRM, and &PRT to form a unique identifier:

&USR
Will be replaced with by your user ID.

&TRM
Will be replaced by the terminal identification.

&PRT
Will be replaced by the interactive partition number.

The tags allow you to specify standard job streams that reference file identifications based on a certain user. They also allow you to base such job streams on a terminal, or on the interactive partition in which they will run. Thus, these tags can be used also in /PAUSE and /TYPE statements, so that the created file identification can be returned to you.

LOC=start,length
The operand applies only to normal VSE disk files. For output files it must be specified. For input files it need be specified only if your system was started with the file-protect (DASDFP) option.

For start, specify the track or block (on FBA devices) number at which the file begins. For length, specify the number of tracks or blocks (on FBA devices) in the file area.

MAXR=nnnnn
This operand applies only to TYPE=ICCF output files. It specifies the maximum number of records for the target member.

For nnnnn, you can specify a value from 1 to 99999 (MAXR has a default of 32767). When the specified number is reached, the job is canceled (unless DTSPROCS is running) and the member is closed. At this point, the file contains MAXR+1 records. All subsequent print or punch output will then be stored in the print or the punch area, respectively.

NAMe=filename
The NAME operand must be specified on all /FILE statements. Except for a TYPE=ICCF file, filename is the 1 to 7 character name used to define the file within the program being run. For TYPE=ICCF

files, the operand specifies the name of the VSE/ICCF library member for input or output. The member must exist in your library. When printing or punching to a library member, VSE/ICCF extends the member if necessary.

PASsword=password
The operand must be supplied only for a TYPE=ICCF file which is password-protected. Specify the member password used to protect the file.

RETAIN=JOB
RETAIN=STEP
This operand applies only to TYPE=ICCF punch or print files. It indicates how long a /FILE statement is to be effective:

STEP
To be effective only during the current job step (this is the dafault).

JOB
To be effective until the end of the current job, until another /FILE statement, or until /OPTION PUCLOSE|PRCLOSE is encountered.

Note: Using a member for different purposes in the same job can lead to unpredictable results.

SERial=vol.-ser-no.
The operand specifies a six-character volume serial number, which may be specified for normal VSE input and output files. For dynamic space allocation files, VSE/ICCF will attempt to find a dynamic space designated area on the specified volume from which to allocate the space:

- For a DISP=KEEP file, if a space cannot be found on the specified volume, the job will be canceled.
- For a temporary file (DISP=PASS), if the space cannot be found on the volume, VSE/ICCF will look for space elsewhere.

SPAce=ntrks
The operand indicates a request for dynamic disk space allocation. Specify only the number of tracks or units of space (a unit of space on an FBA disk is 16 physical blocks) required for your file. If you request more space than the locally defined maximum value, your request will be reduced to that maximum value.

Note: A certain amount of space (up to two tracks) is used by VSE/ICCF to store control information. This space is not available for allocation.

TYPe=Direct
TYPe=Seq
TYPe=Vsam
TYPe=Iccf
For normal VSE files and for dynamically allocated file space, this operand specifies the type of file access: direct, sequential, or VSAM (Seq is the default).

For dynamically allocated file space of VSE/ICCF, TYPE=VSAM should not be specified.

TYPE=ICCF must be specified for VSE/ICCF library members.

UNIt=sysno
UNIt=SYSLST
UNIt=SYSLLG
UNIt=SYSPCH
The UNIT operand can be coded to specify the logical unit that is to be associated with the file.

For a file in dynamically allocated disk space, VSE/ICCF attempts to find a dynamic-space designated area on the disk assigned to that logical unit.

For a DISP=KEEP file, if the space cannot be found on the specified unit, the job is canceled.

For a temporary file, an attempt is made to allocate the space from another area.

For a normal VSE file, the specification for sysno (any numbered unit or a system logical unit) will be used to access the file. Therefore, ensure that the logical unit is assigned to the proper disk unit

before using that logical unit. If you omit the UNIT operand, VSE/ICCF uses a logical unit that it knows to be associated with the specified volume (SERIAL= operand).

If the specified logical unit cannot be found, the /FILE statement is considered as invalid.

For a TYPE=ICCF input file, the UNIT operand must be supplied, and the specified (programmer) logical unit must be the same as the one in the CCB used to access the file. You cannot specify a system logical unit such as SYSIPT.

For TYPE=ICCF punch output to the library, specify UNIT=SYSPCH.

The UNIT=SYSLST, UNIT=SYSLLG and UNIT=SYSPCH operands apply only to TYPE=ICCF members:

- UNIT=SYSLST specifies that only the print output produced for SYSLST is to be stored in the specified member; SYSLOG output is to be stored in the print area and displayed in the usual way.
- UNIT=SYSLLG specifies that output for both SYSLST and SYSLOG is to be stored in the specified member and that SYSLOG data is to be stored in the print area and displayed in the usual way.

If the /FILE statement is used with UNIT=SYSLLG, but without TYPE=ICCF, UNIT=SYSLST is assumed.

Note:

1. The operands RETAIN and MAXR apply to both SYSLST and SYSPCH.
2. The UNIT operand should **always** be specified for job streams that are submitted for execution in a batch partition. This ensures correct assignment for disk volumes which are not accessible from an interactive partition.
3. If UNIT=SYSLST or SYSLLG is specified, the NAME operand should refer to a print-type member (xxxx.P) so that print format data can be obtained via the /LIST command.

VOLume=n
The VOLUME operand applies only to dynamically allocated files. It provides a method of distributing temporary space requests over multiple volumes for more efficient access. For n, specify a number from 0 through 9, corresponding to the ten possible dynamic space areas within the environment (your system may, in fact, only have one, or two, or three).

For a temporary file, VSE/ICCF will attempt to allocate the file space on the specified volume regardless of the serial number. If the space cannot be found, the space will be allocated from any available space.

File information need not be provided for a disk file if the file information for that file (DLBL/EXTENT statements) was present in the VSE/ICCF initialization job stream. A /FILE statement overrides the initialization job stream DLBL/EXTENT information. The three major types of the /FILE statement request file information for files as follows:

1. Sequential or direct files in dynamic space.
2. Normal VSE sequential or direct files or VSE/VSAM files.
3. Sequential files that are members of your VSE/ICCF library.

Each of these types is discussed below.

Defining a File in Dynamic Space

If VSE/ICCF supports dynamic disk space allocation, you can ask the system to find space for your sequential or direct file dynamically. The basic request for file space would be:

```
/FILE NAME=filename,SPACE=ntrks
```

This is all you need to define for a sequential input, an output, or a work file. If you want your file to have a name on disk different from how the file is named in your program, add the IDENT operand:

```
/FILE NAME=filename,SPACE=ntrks,IDENT='fileident'
```

To make a file accessible from one job step to the next, specify DISP=PASS. Otherwise the space will be made available to other users after the step in which it was allocated has been completed. If you want your file to be permanently accessible (from one job or from one day to the next), specify DISP=KEEP.

After a dynamic file has been allocated, you will receive a message telling you where the file was allocated: volume serial number, logical unit, relative disk address, and so on. For a DISP=KEEP file, note this information because you may need it when you want to access the file later, when it will be specified as a normal VSE file.

Space on an FBA disk is allocated by groups of physical records, which consist of 16 physical blocks. Requests for disk space are therefore somewhat independent of the device type.

It is important to note for dynamically allocated files on FBA disk devices that the length field in the location information returned after allocation will be 16 times larger than the requested space. This need not concern you, except when the file is DISP=KEEP. In that case note the location as it appears and use it in exactly the same way on subsequent /FILE statements for this file.

Defining a Normal VSE File or a VSE/VSAM File

For a job that is to be processed under VSE and not under VSE/ICCF, you must provide disk file information in DLBL/EXTENT statement sets. If the job is to be processed under control of VSE/ICCF, this same information can be provided in a /FILE job entry statement. Refer to z/VSE System Control Statements for a more detailed description of the various specifications.

When you specify disk file information for normal VSE input files, two or three operands are required besides the NAME operand (which is mandatory):

- The IDENT='fileident' operand is required if the file identification on disk is different from the file name that you use in your program.
- Use the UNIT=sysno operand if the logical unit for your file must be different from the logical unit of the volume on which your file resides.
- Use the LOC=start,length operand if the file information is for an output file (a file which does not yet exist on disk). The operand gives the relative track address where the file is to be built. You must specify this operand for input files as well as for output files if your VSE system uses the file-protect option (DASDFP option of the IPL SYS parameter). Thus, you might end up with a /FILE statement as follows:

```
/FILE NAME=xxxxxxx,IDENT='fileident',SER=serno,
/     UNIT=sysno,LOC=start,length
```

- Include the DATE=expinf operand if the file information is for an output file and you want the file to be date-protected.

If the file information relates to a direct or VSE/VSAM file specify TYPE=DIRECT or TYPE=VSAM, respectively.

For a VSE/VSAM file you may want to specify the CATALOG or the BUFFER operand or both. CATALOG allows you to specify a catalog to be searched other than the master or user catalog. BUFSIZE allows you to dynamically specify virtual storage to be allocated to the file for I/O buffers.

Defining a VSE/ICCF Member File

Normally, all **punch-directed output** from a program is placed into the punch area. However, you can also have the punch output from a job step placed directly into a library member. The member must be in the primary library and you must specify a /FILE statement of a format as shown:

```
/FILE NAME=membername,UNIT=SYSPCH,TYPE=ICCF
```

If the member is password-protected, you must include also the PASSWORD operand.

You can request a **library member** to be read **into your program** by specifying /INCLUDE for the member in the job stream. However, if it is necessary to read more than one member into the program simultaneously, specify:

```
/FILE NAME=member,TYPE=ICCF,UNIT=sysno
```

The sysno value must match the logical unit number in the CCB that is used to read the file. This function can be used only by a program written in assembler language to read records at the EXCP level without doing a logical IOCS open for the card read file.

The number of VSE/ICCF-type input files a program can use is restricted to two.

You can also save the total print output from a job or a job step in the VSE/ICCF library up to the value specified in MAXR. This allows you to obtain a hard copy of the output by simply sending the member to the printer, using the RELIST macro. The format of the records in such a member corresponds to the format of the records in your print area. Exceptions are records that indicate conversational read requests or full screen write/read requests. These records are replaced in the member by one of the messages:

```
*CONVERSATIONAL READ
*FULL SCREEN WRITE/READ.
```

Examples

- Specify work files for an assembly of the program called ASMB; place the object deck (punch output) into a preallocated member called ASMBOBJ.

```
*READY
/input
/load assembly
/file name=ijsys01,space=10
/file name=ijsys02,space=10
/file name=ijsys03,space=10
/file name=asmbobj,type=iccf,unit=syspch
/option noload,deck,nolist
/include asmb
/endrun
*RUN REQUEST SCHEDULED FOR CLASS=a
```

- Specify file information to access a VSE library.

```
*READY
/input
/load libr
/file name=ijsyssl,id='source.lib',unit=sys070,
/  serial=lib008
   access subl=l1.test
   listdir fregs.a
/endrun
*RUN REQUEST SCHEDULED FOR CLASS=A
```

Note: If the SERIAL operand is not specified, the logical unit (SYS070 in the example) must be assigned to the CICS/ICCF partition.

- Create a direct file named SPECLIB and date protect it.

```
/input
/load specbld
/file name=speclib,ident='wksp.lib',ser=222222,
/  loc=2400,200,date=1999/365,type=direct
/include specdata
/endrun
```

- Dynamically allocate a file (using DYNAMIC Space on FBA) for later use with DATE PROTECTION, merge data from two card type files, write data to the file (program BLDFILE), and read it back to create a listing in a later job.

```
/inp
/load bldfile
/file name=indata,type=iccf,unit=sys005
```

```
/file name=outfile,id=extract,date=10,space=100,volume=3
/include mstrdata
/endr
* RUN REQUEST SCHEDULED
K859I...ALLOCATION FOR OUTFILE -SERIAL=WORKDS UNIT=SYS002
       LOC=3200,1600
 ...
 ...                    (File built by the program)
 ...
* *** JOB TERMINATED RETURN CODE 00 NORMAL EOJ
* READY
/inp
/load prntfile
/file name=inprint,id=extract,loc=3200,1600,
/  SER=WORKDS,UNIT=SYS002
/endr
* RUN REQUEST SCHEDULED
 ...
 ...                   (Job completes)
 ...
```

/FORCE Job Entry Statement

The /FORCE job entry statement forces the lines in the print area to be sent to the terminal as soon as the next print activity occurs.

Normally, a job being executed will not start displaying data until one of the following occurs:

The job ends.
A conversational read is encountered.
The print (spool) area is full.

However, sometimes you may need to force output to the terminal before the print area is full. The /FORCE job entry statement allows you to do this.

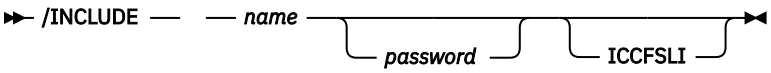
The /FORCE statement can be imbedded within data records, or it can be included with other job entry statements.

Example

```
*READY
/input
/load vfortran
/include primegen
/load loader
/load vfortran
/force                    (Causes previous output to be directed
/include fibonaci         to your terminal before the second
/endrun                   compile is run.)
*RUN REQUEST SCHEDULED
```

/INCLUDE Job Entry Statement

The /INCLUDE job entry statement logically groups library members and/or the contents of work areas into a single source of input.



name
Is the name of the library member that is to be logically included in the job at the time of execution. The inclusion occurs at the point where the /INCLUDE statement appears. For name, you can also specify $$PUNCH, $$PRINT, or $$LOG. This includes the contents of the specified area.

password
Is required only if the member being included is password-protected.

ICCFSLI
Applies only to a job stream that is submitted to VSE/POWER via the SUBMIT procedure. The operand causes the /INCLUDE statement to be transformed into the VSE/POWER JECL statement * $$ SLI.

If your /INCLUDE statement has no ICCFSLI operand, VSE/ICCF first reads the member from the VSE/ICCF library file and then writes it to the VSE/POWER reader queue; from there, the member is finally read into the program.

If the ICCFSLI operand is specified, the SUBMIT program transforms the /INCLUDE into an * $$ SLI statement. As a result, the member to be included will not be read into the reader queue. Rather, VSE/POWER will read the member directly from the VSE/ICCF library file prior to execution time. This improves the response time from the SUBMIT procedure and reduces the I/O load of your system.

If the /INCLUDE statement has the ICCFSLI operand, /INCLUDE statements within the member to be included (nested /INCLUDEs) are later resolved by VSE/POWER, regardless of whether they have the ICCFSLI operand. The number of nesting levels is unlimited. /INCLUDE statements in compressed members as well as in noncompressed members will be resolved.

The * $$ SLI statement which the SUBMIT program generates from the /INCLUDE has the following format:

```
* $$ SLI ICCF=(member[,password]),LIB=(n1[,n2[,n3]])
```

where:

member[,password]
Indicates the name of the member to be included, together with its password if it is password-protected.

LIB=(n1,n2,n3)
Specifies the numbers of the VSE/ICCF libraries to be searched. The VSE/ICCF library numbers reflect the terminal user's search chain when the submit was issued, that is: primary library, connected library, common library.

The /INCLUDE with the ICCFSLI operand cannot be used if the job stream is submitted to a remote node, unless the member exists at the remote system under the same member name and in the same VSE/ICCF library number.

A job stream containing * $$ SLI statements with the ICCF operand can be entered from a card reader.

The VSE/ICCF library file is not available for the ICCFSLI function whenever one of the library modification processes of the DTSANALS or DTSUTIL utilities is active. It is possible that a submitted job requesting an SLI inclusion is canceled due to a library modification currently in progress.

A * $$ JOB and/or a * $$ EOJ statement in a member that is included with the ICCFSLI option is ignored.

When you build a job stream, you may find it inconvenient to have the program, data, and job entry statements all together in the input area or in one library member. The /INCLUDE statement allows you to separate some or all job entry statements from the program or from the data to which the program refers. An /INCLUDE statement in the job stream has the same effect as if all of the included data were physically located at the point of the /INCLUDE.

An /INCLUDE statement must be within a job stream and cannot be entered into a job stream via a /PAUSE statement.

/INCLUDE references can be nested up to eight levels. The /INCLUDE statements can reference library members which themselves contain /INCLUDE statements. A member can chain to the next member by having an /INCLUDE as its last line. Up to 256 library members can be chained together in this way.

For a FORTRAN compile, you would normally run with the following job entry statements and data.

```
/LOAD VFORTRAN
 ...
 ...                    (FORTRAN source program)
 ...
/DATA
 ...
 ...                    (Input data)
 ...
```

However, if the data is in a separate library member named FORTDATA, the job stream would look as follows:

```
/LOAD VFORTRAN
 ...
 ...                    (FORTRAN source program)
 ...
/DATA
/INCLUDE FORTDATA
```

The program itself can be in yet another member named FORTPROG. The job entry statements would then look as follows:

```
/LOAD VFORTRAN
/INCLUDE FORTPROG
/DATA
/INCLUDE FORTDATA
```

Now if the FORTRAN program were large and were stored in three library members named FORTPR1, FORTPR2 and FORTPR3, the job stream would be:

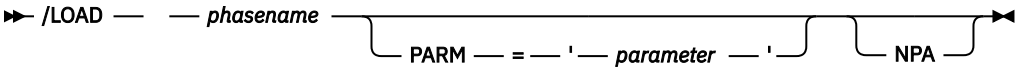
```
/LOAD VFORTRAN
/INCLUDE FORTPR1
/INCLUDE FORTPR2
/INCLUDE FORTPR3
/DATA
/INCLUDE FORTDATA
```

Example

```
*READY
/input
card 1
card 2
/end
*READY
/save group1
*SAVED
*READY
/input
line 1
line 2
/end
*READY
/save group2
*SAVED
*READY
/input
/load object            (This program reads cards and
/option nogo             transfers them to the terminal.)
/upsi 1
/include group1
/include group2
/endrun
*RUN REQUEST SCHEDULED
CARD 1
CARD 2
LINE 1
LINE 2
****JOB TERMINATED - ICCF RC 00, EXEC RC 0000, NORMAL EOJ
```

/LOAD Job Entry Statement

The /LOAD job entry statement specifies the name of a language processor (assembler, compiler), a utility program, or some other program that is to be loaded and run.



phasename
Is the name of a phase that is cataloged in a VSE library. This name will often be one of the following:

ASSEMBLY
Specifies that the following lines are to be processed by the assembler program and that the resulting object program is to be run unless otherwise specified.

RPGII
Specifies that the following lines are to be processed by the RPG II Compiler and that the resulting object program is to be run unless otherwise specified.

OBJECT
Specifies that the lines following in the job stream are an object deck (not BASIC) which is to be written into the punch area and run. OBJECT will also transfer lines to the print area if an UPSI switch is set on. An execution of OBJECT is implied if an object deck is found in the input job stream and the deck is not preceded by a /LOAD. For more information about this option see the section “OBJECT Utility” on page 306.

LINKNGO
Specifies that the contents of the punch area are to be loaded into storage for execution. In most situations, /LOAD LINKNGO need not be specified since its use is often implied. For more information about this option see the section “LINKNGO Utility” on page 303.

DTSPROCS
Specifies that the lines following in the job stream are to be read and interpreted by the Procedure Processor (DTSPROCS). For more information about using procedures, see Chapter 9, “Writing a Procedure or a Macro,” on page 273.

Note:

1. Considerations for specific compilers are described in Chapter 11, “Job Entry Considerations,” on page 309.
2. It is possible to load together (via LINKNGO) object programs created by the above compilers. For the loading of programs and subprograms, the standard VSE linkage conventions apply. These conventions are usually given in the *Programmer's Guide* published by IBM for the programming language.
3. Run requests for any of the above language compilers can be grouped together in a single job.
4. You can use certain VSE programs only if your user profile indicates that you are an authorized user.

PARM='parameter'
Allows information of up to 100 characters to be passed to the loaded program. If the information does not fit into the first 71 positions of the 80-byte record, a continuation record can be used. Put a continuation character into column 72 and start the continuation record with a slash (/) followed by two blanks.

The information is passed to the loaded program via register 1. This register points to a field that contains a one-byte parameter status switch and a three-byte pointer to the information field. The switch is:

X'80'
Parameter has been passed.

If a parameter value is not available, the three-byte pointer is set to zero. The value field starts with a 2-byte length field for the parameter value followed by the 1 to 100 byte parameter value. Parameters

with zero length also have the X'80' bit on. To identify zero length parameters, you should test the 2-byte length field to check if the parameter has a value of all zeros. This is in contrast to the // EXEC statement of z/VSE, where a zero-length parameter is flagged with X'00' in the information field. For details, refer to the z/VSE Guide to System Functions.

To test whether a PARM value has been passed, you should compare the values of registers 1 and 15. If their contents are the same, a PARM value was not specified.

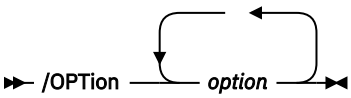
The /LOAD statement should be the first statement in any job step. Any lines in the job stream prior to the first /LOAD statement are ignored. Likewise, if a step of a multistep job terminates before having read all of its data, any lines in the job up to the next /LOAD statement are ignored.

Example

```
*READY
/input
/load assembly
/tabset asm
/set tab=;
main;start;0
;balr;5,0
;using;*,5
;excp;conccb
;wait;conccb
;eoj
conccb;ccb;syslog,conccw
conccw;ccw;9,condta,0,l'condta
condta;dc;c' this means the program worked'
;end
/endrun
*RUN REQUEST SCHEDULED
(assembler and LINKNGO output)
THIS MEANS THE PROGRAM WORKED
**** JOB TERMINATED - ICCF RC 00, EXEC RC 0000, NORMAL EOJ
*READY
```

/OPTION Job Entry Statement

The /OPTION job entry statement is used to alter the standard setting of certain job processing options.



option

Are options which are to be set for a certain job step or a series of steps. As many options can be specified in one statement as fit into 72 columns. The options must be separated by at least one delimiter. Multiple /OPTION statements can be included in a single job step.

An option set for a job, remains in effect until it is is reset by another /OPTION statement or until the end of the job. All options are reset to their default status before the first step of a job.

Certain options apply to the general operation of VSE/ICCF, while others apply to individual language processors. If the information concerning the language processor options given below is not sufficient, refer to the *Programmer's Guide* for that language processor.

The FORTRAN compiler does not check the standard /OPTION statement for language processor options. Instead, the options are specified as follows:

- For the FORTRAN compiler – either on the @PROCESS statement or in the PARM field of the /LOAD statement.

Also, certain of the other language compilers have their own supplementary option statements.

The following are the language processor options which can be specified on the /OPTION statement. Except for RLD/NORLD, the standard default options are shown as the first option, for example NOALIGN

ahead of ALIGN. See the appropriate *Programmer's Guide* for the exact meaning of the option. Once again, these options do not apply to the FORTRAN compiler.

NOALIGN
ALIGN
Is used by the assembler to control the alignment of fullword and doubleword constants and of storage areas to their appropriate boundaries.

DECK
NODECK
Causes an object program to be placed into the punch area from which it can be read in and run by the LINKNGO program. NODECK causes the compiler or assembler to run faster, but subsequent execution of the compiled (or assembled) program is not possible.

NOLIST
LIST
Is used by some compilers to control the listing of the input source program on SYSLST (that is, on the terminal). NOLIST is the default for all compilers because a listing of the source has usually already been made using the /DISPLAY or /LIST commands.

NOLISTX
LISTX
Is used by certain language compilers. LISTX requests that a hexadecimal representation of the generated object program is to be produced on the terminal. NOLISTX, the default, suppresses this type of output, which usually is lengthy and of questionable value. It is not used by the assembler or RPG II.

SUBLIB=AE
SUBLIB=DF
Is used by the assembler to control the source statement sublibraries which are accessed for COPY statements and macro definitions.

NOSYM
SYM
Is used by certain language compilers. SYM requests that a printout of symbolic names defined in the program be produced on the terminal. This option is not used by the assembler or RPGII.

NOXREF
XREF
Is used by the assembler and some language compilers. XREF requests that a cross-reference listing of symbolic names defined within the program be produced on the terminal.

RLD
NORLD
Is used by the assembler to control whether or not the relocation dictionary is printed at the end of an assembly. The default is the value set within your installation.

The following options apply to the control of jobs within the interactive environment. These options are unique to VSE/ICCF or have unique meanings within VSE/ICCF.

ANYPHASE=nn
Normally the COMREG (communication region) field which defines the high end of any phase with the same first four characters in its name as the initially loaded phase is set to the end of the phase loaded first. If this address proves unsatisfactory, you can set this value to any arbitrary point within the non-GETVIS portion of the interactive partition. For example, ANYPHASE=40 will set this address to the 40K point within the interactive partition.

CLEAR
NOCLEAR
Unless NOCLEAR is specified, the entire interactive partition address space (except for the first 6K) is cleared whenever a job ends. If for some reason you want the area to be retained, specify the NOCLEAR option.

NOCONT
CONTINUE

CONTINUE causes the job to be placed in continuous output mode just as if the /CONTINU command had been entered after the /RUN or /EXEC. In continuous mode, print output is displayed continuously; you need not press ENTER, or the Carriage Return key, to obtain each new screen of data.

NODUMP
DUMP

When the DUMP option is set, an abnormal end of a job in an interactive partition causes the DUMP program to be invoked. The DUMP program allows you to display registers and storage areas within your program. The DUMP program is invoked for VSE/ICCF return codes 02, 07, 08 and 14. These codes are described in z/VSE Messages and Codes 2.

The dump program uses a work area of 3K in the interactive partitions's GETVIS area. If the address of this area is destroyed, the first 3K block of the interactive partition is taken as work area.

EOFPRT
NOEOFPRT

When the NOEOFPRT option is set, no end of file record (/* */) is placed into the print area at the end of the print output after the job has terminated. This option may be useful, for example, in handling conversational reads, which reset the pointer to the start of the print area. In this case, the end-of-file record would be written at the start of the print area, thus effectively destroying the contents of the area.

GETVIS=nnn
GETVIS=P-nnn
GETVIS=AUTO

Normally in VSE/ICCF, 48K of your background storage area is reserved as GETVIS space. You can alter this default for a job step by specifying the actual amount of storage to be reserved as GETVIS area. For nnn, specify the amount of GETVIS area to be reserved in increments of 1024 bytes. Depending on the environment in which VSE/ICCF runs, the amount which you specify is rounded up to the next multiple of the page size. GETVIS=64 would reserve 64K (65,535) bytes. GETVIS=0K would indicate that a minimum GETVIS space of 48K is to be reserved.

You can also specify the GETVIS area by specifying the entire partition minus some amount (P-nnn). For example, if your program requires 52K, you can specify GETVIS=P-52. This would reserve 52K for your program (actually, slightly less than 52K due to some VSE/ICCF save areas at the low end of the interactive partition). The rest of the interactive partition would be allocated to the GETVIS area.

GETVIS=AUTO tells VSE/ICCF to calculate GETVIS space according to the largest phase in the program to be loaded. VSE/ICCF takes the phase with the highest ending address that starts with the same four characters as the program to be loaded. The remaining space in the interactive partition becomes GETVIS space, provided minimum and maximum requirements are met.

The minimum value for the GETVIS area is 48K. Requests for less than this will be set to 48K without any error messages. The GETVIS area can never exceed the partition size minus 20K; it is reset to 48K at the start of every job step.

GO
NOGO

This option controls the automatic invocation of the LINKNGO program. Normally, it is invoked automatically at the end of a job when a compiler has produced an object module which has not yet been run or when a /DATA job entry statement is encountered following the input to a compiler. However, when the NOGO option is set, LINKNGO will not be invoked automatically (this does not preclude explicit invocation). This option is useful if you want to produce the object program but not load and run it. For example, you may only want to save it in the library.

The options LOAD and NOLOAD are accepted to provide compatibility with ETSS II; they have the same meaning as GO and NOGO.

NOINCON
INCON

This option controls where card read data, VSE units SYSIPT and SYS005 (or other default unit), is obtained. Normally, card read data is obtained from the job stream itself. It is processed sequentially until all data has been read and processed.

When INCON is specified, any input to be read from VSE units SYSIPT or SYS005 (or other default unit) will be directed instead to the terminal for conversational input. This is one way of writing a conversational program in programming languages that do not support input from the console (SYSLOG or SYS005). This option has the same effect on programs run by the /LOAD function as the /DATA INCON option has on programs loaded by LINKNGO from object modules.

NOJSDATA
JSDATA

Normally, a /LOAD job entry statement marks the end of one job step and the beginning of the next. Also a /DATA statement signals end of file to the job reading data from the job stream. Occasionally, however, it may be useful to be able to read actual VSE/ICCF job streams into a program. This would require that the checks for /LOAD and /DATA statements be bypassed. The JSDATA (Job Stream as Data) option causes such a bypass. All remaining 80-character records in the job stream (beyond the first set of job entry statements) can be read into the running program without interpretation by VSE/ICCF. Note, however, that the /INCLUDE statement will still be interpreted normally, regardless of the setting of this option.

NOLOG
LOG

When the LOG option is specified, all job entry statements encountered by the job scheduler are displayed. This may be useful for the debugging of a job stream.

NOOBJECT
OBJECT

When VSE/ICCF encounters an object deck in a job stream, it normally forces end of file for the previous step's data input and then loads the object deck for execution as the next step of the job. However, you may sometimes want to process an object deck as ordinary data, for example when running the VSE Librarian to catalog an object deck. By specifying the OBJECT option, the object deck in the job stream is treated as ordinary data.

NOPERM
PERMFILE

Normally, an OPEN request for a file named IJSYS0n will be converted to a request for IKSYSpn where 'p' is the interactive partition identifier. Sometimes, however, you may want to actually open the file under the IJSYS0n name. This often occurs in FORTRAN when permanent files must have the IJSYS0n name. When the PERMFILE option is set, the file name will not be altered and the OPEN will proceed for the unaltered file name. The PERMFILE option does not apply to a compilation; it applies only to the execution of a program other than a compiler.

PROMPT
NOPROMPT

Normally, when a conversational read is encountered, VSE/ICCF causes a prompt to be issued to the terminal. For typewriter terminals the prompt is a question mark. For 3270 terminals the prompt is '*ENTER DATA'. When NOPROMPT is specified, the issuance of the prompt is bypassed, which can be useful if the program being run issues its own prompt.

NOSAVE
NORESET
SAVE
RESET

The SAVE and RESET options together control the setting of the punch-area EOF pointer and current (or next) record location pointer.

Normally, at the start of a job, the punch area is set to a null condition. This means that its pointers are reset and it is reused from its beginning. Each step that places data in the punch area places this data immediately behind the punch area from the previous step. In this way, for example,

successive compilations can create several subroutines which are to be linked together via the LINKNGO program. After a request for the LINKNGO program has been processed, the line pointer is set back to the beginning of the punch area for later compilations and executions.

When the SAVE option is used in the first step of a job it causes the automatic 'set to null' function (which is usually performed during the initiation of the initial job step) to be bypassed. Thus, the punch area output of a previous job (not step) can be made available to the current job. This is useful, for example, if you want to access the object deck from a compilation performed in a previous job. The SAVE option is automatically set when the DTSPROCS utility is run.

Normally, the next record pointer in the punch area will be set to the next output record past the output of the previous step within the job. When the RESET option is on, the pointer will be set back to the first record within the punch area. This option is useful, for example, if you want to reset the pointer to the beginning of the punch area since you have finished processing the existing data within the punch area.

NOSPECIAL
SPECIAL

The SPECIAL option must be set if the job to be run uses any special programming techniques such as rewriting job stream data to disk (read-no-feed followed by write) or reading backward in the job stream. If this option is used, it is your responsibility to check for special conditions, such as end of data when reading backward.

TIME=mm
TIME=mm,nn

This option may be used to set the execution time monitoring factors for the job. mm is specified in execution units which closely approximate seconds of elapsed time. If the job exceeds this value, it is automatically canceled. nn is specified in seconds. It represents the total amount of time the job may be in the interactive partition whether running or not. The maximum values for mm and nn are 32767 and 65535, respectively. This option, which you can specify within a job stream, has the same effect as the /SETIME command.

NOTRUNC
TRUNC=nn

This option causes printer output to be truncated to a length of 78 characters. This can be useful if the screen of your terminal is not wide enough to accommodate the print line. The use of the TRUNC option makes the output easier to read.

If TRUNC is not in effect and the /SHIFT setting is reset by a /SHIFT OFF command, any lines whose width exceeds the capacity of the device will be displayed on two lines. If nn is not specified or is specified as '00', the first 78 characters of the print line will be written to the print area. If nn is specified, nn columns are removed from the front of the print line and the next 78 bytes are written to the print area. Thus, TRUNC=20 would cause print positions 21 through 98 to be displayed.

The TRUNC option can also cause a job to run faster since less data need be written to the print area and to the terminal.

PRCLOSE
PUCLOSE

The PRCLOSE option closes a member specified in a previously encountered /FILE statement that specified UNIT=SYSLST or UNIT=SYSLLG. The option directs further print output to the print area behind any records that might already be in the print area. PUCLOSE has the same effect for a /FILE statement that includes UNIT=SYSPCH; the option directs further punch output to the punch area.

SYSLOG
SYSIPT

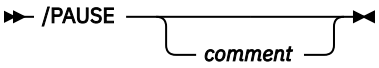
This option allows you to let the VSE librarian read its input from SYSLOG or from SYSIPT. When the VSE librarian runs in a batch partition, it reads from SYSIPT or SYSLOG depending on the VSE job control statements by which it is invoked. When running in an interactive partition, the VSE librarian takes its direction from the SYSLOG/SYSIPT option. SYSLOG means that the librarian prompts you for its input. SYSIPT causes the librarian to read its input from SYSIPT. This option can be used also for other programs that optionally read from SYSLOG or SYSIPT.

Example

```
*READY
/input
/load vfortran
/option list listx xref trunc
/option getvis=auto
     write (3,10)
10   format ('VSE/ICCF demo')
     end
/end
*READY
/run
*RUN REQUEST SCHEDULED
```

/PAUSE Job Entry Statement

The /PAUSE job entry statement is used in a job stream to cause the scheduler to display a message (if 'comment' is specified) and then to halt.



comment
Is a message that is displayed on the terminal when the pause occurs.

In the comment, you can use the tags &USR, &TRM, and &PRT These tags will be replaced by information as shown below and before the message is written to your terminal:

&USR
By your user identifier

&TRM
By the terminal identification.

&PRT
By the interactive partition number.

During the halt you can type in job entry statements to further define the job or merely press the ENTER key to cause the scheduler to continue processing. /INCLUDE statements cannot be entered at a /PAUSE; they must be within the job stream.

Example

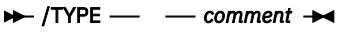
```
*READY
/input
/load spcutil
/file name=utlwork,tracks=10
/pause ** enter appropriate upsi **
/endrun
*RUN REQUEST SCHEDULED FOR CLASS=A
*    - ALLOCATION FOR UTLWORK - SERIAL=111111 UNIT=SYS001
       LOC=3267,10
**ENTER APPROPRIATE UPSI**
?
/upsi 1011
?
(press ENTER)
 ...
 ...                    (Processing of the job continues)
 ...
```

/RESET Job Entry Statement

The /RESET job entry statement is used in a job stream to cause any prior /ASSGN statements to revert to the installation defaults.

/TYPE Job Entry Statement

The /TYPE job entry statement causes a comment to be displayed.



comment
Is a line of comment data that you want to be displayed during the execution of the job in which the command occurs. The comment string may include double-byte characters.

The comment appears on your screen at the point in the job stream at which the command occurs.

In the comment, you can use the tags &USR, &TRM, and &PRT These tags will be replaced by information as shown below and before the comment is written to your terminal:

&USR
By your user identifier

&TRM
By the terminal identification.

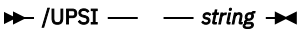
&PRT
By the interactive partition number.

Example

```
*READY
/input
/load listutil
/file name=savdta,ser=etspak,id='jc.saved.data'
/type **********
/type *the following is a
/type *listing of the file
/type **********
list file=savdta
/endrun
*RUN REQUEST SCHEDULED
**********
*THE FOLLOWING IS A
*LISTING OF THE FILE
**********
ABRAMS, JOHN 234 ANY STREET
BAKER, B.A. 1912 FOREST DR
 ...
 ...
 ...
```

/UPSI Job Entry Statement

The /UPSI job entry statement is used to set the VSE User Program Switch Indicators from the job stream.



string
Is an eight-character string. Each of these characters represents a user program switch indicator and sets the switch. If the character is:

0
The corresponding switch is set off.

1
The corresponding switch is set on.

X
The setting of the corresponding switch remains unchanged.

If a position is not coded, an 'X' is assumed.

Up to eight user program switch indicators (UPSI) can be set in the job stream and tested by your program.

All UPSI switches are set off (zero) at the beginning of each job. Once an UPSI switch is set on during a job, it remains on until it is turned off by another /UPSI statement.

The UPSI switches, generally called UPSI-0 through UPSI-7, can be referred to in the various programming languages as follows:

In assembler language via the COMRG macro
In RPG II via the indicators U1 through U8

The FORTRAN user can refer to UPSI switches via an assembler subroutine.

Programs which will be compiled, loaded (by the LINKNGO utility) and run in interactive partitions should not use UPSI-0. This UPSI is used in LINKNGO for control of where to read the object deck.

Examples

1. Set UPSI-0 switch off and UPSI-1 and UPSI-2 on. Leave switches UPSI-3 through UPSI-7 unchanged:

```
/UPSI 011
```

2. Set UPSI-3 switch on, leave all other settings unchanged:

```
/UPSI XXX1
```

3. Check the UPSI setting in an assembler program and set UPSI at program invocation:

```
*READY
/input
/load     assembly
upsame    start       0
          print       nogen
          balr        5,0
          using       *,5
          comrg
          mvc         msg+10(3),=c'off'
          tm          23(1),X'40'        test upsi-1
          bz          swoff
          mvc         msg+10(3),=c'on '
swoff     excp        msgccb
          wait        msgccb
          eoj
msgccb    ccb         syslog,msgccw
msgccw    ccw         9,msg,0,l'msg
msg       dc          c'switch is xxx'
          end
/load linkngo
/upsi 01  (this will set on UPSI-1)
/endrun
*RUN REQUEST SCHEDULED
(assembler, LINKNGO and execution output)
```

Chapter 8. Dump Commands

This section discusses the available dump commands and gives examples illustrating their use.

You can enter a dump command only while the VSE/ICCF dump program is in control. That program is invoked automatically if:

1. The dump option (/OPTION DUMP) has been specified in a job stream and
2. The interactive partition execution ends because of an invalid condition, program check, or VSE/ICCF return code 02, 07, 08 or 14. These return codes are described in z/VSE Messages and Codes 2.

You can enter the dump command when the message:

```
K404D ENTER DUMP COMMAND
```

appears on your screen.

System commands which are effective when a conversational read is outstanding (such as /SKIP and /CANCEL), are effective also at this time.

The Null Command: If you press ENTER without having typed in a command, VSE/ICCF assumes a display forward dump command as follows:

```
DISPFWD * +256
```

This results in a display of the first or next 256 bytes of your program storage in hexadecimal and character format. Pressing ENTER again causes the next 256 bytes to be displayed, and so on.

The Scan/Locate Pointer: Certain dump commands set what is called the scan/locate pointer. This pointer is initially set at the program load point and can be varied by dump commands such as SEARCH, LOCATE, and TOP. The address value of the scan/locate pointer can be referenced in dump commands by the character '*'.

For example, to locate and display data areas or constants which are in static storage locations you might precede certain groups of key constants or data areas with a unique identifier as in the following example:

```
DC C'**DBG**'
```

Then if your program terminates abnormally, you can issue the following commands to locate and display the area.

```
K404D ENTER DUMP COMMAND
locate '**DBG**'
403756 0008CE 0008CE   5C5CC4C2C75C5C ...   **DBG** ...
K404D ENTER DUMP COMMAND
dispfwd * +128
 ...
 ...                    (Storage display)
 ...
```

Reference Area/Origin Command: Whenever a dump command calls for an address operand, it is possible to specify a hexadecimal relative address. Usually the address which you specify is obtained from a compiler listing and thus will normally be relative to the program's load point. This is the assumption made by the Dump Program. However, it is possible to vary the reference area by using the ORIGIN dump command.

The ORIGIN command allows you to vary the base address for relative address calculation. This can be useful when displaying fields within an area when all you have is a listing of fields relative to a certain base point. This can be useful also when you are debugging a subroutine or subprogram loaded with the main program. By setting the ORIGIN to the start of the routine, all relative addresses would then correspond to the program listing for the subroutine.

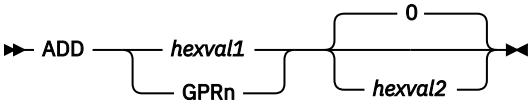
Another use might be to set the origin or reference area to the start of the main program area when this area is, for some reason, not the first control section loaded; in other words, when the main portion of the program does not start at the program load point.

Summary of Dump Commands

Command	Function
ADD	Adds two hexadecimal numbers or one hexadecimal number and the contents of a general purpose register.
Backward	Reduces the scan/locate pointer by a given number of bytes.
CANcel	Ends the dump program and the displays status.
DA	See the DISPLAY command.
DC	See the DISPLAY command.
DEC	Converts a hexadecimal value to decimal.
DF	See the DISPLAY command.
DIN	See the DISPIND command.
DISPAct	Same as DISPLAY, but assumes actual rather than relative addresses.
DISPChar	Same as DISPLAY, but only character representation of data is displayed rather than both character and hexadecimal.
DISPFwd	Same as DISPLAY, except that the scan/locate pointer is advanced by the length of the display.
DISPInd	Displays an area of storage whose address is determined by a base and a displacement. The base address can be the contents of a general register.
Display	Displays the contents of program storage, general registers or floating point registers.
DUmp	Obtains a display of all general and floating point registers as well as all user program storage.
End	Ends the dump program.
Eoj	Same as the END command.
Forward	Advances the scan/locate pointer by a specified number of bytes.
HEX	Converts a decimal value to hexadecimal.
Locate	Locates a string of data characters within your object program area.
ORigin	Sets the basis for relative to actual address calculation to a location other than the program load point.
Point	Sets the scan/locate pointer to a specified address.
SAVE	Obtains a hardcopy dump.
SEarch	Locates a string of data characters within your program area.
SHow	Obtains various status displays.
STatus	See the SHOW command.
SUB	Subtracts one hexadecimal number from another or subtracts one hexadecimal number from the contents of a general register.
Top	Sets the scan/locate pointer to the first position within the program or defined reference area.

ADD Command

The ADD command is used to add two hexadecimal values and to display the sum. It can also be used to add the contents of a general purpose register and a hexadecimal value.



hexval1
Is a one to eight hexadecimal digit value to be added to hexval2.

GPRn
Is the designation of a general purpose register, where n can be any value from 0 through 9 or A through F. The contents of the specified register are added to the value specified as the second operand.

hexval2
Is a one to eight hexadecimal digit value. After having added to this value the value you supplied by the first operand, VSE/ICCF displays the result on your screen. If the (hexadecimal) result is an address within your program, its value relative to the start of your program (or to the address specified in an ORIGIN command) is also displayed.

This command is useful when dealing with base-displacement addressing. For example, a field in storage may be at a displacement of X'04C' from general register 6, the command

```
ADD GRP6 04C
```

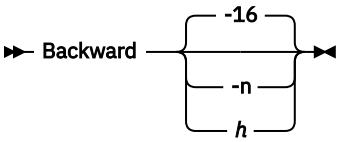
will return the actual and relative addresses of the desired field (see also the section "DISPIND Command" on page 261).

Example

```
K404D  ENTER DUMP COMMAND
*ENTER DATA?
add 24A 137C
K439I  HEX VALUE IS 000015C6
K404D  ENTER DUMP COMMAND
*ENTER DATA?
add gpr6 4C
K439I  HEX VALUE IS 00B76A8 REL=000640
```

BACKWARD Command

The BACKWARD command causes the scan/locate pointer to be reduced by the specified number of bytes.



-n
h
For n, specify a decimal number indicating the number of bytes by which the scan/locate pointer is to be reduced.

For h, specify a hexadecimal number indicating the number of bytes by which the scan/locate pointer is to be reduced.

If the specified number exceeds the number of bytes between the current ORIGIN command point (or load point) and the current scan/locate pointer, the pointer is set to the current origin (usually the load

point). That is, the pointer cannot be backed up farther than the start of the reference area (ORIGIN command point).

Other dump commands which affect the scan/locate pointer are FORWARD, POINT, TOP, SEARCH, LOCATE, and DISPFWD.

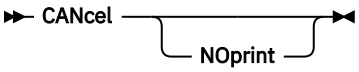
After having set the scan/locate pointer, you can reference the new value by an asterisk (*) as shown in the example below.

Example

```
K404D  ENTER DUMP COMMAND
*ENTER DATA?
display
080170 000030 000030 4110001095007DF5...
K404D  ENTER DUMP COMMAND
*ENTER DATA?
back -24
K441I  ACTUAL=08158  REL=000018
K404D  ENTER DUMP COMMAND
*ENTER DATA?
disp *
080158 000018 000018 D203850A7C76...
```

CANCEL Command

The CANCEL command is used to terminate the dump program.



NOprint
Specifies that all displays are to be bypassed and only the termination messages are to be displayed.

When the CANCEL command is entered, you will receive the following information:

- The termination status
- The type of halt condition that caused the termination as well as its location in the program
- The contents of the 16 general purpose registers.

Example

```
K404D  ENTER DUMP COMMAND
*ENTER DATA?
cancel
K401I  LOAD HI-PTN HI-PHS ORIGIN SCAN *** STATUS ***
ACTUAL:   0B7870 0B8C1D 0B7870 0B789E
RLATIV:   000000 0013AD 000000 00002E
K442I  PSW= ...
           ...
GP 0-F    00000050 000B78D8 ...
          A00B2E40 0A16180C ...
K433I  DUMP PROGRAM CANCELED
```

DA Command

See the section “DISPLAY Command” on page 262.

DC Command

See the section “DISPLAY Command” on page 262.

DEC Command

The DEC command gets the decimal equivalent of a hexadecimal number.

hexval
Is a 1- to 8-digit hexadecimal number.

If this number is greater than 7FFFFFFF, it is assumed that the value represents a two's complement value and a negative decimal number is returned.

Examples

```
K404D  ENTER DUMP COMMAND
*ENTER DATA?
dec 1ac
K440I  DECIMAL VALUE IS 0000000428
K404D  ENTER DUMP COMMAND
*ENTER DATA?
dec ffffffff
K440I  DECIMAL VALUE IS -0000000001
```

DF Command

See the section “DISPLAY Command” on page 262.

DIN Command

See the section “DISPIND Command” on page 261.

DISPACT Command

See the section “DISPLAY Command” on page 262.

DISPCHAR Command

See the section “DISPLAY Command” on page 262.

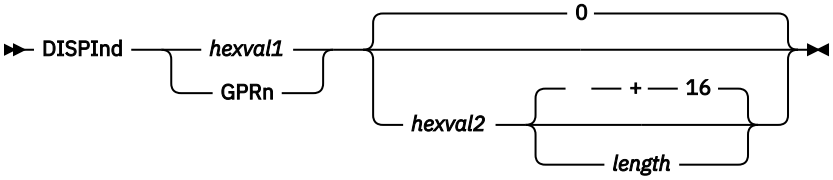
DISPFWD Command

See the section “DISPLAY Command” on page 262.

DISPIND Command

The command can be used to display an area of storage if, instead of the exact address to be displayed, only the offset or displacement from another address is known. That other address may be in a general purpose register.

The operands shown here for the DISPIND command are valid also for the DIN command.



hexval1
GPRn
For hexval1, specify a one- to six-digit hexadecimal value which is interpreted as an actual base address.

For n in GPRn, specify the general purpose register (one of 0 through 9 or A through F) whose contents are a base address.

VSE/ICCF adds the base address to the displacement address that you specify, explicitly or by default by hexval2. The result is a location to be displayed.

hexval2
Is a one- to six-digit hexadecimal value or an equate to such a value; zero is assumed if you omit the operand. A value for hexval2 must be specified if length is specified, even if that value is 0.

length
Can be a hexadecimal or a decimal value preceded by a plus sign, which indicates the amount of data to be displayed.

This command is useful with base-displacement addressing. For example, a field in storage may be at a displacement of X'04C' from general register 6, the command DISPIND GPR6 04C will cause the field to be displayed.

Note: The /ATTEN command or Attention key [7] can be used to end display functions and redisplay the dump command prompt.

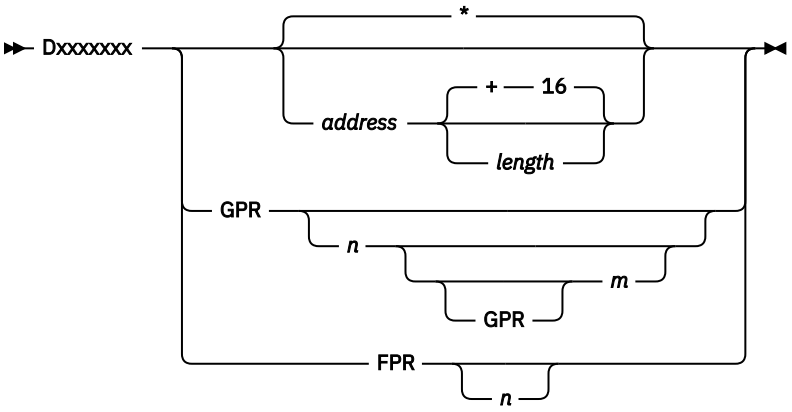
Example

```
K404D  ENTER DUMP COMMAND
*ENTER DATA?
dispind gpr6 4c
K439I  HEX VALUE IS 000B76A8 REL=000640
0B76A8 000640 000640  00374C01982D ...
```

DISPLAY Command

The DISPLAY command is used to view the contents of program storage, of the general purpose registers, or of the floating point registers.

The operands shown below for the DISPLAY command apply also to the various forms of the command discussed following the format description of the command.



For **operation**, shown above as Dxxxxxxx, you can use any of the following:

[7] The definition of the Display/Attention key is a VSE/ICCF tailoring option. The default is PA3. It may be different for your system.

Display
The specified address is assumed to be within your program area, relative to the start of operand1, it is assumed that this address is relative to the start of your program or to the address specified in an ORIGIN command. The data at the referenced storage location is displayed in both hexadecimal and character representation. The command does not change the scan/locate pointer.

DISPAct
DA
In the (display actual) command, any specified storage address is assumed to be an actual (not relative) address.

DISPChar
DC
The (display character) command causes only the character representation of storage to be displayed. Instead of 32 hex digits and 16 characters per line, the display is 32 characters per line.

DISPFwd
DF
The (display forward) command advances the scan or locate pointer to the ending display location plus one. This permits repeated DF '*' commands to return successively higher storage locations.

A description of the **operands** follows:

address
Is the first storage address to be displayed. If this operand is omitted, the address of the scan/locate pointer is used. The operand can be a hexadecimal address or an asterisk (*), the locate pointer reference.

length
is the hexadecimal or decimal length of the data to be displayed. If a decimal length is intended, it must be preceded by a plus sign.

GPR
Requests the display of all general purpose registers.

GPRn
Requests the display of general purpose register 'n' only if the GPRm operand is omitted. GRPn is the first of a series of general purpose registers to be displayed if the GPRm operand is given. For more details, see the description of the GPRm operand, below.

For n, specify the number of the first (or only) register as a hexadecimal value (0 through 9 and A through F).

GPRm
Specifies the last of a series of general purpose registers to be displayed, the first in the series being register n of the GRPn operand. The characters 'GPR' may be omitted; 'm' by itself is enough.

For m, specify the number of the last register as a hexadecimal value (0 through 9 and A through F).

When GPRn GPRm is specified, the general registers are displayed in the order as indicated. Thus, 'GPRE GPR1' results in a display of registers 14, 15, 0, and 1.

FPR
Displays all four floating point registers.

FPRn
Displays a pair of floating point registers, where n can be 0, 2, 4, or 6.

When the display commands are used to print the contents of storage, the display always begins with three 6-digit hexadecimal addresses: the actual storage address of the data, the address relative to the program's load point, and the address relative to the beginning of your reference area (initially set at the load point and optionally modified by the ORIGIN command). Following these addresses is the data itself. If more than 16 bytes are displayed, the hexadecimal display is grouped into fullwords with each word separated by a blank.

Note:

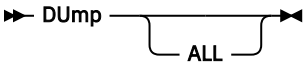
1. The DISPLAY command can be used to obtain the actual equivalent of a relative address, since both actual and relative addresses are displayed on the print line. Conversely, the DISPACT command can be used to obtain the relative address equivalent of an actual address.
2. The /ATTEN command or Attention key [8] can be used to end display functions and to cause the ENTER DUMP command prompt to be issued again.

Examples

```
K404D  ENTER DUMP COMMAND
*ENTER DATA?
display 134A +200
K446I  ACTUAL LOAD/REL ORG/REL - STORAGE DISPLAY
0B4730 001340 001340 05F00700 ...  *.0.. ...
0B4740 001350 001350 00085C80 ...  *..*. ...
 ...
K404D  ENTER DUMP COMMAND
*ENTER DATA?
disp
0B3350 000000 000000 05F047F0F03E ... .0.00.
K404D  ENTER DUMP COMMAND
*ENTER DATA?
d gpr
GP 0-F 00000050 00098FFF ...
       900F28A2 000B6400 ...
K404D  ENTER DUMP COMMAND
*ENTER DATA?
d gpr3 5
GP 3-5 0000FF90 000B5DC8 500B6436
K404D  ENTER DUMP COMMAND
*ENTER DATA?
locate 'this'
00B39C 00004C 00004C  E3C8C9E240C9 ... THIS IS...
K404D  ENTER DUMP COMMAND
*ENTER DATA?
df 5C
00B3AC 00005C 00005C  C6D6D9 ... FOR...
K404D  ENTER DUMP COMMAND
*ENTER DATA?
df
00B3BC 00006C 00006C  C5D5C440 ... END ...
K404D  ENTER DUMP COMMAND
*ENTER DATA?
back -16
K441I  ACTUAL=00B3BC REL=00006C
K404D  ENTER DUMP COMMAND
*ENTER DATA?
dc
00B3BC 00006C 00006C  END OF MESSAGE...
```

DUMP Command

The DUMP command is used to display all general and floating point registers as well as all program storage areas.



ALL
Specifies that the entire interactive partition area is to be displayed.

If no operand is specified for the DUMP command, all general and floating point registers as well as the entire program area up to the end of the last phase loaded will be dumped. To get a hard copy of the dump, use the SAVE command as described under “SAVE Command” on page 268.

The format of the dump is the same as that obtained by using the DISPLAY command, except that areas of storage which are all binary zeros are not displayed.

[8] The definition of the Display/Attention key is a VSE/ICCF tailoring option. The default is PA3. It may be different for your system.

Example

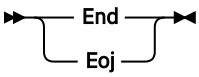
```
K404D  ENTER DUMP COMMAND
dump
```

END Command

See the section "EOJ Command" on page 265.

EOJ Command

The EOJ (and also END) command ends dump-program execution.

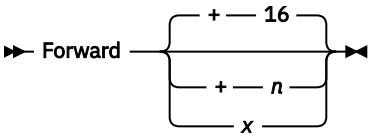


Example

```
K404D   ENTER DUMP COMMAND
*ENTER DATA?
e
K434I   DUMP PROGRAM ENDED
```

FORWARD Command

The FORWARD command advances the scan/locate pointer by the specified number of bytes.



+n
x

The operand specifies a number indicating the number of bytes by which the scan/locate pointer is to be advanced.

The number can be decimal, which is indicated by a preceding plus sign. This is shown above as +n.

The number can be in hexadecimal notation, which is indicated by no preceding plus sign. This is shown above as x.

If the specified number would advance the pointer beyond the end of the program, a message is issued and the pointer is not changed.

Other dump commands which affect the scan/locate pointer are: BACKWARD, POINT, TOP, SEARCH, LOCATE and DISPFWD.

After having set the scan/locate pointer, you can reference it by an asterisk (*) as shown in the example below.

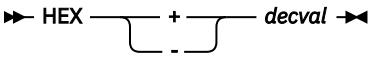
Example

```
K404D  ENTER DUMP COMMAND
*ENTER DATA?
locate d207
080EB8 000E48 000E48 D20756AC543A...
K404D  ENTER DUMP COMMAND
*ENTER DATA?
forward +4
K441I  ACTUAL=080EBC REL=000E4C
K404D  ENTER DUMP COMMAND
```

```
*ENTER DATA?
disp *
080EBC 000E4C 000E4C 543A41110008...
```

HEX Command

The HEX command is used to obtain the hexadecimal equivalent of a decimal number.



+decval
-decval
For decval, specify the 1- to 8-digit decimal number that is to be converted to hexadecimal notation.

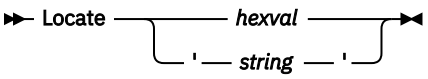
The hexadecimal number is returned as a fullword.

Example

```
K404D  ENTER DUMP COMMAND
*ENTER DATA?
hex +128
K439I  HEX VALUE IS 00000080
K404D  ENTER DUMP COMMAND
*ENTER DATA?
hex -2
K439I  HEX VALUE IS FFFFFFFE
```

LOCATE Command

The LOCATE command is used to find a certain combination of characters within a program.



hexval
Is from 2 to 16 hexadecimal digits, 0 through 9 or A through F. An even number of hex digits is required.

'string'
Is a string of 1 to 8 EBCDIC characters enclosed within a pair of single quotation marks. The trailing quotation mark is required only if the character string has one or more embedded or trailing blanks.

The command alters the scan/locate pointer. The other commands which alter the scan/locate pointer are: BACKWARD, DISPFWD, FORWARD, POINT, SEARCH, and TOP.

The scan for the desired data always begins at the location of the scan/locate pointer and proceeds to the end of your program. If the desired data is not located, a message is issued, and the scan/locate pointer is set to the beginning of the program area (current ORIGIN value).

If you want to begin your search at the beginning of your program rather than at the scan/locate pointer address, issue the TOP command prior to the LOCATE, or use the SEARCH command.

If you are scanning for more than one occurrence of the same string, you must advance the pointer before each LOCATE; otherwise you will continually find the data at the same location. Use the FORWARD command (F +1) to do this.

Note:

1. The LOCATE command is useful in locating data within a program when a symbol map is not available. Unique identifiers can be included as constants within or in front of the actual data areas in the source

program. The LOCATE can then be used to find the data locations. For example, you might code the following:

```
    CTLFIELD DC'*CTL*'
```

The LOCATE can be used to find the field named CTLFIELD as in the first example below.

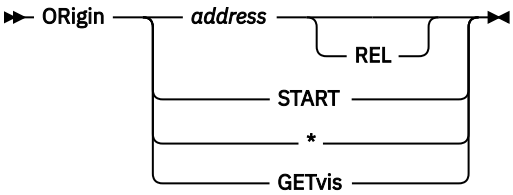
2. The /ATTEN command (or Attention key [9]) ends a LOCATE function that is in progress.

Example

```
K404D  ENTER DUMP COMMAND
*ENTER DATA?
top
K404D  ENTER DUMP COMMAND
*ENTER DATA?
locate '*ctl*'
0D348A 000B82 000B82 5CC3E3D35C...   *CTL*...
K404D  ENTER DUMP COMMAND
*ENTER DATA?
forward +32
K441I  ACTUAL=0D34AA  REL=000BA2
K404D  ENTER DUMP COMMAND
*ENTER DATA?
dispfwd * +64
 ...
```

ORIGIN Command

The ORIGin command is used to change the start of the reference area that is used as the base in calculating relative to actual addresses.



address
Is the actual address of the start of the reference area.

REL
Specifies that the previous address is not an actual address but is relative to the program load point.

START
Indicates that the reference area (origin) is to be re-established at the load point of the program which is its initial value prior to any ORIGIN commands.

*****
Indicates that the reference area origin will be the scan/locate pointer value.

GETvis
Indicates that the reference area origin will be the start of the interactive partition's GETVIS area.

This command can be used to dump storage in a control section, subprogram or subroutine other than the primary program being tested. The actual address where the subroutine was loaded can be obtained from the link edit or load map and specified as the operand for this command. From this point on, all relative addresses used in commands would be relative to the start of the subroutine.

The command could also be used in referring to a very large record area whose format you know. The origin could be set to the start of this area.

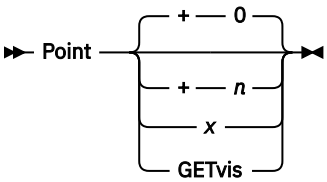
[9] The definition of the display/attention key is a VSE/ICCF tailoring option; the default is PA3. It may be different for your system.

Example

```
K404D  ENTER DUMP COMMAND
*ENTER DATA?
org 86c80
K420I  ORIGIN ADDRESS=086C80, REL=003410
K404D  ENTER DUMP COMMAND
*ENTER DATA?
disp 1ba
086E3A 0035CA 0001BA  F0F0F3F7F8C2...   00378B...
K404D  ENTER DUMP COMMAND
*ENTER DATA?
org start
K420I  ORIGIN ADDRESS=083870, REL=******
```

POINT Command

The POINT command sets the scan/locate pointer to the specified relative location within the program (reference area).



+n
x
: The operand specifies a number indicating the number of bytes by which the scan/locate pointer is to be advanced into the reference area.

  The number can be decimal, which is indicated by a preceding plus sign. This is shown above as +n.

  The number can be in hexadecimal notation, which is indicated by no preceding plus sign. This is shown above as x.

  If the specified number would advance the pointer beyond the end of the program, a message is issued and the pointer is not changed.

GETvis
: Specifies that the scan/locate pointer is to be set to the beginning of the interactive partition GETVIS area.

Other dump commands which affect the setting of the scan/locate pointer are: BACKWARD, DISPFWD, FORWARD, LOCATE, SEARCH, and TOP.

If no operand is specified in the POINT command, its effect is the same as that of the TOP command.

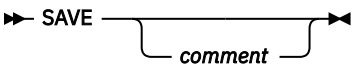
If the operand specifies a value which would advance the scan/locate pointer to an address outside the program, a message is issued and the pointer is not changed.

Example

```
K404D  ENTER DUMP COMMAND
*ENTER DATA?
point 13a
K441I  ACTUAL=0D348A  REL=000882f
K404D  ENTER DUMP COMMAND
*ENTER DATA?
dispchar *
0833AA 00013A 00013A  THE MESSAGE IS H ...
```

SAVE Command

The SAVE command is used to obtain a hardcopy dump of the interactive partition.



comment
Is a character string or up to 24 characters. The comment could, for instance, be a short description of the error as a reminder of why the dump was taken.

The dump is first written to the VSE dump library; it can then be displayed or printed with Info/Analysis.

Examples

1. Before the dump is written to disk, a message with the dump identification appears on the screen. This is followed by another message with the same identification indicating that the dump is now complete.

```
K404D  ENTER DUMP COMMAND
*ENTER DATA?
save invalid addr in pgmabc
0S30I DUMP STARTED. MEMBER=00000002. DUMP IN SUBLIB=SYSDUMP.F2
1I51I DUMP COMPLETE.
K442I DUMP 00000002 SAVED
K404D  ENTER DUMP COMMAND
*ENTER DATA?
```

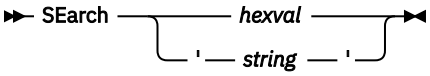
   The identification SYSDUMP.F2.00000002 is the selector to be used for displaying or printing the dump via Info/Analysis.

2. Make sure that the dump library is assigned (`LIBDEF  DUMP,CATALOG=SYSDUMP.F2,PERM`), and that it exists on your system. The library should contain enough free space if you want a hardcopy dump to be taken. In case the dump cannot be written to disk for lack of space, an appropriate message will be returned.

```
K404D ENTER DUMP COMMAND
*ENTER DATA?
save pgm check in calc95
K436I DUMP LIBRARY FULL OR NOT DEFINED - DUMP NOT SAVED
K404D Enter DUMP command
*ENTER DATA?
```

SEARCH Command

The SEARCH command is used to find a particular character string within the program. It begins its scan operation at the first location in your program (or the first location in an area specified by the ORIGIN command). Thus, the command is equivalent to a LOCATE command preceded by a TOP command.



hexval
Is a value of 2 to 16 hexadecimal digits from 0 through 9 and A through F. An even number of hexadecimal digits is required.

'string'
Is a string of up to eight EBCDIC characters within single quotation marks. The ending quotation mark is required only if the character string includes trailing blanks.

The SEARCH command sets the scan/locate pointer to the address where a match occurred. If no matching string can be located, the scan/locate pointer is set to the first position in the scan area. Other dump commands which set the scan/locate pointer are: BACKWARD, DISPFWD, FORWARD, LOCATE, POINT, and TOP.

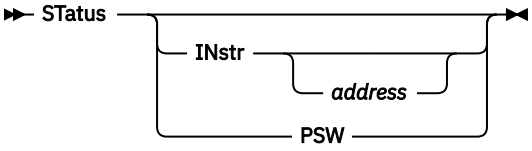
See the description of the LOCATE command for an example of how to use this command.

SHOW Command

See the section "STATUS Command" on page 270.

STATUS Command

The STATUS command is used to obtain various displays of information within the program. The operands shown here for the STATUS command apply also to the SHOW command.



INstr address

INstr
tells VSE/ICCF to display information about the termination instruction, unless you specify also an address.

For address, you can specify the relative hexadecimal address of an instruction within your program.

PSW
Specifies that the termination program status word is to be displayed.

Examples

1. No operand is specified in the STATUS command. VSE/ICCF displays the actual and relative addresses of all key program control factors together with information concerning the program termination status.

```
K404D  ENTER DUMP COMMAND
*ENTER DATA?
status
K401I     LOAD HI-PTN HT-PHS ORIGIN  SCAN*    ***STATUS***
ACTUAL: 0836E8 0872BB 0839B8 0836E8 0836F0
RLATIV: 000000 003BD3 0002D0 000000 000008
 ...
```

2. The INSTR operand is specified. This causes the dump program to decode the termination instruction VSE/ICCF displays:

   The actual and relative data locations
   The lengths of the related data fields
   The contents of the data fields.

   If also an address is specified, VSE/ICCF will decode the instruction at that address location.

```
K404D  ENTER DUMP COMMAND
*ENTER DATA?
st instr
K443I  INSTR=FA2150A950AC
K445I  OPRND2 ACT=083796 REL=0000AE LEN=002 CONTENTS=
083796 0000AE 0000AE   010C0000341C ...
K445I  OPRND1 ACT=083793 REL=0000AB LEN=003 CONTENTS=
083793 0000AB 0000AB   00370C010C00 ...
```

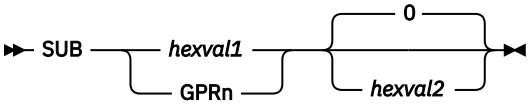
3. The PSW operand is specified. This causes VSE/ICCF to display the PSW of the terminated program. The rightmost six hexadecimal digits of the PSW contain the instruction address at which the program terminated.

```
K404D  ENTER DUMP COMMAND
*ENTER DATA?
status psw
PSW=071D0007D008375E
```

```
ACTUAL=083758 REL=000070
INSTR=FA2150A950AC
```

SUB Command

The SUB command is used to subtract one hexadecimal value from another or from the contents of a general purpose register, and to display the result.



hexval1
Is a 1- to 8-digit hexadecimal number.

hexval2
Is a 1- to 8-digit hexadecimal number.

GPRn
Is the designation of a general purpose register where n can be any of the characters 0 through 9 and A through F.

The second operand is subtracted from the first operand and the hexadecimal result is displayed. If the hexadecimal result is an address within your program, its value relative to the start of the program (or your ORIGIN value) is also displayed.

Example

```
K404D  ENTER DUMP COMMAND
*ENTER DATA?
sub 234C 82A
K439I  HEX VALUE IS 00001B22
K404D  ENTER DUMP COMMAND
*ENTER DATA?
sub gpr5 800
K439I  HEX VALUE IS 000BA64C REL=0035D4
```

TOP Command

The TOP command sets the scan/locate pointer to the first position in the current reference area.

If no ORIGIN command has been issued, the current reference area begins at the original load point of the program being tested.

The TOP command can be used before a LOCATE command to ensure that the search begins at the start of the reference area.

Example

```
K404D  ENTER DUMP COMMAND
*ENTER DATA?
top
K404D  ENTER DUMP COMMAND
*ENTER DATA?
loc 'findit'
K423I  CANNOT LOCATE DATA TOP FORCED
```

Chapter 9. Writing a Procedure or a Macro

This section discusses

- The general characteristics of procedures and macros.
- A summary of the IBM-supplied procedures and macros.
- Rules for calling procedures and macros.
- Rules for writing your own procedures and macros.

General Characteristics

Procedures and macros are VSE/ICCF library members that contain any combination of the following: VSE/ICCF system and editing commands, job entry statements, and data. The order of these commands (statements, data) in a member is the same as if they had been entered directly from the terminal. They perform frequently used functions, such as compilations, loading and running programs, and sorting library members.

Procedures and macros save you work. Once a procedure or macro is tested and stored in your library, it can be run at any time simply by typing its name and pressing ENTER. You thus do not have to reenter the statements and commands each time you need a certain function.

The main difference between procedures and macros is this:

- Macros are run like normal commands, they can be invoked in edit mode. They are processed faster than a procedure, but have a limited logic capability.

  You can automate editing tasks by using macros. For example, you can use:

  – Edit macros in full-screen edit mode.

    Advantage: a macro executes fast. An edit macro may contain full-screen editor commands.

    Limitations: return messages from edit commands cannot be checked; variable user input (for substitution in edit commands or to control conditional skipping of statements) is possible only through the passing of parameters.

  – Macros calling the context editor in command mode.

    Advantage: a macro executes fast; checking of return messages from edit commands is possible.

    Limitations: variable input (for substitution in edit commands or to control conditional skipping of statements) is possible only through the passing of parameters.

- Procedures are run in command mode; they require the procedure processor to be started in an interactive partition. They have more control over the flow and execution of its commands than a macro.

  You can automate editing tasks by using procedures. For example, you can use procedures calling the context editor in command mode.

  Advantage: system variables and internal variables can be used; via a dialog with the user, the procedure can request variable data for control or substitution.
  Limitation: the procedure needs an interactive partition, and none might be available for some time.

Macros and procedures can be stored permanently as library members. Macros (but not procedures) can be built in the stack area. This is quite useful while the terminal is in edit mode.

Certain macros and procedures are supplied by IBM as part of VSE/ICCF, but you can also write your own.

For a discussion of how to write your own procedures and macros, refer to the sections “Writing a Procedure” on page 275 and “Writing a Macro” on page 290.

Most of the IBM-supplied procedures and macros are described in Chapter 3, “System Commands, Procedures, and Macros,” on page 31. You find them there in alphabetical order, together with the system

commands. Editor macros are described in Chapter 4, “General Information about the Editor,” on page 133.

Your location may not have installed all of these procedures and macros. To find out whether a certain macro or procedure exists, request it to be listed. For example, to check for the STORE macro, enter the command:

```
/LIST STORE
```

Rules for Calling Procedures and Macros

This section describes some general rules for calling procedures and macros.

Implied Execute Function for Procedures

To run procedures by simply typing the procedure name and pressing ENTER, the implied execute (IMPEX) function must be turned on. For details see the /SET IMPEX system command in “To Set VSE/ICCF Features” on page 105. If this function is turned off, the /EXEC CLIST command must be used. For example, instead of typing:

```
ASSEMBLE PROGA
```

you would have to type:

```
/EXEC ASSEMBLE CLIST PROGA
```

if the implied execute feature had been set off.

Rerunning a Procedure

When a procedure is invoked, the VSE/ICCF job stream to be run is built in the input area. Thus, as long as the options do not change, the procedure can be run whenever it is needed simply by entering /RUN (or $). Also, if the job stream thus created is to be used again, the input area can be saved (/SAVE command) and rerun by entering /EXEC followed by the name of the library member where the job stream is saved. Rerunning a procedure in this way is more efficient than reentering the procedure name and operands since the overhead of the procedure processor execution need not be repeated.

Macros in Edit or Command Mode

Macros issued in edit mode must be prefixed with '@'; in command mode, this prefix can be omitted.

Input Area

All procedures and some macros cause the contents of the input area to be lost. Thus, if the input area is to be retained, save its contents before you invoke the procedure or macro.

Writing Your Own Procedures and Macros

This section gives some general rules at first and then goes into more detailed discussions under “Writing a Procedure” on page 275 and “Writing a Macro” on page 290

Spacing and Delimiters

The spacing of operands and the use of delimiters between operands is the same as for normal system commands with the following exceptions:

1. Operands in macros cannot use parentheses as delimiters.

2. Operands in procedures can be bounded with apostrophes if the operand itself contains delimiter characters.

Passwords

Not all of the IBM supplied procedures are set up to handle member passwords. If the members you are dealing with are password-protected, you must enter the commands and job entry statements individually or by building your own alternate procedures.

Operand Limitations

Procedures may have open-ended parameter lists ending with one or more options. Note that the options specified must not cause the total number of operands to exceed 9. If the number of operands would otherwise exceed 9, two or more of the options can be specified as a single operand by grouping them together within apostrophes. Thus, 'NODECK LIST XREF' is considered to be one operand.

Writing a Procedure

Writing a procedure yourself means creating a VSE/ICCF library member that can be run whenever you need an often-used function. The file can contain VSE/ICCF system and editing commands, job entry statements, procedure processor orders and/or data. The procedure orders must not have sequence numbers in columns 73-80.

A Simple Procedure

For example, to write a procedure which would save punch output from a previous job into the library under the name JCPUNCH, you might write the following procedure:

```
/INPUT
/INSERT $$PUNCH
/END
/PURGE JCPUNCH
/SAVE JCPUNCH
```

If you entered these statements as input (in edit mode) and saved them in your library as a member named SAVPUN, you could have processed them as a procedure – assuming that the implied-execute feature is set on (see “To Set VSE/ICCF Features” on page 105) – simply by entering:

```
SAVPUN
```

At this point the procedure processor (DTSPROCS) would be called and the commands in the procedure would be executed one by one until the last line was processed, just as if the commands had been entered from the terminal.

Variable Parameters in Procedures

A procedure as described above is useful. However, it would be difficult to build truly generalized procedures without supplying some type of data to the procedure. In the above example, the punch output can be saved in the library only by the name JCPUNCH. The procedure would be more useful if you could specify the name of the member in which to save punch output at the time the procedure is invoked. You can achieve this by specifying a variable parameter name (for example, &&PARAM1) instead of JCPUNCH as shown below:

```
/INPUT
/INSERT $$PUNCH
/END
/PURGE &&PARAM1
/SAVE &&PARAM1
```

If this procedure were run, a name must be specified to replace the variable parameter (&&PARAM1) when it is encountered in the procedure. Thus, the procedure would have to be invoked as shown by the example below:

```
SAVPUN OBJMOD
```

Then, when the procedure processor encounters the /PURGE and /SAVE commands, the value OBJMOD will replace the parameter &&PARAM1, and the punch output will be saved in the library as a member named OBJMOD.

Up to 18 variables (&&PARAM1 through &&PARAM9 and &&PARAMA through &PARAMI) may be used in this way to supply variable information to a procedure. (Even more than 18 variables can be accessed via the &&SHIFT order.)

Logic in Procedures

Occasionally, it may not be enough to simply replace variables in a procedure. You may want to examine the variables themselves, so that you can alter the flow of commands as they are processed within the procedure.

Suppose in the preceding example you wanted to save the punch output in the library by the name specified as above but, if no name was specified, to have the punch output saved as the member named JCPUNCH. In the above procedure, if the member-name operand were omitted, the parameter &&PARAM1 would be replaced by blanks and the /PURGE and /SAVE commands would be invalid. To avoid this, the procedure is modified as shown:

```
/INPUT
/INSERT $$PUNCH
/END
&&IF &&PARAM1 EQ ' ' &&GOTO AAAA
/PURGE &&PARAM1
/SAVE &&PARAM1
&&EXIT
&&LABEL AAAA
/PURGE JCPUNCH
/SAVE  JCPUNCH
```

In the example, the &&IF order is used to find out whether a name for &&PARAM1 has been omitted. VSE/ICCF, in fact, checks to see if the member-name operand is blank. If so, a branch (&&GOTO order) is taken to the $$LABEL order that marks the label AAAA. At this point, VSE/ICCF finds and executes the commands for saving the output as a member named JCPUNCH.

The Procedure Processor

This section discusses the various types of procedures, the procedure processor (program DTSPROCS) itself, and the facilities offered by this program through procedures.

The Simple Procedure

A procedure is a library member containing a sequence of some or all of the following: system commands, edit commands, job entry statements, procedure-processor orders, data. A procedure, when processed, performs specific functions; it is processed in an interactive partition by the procedure processor (utility program DTSPROCS).

The /LOAD for DTSPROCS can be the first statement of a procedure, in which case the procedure can be invoked by entering a simple /EXEC command (for example /EXEC PRTPCH). If the /LOAD DTSPROCS is not the first statement in a procedure, this procedure is to be called in either of the following ways:

- By using the CLIST operand in the /EXEC command — For example:

  ```
  /EXEC PRTPCH CLIST
  ```

- By setting on the implied-execute function (see the /SET IMPEX system command in “To Set VSE/ICCF Features” on page 105). If you do this, you can call a procedure by simply typing its name and pressing ENTER. For example:

  ```
  PRTPCH
  ```

The Variable Procedure

A variable procedure is similar to a simple procedure. However, certain command operands are represented by variable symbols in the form &&PARAMn, where n is a digit from 1 to 9 or a letter from A to I. The variable procedure can also include procedure orders, which are described under "Procedure-Processor Orders" on page 282. A procedure order is interpreted by the procedure processor to perform a logical function or to alter the flow of the procedure.

Unless the implied-execute feature is on, a variable procedure must be invoked with the CLIST form of the /EXEC. The actual values to be substituted for the variable symbols are coded as operands on the /EXEC command. For example:

```
/EXEC LOAD CLIST FORTPROG DATA FORTDATA
```

If the implied execute function is on, it is sufficient to enter:

```
LOAD FORTPROG DATA FORTDATA
```

The Procedure Processor (DTSPROCS)

The DTSPROCS program reads the system and edit commands as if they were data and passes them to the proper processing module just as if they had been entered from the terminal. The DTSPROCS program also monitors the execution of /RUN and /EXEC requests, which themselves are part of the procedure.

The ability to issue /RUN or /EXEC requests within a procedure is a standard facility of DTSPROCS. However, when more than one execution is requested in a procedure, two interactive partitions are required to process the request: one for DTSPROCS itself and one for the executions requested by the /RUN or /EXEC commands. The CLIST form of the /EXEC command cannot be used in a procedure. That is, a procedure cannot invoke another procedure.

The DTSPROCS program scans the procedure for parameters (&&PARAMn) and for procedure orders (for example &&IF). When a parameter is encountered in the procedure, the actual value from the invoking /EXEC command is substituted. If no parameter values are supplied in the /EXEC command, the parameters are set to blanks.

DTSPROCS can also be used to transfer 80-character records from the library into the punch area. If the UPSI-0 switch is set on, the printed data will be transferred also to the punch area if a /DISPLAY, /LIST or /LIBRARY command occurs in the procedure. This facility is useful if, for example, you want to process the output of the /LIBRARY command.

If the UPSI-1 switch is set on, the punch data will not be displayed.

If the UPSI-2 switch is set on, the normal display output from the command processor will not be displayed.

DTSPROCS is, to some extent, a typewriter-terminal user of its own. It has its own control blocks and its own auxiliary areas (input, print, punch, stack, log area). Only the contents of the print area are transferred to your print area automatically. For the contents of the other areas you have to take care in the procedure. For example, if you started logging in a procedure, you would have to list the log area in the same procedure because, at procedure end, the log area would be lost.

Note:

1. The input to DTSPROCS consists of 80-character records. It is best to avoid sequence numbers or flagging information in columns 72 through 80 because they may be interpreted as data. Characters in column 72 are not interpreted as continuation characters.
2. DTSPROCS always assumes an IBM 2741 terminal. Commands that require a 3270 terminal, such as /HARDCPY, cannot be issued from a procedure. This explains why a procedure cannot get control in list mode. All listing operations proceed without pause to their end. Thus, you cannot give system commands valid in list mode within procedures: for example /COMPRESS, /CONTINU, /SKIP.
3. An input area created by a procedure (for example by the /INPUT command) is lost at the end of the procedure. To save the area's contents include a /SAVE command in the procedure. An exception

to this rule is when a /RUN or /EXEC is followed by a /PEND control statement. Then the input area remains available to the job as if it had been built via terminal input.

4. If you password protect a procedure you rule out implied execution. The password would be taken as the first parameter value.
5. PF keys cannot be set or shown from a procedure.

The CLIST Operand of the /EXEC Command

The /EXEC command in the form:

```
/EXEC listname CLIST param1 ...
```

or (if the implied-execute feature is on)

```
listname  param1 ...
```

places the following statements into the input area:

```
/LOAD DTSPROCS
/PARM param1 ...
/INCLUDE listname
```

Then the job is run. Thus, the CLIST option of the /EXEC command is nothing more than an automatic way of invoking DTSPROCS. The /PARM control statement is generated only if parameters are specified following the CLIST.

The /PARM Control Statement

The statement is used to specify the actual values to be substituted for any variable parameters encountered within the 80-character records by DTSPROCS. The format of the statement is:

```
/PARM param1 param2 ...
```

Normally, there is no need for you to code a /PARM statement because it is generated automatically by the /EXEC or an implied /EXEC with one or more specified parameters. However, for reasons of your own you might want to invoke DTSPROCS explicitly and thus code your own /PARM statement.

The values specified in the /PARM statement can be separated by blanks or commas. Each value can consist of up to 30 characters. If a parameter contains delimiter characters such as commas or blanks, it may be enclosed in quotes. The first value (param1 above) will replace the variable parameter &&PARAM1, the second value will replace &&PARAM2, and so on. If you want a parameter to be omitted, indicate the omission by specifying a single ampersand (&).

The total length of all values and delimiters may not exceed 72 characters, and the number of values may not exceed 20.

The /PEND Control Statement

If you include in your procedure a /PEND statement prior to the last or only /RUN or /EXEC statement, there is no need for two interactive partitions for the execution of the final (or only) step.

When the PEND condition is on, all records following a /RUN or /EXEC are flushed (PEND conditions: either a preceding /PEND statement was present or the MULTEX option is off).

Ampersand (&) Coded Job Entry Statements

When the background reader encounters a job entry statement such as /LOAD or /INCLUDE, the statement is processed at once. That is, the /LOAD will mark the end of one step and the start of the next, or an /INCLUDE referenced member will be logically included at the point where the /INCLUDE occurs.

However, it may be desirable to build a job stream within a procedure. Thus, we would not want an /INCLUDE or /LOAD processed at the point it is encountered as data but, rather, later as a job entry

statement in the job stream being built within the procedure. To provide this capability, job entry statements to be read as data by DTSPROCS can be specified starting in column 2 and with an ampersand in column 1. Thus, /LOAD becomes &/LOAD; In fact, any statement encountered by the procedure processor with &/ in columns 1 and 2 will be shifted left 1 column.

Note: Since /* or /& terminates a procedure, use &/* or &/&; instead.

Procedure Processor Variables

The variables recognized by the procedure processor fall into three major categories:

Variable parameters (&&PARAMn)
Internal variables (&&VARBLn and &&COUNTn)
Special variables (like &&DEVTYP)

Any variable's value can be tested by specifying the variable as the first operand of an &&IF order. In addition, all variables except &&CURLN can be specified in any command, data, or order input to the procedure processor. The value assigned to the variable will replace the variable name before the statement is processed by the procedure processor.

Note: Variables are not substituted in the job entry statements /DATA, /INCLUDE, and /LOAD.

A variable can be concatenated to another variable or to a character string simply by specifying the variables adjacent to each other or to a character string with no intervening blank.

The following paragraphs describe the variables which you can reference in your procedures:

- Variable parameters:

```
&&PARAMn (where n is 1 through 9 and A through I)
```

  The &&PARAMn variables are set in a /PARM statement from values which were specified when the procedure was invoked. For example, if /EXEC PROCA CLIST 25 67 were entered, the value of &&PARAM1 would be 25 and the value of &&PARAM2 would be 67. The specified value may not be longer than 30 characters. The data you enter will be translated to uppercase. Via the &&READ order, a &&PARAMn variables can be reset to a new value while the procedure is running.

- Internal Variables:

```
&&VARBLn (where n is 0 - 9 and A - I)
```

  These internal, alphameric variables for temporary storage of intermediate values can be set by using the &&SET order.

```
&&COUNTn (where n is 0 - 9 and A - I)
```

  These internal, numeric variables for temporary storage of intermediate values can be set, incremented, or decremented by using the &&SET order.

- Special Variables:

```
&&CURLIB
&&CONLIB
&&COMLIB
```

  These variables return the numbers of the current (primary), the connected and the common library, respectively. If the library is not assigned, a value of zero will be returned.

```
&&CURDAT
```

This variable contains the current date. The format of the date is the same as defined in the job control //DATE statement.

```
&&CURDTX
```

This variable is the same as &&CURDAT but with a 4 digit year.

```
&&CURTIM
```

This variable contains the time of day in the form hhmmss (hh = hour; mm = minute; ss = second).

This variable can be used only as the first operand of an

```
&&CURLN(mm,nn)
```

&&IF order. Its use implies that the current line of the editor is to be tested for a character string whose starting position on the line is indicated by mm (any value from 01 to 80) and whose length is indicated by nn (any value from 01 to 32). If mm is set to 00, every column on the line is tested for the string. This variable allows you to build editor procedures which apply logic to the data being edited.

```
&&DEVTYP
```

This variable is set to reflect from where the procedure is being run. Thus, the procedure can interrogate the terminal device type if the procedure logic depends on the terminal type. The variable is set to:

00
If the procedure is run from a sequential device acting as a terminal.

40
If the procedure is run from any other typewriter terminal

41
If the procedure is run from an IBM 2741 (or 3767 in 2741 mode)

70
If the procedure is run from an IBM 3270

```
&&EXRC
```

This variable contains the job execution return code as set by the program that was called (as opposed to &&RETCOD which is the VSE/ICCF return code). Compilers, utilities, and also some user programs return a job execution return code.

The return code is a decimal number between 0 and 4095. If the program was called from a procedure (or a macro), this return code is passed to the procedure processors (macro) as a 4-byte character string. You as the terminal user might get, for example, the following completion message

```
***** JOB TERMINATED - ICCF RC 00, EXEC RC 0004, NORMAL EOJ
```

where:

ICCF RC
Is the VSE/ICCF return code as documented in z/VSE Messages and Codes 2

EXEC RC
Is the job execution return code as set and documented by the program that was called.

```
&&LINENO
```

The value of this variable is set to the number of the current line being processed within the procedure.

```
&&PARMCT
```

The value of this variable is set to the number of operands specified when the procedure was invoked.

```
&&RDDATA
```

This variable contains the first word of the line which was entered in response to a &&READ order with no operands.

```
&&RETCOD
```

This variable contains the VSE/ICCF return code. The contents of the variable is one of the following:

- If you operate in edit mode – the first word of the response message issued by the execution of the preceding command.
- For normal system commands – Either *READY or the first word of the last * response message before the *READY or *END PRINT message.

The return code after an execution (/EXEC or /RUN) requested within a procedure will be *EOJ-xx, where xx is the VSE/ICCF return code or 00 (normal end of job). In any case, the &&RETCOD value does not exceed 8 characters.

The &&RETCOD variable can be tested to determine if a command was executed successfully.

Example:

```
/switch 12
*SWITCHED
*READY
```

If you want to test for successful switching, check the character string '*SWITCHE' in &&RETCOD

```
&&TERMFT
```

This variable contains the string 'DBCS' or 'KATAKANA', if the procedure has been invoked from an IBM 5550 with 3270 Emulation. It can only be used as the first operand of an &&IF order.

```
&&TERMID
```

This variable contains the identification of the terminal from which the procedure is being run.

```
&&USERID
```

This variable contains the identifier of the user who invoked the procedure.

Procedure-Processor Orders

When you write procedures that perform logical operations, you can use orders which are recognized only by the procedure processor. For example, with the &&IF order you can examine operands that you have specified and then alter the flow of control in the procedure based on the outcome of the test.

The available procedure-processor orders are summarized below; they are described following this summary.

Order	Function
*	Is used to place comments within the procedure.
&&EXIT	Causes an exit from the executing procedure.
&&GOTO	Causes the transfer of control to a specific location in the procedure.
&&IF	Is used to conditionally execute statements within the procedure.
&&LABEL	Is used to identify branch points within a procedure.
&&MAXLOOP	Is used to alter the maximum loop value that determines when the procedure will be canceled.
&&NOP	Causes all variable parameter substitution and procedure order checking to be bypassed.
&&OPTIONS	Sets certain options that control the execution of the procedure processor.
&&PUNCH	Causes any operand data in the order to be placed in the punch area.
&&READ	Reads a single line of data from the terminal.
&&SET	Is used to alter the contents of the internal variables &&VARBLn and &&COUNTn
&&SHIFT	Shifts parameters to the left to make variable parameter processing easier.
&&TYPE	Enables you to display data.

* Order

The * order is used for comments within the procedure.

When an * order is encountered in command mode it will not be treated as a VSE/ICCF command but as a comment to be ignored, although it will not be ignored for input, edit and execution modes.

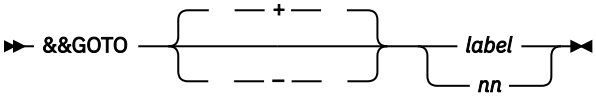
&&EXIT Order

The &&EXIT order causes an exit from the executing procedure.

The &&EXIT order can be used when you want to end processing in a procedure before actually reading through to the end of the procedure. For example, you might use the &&EXIT order within a loop when the condition for exiting from the loop is recognized.

&&GOTO Order

The &&GOTO order causes the transfer of control to a specific location in the procedure.



+
–

Specify &plus if the &&LABEL point to be branched to follows the &&GOTO that references the point.

Specify – if the &&LABEL point to be branched to precedes the &&GOTO that references the point.

label
Specifies the name on a &&LABEL order elsewhere in the procedure.

nn
Specifies a number from 1 to 255, indicating the number of statements to be skipped. For example:

- &&GOTO –1 would cause the preceding statement to be repeated.
- &&GOTO +2 would cause the next statement to be bypassed.

A &&GOTO order must reference a statement in the same member. It may not branch over an /INCLUDE statement.

A &&GOTO order cannot be the last statement in a procedure. This restriction can be overcome by a dummy &&LABEL order following the &&GOTO order at the end of the procedure.

&&IF Order

The &&IF order is used to conditionally execute statements within the procedure.

variable
Can be any of the variables described under “Procedure Processor Variables” on page 279.

operator
Can be any of the following acronyms or symbols:

Acronym	Symbol	Meaning
EQ	=	Equal to
NE	¬=	Not equal to
LT	<	Less than
GT	>	Greater than
LE	<=	Less than or equal to
GE	>=	Greater than or equal to

Specify the operator that describes the type of condition to be tested.

data
Can be a variable (&&PARAMn, &&VARBLn, &&COUNTn, &&CURLIB, &&CONLIB, &&COMLIB, &&RDDATA, &&RETCOD, &&LINENO, or &&PARMCT), a numeric literal (for example, 123 or –14), or an alphameric literal. If blanks are in the string the literal must be within quotes (for example READY or `'END OF LINE'`).

then-clause
Can be a system or editing command, a line of data, another order or any other line or command which would be processible in the procedure at the point where the &&IF order occurs.

The &&IF order tells the procedure processor to determine whether the expressed condition is true or false. If the expressed condition is false, no further processing of the &&IF order takes place and the next sequential line in the procedure is processed.

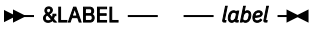
If the condition is true, the then-clause is shifted into column 1 and reevaluated as a command, data line or another order.

Compound conditions can be built on the same statement by coding multiple &&IF orders on the same line.

The comparison between the first operand and the third is alphameric, unless the first operand is specified as a numeric variable such as &&COUNTn, &&PARMCT or &&LINENO. In this case, the comparison is algebraic. In alphameric comparisons the shorter operand is right padded with blanks to the length of the longer operand before the comparison takes place.

&&LABEL Order

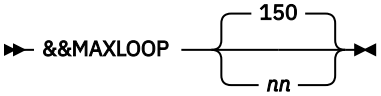
The &&LABEL order is used to identify branch points within a procedure.



label
Must begin with an alphabetic character and must be from 1 to 8 characters long. Any branch-to point specified as a label on a &&GOTO order must be identified by a &&LABEL order.

&&MAXLOOP Order

The &&MAXLOOP order is used to alter the maximum permissible loop value before the procedure is automatically canceled.



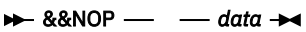
nn
Is a decimal number which can be any size you wish. However, you are recommended not to exceed a value of 4096 (4K).

The &&MAXLOOP order can appear only before the first &&LABEL order within a procedure.

Each time a &&GOTO order is encountered within a procedure, a counter is incremented by 1. When this counter reaches the maximum loop value the procedure will be terminated.

&&NOP Order

The &&NOP order causes all variable parameter substitution and procedure order checking to be bypassed for the entire statement.

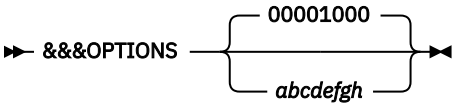


data
Specifies that the data beginning in column 7 will be shifted into column 1 and the resulting statement will be passed to the command processor.

If you want to pass data to the command processor but this data began, for example, with &&IF or some other characters which the procedure processor would recognize as an order, the data should be coded beginning in column 7 of a &&NOP order.

&&OPTIONS Order

The &&OPTIONS order sets options that control the execution of the procedure processor.



The characters 'a' through 'g' refer to options that can be set on by specifying 1 or off by specifying 0. An 'x' leaves the option unchanged.

The options have the following meanings:

a
If set on, this option causes display output from list type commands such as /LIST, /DISPLAY, /LIB or PRINT (editor) to be placed into the punch area rather than displayed. This option is also set by the UPSI 10000000 option.

b
If option a is set on, the punch data will normally be displayed as well as punched. However, if this option is set off, the data will not be displayed. This option is also set by the UPSI 01000000 option.

c
If this option is set on, the normal command processor display output will not be displayed. Only critical procedure error messages and &&TYPE order requests will be displayed; all other display output will be bypassed. This option is also set by the UPSI 00100000 option.

d
If this option is set on, no shifting of data takes place. For example: if the three-character value *** replaces the eight-character variable &&PARAM1 in the string 123&&PARAM1456, the resulting string is 123***bbbbb456; if this option is off, the resulting string is 123***456.

e
This option (which is called the MULTEX option) will initially be set to '1' or on. When the option is off, any /RUN or /EXEC within the procedure will signal the end of the procedure. The procedure processor will be terminated and the execution request will be processed in the same interactive partition. When this option is on, the procedure processor will stay in the interactive partition (to process any possible commands following the /RUN or /EXEC) and the execution request will be scheduled in a second interactive partition. However, a prior /PEND control statement has the same effect as if this option had been set off.

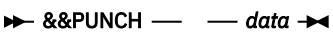
f
When this option is off, user control options such as tabs, control characters, etc., will be set and maintained just as if the commands were entered from the terminal. If this option is on when the procedure terminates, these controls are no longer in effect. They are set for and used only in the procedure.

g
When set off, this option causes the procedure processor to halt immediately prior to any /RUN or /EXEC request encountered in the interactive partition; any output in the print area will be forced out to the terminal. When this option is on, the procedure processor will not halt and output to the print area will not be forced to the terminal.

h
Not used.

&&PUNCH Order

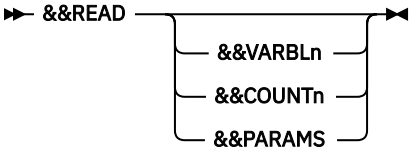
The &&PUNCH order causes any operand data on the order to be placed in the punch area.



data
Specifies data that is to be shifted to column 1 of an 80-character record. This data can include variable parameters, in which case normal variable parameter substitution will occur.

&&READ Order

The &&READ order causes a single line of data to be read from your terminal. Any lowercase characters included in this line of data are translated to uppercase.



&&VARBLn
Specifies that the data read will be placed into the internal variable. The data entered must not exceed eight characters.

&&COUNTn
Specifies that the data read will be placed into the internal variable. This data must be a decimal value and can be preceded by a plus or minus sign.

&&PARAMS
Specifies that a new set of parameters will be read in to replace &&PARAM1 through &&PARAM9 and &&PARAMA through &&PARAMI The old parameters are first cleared, then the new ones are read in. You indicate the omission of a parameter by specifying a single ampersand (&).

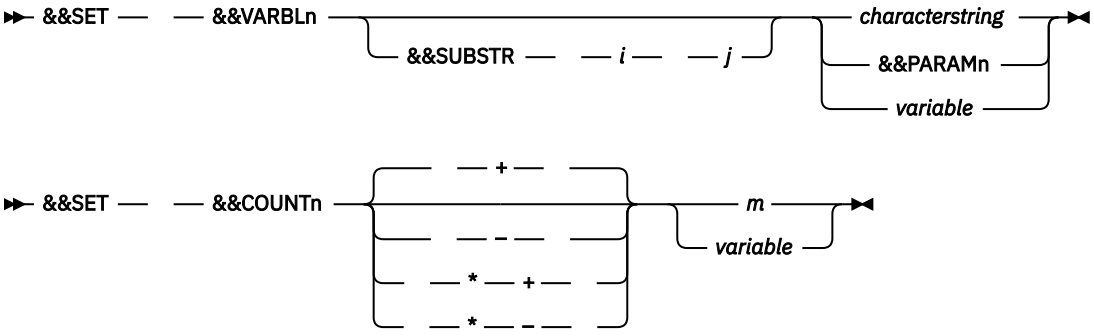
If you do not specify an operand, the procedure processor reads and processes the entire line just as if the command or data line had been in the procedure at the point where the &&READ occurred.

Following the &&READ, the first word read can be tested using the &&RDDATA variable.

Note: If a valid command is entered in response to ENTER DATA, the command is processed by the foreground command processor. The procedure processor does not get control. When command execution is finished, 'ENTER DATA' will be displayed again.

&&SET Order

The &&SET order is used to alter the contents of the internal variables &&VARBLn and &&COUNTn It can be used to increment or decrement the &&COUNTn variable from its current value. The &&SET order has two distinct formats as shown below.



n
Can be a value as follows:

From 0 through 9 or A through I for &&VARBLn or &&COUNTn
From 1 through 9 or A through I for &&PARAMn

&&VARBLn
When setting one of the &&VARBLn internal variables, you can specify a character string of up to eight characters. If the character string includes delimiter characters, it must be specified between single quotation marks.

You may set the internal variable to a parameter (&&PARAMn). If &&PARAMn consists of more than 8 characters, only the first 8 characters will be used to set the variable. &&VARBLn can also be set to another internal variable (&&VARBLn or &&COUNTn) or any other special variable except for &&CURLN (e.g., &&RETCOD or &&LINENO).

&&SUBSTR i j
The &&SUBSTR order can immediately follow the first operand to set an internal variable (&&VARBLn) to a substring of the character string (or parameter or any variable, except &&CURLN) specified as the last operand.

For i, specify the starting location within the character string; it may be any number from 1 to 32.
For j, specify the number of characters to be set; it may be any number from 1 to 8.

Thus, if &&PARAM1 contains ABCDEF, the order

```
&&SET &&VARBL3 &&SUBSTR 2 3 &&PARAM1
```

causes the variable &&VARBL3 to be set to BCD.

&&COUNTn +m
&&COUNTn –m
&&COUNTn *+m
&&COUNTn *–m
&&COUNTn variable

When setting a numeric internal variable (&&COUNTn), you can specify:

- A plus or minus decimal integer (+m or -m) to cause the variable to be set to a given value.
- *+m increments the value of the variable by m.
- *-m decrements the value of the variable by m.
- Any of the other variables (&&PARAMn, &&VARBLn, &&COUNTn, &&LINENO, and so on) as long as the specified variable has a numeric value.

Note: Only character strings shorter than eight characters can be enclosed in apostrophes. Longer strings may not contain apostrophes, blanks, commas, asterisks or brackets.

&&SHIFT Order

The &&SHIFT order causes parameters to be shifted one position to the left as a means of making variable parameter processing easier.

n

Is a number from 1 to 9 indicating the parameter at which point the shift is to begin.

Given the parameter list A B C D E F, a procedure could reference these internally as &&PARAM1, &&PARAM2, ... &&PARAM6 If &&SHIFT 3 is specified, the parameter list then is A B D E F. Every parameter beyond &&PARAM3 is shifted to the left one logical location.

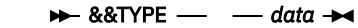
A &&SHIFT order can be useful in procedure processing where a variable number of keywords can be coded as parameters in any order. If the first non-fixed parameter were &&PARAM3, you could check &&PARAM3 for each of the variable keywords. Then when one was found, you could process it or save the fact that it was entered in a variable (&&VARBLn) and then say &&SHIFT 3 and repeat your test loop. The parameters that had been &&PARAM4, 5, 6, etc., will now be &&PARAM3, 4, 5, etc., so &&PARAM3 can be tested again and again until there are no more parameters (&&PARAM3 equals to a blank).

Up to 19 parameters can be made available in this way.

The &&SHIFT order reduces the value of the &&PARMCT variable by one.

&&TYPE Order

The &&TYPE order enables you to display data.



data

Can consist of any (alphanumeric or double-byte) data characters including variables which will be replaced by their current values before the display takes place.

Even if the no print option is in effect (see &&OPTIONS order) a &&TYPE order will cause data to be displayed.

Examples of Procedures

The examples include short explanations within parentheses.

Example 1 – Assemble a Program

The following procedure will assemble a source module and, optionally, save an object module in a library member rather than in the punch area. You can also type in additional operands which are treated as assembler or VSE/ICCF options.

```
* ************************************************************
* PROCEDURE TO ASSEMBLE A PROGRAM
*
* 1ST PARM IS THE 'NAME-OF-MEMBER' CONTAINING THE SOURCE
*
* 2ND PARM IS OPTIONAL KEYWORD 'OBJ' FOLLOWED BY 'NAME-
*  OF-MEMBER' INDICATING THE VSE/ICCF MEMBER TO CONTAIN
*  THE OBJECT DECK -- IF NOT ON THE FILE IT WILL BE
*  CREATED
*
* ADDITIONAL PARMS ARE TREATED AS OPTIONS AND ARE PLACED
*  ON THE /OPTION STATEMENT
* ************************************************************
&&IF &&PARAM2 NE OBJ &&GOTO INLIB
/LIST 1 1 &&PARAM3       (See if object member in library)
&&IF &&RETCOD EQ *READY &&GOTO INLIB   (If a normal return, it
/INP                                    is in the library)
DUMMY
/SAVE &&PARAM3   (build dummy member for object module)
&&LABEL INLIB
/INP
&/LOAD ASSEMBLY
&&IF &&PARAM2 EQ OBJ &&GOTO +OBJ1  (Test if object deck to
                                   library member)
/OPTION &&PARAM2 &&PARAM3 &&PARAM4 &&PARAM5 &&PARAM6 &&PARAM7
&&LABEL OBJ1
&&IF &&PARAM2 NE OBJ &&GOTO +NOOBJ1
/OPTION &&PARAM4 &&PARAM5 &&PARAM6 &&PARAM7
/FILE TYPE=ICCF,UNIT=SYSPCH,NAME=&&PARAM3  (Define member
&&LABEL NOOBJ1                              for object deck)
/FILE NAME=IJSYS01,SPACE=40,VOL=0
/FILE NAME=IJSYS02,SPACE=40,VOL=1
/FILE NAME=IJSYS03,SPACE=20,VOL=2
&/INCLUDE &&PARAM1
/END
/PEND
/RUN
```

If the above procedure is invoked by:

```
 assemble beep obj myobj list xref
```

the following lines will be generated and passed to the routines that are processing the commands (asterisk lines are responses from those routines):

```
/LIST 1 1 MYOBJ
*NOT IN LIBRARY
*END PRINT
*READY
/INP
DUMMY
/SAVE MYOBJ
*SAVED
*READY
/INP
/LOAD ASSEMBLY
/OPTION  LIST XREF
/FILE TYPE=ICCF,UNIT=SYSPCH,NAME=MYOBJ
/FILE NAME=IJSYS01,SPACE=40,VOL=0
/FILE NAME=IJSYS02,SPACE=40,VOL=1
/FILE NAME=IJSYS03,SPACE=20,VOL=2
/INCLUDE BEEP
/END
*READY
/RUN
```

Example 2 – Execute a Program

This LOAD procedure can be used to load and run object decks from the punch area or from a library member.

```
* ************************************************************
* PROCEDURE TO EXECUTE OBJECT PROGRAMS
*
* 1ST PARM IS NAME OF MEMBER CONTAINING OBJECT DECK -OR-
  '*' INDICATING DECK IS IN PUNCH AREA
*
* OPTIONAL KEYWORD 'JES' FOLLOWED BY 'NAME-OF-MEMBER'
*  JOB ENTRY STATEMENTS ARE IN 'NAME-OF-MEMBER' -OR-
*  '*' INDICATES THE STATEMENTS ARE TO COME FROM THE
*  TERMINAL
*
* OPTIONAL KEYWORD 'DATA' FOLLOWED BY 'NAME-OF-MEMBER' -OR-
*  '*' DATA IS IN 'NAME-OF-MEMBER' '*' INDICATES THE
*  'INCON' OPTION IS TO BE USED
* ************************************************************
&&SET &&VARBL0 &&PARAM1
&&IF &&PARAM2 EQ JES &&SET &&VARBL1 &&PARAM3
&&IF &&PARAM4 EQ JES &&SET &&VARBL1 &&PARAM5
&&IF &&PARAM2 EQ DATA &&SET &&VARBL2 &&PARAM3
&&IF &&PARAM4 EQ DATA &&SET &&VARBL2 &&PARAM5
&/INP;
&/LOAD LINKNGO
&&IF &&VARBL1 EQ '*' /PAUSE -- TYPE JOB ENTRY STMTS HERE
&&IF &&VARBL1 NE ' ' &&IF &&VARBL1 NE '*' /INCLUDE &&VARBL1
&&IF &&VARBL0 NE ' ' /UPSI 1
&&IF &&VARBL0 NE ' ' /INCLUDE &&VARBL0;
&&IF &&VARBL2 EQ '*' &&SET &&VARBL3 INCON
&&IF &&VARBL2 NE ' ' /DATA &&VARBL3
&&IF &&VARBL2 NE ' ' &&IF &&VARBL2 NE '*' /INCLUDE &&VARBL2
/END
/PEND
/RUN
```

If the above procedure is invoked as follows:

```
 load myobj jes myjes data mydata
```

the following lines will be generated and passed to the routines that are processing the commands.

```
/INP
/LOAD LOADER
/INCLUDE MYJES
/UPSI 1
/INCLUDE MYOBJ
/DATA
/INCLUDE MYDATA
/END
/RUN
```

Example 3 – Edit a File

The following procedure causes a file (&&PARAM1) to be scanned. Any line in the file that includes the character string R1 or R7 will be located by the procedure and displayed on the terminal. The terminal will be prompted for an editing command. At this point you can enter a command to cause the data to be changed.

```
/ED &&PARAM1
BRIEF
NEXT
&&LABEL AAAA
&&OPTIONS XX1         (Set printing off)
BRIEF
NEXT
&&IF &&RETCOD EQ INVALID &&EXIT
&&IF &&CURLN(00,2) EQ R1 &&IF &&CURLN(0,2) EQ R7 &&GOTO CCCC
&&GOTO -AAAA
&&LABEL CCCC
&&OPTIONS XX0         (Set printing on)
VERIFY
&&TYPE ENTER EDIT COMMAND
```

```
&&READ                (Read the edit command)
&&IF &&RDDATA EQ QUIT &&EXIT (Test the entered command)
&&IF &&RDDATA EQ Q &&EXIT
&&IF &&RETCOD EQ *READY &&EXIT
BRIEF
&&GOTO -AAAA
&&LABEL DUMMY
```

Writing a Macro

A macro is a library member containing VSE/ICCF commands, procedure invocations and macro orders. Macros make it easy and efficient for you to invoke often-used, complex functions. You simply issue the macro name like a command – prefixed with the commercial-at sign (@) if you are in edit mode. In command mode this prefix is not necessary, but it is still valid. To invoke a macro is allowed only on the last statement of the current macro.

A macro cannot have a sequence number or flag field in columns 73 through 80.

Variable Parameters

Macros can include variable parameters &&PARAM1 through &&PARAM9 as operands of VSE/ICCF commands. These parameters are replaced with the actual operands that you later specify when the macro is called. Each parameter value can be eight bytes long, plus the number of spaces that you place between parameters when you write the macro. Shorter parameters are padded with blanks. Variable parameters may no longer be available after commands that cause execution or list mode to be entered.

The operands in macros must not contain blanks or commas because these would be treated as delimiters. If you want to omit an operand in a macro call, you have to specify && as a place holder, unless this is the last operand. The operand(s) at the end can be omitted.

Temporary Macros

Macros can be built in the punch area or editor stack area and invoked by specifying @$$PUNCH or @$$STACK. This is useful when you are using the editor and want to repeat a series of commands. To do this:

1. Put the commands (and/or data) into the stack, using the STACK command.
2. Then, when you want to execute them, enter @$$STACK (remember that the stack area is part of the punch area).

Note: A macro built in the stack area may not include the @MACRO statement as the first statement in the macro.

Chaining Macro Calls

A macro cannot be called from another macro unless the call appears as the last statement of the calling macro. When a macro call is encountered within a macro, it is treated as if it were the last statement in the containing macro.

Abnormal Termination of a Macro

When an invalid command condition occurs (for example an ADD command without a parameter), the macro execution terminates immediately.

Macro Orders

The first statement within a macro must be

```
@MACRO
```

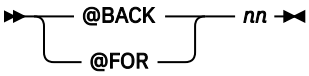
This statement and all macro orders must start in column 1.

The '@NOPRINT' macro order can be inserted in a macro to prevent the normal output from the commands in the macro from reaching the terminal. This helps eliminate unwanted intermediate terminal output. The '@PRINT' macro order reverses this effect.

A macro order must not be the last statement in a macro, otherwise it would be treated as a macro call.

@BACK and @FOR Macro Orders

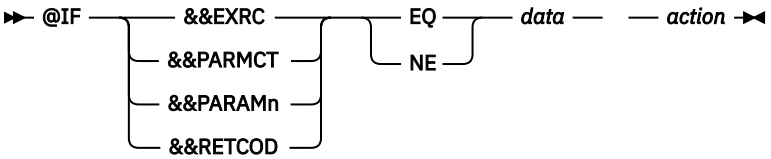
The @BACK and @FOR macro orders are used to cause a backward or forward branch within the macro to re-execute certain statements. These orders can be used with the @LIMIT or @IF orders.

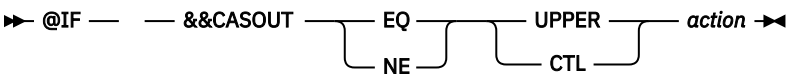


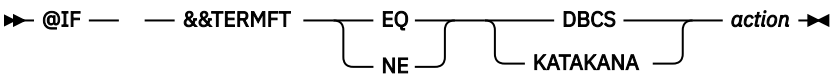
nn
Is the number of statements to branch forward or backward. It can be a value between 1 and 99999999. For nn, you can also specify a variable parameter.

@IF Macro Order

The @IF macro order is used to test whether the preceding command in your macro has been executed successfully. If the @IF condition is false, the next sequential statement in your macro is executed. If the @IF condition is true, you can either branch forward or backward a specified number of statements.







&&EXRC
Contains the return code of the last interactive-partition execution.

&&PARMCT
Allows you to check for the existence of parameters at macro invocation time. You can compare only with 0: if no parameters were specified, &&PARMCT EQ 0 holds, else &&PARMCT NE 0 is true.

&&PARAMn
Allows you to check the contents of the passed parameters &&PARAM1 through &&PARAM9

&&RETCOD
Contains the first word of the response message issued by the execution of the preceding command (for a more detailed description of &&RETCOD see “Procedure Processor Variables” on page 279).

data
Can be a variable (&&PARAMn) or a literal of up to 8 characters. If blanks are in the string, the literal must be within single quotation marks.

&&CASOUT
Allows you to check the output case settings: UPPER and CTL.

&&TERMFT
Contains the string 'DBCS' or 'KATAKANA', if the macro was invoked from an IBM 5550 with 3270 Emulation.

action
Is one of the following macro orders:

@BACK nn
@FOR nn
@EXIT
command

@EXIT Macro Order

The @EXIT macro order causes an exit from a macro before the physical end of the macro is reached. @EXIT at the physical end of a macro is ignored.

command
Can be any VSE/ICCF command, but not a macro order.

Examples

1. Exit the macro and edit member MYFILE:

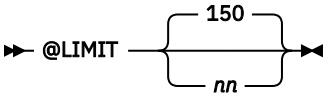
```
@EXIT /ED MYFILE
```

2. Exit both the macro and the editor if the comparison is equal:

```
@IF &&RETCOD EQ DATA @EXIT QUIT
```

@LIMIT Macro Order

The @LIMIT macro order is used to set the maximum number of statements that will be processed in the macro before it is terminated.



nn
Is the maximum number of statements that will be processed. nn can be any value from 1 to 32767. For nn, you can specify also a variable parameter.

@MACRO Macro Order

The @MACRO macro order must be the first statement within a macro.

@NOPRINT Macro Order

The @NOPRINT macro order is used to eliminate the normal output from the commands in the macro from reaching the terminal. This helps avoid unwanted intermediate terminal output.

@NOPRINT is reset by a conversational read or a full-screen read/write operation. A program or a procedure started from within a macro can issue such a terminal I/O request.

@PRINT Macro Order

The @PRINT macro order is used to route normal output from the commands in the macro to the terminal.

Examples of Macros

1. The following macro puts your terminal into edit mode and sets up all your options and defaults. You invoke this macro (called ED) by specifying ED xxxx, where xxxx is the name of the member to be edited.

```
@MACRO
@NOPR
/ED &&PARAM1
NEXT
SET LOG ON
SET HEX=<
SET TAB=;
SET END=:
SET SCR 4
TABSET 10 16 36 68
VERIFY LONG 80 15 9
LINEMODE 73,6
PROMPT 4
INDEX
SET PF1ED OC68 CLR//
SET PF2ED UP 10
SET PF10ED STACK OPEN
SET PF11ED STACK CLOSE
SET PF4ED @$$STACK
SET PF12ED P $$STACK
SET PF8ED GETFILE $$STACK
SET PF9ED P $$LOG
SET PF3ED PF 10,80
PRINT
SET SCR CLEAR
ECHO *READY TO EDIT &&PARAM1
```

2. The following example illustrates the use of a macro within a macro that will convert a library member from lower- to uppercase. This might be useful if you had accidentally entered data in lowercase mode when you had intended it to be uppercase. The macro builds a temporary macro and finally executes it.

```
@MACRO
@NOPR
/INP
/INSERT &&PARAM1
/END
/SET CASE UPPER
/ED
I @MACRO
I @NOPR
I /INP
B
I /END
I /REPL &&PARAM1
I @PRI
I /ECHO *&&PARAM1 IS NOW UPPER CASE
SAVE TMP$$$$$
@TMP$$$$$
```

Chapter 10. Utility Programs

The VSE/ICCF utility programs listed below are available to you at the terminal. You can use some of them directly; others are available via the submit-to-batch facility. The available utilities are:

- DTSAUDIT

  Displays the changes that were made to a given library member following the last system checkpoint.
- DTSBATCH

  Allows you to execute a limited number of foreground commands in a VSE partition. DTSBATCH can only be invoked via the submit-to-batch facility.
- DTSCOPY

  Copies one sequential file to another.
- DTSDUMMY

  Performs no processing function; it goes immediately to end of job. Is useful in procedures where you might want to sometimes run a program and sometimes not, depending on the setting of variables.
- DTSSORT

  Performs a fast sort on data from the input area or the punch area.
- LINKNGO

  Links programs and subprograms together to form an executable program, which is then run.
- OBJECT

  Transfers 80-character records from the job stream to the punch area. Its main purpose is to transfer object programs or LINKNGO control statements to the punch area for later loading and execution.

DTSAUDIT Utility

The utility displays added, deleted, and changed records in library members. You would use it to monitor changes to data within your library. The utility displays only those changes that have been made since the last file checkpoint – the most recent DTSUTIL backup and restore run for the VSE/ICCF library file.

The DTSAUDIT utility allows you to:

- Scan a given member, or a group of members for changes.
- List only the changes to a member, or the entire member along with its alterations (additions, deletions, changes), including where they occur within the member.
- Select one or any combination of the possible scan options for a given member. These options are:
  - A **physical scan option**, which displays all physical additions and deletions. This scan also indicates whether a member is entirely new. All physical scanning is based on a checkpoint concept. A physical scan can be made on any library member.

    This scan option does not apply to compressed library members.
  - A **logical scan option**, which indicates all logical additions, changes, and replacements to the member. The scan is based on the VSE/ICCF editor flagging option; it is meaningful only if the member being scanned has been changed with the editor change flagging option set on. In that case, each altered record is flagged to indicate the date and type of the change and the user making the change.

    The option causes a member to be scanned for the editor flag. When a flagged record is found, it appears in the report. The report indicates the date and type of change (addition, change, replacement) and the identifier of the user who made the change. The logical scan option can be used also for compressed members.

To set on the flag option, use the /PROTECT command. For a description of the command, see the section "/PROTECT Command" on page 83.

– A **sequence-number scan option**, which displays any deletions or additions to a member as indicated by changes in the normal sequence-number progression within the member. The scan is based on a fixed increment associated with the last resequencing of the member. This type of a scan is appropriate for members with sequence numbers imbedded in the records.

  The sequence-number option described applies also to compressed members.

Invoking the DTSAUDIT Utility

The basic job stream for executing DTSAUDIT in an interactive partition would be:

```
/INPUT
/LOAD DTSAUDIT
 ...
 ...            (One or more commands)
 ...
/ENDRUN
```

To type in commands interactively, begin execution by entering:

```
$DTSAUDIT CONSOLE
```

Then, when the conversational read is requested, enter the appropriate PRINT command.

Commands

The DTSAUDIT utility recognizes several commands. They all must begin in column 1 of the input control record, and they must be wholly contained within a single 80-character record. The commands can be abbreviated down to the minimum form as shown in the format descriptions. All operands must be separated from each other by commas or blanks. Multiple commas or blanks are treated as single ones. All operands can be abbreviated down to the point where they become nonunique among the set of all operands.

The commands recognized by the utility are:

- PRINT

  The command controls whether the physical, logical, or sequence-number scan option (or a combination of these options) will be performed on the requested data.

- OPTION

  The command allows the user to set several of the DTSAUDIT-program options on a global basis. That is, once an option is set via the command, it remains in effect for all later PRINT commands until reset by another OPTION statement. Thus, when an option is set this way, it need not be respecified on each later PRINT command.

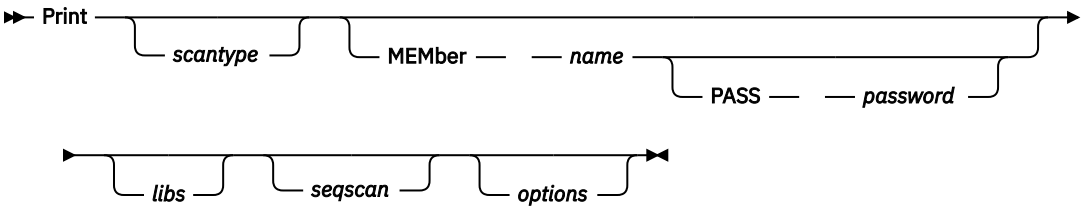
- CARD

  The command has no operands. It can be used to set the control command input device from the console to the card reader.

- END

  The command ends command input. It is used primarily when you enter commands from the console since the /* or end of file normally terminate commands entered via SYSIPT.

PRINT Command



scantype
Is the physical/logical scan type to be performed. The options are:

LOGICAL
Performs a logical scan.

PHYSICAL
Performs a physical scan.

BOTH
The default. Performs a logical and a physical scan.

NEITHER
Performs neither of the above scans.

MEMber name
MEMber name PASS password
Indicates which member within the libraries specified in the libs operand will be scanned. For name, specify the name of the member that is to be scanned.

If the utility runs in an interactive partition, this operand is the only one you can specify. Moreover, you must provide the password if you are not the VSE/ICCF administrator and the member is password-protected.

libs
May be specified as:

ALL
The default. It indicates that the member specified by the MEMBER operand is to be scanned in all libraries within the VSE/ICCF library file.

LIBRARY nnnn
It indicates that the member specified by the MEMBER operand is to be scanned only in library nnnn, where nnnn is the number of the library.

seqscan
Indicates that a sequence number scan is wanted. This option can be used by itself or with either the physical/logical or both options. It is specified as 'SEQUENCE n1 n2 n3' where the parameters represent the following values:

n1
Is the starting column number in the record where the sequence number occurs.

n2
Is the number of columns in the sequence number field.

n3
Is the increment used during the last resequence of the member.

options
May be specified as one or several of the following operands:

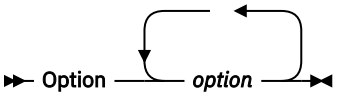
SORTED
Indicates that library directories are to be sorted by member name before processing.

EJECT
Indicates that the printer should be skipped to head of form before printing the output report for each member.

FULLPRINT
Indicates that all lines within the members being scanned should be printed rather than the default, which is printing only the added, changed, or deleted lines.

RESET
This option is effective only in a VSE batch partition and when VSE/ICCF is not running. The option resets the editor flagged records in any member being scanned using the logical scan option. Thus, the next time the DTSAUDIT program is run, the previously flagged records will no longer be indicated as changed records. RESET causes the editor flag area within a flagged record to be overlayed with four asterisks and the month and day of the RESET.

OPTION Command



option
Are options to be set for all PRINT commands. Thus, equivalent options need not be specified on each PRINT command. All OPTION command settings are reset each time a new OPTION command is read. The following options can be specified:

SORTED
Indicates that all library directories processed by the DTSAUDIT utility are to be sorted alphabetically by member name prior to processing.

EJECT
Indicates that a skip to the head of the form is to be performed prior to printing the change report for each member processed by the DTSAUDIT Utility.

NOEJECT
Indicates that a skip to head of form is not to be performed for each new PRINT command. Normally, each new PRINT command read from the command input device advances the printer to the head of the form. This option causes this skip to head of form for each PRINT command to be bypassed.

FULLPRINT
Causes both the changes and the entire member to be printed. Changed lines will be flagged to distinguish them from the unchanged lines.

RESET
Indicates that, after a logical scan, editor flagged records will have the editor flag area reset or overlayed within the records so the records do not appear as changed on later DTSAUDIT reports.

DTSAUDIT Command Examples

1. Scan all libraries for all occurrences of a member named XFER. Perform a physical, logical and sequence number scan on these members. Print all records, not just flagged records.

```
    PRINT MEM XFER SEQ 1 6 100 FULL
```

2. Print listing of the member called MYTEST from library 5. Do not perform any of the scanning options.

```
    PRINT NEITHER FULLPRINT MEM MYTEST LIB 5
```

3. Perform a logical scan on the member called VSMCOB in library 4, and print all records within the member.

```
    PRINT LOGICAL FULL MEMBER VSMCOB LIB 4
```

4. Determine in which library or libraries the member named ADRSLST occurs.

```
PRINT NEITHER MEMBER ADRSLST
```

5. Perform miscellaneous scans on various library members.

```
PRINT LIB 1 MEM ASMPRGA SEQ 73 6 100
PRINT LIB 1 MEM ASMPRGB SEQ 73 6 100
PRINT LIB 1 MEM ASMPRGC SEQ 73 6 100
PRINT LIB 4 MEM COBPRGA SEQ 1 6 100
PRINT LIB 4 MEM COBPRGB SEQ 1 6 100
PRINT LIB 6 MEM MYPROC PHYSICAL
```

DTSBATCH Utility

The DTSBATCH utility cannot be invoked as part of an interactive partition job stream. It can be used only with submit-to-batch; that is, it can be run only via the SUBMIT procedure, and not with a /RUN or /EXEC command.

The DTSBATCH utility allows you to execute foreground system and edit commands, except the following:

/EXEC
/ENDRUN
/MSG
/RUN
/SEND
VSE/POWER interface (like /LISTP and /CTLP)
full-screen editor

The main purpose of the utility is to (1) let you print members from the library on the printer or (2) punch out members into 80-character records.

The other features of the DTSBATCH utility are not so important to you because you can perform the same functions by entering the actual commands. However, you can set up lists of commands (not including those excepted above) which can be processed offline in batch mode.

Control Commands

The control commands for the DTSBATCH utility are 80-character records of any foreground command you want to be executed against the VSE/ICCF file in batch execution mode.

Bits 3 and 4 of the UPSI byte are used to control punching out of library members:

/UPSI 0001
If specified, any of the commands /LIST, /DISPLAY, and /LIB cause the output to be directed to the card punch as well as to the printer.

/UPSI 00011
If specified, the normally listed output is directed to the card punch, and the printing is suppressed.

Note:

1. Because in normal input mode, system commands cannot be entered as data, editor input mode must be used for entering the commands. For example, if you are in system input mode and you wanted to enter a command (/LIST member) as data, the /LIST command would be interpreted and the listing would be produced immediately. However, in the input submode of the editor, system commands can be entered and will be accepted as normal data.
2. When setting up job streams using DTSBATCH, the /LOGON and password must be entered to get batch access to the system. Because the logon password must be coded in the data, it may be wise to password-protect the job stream, especially if you share a library with others. This protects the security of your logon password.

DTSBATCH Utility Examples

1. Punch card decks from the members stored in your library by the names FORTPROG and FORTDATA

```
/INPUT
/CANCEL                 (Ensure that input area clear)
ED
INPUT                   (Enter editor input mode)
/LOAD DTSBATCH    *
/UPSI 00011       *     (Punch decks, suppress print)
/LOGON USRJ       *     (Logon with your user ID)
PASWDJ            *     (Your logon password)
/LIST FORTPROG    *
/LIST FORTDATA    *
/LOGOFF           *
                        (Null line to return to edit mode)
SAVE BATCHEX            (Save commands in the library)
```

   The lines flagged with an asterisk (*) are saved as a library member and form the basis for the batch execution.

2. Produce full directory listing and list the members COBSAMP and BASCSAMP with line numbers.

   Enter input submode of the editor as above.

```
/LOAD DTSBATCH
/LOGON JACL
JACJUN
/LIB FUL,ALL
/DISPLAY COBSAMP
/DISPLAY BASCSAMP
/LOGOFF
```

DTSCOPY Utility

The DTSCOPY utility program copies a disk file from one file area to another (input may also be a tape file). The source and target files may be on the same disk device type or on different ones.

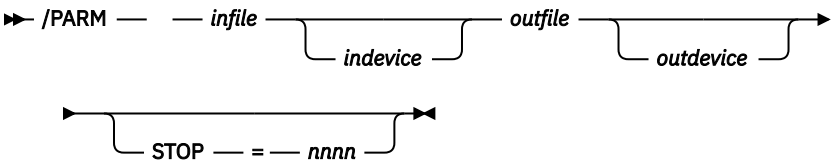
You can copy any file created by using the sequential access method of VSE for disk. Direct access files without keys can also be copied if the file area contains an EOF record at the end of the usable area or if you supply a record count on the control command discussed below.

The utility supports a block size of up to 8000 bytes.

A typical use for this program is transferring permanent data into a temporary file area (IJSYS00 through IJSYS04). This may be desirable when you are testing an application and do not want to endanger the permanent file. Copying may also be useful when processing in a programming language such as FORTRAN, which does not offer complete flexibility in naming files.

Control Command

The DTSCOPY utility requires one control command to be read. Its format is:



The operands must be separated by blanks or commas. They must be specified in the order as indicated.

infile
Is the name of the input file; that is, the file to be copied. If you specify IJSYSIN as the infile operand, the statements immediately following the /PARM statement in the input job stream will be written to the output file in blocks of 800 bytes. If you specify IJSYSTP, the **unlabeled** tape file assigned to SYS004 will be written to the output disk file.

indevice
Is optional. It is the device type on which the file resides. The possible operands are 3375, 3380, 3390 or FBA. If this operand is omitted, FBA is assumed.

outfile
Is the name of the file area into which the input file is to be copied. The name can be from one to seven characters in length.

Note: The CISIZE specification of an 'outfile' residing on an FBA device should be 4K or greater, or omitted entirely. If the size specification is omitted, 4K is assumed. Therefore, a file may require more space than before if it originally had a smaller CISIZE.

outdevice
Is the device type on which the output file area resides. It can be specified as 3375, 3380, 3390 or FBA. If the outdevice specification is omitted, the input device type is assumed.

STOP=nnnn
Can be used to specify the maximum number of data blocks to be transferred; it can be a value of up to 9999. You can use the operand to end the copy operation if there is no EOF record at the end of the file. The operand may also be useful if you want to transfer only a portion of a file for test purposes.

DTSCOPY Utility Examples

1. Transfer the contents of a file named NAMEFIL into the temporary file area IJSYS04. The default device type for the installation is used and all records are transferred. It is assumed that work files IJSYS01,2,3 and 4 are preallocated and require no /FILE statements.

```
/INPUT
/LOAD DTSCOPY
/PARM NAMEFIL IJSYS04
/ENDRUN
```

2. Copy the temporary file area IJSYS00, which is on a 3380 device, to the permanent file area named INVMAST which is on 3390. It is assumed that IJSYS00 is a preallocated work file requiring no /FILE statement.

```
/INPUT
/LOAD DTSCOPY
/FILE NAME=INVMAST,SER=INVPAK,ID='INVENTORY.MASTER'
/PARM IJSYS00 3380 INVMAST 3390
/ENDRUN
```

3. Copy the first 20 blocks of data from a file named PAYMAST to a file named PAYTEST, which will be allocated from dynamic space.

```
/INPUT
/LOAD DTSCOPY
/FILE NAM=PAYMAST,SER=PAYPAK,ID='PAYROLL.MASTER'
/FILE NAM=PAYTEST,SPACE=6,DISP=KEEP
/PARM PAYMAST PAYTEST STOP=0020
/ENDRUN
```

DTSDUMMY Utility

When run, this utility goes immediately to end of job. It may be useful if you want to set /OPTION statement options or /UPSI statement switches in a job step prior to the step which tests the condition.

The DTSDUMMY utility can be useful also in procedures that contain execution dependencies. Here, depending on /EXEC command operands, you may want a step to be run, and sometimes not. A /LOAD for a variable parameter can be set up. The default for the variable parameter can be defined as DTSDUMMY. If a member name is supplied on the /EXEC command, the program contained in the member is run. Otherwise, DTSDUMMY is run.

Another use of DTSDUMMY is to force terminal print output at a certain point in a job stream, for example between a compile and an execution, where otherwise the correct placement of the /FORCE command would be difficult.

DTSDUMMY Utility Examples

1. Use DTSDUMMY to set options and UPSI switches:

```
/LOAD DTSDUMMY
/UPSI 101
/OPTION SAVEPUNCH,TIME=500
/LOAD ...                (Next step)
```

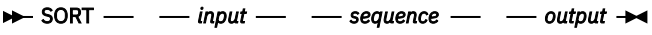
2. Use DTSDUMMY to force print output between a compilation and an execution:

```
/LOAD VFORTRAN
 ...
 ...                     (FORTRAN Source)
 ...
/LOAD DTSDUMMY
/FORCE
/DATA
 ...
 ...                     (Input data)
 ...
```

DTSSORT Utility

The DTSSORT program is a single-pass sort program which can be invoked to sequence records from your input area or punch area. Records from a member can be sorted if they are placed in the input area (with the /INSERT command or the /INCLUDE statement).

The sorting sequence is established through the use of a SORT control statement, which is read from the input area. It has the following format:



The operands must be separated from the SORT keyword and from each other by at least one blank. They must be coded in the sequence as shown.

SORT
This keyword must begin in column 1.

input
The operand specifies the source of the input records. For input, specify either of the following:

INPUT
If the records for the sort are to be read from your job stream. The 80-character records (or an /INCLUDE for the records) to be sorted must immediately follow the SORT control statement.

PUNCH
If the contents of the punch area are to be used for input.

sequence
For sequence, you can specify up to four subfields for sequencing, ranging from major to minor, as follows:

- Each subfield specification begins with an A (ascending) or D (descending) followed by the two digit starting column number and the two digit length of the subfield.
- No spaces are permitted between subfield specifications.
- The fields can overlap, but they may not extend beyond the 80th column.

output
The output of the sort is normally to the punch area. It may also be routed to the terminal as display data by specifying PRINT following the sort sequence operand. By supplying a suitable /FILE statement, output can be an existing VSE/ICCF library member (see Example 5 below).

Default Sort

If the SORT control statement is not read as the first record, then a default control statement is used. The default causes records from the input area to be sorted on columns 1 through 15 in ascending sequence and the sorted output to be placed into the punch area.

DTSSORT Utility Examples

1. Sort a portion of a member – output to punch:

```
/INPUT
/LOAD DTSSORT
SORT INPUT A1005
/INSERT MYFILE 2 48
/ENDRUN
```

2. Sort an entire member – output to terminal:

```
/INPUT
/LOAD DTSSORT
SORT INPUT A1005D1503 PRINT
/INCLUDE NAMEFILE
/ENDRUN
```

3. Sort data in punch area – output to punch:

```
/INPUT
/LOAD DTSSORT
/OPTION RESET
SORT PUNCH D0105A1005A4310
/ENDRUN
```

4. Default Sort – input from input area, output to punch in ascending sequence of columns 1 through 15:

```
/INPUT
/LOAD DTSSORT
 ...
 ...                (Data or /INCLUDE referencing the data)
/ENDRUN
 ...
```

5. Sort an entire member – output overlays original member in the library:

```
/INPUT
/LOAD DTSSORT
/FILE NAME=NAMEFILE,UNIT=SYSPCH,TYPE=ICCF
SORT INPUT A1005
/INCLUDE NAMEFILE
/ENDRUN
```

LINKNGO Utility

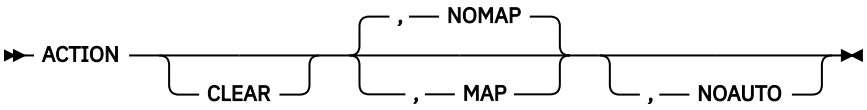
This program is used to bring the object code from a compiler into an executable form and then to transfer control to the entry point for execution. The utility is invoked implicitly whenever a /RUN or /EXEC request with OPTION DECK or GO is made for a compiler or the assembler. It can be invoked explicitly through the control statement /LOAD LINKNGO.

Input to LINKNGO is normally from the punch file, but this can be changed to the input area through the use of the /UPSI 1 statement.

LINKNGO automatically finds any subroutines or system support modules that may be required for program execution. This action is normally transparent to you. It should concern you only if the message UNRESOLVED EXTRN appears as part of the output in the load information. Valid input to LINKNGO consists of optional control statements and object modules. All LINKNGO control statements use the standard VSE link-edit control statement format.

Control Statements

Following is a discussion of the valid control statements and their formats. Any of these statements must be preceded by at least one blank column:



This control statement sets the mode of processing for the current run of the program. Only one ACTION statement is permitted per run, and the statement must precede all other control statements in the input stream. The specified actions must be separated by commas if more than one is used.

CLEAR
Causes storage to be cleared to binary zeros up to the end of the program area. Nothing beyond the end of the last control section or the end of common (if present) is cleared. This ensures that uninitialized areas have binary zeros as their initial values.

MAP
NOMAP
MAP produces a listing of the location of each control section of the program, along with its name and all entry points within the control section.

NOMAP (which is the default) suppresses the MAP option.

NOAUTO
Suppress the automatic lookup in the VSE library for any unresolved external names. This option would be used to ensure that only object modules specifically requested are included in the program.

Other action codes valid to VSE are ignored. They pertain to situations which are not relevant in an interactive system.

The statement gives a name to your program. It allows you to specify page boundary alignment or any offset from the start of the interactive partition; it allows you to suppress automatic lookup in the VSE library. Any other options which are valid under VSE are ignored. This statement must be processed before any object modules are processed.

name
A string of one to eight alphameric characters. This string is used as the name of your program. The operand is required if you submit a PHASE statement.

origin
Your program starts at the beginning of the user's storage area plus 56 character positions, unless an additional offset is provided. Valid origins are '*' and 'S', and either must be present.

An offset, if present, must immediately follow the origin and has the format +X ' nnnnnn ' . For nnnnnn, specify the offset as a six-digit hexadecimal value. This value will be added to the normal program start value and is the actual starting point of the program.

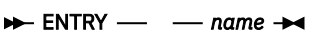
actions
These are additional actions which pertain to the whole operation of LINKNGO. These actions are:

NOAUTO
The same as in an ACTION statement – Suppresses the automatic lookup for any unresolved external names.

PBDY
Depending on the environment in which VSE/ICCF is running, LNKNGO sets the start address of the program to a multiple of 2K or 4K.

This control statement causes inclusion of the named module from a VSE library.

name
Must be the name of a module in a VSE sublibrary. Submodular naming is not permitted.



The statement, if present, denotes the end of the input stream. It specifies an external name which should be given control when the load function is complete.

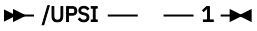
This statement is used **within** an object module to replace data that has already been assembled or compiled. The statement allows you to change instructions or the values of constants within an object module. The statement must be located after the data it is to replace or it will have no effect. The best position for this statement is before the END statement of an object module. There is no limit to the number of REPLACE statements allowed in an object module.

***REPbb**
These must be the first six characters of the statement, where bb is two blanks. The VSE format (first position X'02') is also acceptable but difficult to enter on most terminals.

aaaaaa
Is a six-byte hexadecimal address of the start of the data to be replaced. This address is taken from the program listing. Any necessary relocation is done by the loader program.

iii
Is the identification number of the control section containing the data to be replaced. It is always hexadecimal and, except for a multiple control section program, always 001. In the case of a multiple control section program, the number can be read from the external symbol dictionary listing on the compiler output.

data
This field immediately follows the III field (no intervening blank). It is always in groups of 2 bytes (four hexadecimal characters) separated by commas. A maximum of 24 bytes can be replaced by a single REPLACE statement.



The /UPSI 1 job entry statement causes LINKNGO to read its input from the job stream rather than from the punch area.

Note: The LINKNGO program uses the installation default unit (usually SYS008 but it may be some other) for reading object decks from the punch area, unless /UPSI 1 is specified. Therefore, if your program also reads from the punch area using a different SYS number via an /ASSGN, the object deck must be read from SYSIPT using /UPSI 1.

LINKNGO Utility Examples

1. Normal compile and run (implicit use of LINKNGO):

```
/INPUT
/LOAD VFORTRAN               (/OPTION DECK assumed)
(source program)
/DATA
(input data)
/ENDRUN
```

2. Combining a module from a VSE library (SUBRTN) with the output of an assembly:

```
/INPUT
/LOAD OBJECT
  ACTION MAP,CLEAR,NOAUTO   |  These three state-
  PHASE MYTEST,*            |  ments are written
  INCLUDE SUBRTN            |  to the punch area.
/LOAD ASSEMBLY
```

```
 ...
 ...                          (Program)
 ...
/DATA
 ...
 ...                          (Data)
 ...
/ENDRUN
```

3. Running entirely from a VSE library:

```
/INPUT                        (Implicit use of LINKNGO)
/LOAD OBJECT
  INCLUDE PROGA
/ENDRUN
```

or

```
/INPUT                        (Explicit use of LINKNGO)
/LOAD LINKNGO
/UPSI 1
  INCLUDE PROGA
/ENDRUN
```

4. Executing an object program saved in the VSE/ICCF library file:

```
/INPUT
/LOAD OBJECT
/INCLUDE name
/ENDRUN              (Implicit invocation of LINKNGO)
```

or

```
/INPUT
/LOAD LINKNGO
/UPSI 1              (Tells LINKNGO to read from job stream)
/INCLUDE name
/ENDRUN
```

The format of the /INCLUDE member can be either:

```
 ...
 ...                 (Object deck)
 ...
```

or

```
 ...
 ...                 (Object deck)
 ...
/DATA
 ...
 ...                 (Data)
 ...
```

OBJECT Utility

The OBJECT utility is primarily used to transfer object programs and LINKNGO control statements from the job stream to the punch area, from where they are loaded into storage for execution.

OBJECT behaves somewhat like a compiler. It places object programs in the punch area and sets a flag. This flag causes the LINKNGO program to be automatically invoked at the end of the job stream or when a /DATA statement is encountered.

OBJECT may be used also to transfer data from the job stream to the punch area. Setting the NOGO option prevents the LINKNGO program from being invoked if it is not needed.

When UPSI-0 is set on, the OBJECT program transfers the contents of the job stream to the terminal, rather than to the punch area. It thus may be used as an 80-80 list program.

When the NODECK option is set, OBJECT will not write any data to the punch area.

OBJECT Utility Examples

1. Cause a subsequent execution of LINKNGO to produce a CSECT map by writing ACTION MAP to the punch area prior to a compilation.

```
/LOAD OBJECT
 ACTION MAP
/LOAD VFORTRAN
 ...
 ...                    (FORTRAN Source Program)
 ...
/DATA
 ...
 ...                    (Input data records)
 ...
```

2. Cause LINKNGO to load and run a module from a VSE library.

```
/LOAD OBJECT
 INCLUDE  MYPROG        (Name in VSE library)
/DATA
 ...
 ...                    (Input data for execution)
 ...
```

3. Cause LINKNGO to load and run an object program stored in the VSE/ICCF library.

```
/LOAD OBJECT
/INCLUDE  MYMOD         (Name of library member)
/DATA
 ...
 ...                    (Input data for execution)
 ...
```

4. Write data to the punch area but prevent LINKNGO from being invoked.

```
/LOAD OBJECT
/OPTION NOGO
 CARD 1
 CARD 2
 CARD 3
  ...
```

5. Write 80-character records from the library (members named MEM1, MEM2, MEM3) to the terminal.

```
/LOAD OBJECT
/OPTION NOGO
/UPSI 1
/INCLUDE MEM1
/INCLUDE MEM2
/INCLUDE MEM3
```

Subroutines

Two subroutines are available as part of VSE/ICCF: DTSSNAP and DTSPGMCK. Your VSE/ICCF administrator may give you additional subroutines.

Subroutine DTSSNAP

This subroutine can be used to display a portion of your storage area in storage dump format. This subroutine can be used in assembler and FORTRAN via the CALL statement.

You can call the subroutine anywhere in your program to obtain a hexadecimal and character display of storage areas and general registers (except 0 and 1).

The subroutine requires two parameters: the labels of the beginning and ending areas to be displayed.

Example

- Assembler example

```
/LOAD ASSEMBLY
MYPROC    CSECT
          BALR  5,0
          USING *,5
          CALL  DTSSNAP,(TABLE,TABLND)
          EOJ
TABLE     DC    XL256'00'
TABLND    EQU   *
          END
```

DTSPGMCK Subroutine

This subroutine, if called at the beginning of a program, causes a dump of general purpose registers 2 through 13 and the storage area to be displayed on the terminal if a program check interrupt occurs.

The dump is displayed in both character and hexadecimal formats. The storage area displayed is the area near the location of the program check.

The subroutine should be issued only once per program.

If the subroutine is run while some other program check facility (such as the FORTRAN debug facilities) is active, this subroutine has no effect. In other words, the debug facilities of the compilers FORTRAN and RPG II are more useful than this subroutine.

This subroutine can be very useful if you are programming in assembler language.

Example

```
/LOAD ASSEMBLY
 TESTPRG   CSECT
           BALR 5,0
           USING *,5
           CALL DTSPGMCK
```

Chapter 11. Job Entry Considerations

This publication provides you with basic information on building a job stream and on the execution of jobs on an interactive system. A section is provided for each supported language. In that section, any unique requirements for compiling and executing the programming language will be noted.

General Considerations

Interactive Partitions

When an /EXEC or a /RUN command has been successfully processed, the system attempts to find an available virtual storage area into which to schedule the job. This storage area is referred to as an interactive partition. It appears to have the same characteristics as a normal VSE partition; for example, its own work files, its own GETVIS area, and its own storage protection key. The storage protection key protects the programs of one interactive partition from accidental destruction by programs in another interactive partition.

GETVIS Area

A certain amount of each interactive partition is reserved for dynamic allocation; this part is referred to as GETVIS space. The amount normally reserved is 48K, which is also the minimum.

Some programs and compilers need more than 48K of GETVIS space. It depends on how many modules are loaded and the size of the I/O areas and tables in the GETVIS area.

You can alter the default GETVIS allocation by use of the GETVIS operand on the /OPTION statement. The maximum amount of GETVIS space that you can allocate is your interactive partition size minus 20K.

Time-Sliced Execution

Once a job has been successfully scheduled (in an interactive partition), it is made eligible for execution. When a job begins execution, it continues to run until one of the following occurs:

- A conversational read is encountered.

  The background job execution is suspended. Any print lines in the print area are written to the terminal, and the request for conversational input (? on an IBM 2740/3767 or ENTER DATA on an IBM 3270) is displayed. After you have entered the conversational input, the job is again made eligible for execution.
- The print area is full or has been forced (/FORCE statement).

  The background job execution is suspended until all print lines have been transferred to the terminal, at which point the job again becomes eligible for execution.
- The job ends.

  The interactive partition becomes available for other users, and the remainder of the print output is transferred to the terminal.
- The interactive partition's time slice elapses.

  Execution is suspended, but the job is made eligible for execution the next time a time slice becomes available.

Job Streams

Job streams comprise job entry statements, source programs and input data. Actually, the source programs or data (or both) need not be a part of the job stream; they can be included by the /INCLUDE job

entry statement. Job entry statements can be specified also in members that are included by /INCLUDE statements.

Job Entry Commands

Two commands can be used to initiate execution of background jobs: /RUN (or /ENDRUN) and /EXEC. If /RUN is used, the job stream or an /INCLUDE referencing the job stream must be in the input area. If /EXEC is used, the job stream (or command list) must be a library member.

The Input Area

This is a temporary disk work space into which you can enter job entry statements, programs, and data. See “Temporary Storage Areas” on page 6 for more information about the input area.

The Punch Area

The punch area is a temporary disk work space into which the language compilers write their object program output. You can also place 80-column data into the punch area (unit SYSPCH/SYS007 defined below) and read it later. Data in the punch area can be saved in your library by executing the STORE macro or by inserting the data into the input area and saving it as shown by the examples for the “[/]SAVE Command” on page 96. See “Temporary Storage Areas” on page 6 for more information about the punch area.

Input/Output

Your program can obtain input from one of the following sources:

- Job stream
- The terminal in a conversational manner
- A library member
- A previously written temporary disk file
- A permanent disk file.

The output can be written to one of the following:

- The terminal
- The punch area
- The print area
- A VSE/ICCF member
- A temporary disk work file
- A permanent disk file.

The choice of input/output type depends on your needs. The type of input/output in most programming languages is decided by how you specify file names and/or VSE SYS-unit designations.

The following input/output choices are available to all users:

SYSIPT
: Indicates that 80-character records are to be read from the job stream. The data is in the job stream at the logical point where it is to be read. For programs like language compilers, this point would be after the /LOAD or other job entry statements. For your own program compilations and executions this point would be after the /DATA statement. If INCON is specified on an /OPTION or a /DATA statement, the input request is converted to a conversational read from the terminal.

  When defining this unit in a program, you should always set it up as if it were an actual card reader.

  In FORTRAN, SYSIPT corresponds to Unit 5.

SYSLST
: Indicates that print lines are to be directed to the terminal for output.

In FORTRAN, SYSLST corresponds to Unit 6. In other programming languages, define this unit as if it were an actual printer.

All line spacing will be suppressed and only single-spacing will be displayed.

SYSPCH
Indicates that 80-character records are to be written to the punch area. Language compilers use this as the unit designation on which the object program is to be written. You can also use SYSPCH (the punch area) for output that is to be stored later in your library.

In FORTRAN, SYSPCH is referenced as unit number 7. In other programming languages,define this unit as if it were an actual punch.

SYSLOG
Indicates input from or output to the terminal. This unit designation can be used to build a conversational program. Any read to SYSLOG will be translated into an input request from the terminal while any write to SYSLOG will be treated as a write to the terminal.

SYSLOG input can be up to 256 bytes long. SYSLOG output can be up to 156 bytes long. If a programming language does not permit input on SYSLOG, SYSIPT with the INCON option may be substituted.

SYS000
Is a temporary disk file on which you can record information to be read in a later job step. The file should be defined within your program as a disk file residing on a disk drive. The choice of drive type depends on the disk storage type which your location uses. Records written to or read from SYS000 can be of any size permitted on the type of the specified device. The file type must be sequential. Since a disk file is identified by a seven-character file name, the file name IJSYS00 must be used to reference the file.

SYS001, SYS002, and SYS003
These are temporary disk files with the same attributes as listed above for SYS000. Their file names are IJSYS01, IJSYS02, and IJSYS03, respectively. The differences between these units/files and SYS000 are: these file areas are used as compiler work areas (but not in FORTRAN). They may not be used to pass data from one job step to another if a compiler step intervenes. (SYS003 is not used as a work file by RPG II.)

SYS004
This is a temporary disk file with the same attributes as SYS001 through SYS003. The file name associated with this unit is IJSYS04. SYS004 differs in that it is used as a work file only by the ANS COBOL compiler. None of the other compilers use SYS004 as an intermediate work file.

If your location does not support preallocated work files for IJSYS00 through IJSYS04, you must supply / FILE information in your job streams to have these files allocated and made available for your use. All work files can be defined as follows:

```
    /file name=ijsys0n,space=20
```

where 'n' is specified as appropriate. The amount of space can be changed according to your needs.

SYS005
Card read (job stream) input. SYS005 can be used in place of SYSIPT (defined above) in programming languages which do not permit reference to SYSIPT. SYS005 has all the same characteristics as SYSIPT. However, you should check whether your system was tailored to have a unit other than SYS005 used as the programmer unit for job stream input.

SYS006
Printer (terminal) output. SYS006 can be used in place of SYSLST (defined above) in programming languages which do not permit reference to SYSLST. SYS006 has the same characteristics as SYSLST. However, you should check whether your system was tailored to have a unit other than SYS006 used as the programmer unit for print (terminal) output. All line spacing is suppressed and only single line spacing is displayed.

SYS007
Punch area output. SYS007 can be used in place of SYSPCH (defined above) in programming languages which do not permit reference to SYSPCH. SYS007 has all the same characteristics as SYSPCH. However, you should check whether your system was tailored to have a unit other than SYS007 used as the programmer unit for punch (terminal) output.

SYS008
Punch area input. SYS008 is used for directly reading the contents of the punch area. This unit indicates that the 80-character records in the punch area resulting from a previous job or job step are to be read. The LINKNGO program uses this unit number as the source of its object program input. When defining this unit in a program, it should always be setup as if it were an actual card reader. Note that the contents of the punch area may also be read on SYSIPT/SYS005 via an /INCLUDE $$PUNCH statement.

Check whether your system was tailored to have a unit other than SYS008 used as the programmer unit for reading the punch area.

SYS009
Terminal input or output. SYS009 can be used in place of SYSLOG (discussed above) in programming languages which do not permit reference to SYSLOG. SYS009 has all the same characteristics as SYSLOG.

Check whether your system was tailored to have a unit other than SYS009 used as the programmer unit for accessing SYSLOG.

Permanent Disk Files

You can access permanent VSE disk files if disk label information (DLBL/EXTENT for the files) is provided in the job stream that starts VSE/ICCF, or if you include this information for such files in your job stream that accesses the files.

You may specify file information (via the /FILE statement) for normal VSE files if the following conditions are met:

- Your user profile allows you to specify VSE file information.
- The file to be accessed is not an ISAM file.
- The file has only one extent.

If these criteria are not met, you can access the requested file only if the label information (DLBL/EXTENT) for the file was specified in the VSE/ICCF startup job stream or is stored in a standard label area.

If permanent disk files are to be referenced in your programs, keep in mind the following points:

1. The name you specify for the file when you define it in your program must be the same as the 'file name' on the DLBL statement or the NAME= operand in the /FILE statement.
2. VSE/ICCF provides no facilities for protecting these files from simultaneous updating. You must therefore provide for this protection yourself.
3. Permanent data sets can be organized as sequential, direct, VSE/VSAM or indexed sequential and accessed according to standard VSE access method conventions.

Overall Job Restrictions

Interactive processing requires that certain limitations be observed. For example, timing devices should not be built into programs. The setting of timer intervals in background jobs is prohibited.

Tape file processing is not generally supported in background jobs.

When defining the unit record type files (SYSIPT, SYSLST, SYSPCH, and SYS005 through SYS009), always specify them as unblocked, fixed length, unlabeled, and single buffered where explicit definition is required.

The assembly (or compilation) and execution of multiphase programs is not supported by the LINKNGO utility. However, a multiphase program may be compiled and linked (via the SUBMIT to VSE/POWER procedure) to a VSE library. Once in the VSE library, the multiphase program can be loaded and run in VSE/ICCF by specifying its name on the /LOAD job entry statement.

The format of the print output produced by the RELIST macro may be different from the format you would get if you run the same program in a VSE batch partition and send the print output directly to a printer or to a VSE/POWER print queue.

For more details on restrictions in interactive partitions, see Appendix A, “Interactive Job Restrictions,” on page 333.

End of Job

When a job ends normally or is canceled by VSE or VSE/ICCF, either of the following messages is printed at the end of the job's terminal output:

```
**** JOB TERMINATED -VSE/ICCF RC nn, EXEC RC nnnn, NORMAL EOJ
```

or

```
**** JOB TERMINATED - ICCF RC nn, JOB CANCELED
```

A VSE/ICCF RC of 00 is the normal job ending code. These codes, indicated above by nn in the message, more fully explain the reason for job termination. A VSE/ICCF return code of 08, for example, is usually preceded by a VSE message. Both, VSE messages and VSE/ICCF return codes are described in z/VSE Messages and Codes 2

The EXEC RC is the job execution return code as set and documented by the program that was called.

An interactive partition is canceled if the job running in it exceeds the time limit set for the partition. If, however, terminal I/O is outstanding when this partition limit is reached, then the partition is not canceled immediately. It will be canceled either after the I/O is done or when the terminal time-out occurs – depending on which event happens first.

Assembler and Compiler Considerations

This section contains information about the use of the assembler and of compilers that are supported by VSE/ICCF.

Following is a list of explicitly supported language translators. The list also shows how you can invoke the translator by way of a /LOAD command. Your location may have chosen not to install all of these compilers or to install compilers which are not in this list. For information about available (and supported) compilers contact your VSE/ICCF administrator.

DOS/VS COBOL, PLI Optimizing Compiler and BASIC Support has been removed with VSE/ESA 2.4.

COBOL/VSE, PLI/VSE and C/VSE are not supported by VSE/ICCF.

Language Translator	/LOAD Command
Assembler	/LOAD ASSEMBLY
FORTRAN compiler	/LOAD VFORTRAN
RPG II compiler	/LOAD RPGII

Program Linkage

The LINKNGO utility combines program segments produced by different language compilers into an executable program. However, standard IBM VSE linkage conventions must be maintained, and program linkage must be supported by the given compiler.

The following job entry statements can be used to compile a FORTRAN program that calls an assembler subroutine which is to be assembled in the same job stream.

```
/LOAD VFORTRAN
 ...
 ...
 ...
 CALL SUBRTN
 ...
 ...
 ...
/LOAD ASSEMBLY
 SUBRTN START 0
 ...
 ...
 ...
        BR 14
/DATA
 ...
 ...                      (Data records, if any)
 ...
```

The program or subprogram to be loaded with other programs or subprograms may appear in the job stream in object (already compiled) form. For example, if SUBRTN alone had been compiled and its object program stored in the library under the name SUBOBJ, the job stream might look as follows:

```
/LOAD VFORTRAN
  ...
 ...
 ...
 CALL SUBRTN
 ...
 ...
 ...
/LOAD OBJECT             (Optional)
/INCLUDE SUBOBJ
/DATA
 ...
 ...                      (Data records)
 ...
```

It is also possible that all programs and subprograms have been separately compiled and stored in the library. The execution job stream would then be:

```
/LOAD OBJECT             (Optional)
/INCLUDE FORTOBJ
 /INCLUDE SUBOBJ
/DATA
 ...
 ...                      (Data records)
 ...
```

The main program should always be the first source or object program in (or included in) the job stream, so that the proper transfer address is obtained.

In the examples above, the optional /LOAD OBJECT causes the object programs to be transferred to the punch area for loading by LINKNGO. If /LOAD OBJECT is omitted and an object program is encountered in the job stream, the OBJECT utility is invoked automatically.

The following sections contain information and restrictions regarding each of the language translators.

Assembler Considerations

VSE/ICCF supports the assembler for program creation and execution.

In addition, subroutines may be written in assembler language and entered through the standard VSE CALL linkage conventions from programs written in FORTRAN or RPG II.

Subroutines are combined with a main program by the LINKNGO utility program. LINKNGO can combine object modules in the punch area with object modules from the library. Unresolved external references are resolved from a VSE library.

Note:

1. At program invocation the VSE register conventions are observed.

   R0
   Load point of loaded phase.

   R1
   Dependent on the /LOAD job entry statement: if any parameter was passed, the register points to the parameter field(s); else, the entry point of the loaded phase.

   R13:
   Pointer to an 18-fullword save area.

   R14:
   Return point to VSE/ICCF.

   R15:
   Entry point of the loaded phase.
2. Because the LINKNGO program produces only single phase programs, the LOAD and FETCH macros can be used only to load (or fetch) a phase from a VSE library.
3. Programs issuing any of the following macros are not supported in interactive partitions.

```
ATTACH                JOBCOM            STXIT(MR|TT)
DETACH                PAGEIN            TESTT
DEQ                   PFIX              TPIN
ENQ                   PFREE             TPOUT
EXCP ccbname,REAL     POST              VIRTAD
EXIT(AB|MR|TT)        REALAD            WAITF
FCEPGOUT              SETPFA            WAITM
FREE                  SETT
```

4. The STXIT OC, STXIT PC, and STXIT AB macros can be used to provide for operator attention, program check, or abnormal termination intercept.
5. End of file on SYSIPT or SYS005 (or equivalent unit) can be recognized in one of the following ways:
   - By testing for /, *, and a blank in columns 1 through 3 of the read input area
   - By testing the unit exception bit in the CCB.

   If you use the SAM macros DTFCD, GET, and PUT, the logical IOCS routines recognize end of file.
6. If invalid CCWs are issued by the program they are ignored.

Examples

The following assembler program will read statements, write them onto a temporary disk file, read them back in and list them on SYSLST (on the terminal). As each record is read the second time, a subroutine is entered to verify the validity of the data.

```
*READY
/input
/load assembly
/include workfile            (If no preallocated work files)
Mainpgm   csect
          balr   5,0                load base reg
          using  *,5                declare base reg
          open   filout             open output file
loopa     excp   rdccb              read a statement
          wait   rdccb              wait on completion
          clc    r(3),=c'/* '       check for last statement
          be     endcard            go to file end routine
          mvc    0(80,2),r          move image to filebuffer
          put    filout             write to output file
          b      loopa              loop through all input
endcard   close  filout             close output file
          open   filin              open the input file
loopb     get    filin              read record
          la     13,savearea        savearea pointer
          lr     1,2                dataarea pointer
          call   subpgm             enter the subroutine
          ltr    1,1                test error condition
```

```
          bz     cardok              bypass message if ok
          mvc    p+85(12),=c'invalid statement'
cardok    mvc    p(80),0(2)          move image to print
          excp   prccb               write to syslst
          wait   prccb               wait i/o completion
          mvc    p,=cl120' '         clear print area
          b      loopb               process next record
enddisk   close  filin               close input file
          eoj
savearea  ds     9d                  save area
p         dc     cl120' '            print area
r         dc     cl80' '             input area
rdccb     ccb    sysipt,rdccw        card read ccb
prccb     ccb    syslst,prccw        printer ccb
rdccw     ccw    2,r,0,80            read command
prccw     ccw    9,p,0,120           print command
ioa       ds     cl408               disk i/o area1
iob       ds     cl408               disk i/o area2
filin     dtfsd  ioarea1=ioa,ioarea2=iob,ioreg=(2),typefle=input,  x
                 blksize=400,recsize=80,                          x
                 eofaddr=enddisk,recform=fixblk
          org    filin+22
          dc     c'ijsys00'          set filename to ijsys00
          org
filout    dtfsd  ioarea1=ioa,ioarea2=iob,ioreg=(2),typefle=output, x
                 blksize=408,recsize=80,recform=fixblk
          org    filout+22
          dc     c'ijsys00'          set filename to ijsys00
          org
          end
```

```
/load assembly
/include workfile
subpgm    csect
*this subroutine is called by the main program to validate data
          save   (14,12)             save calling program regs
          lr     5,15                set reg 5 as base
          using  subpgm,5            declare base
          la     4,5                 5 is max no of data columns
          lr     3,1                 1 contains data area pointer
loop      cli    0(3),c'0'           is column numeric
          bl     errext              no - error exit
          la     3,1(3)              bump to next byte
          bct    4,loop              loop till all done
goodex    lm     14,12,12(13)        restore entry regs
          sr     1,1                 clear reg 1 indicates valid
*                                    card condition
          br     14                  return to caller
errext    return (14,12)             return to caller
          end
/data
/file name=IJSYS00,space=1 (if no preallocated IJSYS00)
03754     this is a valid statement
76843     this is a valid statement
42d68     this is an invalid statement
12345     this is a valid statement
/end
*READY
/run
*RUN REQUEST SCHEDULED
```

In the above example, both FILIN and FILOUT refer to the same file area. One DTFSD is for output, the other for input. The temporary file IJSYS00 is used for storing and retrieving the data. Because you cannot name both DTFs IJSYS00 nor use one DTF for both input and output, an ORG to filename+22 appears after each DTF. This is the area in the DTF macro expansion where the file name is kept. After the ORG, a DC C'IJSYS00' appears. This causes the name in the DTF area to be IJSYS00 for both files. When each file is opened, it will be setup to print to the SYS000 (IJSYS00) temporary file area.

Also, in the above example, if the subroutine SUBPGM had been assembled separately and the resulting object program stored in the library under the name SUBOBJ, the job stream could have been specified as follows:

```
    /LOAD ASSEMBLY
     ...
     ...                    (Source statements for MAINPGM
     ...                    as above)
```

```
/LOAD OBJECT             (This is optional)
/INCLUDE SUBOBJ
/DATA
 ...
 ...                     (Data records as above)
 ...
```

The /LOAD OBJECT is optional because the utility is invoked automatically if an object program is encountered in the job stream.

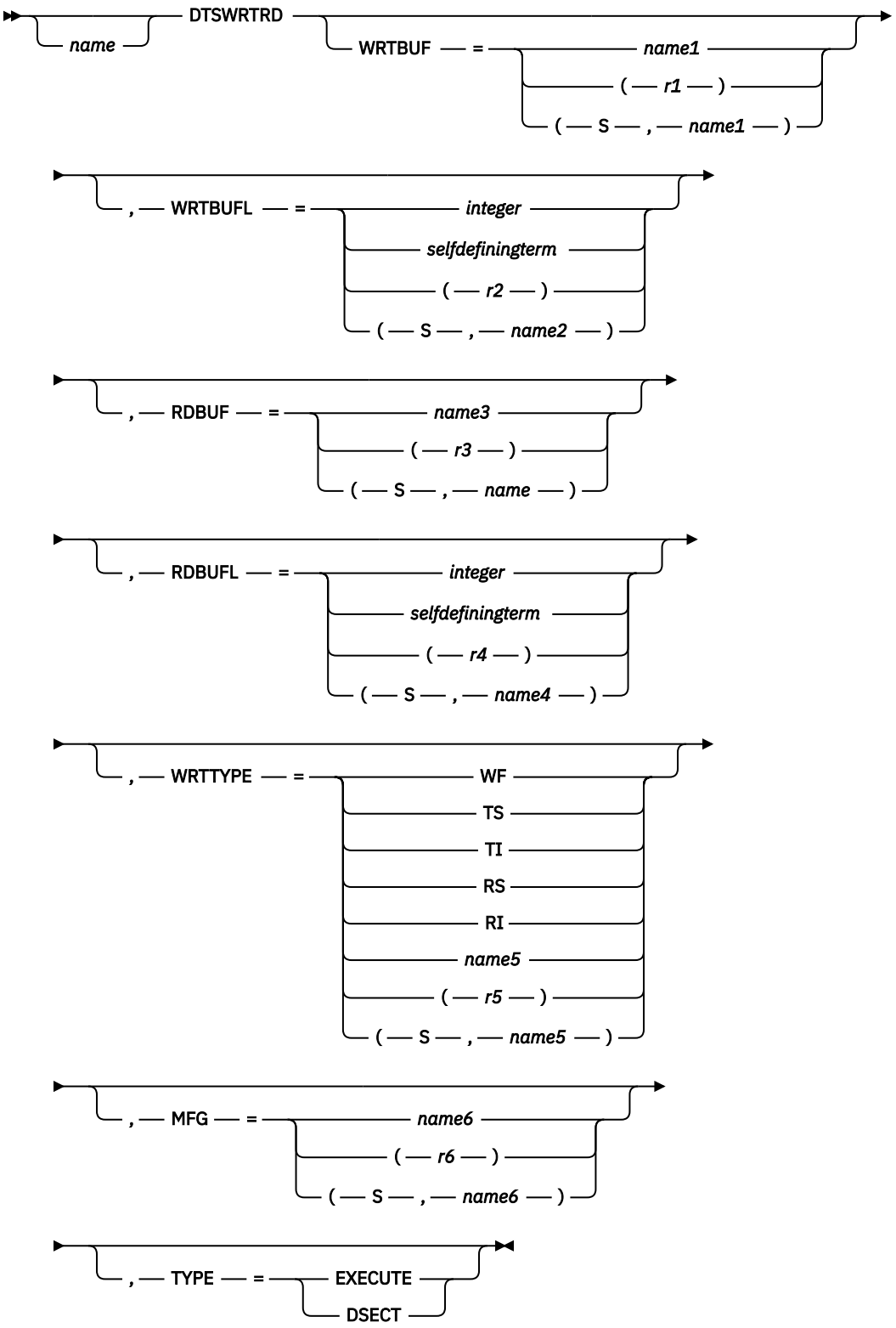
Full Screen Macro (DTSWRTRD)

The DTSWRTRD macro enables your assembler programs in interactive partitions to manipulate the screen of the IBM 3270 Display System. For example, it allows you to dynamically build and display screens or to support the selector pen.

You must provide the output data stream in an output buffer and analyze the input data stream that VSE/ICCF returns to your input buffer.

The output data stream that you build and send to the terminal is not modified by VSE/ICCF. Also, VSE/ICCF does not check or change the input data before transmitting it to your program (that data comprises only those fields which have their modified tags set on).

All operands are optional. A full-screen write does not have to be followed by a read. However, when this occurs, a wait occurs. The macro has the following format (the continuation character in column 72 is not shown):



WRTBUF=name1
WRTBUF=(r1)
WRTBUF=(S,name1)

Specifies the address of the write buffer. The contents of the write buffer can be structured in two ways, depending on the write request which you specify in the WRTTYPE operand. In either case, the data stream is transferred unchanged to the display station. This is discussed below:

WRTTYPE=RS
WRTTYPE=RI
WRTTYPE=TS
WRTTYPE=TI

Can be specified even if your terminal does not support the IBM 3270 Extended Data Stream Feature. Use the DTSSCRN macro to find out about the characteristics of your terminal. This macro is described under "Macro to Identify Terminal Characteristics (DTSSCRN)" on page 324.

The first byte in the write buffer must be the write control character (WCC). If your terminal supports the IBM 3270 Extended Data Stream feature, the following bytes can contain the 3270 extended data stream; else, they should contain the non-extended 3270 data stream. The extended 3270 data stream, which comprises the non-extended 3270 data stream, is described in the *IBM 3270 Data Stream Reference* manual, order number GA23-0059.

WRTTYPE=WF

Can be specified only, if your terminal supports the IBM 3270 Extended Data Stream Feature. In this case, no WCC may occur at the beginning of the write buffer, and the data stream has to be a collection of structured fields. For each one of these fields, specify a length greater than zero. The format of structured fields in outbound data streams is described in detail in the above cited *3270 Data Stream Reference* manual.

The field identifications that you can specify for a structured field are:

X'01'
Read partition

X'06'
Load programmed symbols

X'09'
Set reply mode

If you have a structured field with a read-partition identifier, ensure that:

- This structured field comes last in your data stream.
- You specify 'Query' (X'02') as Type for this structured field.

The output area may be located in the interactive partition issuing the DTSWRTRD macro or in the SVA. If WRTBUF is not specified, it is assumed to have been established previously in the area pointed to by the MFG operand, which is required in this case.

WRTBUFL=integer
WRTBUFL=selfdefiningterm
WRTBUFL=(r2)
WRTBUFL=(S,name2)

Specifies the length of the write buffer. The minimum is 1 byte, the maximum is 32700 bytes. For performance reasons, a value less than the device dependent TIOA size should be specified. IBM recommends that you specify a buffer size as shown below, depending on the terminal device type:

Terminal Type
Buffer Size

IBM 3270 Model 2
2000 bytes

IBM 3270 Model 3
2660 bytes

IBM 3270 Model 4
3560 bytes

IBM 3270 Model 5 *
3650 bytes

* A wide screen

By issuing the DTSSCRN macro (see “Macro to Identify Terminal Characteristics (DTSSCRN)” on page 324), your program can investigate the screen size of the terminal with which it is communicating. If WRTBUFL is not specified, the buffer is assumed to have been setup earlier in the area pointed to by the MFG operand, which is required in this case.

RDBUF=name3
RDBUF=r3
RDBUF=(S,name)

Specifies the address of the read buffer. The contents of the read buffer can be structured in two ways, depending on the write request which you specified in the WRTTYPE operand. If you specified:

- WRTTYPE=RS WRTTYPE=RI WRTTYPE=TS WRTTYPE=TI

  The buffer starts with a two-byte field that contains the length of the data stream. If this length is greater than RDBUFL, the data stream is truncated to the length of RDBUFL minus 2.

  The length field is followed by the inbound 3270 data stream (for a description of an inbound 3270 data stream, see the *IBM 3270 Data Stream Reference* manual, order number GA23-0059). All input data (AID, buffer address, data ...), with the exception of the cursor address, is transferred unchanged from the terminal into the input area following the length field. The cursor address, however, is translated into binary notation before it is placed into the read buffer (the first screen byte, for example, has the cursor address X'00').

  The AID's transferred are the PF keys, the CLEAR key, the ENTER key, and the selector pen attention.

  All PA keys are reserved for use by VSE/ICCF. If a PA key is pressed, the two-byte length field contains X'0000'.

- WRTTYPE=WF (with Read Partition Query)

  The buffer starts with a two-byte field that contains the length of the data stream. If this length is greater than RDBUFL, the data stream is truncated to the length of RDBUFL minus 2. The length field is followed by a one-byte Attention Identifier field, which has a value of X'88'.

  The Attention Identifier field is followed by one or more Query Reply structured fields. The various formats of a Query Reply structured field are described in the *IBM 3270 Data Stream Reference* manual, order number GA23-0059.

If RDBUFL is not 0 and RDBUF is not specified, the buffer is assumed to have been setup earlier in the area pointed to by the MFG operand, which is required in this case.

RDBUFL=integer
RDBUFL=selfdefiningterm
RDBUFL=(r4)
RDBUFL=(S,name4)

Specifies the length of the read buffer (minimum 2, maximum 32K - 1 or 32,767 bytes), including two bytes for the length field. If more data arrives, it is truncated to the length of the read buffer minus 2.

A value of 0 for RDBUFL indicates WRITE without READ, in which case the RDBUF address is ignored.

If WRTTYPE=WF with ID=Read Partition was specified, the minimum length of the read buffer is one byte. Number and length of the returned Query Reply structured fields determine the required size of the read buffer.

If WRTTYPE was specified as RS, RI, TS or TI, the maximum length of an input data stream that a program can read depends on the screen size of the terminal which the program is communicating with. For terminals supported by VSE/ICCF, the maximum input data stream is twice the screen size of the terminal. Your program can investigate the screen size of the terminal with which it is communicating via the DTSSCRN macro (see “Macro to Identify Terminal Characteristics (DTSSCRN)” on page 324).

If RDBUFL is not specified, the read buffer is assumed to have been setup earlier in the area pointed to by the MFG operand, which is required in this case.

WRTTYPE=WF
WRTTYPE=TS
WRTTYPE=TI
WRTTYPE=RS
WRTTYPE=RI
WRTTYPE=name5
WRTTYPE=(r5)
WRTTYPE=(S,name5)

Specifies whether the previous contents of the screen are to be erased before the write buffer is written. It also specifies whether VSE/ICCF is to save and restore the screen (for example when a message arrives and destroys the contents of the screen), or whether your program will be notified via a return code that the screen contents have been destroyed and must be rebuilt if it is still needed.

WRTTYPE=TS
Causes the Erase Write command to be issued.

WRTTYPE=RS
Causes also the Erase Write command to be issued, but a return code is placed in register 15 to inform your program that the screen must be rebuilt if it is still needed.

WRTTYPE=TI
Causes the Write command to be issued without a previous erase.

WRTTYPE=RI
Causes also the Write command to be issued without a previous erase, but a return code is placed in R15 to inform your program that the screen must be rebuilt, if it is still needed.

WRTTYPE=WF
Causes the Write Structured Field command to be issued without a previous erase. If the contents of the screen have been destroyed, a return code is placed in R15 to inform your program that the screen must be rebuilt, if it is still needed.

For all other specifications the address value must point to a field containing the character string TS, TI, RS or RI.

Note:

1. RS and RI should be used, especially on remote lines, because TS and TI increase message traffic and use more storage.
2. A WRITE/ERASE is forced when one of the following occurs:
   - The CLEAR key is pressed.
   - The program issues the first full-screen write request.
   - The program changes the WRTTYPE specification from RS or RI to TS or TI, or vice versa.

MFG=name6
MFG=(r6)
MFG=(S,name6)

Allows the address of a parameter list to be specified. If not specified, the parameter list is generated inline with the macro call, in which case modification and reuse of the parameter list is not possible. If specified, all other values are entered into the applicable slots of the list that your MFG specification points to. This way parameters can be overwritten or reused if specified in a previous macro call to DTSWRTRD.

The length of the parameter list can be determined with the DTSWRTRD macro with the TYPE=DSECT operand. The area in which the parameter list is to be built must be cleared to binary zeros before the first macro is entered.

TYPE=EXECUTE
TYPE=DSECT

Specifies the form of the macro:

EXECUTE
Specifies actual execution of the terminal I/O function. This is the default.

DSECT
Specifies that a DSECT describing the parameter list for DTSWRTRD is to be generated. The name of the DSECT is taken from the macro invocation. If no name is specified, the DSECT name defaults to DTSWRDD.

Some examples for possible operand specifications follow:

```
WRTBUF=BUFF            buffer address
WRTBUF=(2)             register containing the buffer address
WRTBUF=(S,BUFF)        S-type constant for relocating purposes
WRTBUFL=2000           absolute length value
WRTBUFL=L'BUFF         implicit length constant
WRTBUFL=LEN            length equate
WRTBUFL=(S,BUFL)       S-type constant pointing to length value
BUFF DC ...
LEN EQU ...
BUFL DC H(..)
```

If TYPE=DSECT is specified, the following DSECT is produced:

```
DTSWRDD   DSECT ,
DTSWRDO   DS    F      address of output area
DTSWRDOL  DS    H      length of output data stream
DTSWRDOT  DS    CL2    type of write (TS, TI, RS, RI, WF)
DTSWRDI   DS    F      address of input area
DTSWRDIL  DS    H      length of input area
DTSWRDIT  DS    CL2    type of read ('RD')
DTSWRDND  DS    0C     end of table label
DTSWRDL   EQU   *-DTSWRDD
DTSWRDNR  EQU   0      normal return
DTSWRDLI  EQU   4      inp len incorrect short on storage
DTSWRDSR  EQU   8      program has to restore screen
DTSWRDSH  EQU   12     shutdown warning issued
DTSWRDDI  EQU   16     invalid output data stream
DTSWRDNS  EQU   20     EDS not supported
DTSWRDPI  EQU   28     parameter list invalid
DTSWRDDE  EQU   32     data stream error
DTSWRDMS  EQU   64     message suppressed
```

Register Usage

R0
Work register

R1
Contains the address of the parameter list.

R15
Return code register

Code
Meaning

X'00'
Normal return.

X'08'
Occurs only with WRTTYPE=RS or RI and indicates that the program must restore the screen, if necessary.

X'0C'
Indicates that the VSE/ICCF operator command /WARN has been issued. Register 15 contains this code each time DTSWRTRD is issued between /WARN and /WARN RESET or shutdown.

X'10'
Indicates one of the following:

- The structured field in the data stream which you submitted had an identifier other than Load Programmed Symbol, Set Reply Mode, or Read Partition.
- The Read Partition structured field was not the last in your outbound data stream.

- The length of the data stream that was submitted with WRTTYPE=WF was different from the length of the write buffer.
- The length of one of the structured fields in your outbound data stream has not been specified correctly.

X'14'
Occurs only with WRTTYPE=WF. Indicates that your terminal does not support the IBM 3270 Extended Data Stream Feature.

X'1C'
An invalid parameter list was specified.

X'20'
An error was found in the Write Structured Field request.

X'40'
Occurs only if WRTTYPE=TI or TS or WF and if the terminal supports the IBM 3270 Extended Data Stream feature. Indicates that the display of a message pending for the user is suppressed although the user requested automatic message display.

Restrictions and Special Considerations

1. A program that uses the full-screen service may not be called from within a procedure in such a way that it needs a second interactive partition. You can call the program from a procedure if the MULTEX option is off or by issuing /PEND prior to the /RUN or /EXEC request for that program.
2. When a program issues DTSWRTRD, then:
   - The interpretation of VSE/ICCF control characters, like line-end, is skipped.
   - PF key setting as well as scrolling through the print output from the program is skipped.
   - Special screen settings, like number of input lines, screen size specifications, and so on are ignored.
   - Hardcopy mode is ignored.
   - Continuous output mode has no effect.
   - No translation of input and output data is performed.
   - The messages `*ENTER DATA?` and `*PARTIAL END PRINT` are suppressed.
   - No scale line appears on the screen.
   - PA2 key causes /CANCEL AB to be processed.
   - PA1 key causes the OC exit to be entered.

   As soon as the program gets control back from DTSWRTRD, the above settings are in effect again as they were before the program issued DTSWRTRD. However, after DTSWRTRD has been processed, VSE/ICCF does not change the screen contents until the program issues a regular SYSLOG or SYSLST request. This means that, between successive DTSWRTRD requests, VSE/ICCF does not send a scale line to the screen, nor does it manipulate control settings of the terminal, such as keyboard restore.
3. The message

   ```
   *PARTIAL END PRINT BEFORE FULL SCREEN WRITE
   ```

   is issued immediately before the DTSWRTRD output data is displayed on the screen (in case there is other print output from SYSLOG and/or SYSLST in front of DTSWRTRD). Either press ENTER to get the DTSWRTRD output data, or issue any command that can be entered during execution spool print mode.
4. If light pen selection and attention is used, a program must always be prepared to accept the input from the terminal via ENTER, which means that it also receives the data of the modified fields.
5. A Write/Erase will be forced if one of the following occurs:
   - The CLEAR key is pressed,
   - The program issues the first full-screen write request,
   - The program changes the WRITTYPE specification from RS or RI to TS or TI, or vice versa.

6. If the automatic message display feature is set on and a message is pending for the user, the display of the message depends on the value specified in the WRTTYPE operand and on the terminal characteristics:
   - If the terminal does not support the IBM 3270 Extended Data Stream feature, the message is displayed immediately.
   - If the terminal supports the IBM 3270 Extended Data Stream feature and if WRTTYPE=TI|TS|WF is specified, the message is not displayed and return code X'40' is passed to the program.
   - If the terminal supports the IBM 3270 Extended Data Stream feature and if WRTTYPE=RI|RS is specified, the message is displayed. It is the program's responsibility to save and restore the screen contents.

Macro to Identify Terminal Characteristics (DTSSCRN)

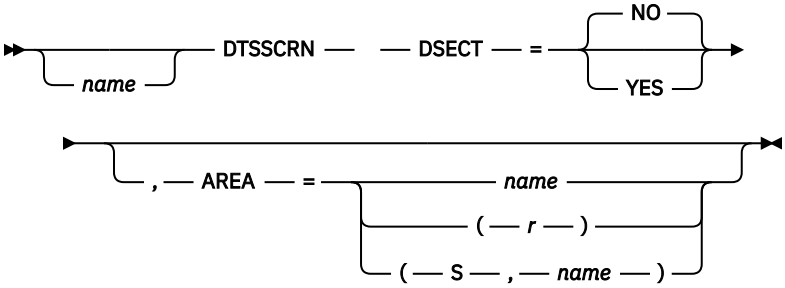
If a program running in an interactive partition communicates with a terminal, it needs to know certain characteristics of that terminal:

- If the program communicates with the terminal via I/O requests to SYSLOG and SYSLST, it needs to know the **logical screen size** which has been defined with the /SET SCREEN command. This size determines the number and the length of output lines which the program generates.
- If the program communicates with the terminal via the DTSWRTRD macro, it needs to know the **physical screen size** (the logical screen size is not considered by the DTSWRTRD macro).

To determine the logical and the physical screen size of a terminal, call the DTSSCRN macro in your program. The macro returns information about the following:

Size of the physical screen
Size of the logical screen
Terminal device type
Extended Data Stream features

The format of the macro is:



AREA
Specifies the address of an area into which VSE/ICCF will place the terminal characteristics. Define this area with a length of DTSCLEN. If the terminal does not have a device type of 327x or 329x, the whole area except DTSTYPE is set to X'00'.

DSECT=YES
DSECT=NO
DTSECT=YES causes a dummy section to be generated for the area that contains the terminal characteristics. The name for this area is taken from the macro specification. If no name was specified, the name defaults to DTSSCRN.

The dummy section has the following format:

```
DTSSCRN  DSECT  ,             SCREEN INFORMATION MAP
*               PHYSICAL SCREEN CHARACTERISTICS
DTSPHLNS DS     C                   NO. OF SCREEN LINES
DTSPHCLS DS     C                   NO. OF SCREEN COLUMNS
*               LOGICAL SCREEN CHARACTERISTICS
DTSLOINP DS     C                   NO. OF SCREEN INPUT LINES
```

```
DTSLOSOT DS     C                   STARTING LINE FOR OUTPUT
DTSLOOUT DS     C                   NO. OF SCREEN OUTPUT LINES
DTSLOSTR DS     C                   SCREEN STARTING COLUMN
DTSLOEND DS     C                   SCREEN ENDING COLUMN
*               TERMINAL DEVICE TYPE
DTSTYPE  DS     C                   TERMINAL DEVICE TYPE
DTSX3270 EQU    1                   327N, 328N, 3290, OR 5550
DTSX2741 EQU    2                   2741
DTSX2740 EQU    3                   2740,1050,....
DTSXANY  EQU    4                   ANY OTHER TERMINAL
                3270 EXTENDED DATA STREAM FEATURES
DTSC32EF DS     C                   3270 EXTENDED FEATURES
DTSCEDS  EQU    X'80'               EXT.DATA STREAM SUPPORTED
DTSCCOL  EQU    X'40'               COLOR SUPPORTED
DTSCPSS  EQU    X'20'               PROGRAMMED SYMBOLS SUPPORTED
DTSCHIL  EQU    X'10'               HIGHLIGHT SUPPORTED
DTSCVAL  EQU    X'08'               VALIDATION SUPPORTED
*
DTSC32E2 DS     C                   3270 EXTENDED FEATURES
DTSCFRL  EQU    X'80'               FIELD OUTLINING SUPPORTED
DTSCMIX  EQU    X'40'               MIXED FIELD SUPPORTED
DTSTIOAS DS     H                   TIOA-SIZE FOR SCREEN
DTSSCEND DS     OC                  END OF TABLE LABEL
DTSCLEN  EQU    DTSSCEND-DTSPHLNS   LENGTH OF DSECT
```

The terminal I/O area size, which the DTSSCRN macro returns in field DTSTIOAS is not the maximum buffer size (32700) that can be used for a single full-screen write to the terminal. For performance reasons, specify this size in the WRTBUFL operand.

Register usage:

R0
work register

R1
address of the terminal characteristics area.

Return codes: None

FORTRAN Compiler Considerations

Most of the language features of the FORTRAN compiler are available to the VSE/ICCF user.

Subroutines written in other languages (for example, assembler) can be called by a FORTRAN program. The FORTRAN program and its subprograms are loaded and linked together by using the LINKNGO utility.

Notes and Restrictions

The /LOAD OBJECT statement is optional. If an object deck is encountered in the job stream, the /LOAD OBJECT is implied.

RPG II Compiler Considerations

RPG* II is an easy-to-use specification-type language that you can use with VSE/ICCF. Together with the DSPLY function it enables you to create conversational programs.

The /OPTION statement is generally used to control compiler execution options. However, you can specify certain other options in the RPG II control card (H in column 6).

The *Language* and *User's Guide* manuals for RPG II contain the information that you need to develop and test RPG II programs.

An RPG II program can call or be called by other programs written in RPG II, PL/I, COBOL, or assembler language as defined in the RPG II language manual. These other programs can be compiled or assembled in the same job stream or in a different job stream and stored in the VSE/ICCF library file or in a VSE library.

Notes and Restrictions

1. The RPG II DSPLY statement can be used to write data to the terminal and to read data from the terminal. The file specification device field must be specified as CONSOLE for the DSPLY file.
2. Records to be read from the job stream can be read from a file setup with device type READ40 and symbolic unit SYSIPT or SYS005 (or equivalent local default).
3. Lines printed on the terminal can be written to a file setup with device type PRINTER and symbolic unit SYSLST or SYS006 (or equivalent local default).
4. Records to be placed in the punch area can be written on a file setup with device type READ40 and symbolic unit SYSPCH or SYS007 (or equivalent local default).
5. To read records from the punch area, you can define a file as device READ40 with symbolic unit SYS008 (or equivalent local default).
6. All unit record I/O (Notes 2 through 5) should be specified as single buffered (no 2 in column 32 of the file specification form) and fixed in length (column 19).
7. If you intend to use one of the temporary file areas, the file name on the file specification form should be IJSYS0n where n is any number from 0 to 4. The symbolic unit should be SYS00n. These file areas can be considered fixed or variable in format and have any block size permitted for the indicated device type. The device type must be specified as FBA, DISK14, DISK30, or DISK40. If preallocated work files are available within the system, these files need not be defined with /FILE statements.
8. The RPG II compiler uses IJSYS0l and IJSYS02 as work files. Thus, when transferring data from one step to another past intervening RPG II compilations, do not use these file areas unless you are dynamically allocating these areas for each execution as DISP=KEEP files (on the /FILE statement).
9. The relocation dictionary buffer for LINKNGO has been defined with a size of 1600 bytes. If you later find that very large RPG II programs are canceling or program checking, submit these jobs to batch for linkage by the VSE linkage editor program.

Submit-to-Batch Capability

If your location uses VSE/POWER and your user profile allows you to use the SUBMIT function, you can run jobs in VSE batch partitions. This VSE batch partition can be either a *static partition* or a *dynamic partition*.

Some reasons why you might want to submit to batch rather than run a program in an interactive partition are given below:

1. You want to use a program which cannot run in an interactive partition (for example LNKEDT, the VSE linkage editor).
2. You want to use a feature which is not supported by your VSE/ICCF (for example, multitasking or interval timer).
3. The job you want to run is not urgent and you are not in a hurry for the results.
4. The job you want to run is very long and would tie up a VSE/ICCF interactive partition for too long a time.
5. The job you want to run produces a lot of print output which would take time to display.
6. Your 3270 display terminal has no printer but you need a hardcopy listing of some type.
7. You want to punch out a member so it can be carried to a different location.
8. You want to link edit a multiphase program for being run in an interactive partition.

Your submit-to-batch job stream may consist of VSE/ICCF job entry statements or of VSE job control language. The print output from execution can be directed to the printer or held in a queue until you have viewed it. If your system is a VSE/POWER-controlled node of a network, you can transmit jobs for processing at other nodes in the network.

What Jobs May be Submitted

A library member consisting of either VSE JCL and VSE/POWER JECL or of VSE/ICCF job entry statements can be submitted for batch execution. A member that contains VSE/ICCF job entry statements gets those statements automatically converted to VSE JCL while they are being submitted. A member consisting of VSE job control statements can include also the VSE/ICCF /INCLUDE statement to avoid that all the data need be part of the job stream member.

A member that consists entirely of object statements (ESD, RLD, REP, END, TXT, or SYM) is treated in the same way as a job containing VSE/ICCF job entry statements and can be submitted successfully. If the member also has control statements (such as ACTION, INCLUDE, and PHASE), the member must be edited and a /LOAD OBJECT must be placed at the front of the deck to ensure proper execution after submission.

How to Submit a Job

The SUBMIT procedure is the basis for requesting the submit-to-batch function. The procedure has two basic options:

- The DIRECT option

  This is the default. The option causes the job to be submitted for batch execution and the output of the job to be routed to the printer.
- The RETURN option

  The option causes all print and punch output to be held in the output queue. You can display the print output and then dispose of it as you see fit. For possible actions, see the "/LISTP Command" on page 74 and "/STATUSP Command" on page 125, respectively. Only the submitter or the user designated in the DEST operand of the VSE/POWER JECL statement * $$ LST may access an output queue entry. An authorized user may specify ANY in the DEST operand (authorization is established in the VSE/ICCF user profile). This makes the output accessible to any user.

You can submit any type of VSE/POWER JECL with your job. However, defaults defined for your system may cause some of your JECL values to be overridden.

Your job stream may include an * $$ RDR statement. This allows you to include data from a diskette.

VSE/ICCF will allow more than one job to be run at a time, using the submit-to-batch function.

Submitting a Job for Output on the Printer

To submit a job for execution in a VSE partition and to have the output written to a system printer, enter:

```
SUBMIT name [DIRECT]
```

For name, specify the name of the VSE/ICCF member that contains the job to be submitted. The job inherits the member name and will be queued for processing. The job will run when the partition becomes available.

To test whether a job is waiting for processing, is being processed, or has completed, enter:

```
/STATUSP jobname
```

The /ERASEP command may be used to cancel the execution of a job before the job starts.

Submitting and Viewing of Output at a Terminal

To submit a job for processing in a VSE partition and to have the output stored in the VSE/POWER output queue so that you can view it later, enter:

```
 SUBMIT name RETURN
```

For name, specify the name of the VSE/ICCF member that contains the job to be submitted.

Use the /STATUSP command to see whether the job has been completed before you attempt to display the job output; else, wait until you are notified by VSE/POWER that processing for your job is finished. Ask for the completion message with the /MSG command or request automatic message display (/SET MSGAUTO ON).

To display the print output, enter the command:

```
    /LISTP jobname
```

The following commands (procedures) are available to control, view, and retrieve job output:

```
    /CANCEL             /HARDCPY
    /COMPRES            /LOCP
    /CONTINU            /SHIFT
    GETL                /SKIP
    GETP
```

When you have finished viewing the output, dispose of it; for example by:

- Erasing the output from the output queue; enter:

  ```
      /ERASEP jobname
  ```

- Routing the print output to the printer; enter:

  ```
  /ROUTEP LST jobname CLASS=class
  ```

  For class, specify an output class for which a VSE/POWER printer task was started.

- Routing the print output to a remote VSE/POWER RJE printer; enter:

  ```
      /ROUTEP LST jobname  REMOTE=remid
  ```

  For remid, specify the number of the remote work station to which the output is to be routed.

- Routing the print output to a 328x hardcopy printer if you are using an IBM 3270; enter:

  ```
      /HARDCPY printer-name
                    (Press the Clear key)
      /LISTP jobname
                    (Press the Clear key)
      /CONTINU
                    (Wait for completion)
                    (Press the Clear key)
      /HARDCPY OFF
  ```

  You can proceed with other work while the print output is being printed on the hardcopy device.

Note: When you issue VSE/POWER interface commands (such as /STATUSP, /LISTP, or /ROUTEP), you get messages from VSE/POWER. These messages are identified by the characters '1Q', '1R', and '1V' followed by a message code. For an explanation of these messages, see z/VSE Messages and Codes 1.

Submitting a Job for Transmission to Another Node

If your system is a node of a VSE/POWER controlled network (PNET), and if you are authorized to supply your own VSE/POWER job entry control language (JECL) statements, you can submit jobs for transmission to another node. To do this you must prepare your JECL to include the proper destination. For more information about the JECL statements, see VSE/POWER Administration and Operation

Receiving of List Output from Another Node

If you expect output from jobs that were not submitted from one of the VSE/ICCF terminals of your system, the following restrictions apply (in particular if the job was submitted at a non-VSE system):

- List queue entries containing non-displayable characters cannot be accessed by the /LISTP command or the GETL procedure. Non-displayable characters are contained in the following types of print records:

SCS print records
BMS mapping records
3270 format records
APA data records

- List queue entries with ASA or machine control characters are accepted up to a record length of 156 bytes. Longer records will be truncated.
- Only the submitter or the user designated in the DEST operand of the * $$ LST statement is allowed to access a list queue entry (the VSE/ICCF administrator has access to all queue entries, of course).
- Jobs submitted via the card reader are protected by an unprintable password if no password is provided. Only the VSE/ICCF administrator can retrieve output from such jobs.
- The job number in the list queue will be different from the job number of the job that created the list output.

Example

User AAAA, logged on to VSE/ICCF at the Node P2, wants to transmit a job for execution at node P1. Both P1 and P2 are VSE systems and nodes of a VSE/POWER controlled network (PNET). User AAAA, a VSE/ICCF administrator, may submit jobs, specify VSE/POWER JECL statements, and use the /CP command.

1. The VSE/POWER JECL statements define where the job is to be run and the destination to which the output is to be routed:

```
/list tranp1_
 ...+....1....+....2....+....3....+....4....+....5....+ .CM
*READY
_
 ...+....1....+....2....+....3....+....4....+....5....+.. .LS
* $$ JOB JNM=TRANP1,CLASS=0,DISP=D,XDEST=P1,NTFY=YES
* $$ LST CLASS=A,DISP=K,DEST=(P2,AAAA)
* $$ PUN CLASS=A,DISP=K
// JOB TRANP1
 . . .
/&
* $$ EOJ
```

   The XDEST operand in the * $$ JOB statement defines the node where the job is to be run. The DEST operand in the * $$ LST statement defines the node and the user to which the list output is to be routed. In this example, the DEST operand could have been omitted (as was done for the * $$ PUN statement) because the originator of the job is the default destination.

2. User AAAA submits the job for processing at node P1:

```
submit tranp1 return_
 ...+....1....+....2....+....3....+....4....+....5....+ .CM
*READY
_
 ...+....1....+....2....+....3....+....4....+....5....+.. .SP
***** START OF PROCEDURE (CLIST) *****
K889I JOB TRANP1 00333 SUCCESSFULLY SUBMITTED ...
*PARTIAL END PRINT
```

   This is a normal submission of the job TRANP1. All routing information is given in the VSE/POWER JECL statements. Job TRANP1 receives the job number 00333 at node P2.

3. VSE/POWER notifies user AAAA when the job has been transmitted.

```
_
 ...+....1....+....2....+....3....+....4....+....5....+ .CM
*READY
_
 ...+....1....+....2....+....3....+....4....+....5....+.. .CM
 02/15-14:39 MSG FROM VSE/POWER:
1RA0I JOB TRANP1  00333 TRANSMITTED TO P1 FOR P1
*END MSG
```

```
*READY
```

User AAAA gets the message automatically (by pressing ENTER from time to time) if automatic message display is in effect (the message display feature is discussed on page “MSGauto” on page 105).

4. User AAAA views the queue entry of the transmitted job in the reader queue of the target node, and transmits the appropriate command to the target node P1:

```
/cp pxmit p1,pdisplay rdr,cuser=aaaa_
 ...+....1....+....2....+....3....+....4....+....5....+ .CM
*READY
   ...+....1....+....2....+....3....+....4....+....5....+....6....+..  CM
02/14-14:40 MSG FROM P1: 1R46I READER QUEUE   P D C S CARDS
02/14-14:40 MSG FROM P1: 1R46I TRANP1   00157 3 D 0      10 FROM=P2(AAAA)
*END MSG
*READY
```

The target node responds with a series of messages, which might be mixed with other messages. Job TRANP1 receives the jobnumber 00157 at node P1. It waits for being processed.

5. The processing node notifies user AAAA when the job has been run. VSE/POWER at user AAAA's own node displays a message when the output was rerouted.

```
_
 ...+....1....+....2....+....3....+....4....+....5....+ .CM
*READY
_
 ...+....1....+....2....+....3....+....4....+....5....+....6... .CM
 02/15-14:52 MSG FROM P1:
1Q5DI  EXECUTION COMPLETED FOR TRANP1   00157 ON P1, TIME=14:52:48
 02/15-14:54 MSG FROM VSE/POWER:
1RB5I OUTPUT TRANP1  00334(00333) RECEIVED FROM P1  FOR P2
*END MSG
*READY
```

The output is given the new job number 00334 at node P2. The number of the job is given within parentheses (00333). The list output can now be displayed in the usual way.

Helpful Commands, Macros, and Procedures

The following commands, macros, and procedures can help you handle VSE/POWER jobs:

Commands

/CTLP
To issue a VSE/POWER command (for the VSE/ICCF administrator only).

/DQ
To display VSE/POWER queues.

/ERASEP
To erase a VSE/POWER queue element.

/LISTP
To display output data from the VSE/POWER LST queue.

/LOCP
To locate a character string in the displayed VSE/POWER list output.

/MSG
To review the VSE/POWER messages sent for you.

/ROUTEP
To route a VSE/POWER queue element.

/SET MSGAUTO ON
To get the incoming VSE/POWER messages displayed automatically.

/SKIP
To skip forward and backward in a VSE/POWER print file (in a /LISTP operation).

/STATUSP
To display the status of a VSE/POWER job.

Macros

RELIST
To transfer the contents of the print area or a library member to the printer.

Procedures

GETL
to retrieve VSE/POWER list queue output.

GETP
To retrieve VSE/POWER punch queue output.

GETR
To retrieve a VSE/POWER reader queue entry.

SUBMIT
To submit jobs to VSE/POWER.

Restrictions and Notes

1. The SUBMIT program accepts only * or // as the beginning of a VSE job. Anything else is considered VSE/ICCF JCL and handled accordingly. Place all of your ASSGN statements immediately behind the JOB statement, or use the PERM option of // ASSGN.
2. SUBMIT replaces `/DATA` by `/*`.
3. If the first data record after VSE/ICCF job entry statements starts with // followed by a blank, SUBMIT moves this record in front of the // EXEC statement of the generated job stream.
4. /LOAD DTSDUMMY in a job stream to be submitted is **not** replaced by a corresponding // EXEC DTSDUMMY statement.
5. The SUBMIT program translates VSE/ICCF JCL into VSE JCL if possible and necessary. `/OPTION GETVIS=nnn` is translated to

   ```
   // EXEC programname,SIZE=mmmK
   ```

   where mmmK is the default size of an interactive partition minus nnn. If nnn is greater than the size of the interactive partition, VSE/ICCF generates

   ```
   // EXEC programname,SIZE=AUTO
   ```

   The statement /OPTION GETVIS=P-nnn causes VSE/ICCF to generate

   ```
   // EXEC programname,SIZE=nnnK
   ```

6. Do not use any of the following names for members that contain jobs for submission to VSE/POWER: ALL, CANCEL, HOLD, FREE, RJE, LOCAL.

   Also, do not use names consisting of only one character. When a job with such a name is submitted to VSE/POWER, this name may conflict with a VSE/POWER operator command.

   You may want a member to be read by the SLI function of VSE/POWER for inclusion in a job. This requires that you store the member in your library by a name which conforms to the naming rules of VSE/POWER: a string of up to eight alphameric characters. An alphameric character is defined as:

   Any of the letters A through Z
   Any of the numerals 0 through 9

The number (#) sign
The dollar ($) sign
The commercial at (@) sign
The punctuation marks hyphen (-), period (.), and slash (/)

7. Columns 73 through 80 of VSE/POWER JECL records are not transferred by SUBMIT. When the submittor's OPTB bit 7 is on and columns 73 through 76 of a VSE/POWER JECL record contain $SLI, SUBMIT transfers the record with all 80 columns unchanged.
8. Because VSE/POWER transmits only 60 characters per message, its messages may be truncated. This may happen for any command listed under “Job Execution Under VSE/POWER” on page 11.
9. If the RETURN option is to be used, specify RBS=0 in the LST or PUN statements to suppress segmentation.
10. Comments on VSE/POWER JECL statements are not submitted to VSE/POWER.
11. Trying to assign a new class to a job by placing a '* $$ CTL' statement in the job stream will have no effect in a job that was submitted via the SUBMIT procedure; the statement is ignored.
12. If a continuation character is encountered in column 72 of a VSE/POWER JECL statement and no correct continuation statement follows, the continuation character is ignored.

Appendix A. Interactive Job Restrictions

The following restrictions apply to programs that are to be run in interactive partitions.

Access Methods and File Definitions

- The track-hold option (HOLD=YES) may not be used with any of the access methods SAM, DAM, and ISAM.
- BTAM-ES and VTAM* cannot run in an interactive partition, nor can a program in an interactive partition link to VTAM.
- A MICR/OCR device cannot be accessed by a program in an interactive partition.
- The support of system files (SYSFIL=YES) does not affect a program in an interactive partition because VSE/ICCF directs the I/O to these logical units, to the terminal, or to the VSE/ICCF library file.
- Except for the unit record devices for which I/O is intercepted by VSE/ICCF, assignments to physical devices must be done at or before VSE/ICCF startup. The assignments are valid as long as VSE/ICCF is up.

  The unit record devices for which VSE/ICCF intercepts I/O are:

  The card readers assigned to SYSIPT and (usually) SYS005
  The printers assigned to SYSLST and (usually) SYS006
  The punch units assigned to SYSPCH and (usually) SYS007
  The punch input file usually assigned to SYS008
  The console assigned to SYSLOG and (usually) SYS009

  This restriction may cause problems for a program that uses access methods other than those for disk. In disk access methods, OPEN automatically replaces the logical unit supplied in the program with the one supplied in the label information. If such a program is run in an interactive partition, the required assignment has to be setup at or before VSE/ICCF startup.

  VSE provides only one set of logical units (SYSnnn) for all interactive partitions, whereas each VSE partition has its own set of logical units. This means, if a program to be run in an interactive partition needs a dedicated logical unit assigned to a file, any other program that needs the same logical unit to be assigned to a different file cannot run in an interactive partition.

  If assignments have not been made, VSE/ICCF sets up dummy assignments for all unit record devices during startup. I/O to these devices is intercepted by VSE/ICCF; that is, to the system logical units and the corresponding default logical units that are defined as VSE/ICCF tailoring options (usually SYS005 through SYS009).

  A default logical unit can be used in a program for a different file only if the default assignment has been changed via the /ASSGN job entry statement.
- Blocking of records is ignored by VSE/ICCF for the unit-record devices for which the I/O is intercepted. VSE/ICCF assumes that a block contains only one logical record; only fixed length records are supported for these devices. VSE/ICCF assumes that the first byte of the record is the first byte of data.
- VSAM space management for SAM files can be used in interactive partitions by specifying TYPE=VSAM on the /FILE statement. These files should be obtained only via the /FILE statement. The RECORDS and RECSIZE operands of the DLBL JCL statement are not supported on the /FILE statement.
- Ensure that your file identifications are unique – for example by specifying the IDENT operand with the parameters &USR, &TRM, and &PRT Make use of VSE/ICCF's dynamic file management to obtain work-file space.
- For conversational reads from terminals, double buffering with overlap of I/O cannot be used; otherwise input and output messages could get out of sequence at the terminal.
- A DAM file that is to be opened by a program in an interactive partition can have up to 97 extents.

- The label information which VSE/ICCF builds from the /FILE job entry statements is written to the beginning of the interactive partition. OPEN reads the file label information from that area. The maximum number of files that you can specify via /FILE statements during one job is 28.
- VSE/ICCF does not provide any more protection against simultaneous access of files than VSE. Thus, data may get destroyed if programs in different interactive partitions access the same file at the same time. VSE/ICCF, however, provides support that allows each interactive partition to have its own set of work files.
- If jobs are to be run a number of times in an interactive partition, ASSGN and EXTENT statements should be specified in addition to the DLBL statement, for example for VSAM files. This prevents the partition from running out of LUBs, which are not freed at EOJ. Another safeguard against this situation is to specify the maximum number of LUBs using the job control command NPGR.
- For VSAM files the default value of GETVIS space is not sufficient. Define more with the GETVIS operand of the /OPTION statement.

Librarian and Linkage Editor

- The VSE Librarian program provides service functions for VSE libraries. VSE/ICCF includes macros that allow you to make use of these service functions from your terminal. The macros are:
  - LIBRC – It allows you to catalog members into a VSE library.
  - LIBRL – It allows you to list a member of a VSE library.
  - LIBRP – It allows you to transfer a member from a VSE library into a VSE/ICCF library.

  Because the macros cause the VSE Librarian program to be run, any restrictions for the VSE Librarian program apply also to these macros.
- The VSE linkage editor cannot run in an interactive partition.
- LSERV can run only in **one** interactive partition at a time.

Sort/Merge Program

The STORAGE=(n,VIRT) operand must be specified in the Sort/Merge OPTION control statement because page fixing (SVC 67) is not supported in an interactive partition.

Note: Supply labels for SORTWK1 and SORTWK2 if necessary. VSE/ICCF changes the file names to IKSYSP3 and IKSYSP4 before OPEN depending on the /OPTION PERMFILE setting.

Considerations for Interactive Partitions

- Of the 15 storage-protect keys available to VSE, VSE/ICCF can use, for protecting its interactive partitions, 15 minus the number of generated VSE partitions. Assuming that the VSE system has been generated with five VSE partitions, VSE/ICCF could then use up to 10 storage-protect keys for its interactive partitions. If VSE/ICCF had been tailored with 20 interactive partitions, then two interactive partitions would always get the same storage-protect key. VSE/ICCF makes sure that interactive partitions with the same storage-protect key are not located adjacent to each other.

  If two or more interactive partitions have the same storage-protect key, a program in one interactive partition can inadvertently damage a program in another with the same storage-protect key.
- Programs in interactive partitions run under control of VSE subtasks. Because there are not enough VSE subtasks available for all the programs that can be run concurrently in interactive partitions, VSE/ICCF performs rollin/rollout of programs on a time-slice basis.

  Rollout means that a subtask is taken away from a program; rollin means that a subtask is assigned to a program. The dispatching priorities are randomly assigned to the programs and usually change during the processing of the program.

  If sufficient VSE subtasks are available for interactive partition execution, VSE/ICCF does not do any rollin/rollout but leaves the subtasks assigned until the programs terminate. If in such an environment

a program with very little I/O activity gets a high dispatching priority, it will monopolize the total background processing of VSE/ICCF.

- The layout of interactive partitions is very similar to the layout of VSE batch partitions, except that up to 4K of space at the beginning of each interactive partition is reserved for VSE/ICCF use.

  You cannot have processor (real) storage allocated to an interactive partition to have sections of a program fixed in storage (by the PFIX and PFREE macros).
- The communication region of an interactive partition is pageable. A program gets the address of its communication region only by issuing the COMRG macro.

Supervisor Services

- The JOBCOM macro is not supported in interactive partitions.
- The PDUMP macro causes up to 8K (or until a print-area-full condition) to be printed to the terminal. Everything beyond this point is ignored.
- The DUMP and JDUMP macros cause the VSE/ICCF DUMP program to be invoked if it had been loaded into the SVA before VSE/ICCF was started. If the VSE/ICCF DUMP program has not been loaded into the SVA, the program is canceled without a dump. If /OPTION DUMP was specified, the work area for the VSE/ICCF DUMP program either is taken from the GETVIS area of the interactive partition or the last 3K bytes of that partition are used as work area.
- All VSE messages, except information (I) type messages supplied from SVA or B-transient routines, are routed to the terminal user requesting the service and to the console operator. Answers to such messages are read only from the operator console.
- The ASSIGN macro is supported in interactive partitions. However, temporary assignments are not reset automatically at the end of a job because they are made in the VSE batch partition. You are responsible for resetting these assignments. If temporary assignments are not reset, the pool of logical units could eventually become exhausted. This, in turn, makes new assignments impossible until CICS Transaction Server is shut down and started up again.
- If a B-transient service produces printer output and this output fills the requestor's print area, any additional output from that B-transient service is ignored.
- Checkpoint/restart is not supported for interactive partitions.
- Table 4 on page 335 shows the SVCs that can be used by a program running in an interactive partition. A program which issues SVCs that are not listed here might be cancelled or get unpredictable results. See z/VSE Systems Macro Reference for details.

Table 4. SVCs Supported in Interactive Partitions

SVC	Macro
0	EXCP
1	FETCH
2	
3	
4	LOAD
5	MVCOM
6	CANCEL
7	WAIT
8	
9	LBRET
10	SETIME

Table 4. SVCs Supported in Interactive Partitions (continued)

SVC	Macro
11	
12	
13	
14	EOJ
16	STXIT (PC)
17	EXIT (PC)
18	STXIT (IT)
19	EXIT (IT)
20	STXIT (OC)
21	EXIT (OC)
24	SETIME
26	
32	WTO, WTOR
33	COMRG
34	GETIME
37	STXIT (AB)
40	POST
44	
51	
52	TTIMER
56	
57	
60	
61	GETVIS
62	FREEVIS
63	
64	
65	CDLOAD
66	RUNMODE
75	SECTVAL
77	
78	CHAP
82	VSE/ICCF special
92	XECBTAB
93	XPOST
94	XWAIT

Table 4. SVCs Supported in Interactive Partitions (continued)	
SVC	**Macro**
98	EXTRACT
99	
101	
102	
103	
104	
105	
106	
107	
108	
109	
110	LOCK/UNLOCK
112	
113	XPCC

Note:

1. VSE/ICCF supports WTO/R issued by SVC 32 or by MVS simulated SVC 131 Code 35.
2. CANCEL ALL is changed by VSE/ICCF to CANCEL.
3. Fast CCW-translation is skipped for interactive partitions.
4. EXCP with the operand REAL is not supported in interactive partitions.
5. A program that uses VSE services or accesses supervisor tables, which are not described as official interfaces in z/VSE Systems Macro Reference might cause problems if it runs in an interactive partition. The program may run without problems in a VSE batch partition.
6. Applications of XPCC that require the use of WAITM (for example, multiple connections) cannot be used.
7. EXTRACT with ID=BDY and PID=pid is not supported in interactive partitions or CICS Transaction Server transactions.

Analysis of CCWs for Unit Record I/O

VSE/ICCF intercepts I/O requests to:

The card reader
The printer
The punch
The punch input file
The operator console

It routes these requests either to the terminal or to the VSE/ICCF library file. To do this, VSE/ICCF analyzes and transforms the CCW of each I/O request.

To avoid a complex CCW analysis routine, which would affect system performance, VSE/ICCF puts restrictions on the CCW chains that it intercepts. Usually these restrictions do not cause problems. However, if your program output looks unusual, you might have violated one of these restrictions. The following list explains these restrictions:

- Analysis of CCW Chains:

  VSE/ICCF generally assumes that each CCW in a chain is adjacent to the former one; that is, VSE/ICCF expects the next CCW to be located at an address which is 8 bytes higher than that of the last CCW. Status modifier commands are ignored with respect to the flow of control through the CCW chain.

  Depending on the chaining flags, VSE/ICCF continues analyzing the CCW chain. A TIC command always causes VSE/ICCF to proceed with the CCW at the address given in the TIC command. CCWs that are chained together via the data chaining flag are handled by VSE/ICCF as if they were CCWs with the same command code chained together via the command chaining flag.

  Command code checking is restricted. The analysis by VSE/ICCF is as follows:

  – For non-DASD channel programs:
    - If bit 6 of the command code is on, VSE/ICCF handles the CCW as read CCW.
    - If the bit 7 is on, VSE/ICCF assumes that the CCW is a write CCW.
    - If bits 6 and 7 are on, VSE/ICCF handles the CCW as control CCW.
    - The command code X'08' is handled as a TIC CCW.
    - VSE/ICCF ignores the modifier bits of the command code.
  – For DASD channel programs (beginning with command code X'07'):
    - Command codes X'05', X'0D', and X'ID' are treated as writes.
    - Command codes X'06', X'0E' and X'IE' are treated as reads.
    - Command code X'08' is handled as a TIC CCW.

  VSE/ICCF ignores commands that do not fall into the above categories. It also ignores commands that are not meaningful for the logical unit; for example, read from SYSLST. VSE/ICCF does not issue a message nor is the I/O request canceled. If such a CCW has a chaining flag on, VSE/ICCF proceeds with the next CCW following the invalid one.
- Write Requests to SYSLST or the Corresponding SYSnnn:

  The maximum number of bytes you can write to your print area with one write CCW is 156. If the data length is longer, the remainder is ignored.

  If you use DTFDI with RECSIZE not equal to 121, VSE/ICCF does not strip off the first character (which is supposed to be the printer control character). This character therefore appears in your print area as data. If you issue your own EXCP requests to the printer, be careful with data lengths of 121, 129, and 133. If the data length is 129, VSE/ICCF strips off the first 9 bytes; if this length is 121 or 133, VSE/ICCF strips off the first byte.

  VSE/ICCF can save the control information of only one command per write request. Other control information gets lost. VSE/ICCF uses the control information to build the CCW chain needed for the central printer. If the command code of a write CCW is "write without spacing," VSE/ICCF saves the command code of the most recent control command it encountered in the CCW chain. If the write CCW is not a "write without spacing," VSE/ICCF usually saves the command code of the write CCW. However, if the most recent control command is a "skip to channel 1" request, VSE/ICCF saves a special code to ensure the generation of the skip-to-channel-1 request in addition to the original write request. For the subsequent write CCWs, VSE/ICCF assumes the most recent control command to be "space 1 line" until it encounters another control command. VSE/ICCF does no error checking for valid printer control commands during CCW analysis.

  VSE/ICCF can build DASD and tape channel programs for print output to SYSLST or the corresponding SYSnnn. However, VSE/ICCF does not transfer printer control information to its channel-program build routine in a meaningful manner.

  The device type codes of the 3800 Printing Subsystem should not be specified because its I/O module buffers the printer output and displays it only when the buffer is full or at close time.
- Read/Write Requests to and from SYSLOG or the Corresponding SYSnnn:

Write requests go to your print area in the same way as they go to SYSLST. They are therefore handled in the same way as write requests to SYSLST.

Independent from the data length you specify in a read CCW, VSE/ICCF always accepts up to 256 characters as input from the terminal. If the input data length from the terminal is longer than that specified in the read CCW, VSE/ICCF stores the difference as residual count to the corresponding CCB and moves the data up to the length specified in the read CCW to the I/O area of your program. If the input data length from the terminal is smaller than that specified in the read CCW, VSE/ICCF sets the residual count to 0 and moves the data up to the length of the input from the terminal to the I/O area of your program. The remainder of the I/O area remains unchanged.

It is meaningful to specify a data length of only up to 256 in the read CCW.

- Read Requests from SYSIPT or the Corresponding SYSnnn:

  If you specified /DATA INCON for your program, which means that a read request from SYSIPT or the corresponding SYSnnn is directed to the terminal, the same rules apply as described for read requests from SYSLOG, except that the data moved to the I/O area of your program can never be larger than 80 bytes. If the input data length from the terminal is larger than 80 bytes, the residual count in the corresponding CCB is set to 1. If the input data length is smaller than 80, the residual count is set to 0.

- Print and Punch Area, and 'SYSPCH to a Member':

  The print and punch areas are allocated by VSE/ICCF on request in multiples of 100 records. Before anything is put into these records, VSE/ICCF initializes them with /* */ characters. These are overridden one after the other by the data you write to the area. If your output needs a number of records not equal to (n*100 - 1), the remainder of the area still contains the initialization characters. Note that data to SYSLST or the corresponding SYSnnn with a data length larger than 78 requires 2 records from the print area.

  If you punch out data to an existing member, the data first overrides the existing member. When all records of the existing member have been used up, then the same allocation and initialization mechanism takes place as for the punch area. This means that, if your punch output consists of more records than the existing member, you may find /* */ records at the end of that output. However, if your punch output consists of fewer records than the existing member, you will find that the old information (from the existing member you punched to) has been deleted.

Appendix B. Context Editor

The context editor, a general purpose text editing program, has functions similar to those of the full-screen editor. The context editor allows you to create and modify files from your display terminal. A file can be a member of your library or data in the input area.

The main difference between the context editor and the full-screen editor is the following:

- When you work with the full-screen editor, you can enter commands in the command area and in the line-command areas; you can enter data anywhere on the screen.
- When you work with the context editor, you can enter commands only on the command input line. The context editor does not include a split-screen function.

For more information about the full-screen editor, see Chapter 4, “General Information about the Editor,” on page 133.

The command sets of the two editors are similar. However, some commands are valid only under the full-screen editor while others are useful only under the context editor.

The context editor must be used if you:

- Work with a 274x terminal.
- Write procedures and macros that perform editing functions.

As with the full-screen editor, all context editing commands take effect immediately.

If you use a display terminal, your position within the member being edited is indicated by a scale line across the middle of the screen. The current line is the one directly below the scale line. You select the number of lines (1-20) that you want to appear before the current line and the number of lines (1-20) that you want to appear after it. These values are selectable during editing. All lines below the scale line are displayed with high intensity. You can turn full-screen verification on or off.

By placing sequence numbers within a member, you can use line number editing as well as context editing. To find a given line in line number editing, you simply type in the line number. To replace a current line or to add a new line, type in the line number followed by the data. The editor automatically positions you to the specified line in the file and makes the requested change.

Line number editing and normal context editing can be used on the same member concurrently. Some hints are included in this appendix and in sections “LINEMODE Command” on page 197 and “nn Command” on page 229.

For editing a large member, you can create an index for the member. This index enables you to move the line pointer quickly. When line number editing is in effect with indexing, finding a certain line in the member is done using the index. Moving and copying data from one part of a member to another is done by the @MOVE and @COPY macros. For example,

```
    @COPY 10 LOC /IDEA/
```

would cause the ten lines at the current position to be copied behind the first line where the string IDEA occurred.

Invoking the Context Editor

You invoke the context editor by issuing the /EDIT command, which places your terminal into context editor ('ED') mode. To edit a library member, specify the member name, followed by the member password if required. Format of the command:

```
    /EDIT membername [passwd]
```

To edit the input area, enter the /EDIT command without member name.

Editing Modes

As with the full-screen editor, you can work with the editor in two basic modes:

- edit mode
- editor-input mode

After having invoked the context editor, your terminal is in context editor mode. You can enter commands to move the line pointer and to locate, insert, add, or replace data. The INPUT command puts your terminal into editor-input mode. Editor input mode is indicated by the message `MORE INPUT?`.

In editor-input mode, you can only enter data. Each line that you enter is added to the file at the position of the line pointer. One 80-character line is created from each input line. To insert a blank line in a file, type at least one space and press ENTER (or the carriage-return key).

You end editor-input mode by entering a null line. The message `INPUT MODE ENDED` indicates that you are back in edit mode.

If a null line is entered in edit mode, it has the same effect as the NEXT command.

Response Modes

There are three editing response modes (or verification modes), which are controlled by the VERIFY command:

- Verify response
- Brief response
- Long verify response

Verify is the normal mode. It automatically displays each line that has been changed or found as a result of a command.

Brief mode does not display the specified lines, so that you must issue a PRINT command to obtain a display.

The long verify mode applies only to 3270 terminals and is most useful in a local 3270 environment. In this mode, the output area of the screen contains a number of lines before and after the line pointer. A scale line indicates the location of the line pointer. The number of lines before and after the scale line can be varied via operands of the VERIFY command.

Long verify mode is automatically set for local 3270 terminals. If the long verify display is lost, it can be restored by entering VERIFY with no operands.

Certain messages are always printed even if you have selected brief message mode.

Summary of Editor Commands Valid in Context Editing

Figure 26 on page 343 lists the editor commands that are valid under the context editor. Most of these commands are described in Chapter 4, “General Information about the Editor,” on page 133. Others are equivalents of system commands with the same name, for example: LIBRARY is the same as the system command /LIBRARY. In these cases, the descriptions in Chapter 3, “System Commands, Procedures, and Macros,” on page 31 apply.

The editor commands basically work the same way under both editors. If a difference exists for a command, this command is flagged, and the section “Specifics About Context Editing” on page 344 gives explanations.

Command	Function
Add	Adds data beyond the end of data within the zone.
ALIgn	Aligns data in the zone to right and left margins.
ALter	Changes a single character to another character in one or more lines.
BAckward*	Moves the line pointer upward a given number of lines; see UP command.
BLank	Blanks out characters in the current line.
Bottom	Moves the line pointer past end of file.
BRief*	See the VERIFY command.
case	controls upper- /lowercase input data translation.
CENter	Centers the data within the current zone.
Change*	Changes one character string to another in one or more lines.
CTL	Sets various VSE/ICCF options; works like the /SET system command.
DELete	Deletes one or more lines.
DELIM	Defines the delimiter character.
DOwn	Moves the line pointer forward a given number of lines; see the NEXT command.
DUP	Duplicates the current line a specified number of times.
ECHO	Writes a message to the terminal; works like the /ECHO system command.
END	Terminates edit mode; see the QUIT command.
FILe*	Saves the file and terminates the edit session.
Find	Performs a column dependent search for specified data.
FLag	Sets change flagging on or off.
FOrward*	Moves the line pointer forward a given number of lines.
GETfile	Copies data from another member or work area into the file that you are editing.
HARdcpy	Places the terminal in hardcopy mode and directs terminal output to a hardcopy printer or to a private destination; works like the /HARDCPY system command.
IMage*	Sets backspace/tab transparency on or off.
INDex	Builds an index of the current file for rapid editing.
INPut*	Sets the editor into input mode for entry of multiple lines of data.
Insert*	Inserts a single line of data into the file.
JUStify	Left or right justifies data within the zone.
LIBRARY	Displays the member names of a library; works like the /LIBRARY system command.
LINemode*	Sets line number editing on or off.
LN	See LOCATE command.
Locate	Locates a string of characters within a file.
LOCNot	Locates the first nonoccurrence of a string; see LOCATE command.
LOCUp	See LOCATE command.
LUp	See LOCATE command.
MSG	Displays messages that have been sent to the terminal; works like the /MSG system command.
Next	Moves the line pointer forward a given number of lines.
Overlay	Overlays the current line with the characters specified in the operand field of the command.
OVERLAYX	Overlays the current line with hexadecimal format data.
OX	See OVERLAYX command.
PF	See PRINT command.
PFnn	Invokes the function associated with PF key 'nn'; works like the /PFnn system command.
POint	Positions the line pointer based on an established index.
Print	Displays a given number of lines.
PRINTFwd	See PRINT command.
PROmpt	Sets or displays prompt increment for line number editing.
Quit	Terminates edit mode.
RENum	Puts new sequence numbers into the file being edited.
REPEAT	Executes a subsequent OVERLAY, BLANK, ALIGN, CENTER, JUSTIFY or SHIFT command a given number of times.
REPlace*	Replaces a library member with all or part of the input area, or with a newly created file; works like the /REPLACE system command.
REStore	Restores various editor settings from previous STACK EDIT.
Rewrite*	Replaces the current line with new data.
RPT	See REPEAT command.
SAve*	Saves the contents of the input area or of a newly created file; works like the /SAVE system command.
Search	Searches the file from the top for a given string of characters.
SET*	Sets various VSE/ICCF options; works like the /SET system command.
SHIft	Shifts data within the zone left or right a given number of columns.
SHow*	Displays the setting of all VSE/ICCF options that can be controlled by the SET command; works like the /SHOW system command.
SPlit	Splits the current line into two lines.
STACk	Places data or commands in the editor stack.
STATUS	Displays the setting of all VSE/ICCF options that can be controlled by the SET command; works like the /SHOW system command.
TABset	Establishes logical tab settings; works like the /TABSET system command.
Top	Positions the line pointer to the null line in front of the first line in the file.
TYpe	Displays a given number of lines; same as the PRINT command.
Up	Moves the line pointer upward a given number of lines.
Verify*	Sets verifications on or off; or sets 3270 full-screen verify; or displays the current line.
Zone	Sets the current zone for editing, or scanning operations.
'nn'	Replaces or locates lines by sequence number.

Figure 26. Editor Commands in Context Editing

Specifics About Context Editing

The editor commands flagged in the above table by an asterisk either work differently under the context editor (as opposed to the full-screen editor), or they are not relevant for full-screen editing. The following sections explain these specifics.

BACKWARD Command

The BACKWARD command in context editing works the same way as the UP command. You can move the line pointer upward a given number of lines. The command does not allow you to scroll upward a specified number of logical pages as in full-screen editing.

BRIEF Command

Refer to "VERIFY Command" on page 347.

CHANGE Command

If you enter this command with no operands instead of moving the cursor to the current line of the screen, the record pointed to by the line pointer appears in the input line of the display. After that, the system offers you several facilities:

- You can alter the contents of the input line by retyping part or all of the line, or by using the Insert, Delete, or ERASE-EOF keys to insert or delete characters.
- When the line is modified, press ENTER. This causes the record on the input line to replace the current line in the output display area.
- If the column suffix is used, the cursor is placed at the location indicated by the column suffix.
- If you do not want to change the line, press the ERASE INPUT key and then the ENTER key and the line will not be changed.

Note: A CHANGE command without operands is invalid at a typewriter terminal.

FILE Command

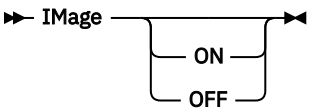
Under the context editor, the command allows you to save **part** of the input area as a library member. This is not possible in full-screen editing. The syntax of the FILE command is the same as for the /SAVE system command (see "[/]SAVE Command" on page 96).

FORWARD Command

The FORWARD command in context editing works the same way as the NEXT command (see "NEXT Command" on page 202). You can move the line pointer forward a given number of lines. The command does not allow you to scroll forward a specified number of logical pages as in full-screen editing.

IMAGE Command

The IMAGE command is used to control how the context editor handles backspace and tab characters within commands or input data.



ON
Causes backspaces or tabs to be removed according to their logical functions. A backspace causes the previous character to be logically overlaid. A tab causes blanks to be copied according to predefined logical tab stops.

OFF
Causes tab and backspace characters to be treated like other characters.

If no operand is specified, the current IMAGE setting is displayed. The command applies only to the current editing session.

Example

The example below assumes that the character # is defined as the backspace character.

```
request:      image on
              insert xxx###zzz
              print

response:     ZZZ

request:      image off
              insert xxx###zzz
              print

response:     xxx###zzz
```

Note: The IMAGE command is available also in full-screen editing, but it has no practical use there.

INPUT Command

In context editing, the command has the same effect as in full-screen editing: the terminal changes from edit mode to editor-input mode. However there are some differences:

- When line number editing is in effect, sequence numbers can be made part of all new lines added to the file (see “PROMPT Command” on page 207). In full-screen editing, the sequence numbers may be overwritten by data.
- If in context editing, you leave input mode without having entered a line, the line pointer remains at the position that it had just before editor-input mode was entered. In full-screen editing, the line pointer is advanced by one line.
- When using the 3270, you can press the Cancel (PA2) key [10] to return from editor-input to edit mode. If the PA2 key is pressed a second time, it has the same effect as the QUIT command (see “QUIT Command” on page 208). Edit mode is terminated and command mode is entered.

INSERT Command

As under the full-screen editor, the command is used to insert a string of characters into the file without entering editor-input mode.

In context editing the INSERT command without operands is invalid, and the message `INVALID OPERAND` is displayed. In full-screen editing, the INSERT command has the same effect as the INPUT command.

For more information about the two commands, see “INPUT Command” on page 193 and “INSERT Command” on page 193.

Note: If you are using line number editing, the insert string will receive a sequence number; under the full-screen editor this does not happen.

LINEMODE Command

Line-mode editing with the context editor differs from this type of editing with the full-screen editor as follows:

- Under the context editor, you are prompted with a line number to enter each line.
- Under the full-screen editor in input mode, the columns occupied by the sequence number should not be used as part of the input lines. Otherwise the sequence number will be overwritten.

[10] The definition of the Cancel key is a VSE/ICCF tailoring option. The default is PA2. It may be different for your system.

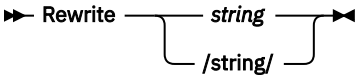
REPLACE Command

In context editing the command works the same way as the /REPLACE system command. You can replace an existing library member with only a portion of the input area. If only a portion of the input area replaces the existing library member, the portion not participating in the replace will remain in the input area.

For the syntax of the REPLACE command, see "[/]REPLACE Command" on page 86.

REWRITE Command

The command is used to replace the current line with a specified string of characters.



string
Is an input line that replaces the current line.

/
Is a delimiter character.

The column suffix can be used with this command.

The logical tab character, the tab key, and the logical backspace character (subject to the IMAGE setting) can be used in 'string'. 'String' is separated from the command by only one blank unless the delimiter characters are used.

The line pointer is not advanced by this request.

The REWRITE command is valid when line number editing is in effect as long as the sequence number field begins in column 1, 73 or beyond.

The keyboard is unlocked. For 3270 terminals, the replacement line is displayed.

Examples

1. R ¬IREG = J + K**2 (where ¬ is the tab character)

```
Line before the request:       IRE555 = 1 - K
Request:                 r ¬ireg = j + k**2
Line after the request:        IREG = J + K**2
```

   The string specified with the request replaces the current line. Assuming that the logical tabs are set for a FORTRAN file type, the statement IREG = J + K**2 begins in column 7.

2. RC16 THIS IS THE NEW LINE

```
Line before the request: THIS IS THE ORIGINAL LINE
Request:                 rc16 this is the new line
Line after the request:                  THIS IS THE NEW LINE
```

   The specified string becomes the new line beginning at column 16. Columns 1 through 15 are replaced by blanks.

Note: The REPLACE command is available also in full-screen editing, but it has no practical use there.

SAVE Command

While the terminal is in context edit mode, the SAVE command has the same effect as the /SAVE system command.

The differences between the two editors are as follows:

- In context editing you are able to save only a part of the contents of the input area. All lines not saved remain in the input area, and all lines saved are no longer in the input area at the end of the SAVE operation.

- Under the context editor, the 'name' operand must be specified.

For further explanations, see “[/]SAVE Command” on page 96.

SET Command

The SET command offers the same functions under both editors. However, the following options are not available in context editing:

NULls
NUMbers
PFC
REPoption

For a full discussion of the SET command, see “[/]SET Command” on page 100 and “SET Command” on page 216.

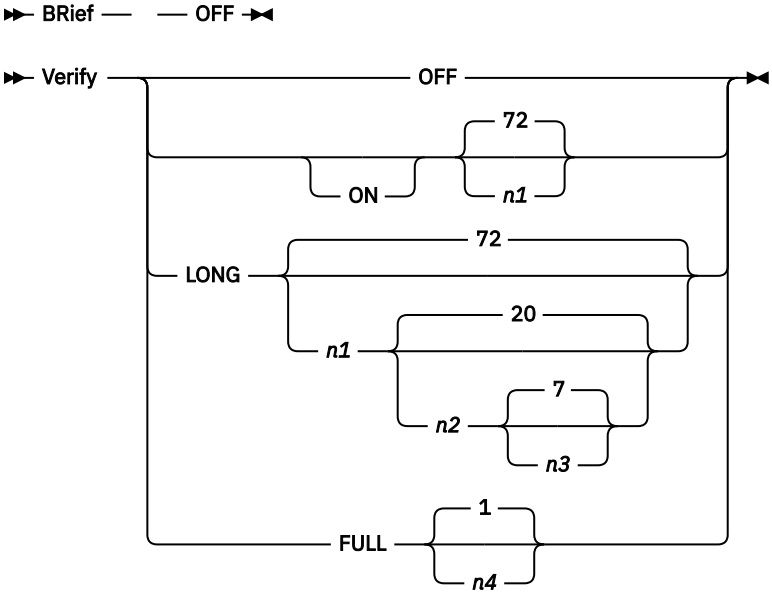
SHOW Command

In context editing the command does not accept the NAMES operand. This is possible only in full-screen editing.

VERIFY Command

The VERIFY command controls what is displayed while you are entering editor commands that modify lines within the file.

The BRIEF command has no effect when used during full-screen editing.



n1
Is a decimal number from 1 to 80 indicating the number of columns of data to be displayed when verification mode (the opposite of brief mode) is in effect. The default is the current line size value at the time the editor is entered, which is usually 72.

n2
Is a decimal number indicating the total number of lines to be displayed on the screen when LONG verify mode is set. The values for the various screens are:

Type and Model	Range	Default
3278 Model 3	5 to 28	28
3278 Model 4	5 to 39	39

```
3278 Model 5         5 to 23          23
All others           5 to 20          20
```

n3
Is a decimal number from 1 to n2 indicating the number of lines (of the total displayed in LONG verify mode) which are to be displayed before the current line. (The scale line which precedes the current line is included in this value.)

n4
Is a decimal number from 1 to 16 indicating the number of lines prior to and including the current line to be displayed on the full-screen edit or logical screen.

There are three basic verification modes:

BRief mode
Changed lines are not displayed.

Verify mode
Changed lines are displayed. This is the default for hardcopy terminals.

Long verify mode (IBM 3270 only)
The specified or defaulted number of lines before and after the current line are displayed. This is the default for 3270 terminals.

If you specify one of:

```
VERIFY
VERIFY [ON] nn
```

then verify mode is set if brief mode was in effect (set by the BRIEF or VERIFY OFF commands). If verify mode or long verify mode was in effect, that mode is retained. The editor writes the verify information to the terminal as follows:

- If you specify nn

  The editor sets the width of the verification display to the number of columns you specified for nn.
- If you do not specify nn

  The editor sets the width of the verification display to the value you specified for nn in a previous VERIFY command, if there was one, or to the default value (setup in your profile or by a /SET LINESIZE).

Either of the above forms sets the verify mode, causing the automatic display of lines changed or searched for by other edit commands.

If you specify one of the following (for IBM 3270s only):

```
VERIFY LONG
VERIFY LONG n1 [n2 n3]
```

then long verify mode is set if brief or normal verify mode was in effect. If long verify mode was already in effect, this mode is retained. The editor displays the long-verify screen.

If you specify any of the additional operands (n1, n2, n3), the editor sets the width and length of the verify screen, and also its start line, to the specified values. If you do not specify these operands, the editor sets the verify screen to the values you specified in the preceding VERIFY LONG, if there was one, or uses the applicable default values.

If you specify either of:

```
VERIFY OFF
BRIEF
```

and verify long mode was in effect, then:

1. The first VERIFY OFF or BRIEF command causes verify mode to be entered.
2. The second VERIFY OFF or BRIEF command causes brief mode to be entered.

The current line (or long verify screen) can always be redisplayed by issuing the VERIFY command with no operands.

In brief mode, lines that are changed in the file are not displayed. The mode affects commands such as BLANK, CHANGE, FIND, LOCATE, SEARCH, NEXT, OVERLAY and UP.

VERIFY FULL initiates the full-screen editor and, optionally, sets the number of lines to be displayed prior to and including the current line. It is not possible to invoke the full-screen editor in a procedure.

Examples

1. Stop context editing and start full-screen editing for the member being edited.

```
        verify full 5
```

2. Set verify mode and cause 60 columns of the current line to be displayed after location and change commands.

```
        verify 60
```

3. Set long verify mode with 12 total lines, 5 of which are lines prior to the current line.

```
        verify long 72 12 5
```

4. CHange command in Verify and BRief Modes. Line before the request:

```
        020 INPUT I,J,K

    Request in Verify Mode:     ch /j,k/k,l
    Response:                   020 INPUT I,K,L

    Request in BRief Mode:      ch /k,l/m,n
    Response:                   (none)
```

Appendix C. Understanding Syntax Diagrams

This section describes how to read the syntax diagrams.

To read a syntax diagram follow the path of the line. Read from left to right and top to bottom.

- The ▶▶— symbol indicates the beginning of a syntax diagram.
- The —▶ symbol, at the end of a line, indicates that the syntax diagram continues on the next line.
- The ▶— symbol, at the beginning of a line, indicates that a syntax diagram continues from the previous line.
- The —▶◀ symbol indicates the end of a syntax diagram.

Syntax items (for example, a keyword or variable) can be:

- Directly on the line (required)
- Above the line (default)
- Below the line (optional)

Uppercase Letters

Uppercase letters denote the shortest possible abbreviation. If an item appears entirely in uppercase letters, it cannot be abbreviated.

You can type the item in uppercase letters, lowercase letters, or any combination. For example:

In this example, you can enter KEYWO, KEYWOR, or KEYWORD in any combination of uppercase and lowercase letters.

Symbols

You must code these symbols exactly as they appear in the syntax diagram.

*****
Asterisk

:
Colon

,
Comma

=
Equal sign

-
Hyphen

//
Double slash

()
Parenthesis

.
Period

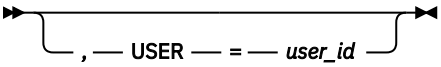
+
Add

For example:

```
* $$ LST
```

Variables

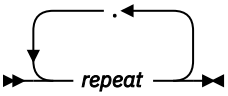
Highlighted lowercase letters denote variable information that you must substitute with specific information. For example:



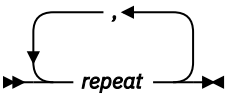
Here you must code USER= as shown and supply an ID for user_id. You can enter USER in lowercase, but you should not change it otherwise.

Repetition

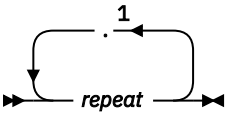
An arrow returning to the left means that the item can be repeated.



A character within the arrow means you must separate repeated items with that character.



A footnote (1) by the arrow references a limit that tells how many times the item can be repeated.

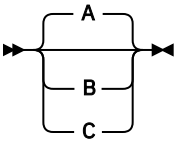


Notes:

[1] Specify *repeat* up to five times.

Defaults

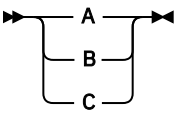
Defaults are above the line. The system uses the default unless you override it. You can override the default by coding an option from the stack below the line. For example:



In this example, A is the default. You can override A by choosing B or C.

Required Choices

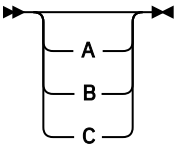
When two or more items are in a stack and one of them is on the line, you must specify one item. For example:



Here you must enter either A or B or C.

Optional Choice

When an item is below the line, the item is optional. You can only choose one item. For example:



Here you can enter either A or B or C, or omit the field.

Required Blank Space

A required blank space is indicated as such in the notation. For example:

```
* $$ EOJ
```

This indicates that at least one blank is required before and after the characters $$.

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Programming Interface Information

This publication documents intended Programming Interfaces that allow the customer to write programs to obtain services of z/VSE.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IPv6/VSE is a registered trademark of Barnard Software, Inc.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Terms and Conditions for Product Documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein. IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed. You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/VSE enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using Assistive Technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/VSE. Consult the assistive technology documentation for specific information when using such products to access z/VSE interfaces.

Documentation Format

The publications for this product are in Adobe Portable Document Format (PDF) and should be compliant with accessibility standards. If you experience difficulties when you use the PDF files and want to request a web-based format for a publication, you can either write an email to s390id@de.ibm.com, or use the Reader Comment Form in the back of this publication or direct your mail to the following address:

```
IBM Deutschland Research & Development GmbH
Department 3282
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany
```

In the request, be sure to include the publication number and title.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Glossary

This glossary includes terms and definitions for IBM z/VSE.

The following cross-references are used in this glossary:

1. See refers the reader from a term to a preferred synonym, or from an acronym or abbreviation to the defined full form.
2. See also refers the reader to a related or contrasting term.

A

Access Control Logging and Reporting

An IBM licensed program to log all attempts of access to protected data and to print selected formatted reports on such attempts.

access control table (DTSECTAB)

A table that is used by the system to verify a user's right to access a certain resource.

access list

A table in which each entry specifies an address space or data space that a program can reference.

access method

A program, that is, a set of commands (macros) to define files or addresses and to move data to and from them; for example VSE/VSAM or VTAM.

account file

A disk file that is maintained by VSE/POWER containing accounting information that is generated by VSE/POWER and the programs running under VSE/POWER.

addressing mode (AMODE)

A program attribute that refers to the address length that a program is prepared to handle on entry. Addresses can be either 24 bits, 31 bits, or 64 bits in length. In 24 bit addressing mode, the processor treats all virtual addresses as 24-bit values; in 31 bit addressing mode, the processor treats all virtual addresses as 31-bit values and in 64-bit addressing mode, the processor treats all virtual addresses as 64-bit values. Programs with an addressing mode of ANY can receive control in either 24 bit or 31 bit addressing mode. 64 bit addressing mode cannot be used as program attribute.

administration console

In z/VSE, one or more consoles that receive all system messages, except for those that are directed to one particular console. Contrast this with the user console, which receives only those messages that are directed to it, for example messages that are issued from a job that was submitted with the request to echo its messages to that console. The operator of an administration console can reply to all outstanding messages and enter all system commands.

alternate block

On an FBA disk, a block that is designated to contain data in place of a defective block.

alternate index

In systems with VSE/VSAM, the index entries of a given base cluster that is organized by an alternate key, that is, a key other than the prime key of the base cluster. For example, a personnel file preliminary ordered by names can be indexed also by department number.

alternate library

An interactively accessible library that can be accessed from a terminal when the user of that terminal issues a connect or switch library request.

alternate track

A library, which becomes accessible from a terminal when the user of that terminal issues a connect or switch (library) request.

AMODE

Addressing mode.

APA

All points addressable.

APAR

Authorized Program Analysis Report.

appendage routine

A piece of code that is physically located in a program or subsystem, but logically and extension of a supervisor routine.

application profile

A control block in which the system stores the characteristics of one or more application programs.

application program

A program that is written for or by a user that applies directly to the user's work, such as a program that does inventory control or payroll. See also batch program and online application program.

AR/GPR

Access register and general-purpose register pair.

ASC mode

Address space control mode.

ASI (automated system initialization) procedure

A set of control statements, which specifies values for an automatic system initialization.

attention routine (AR)

A routine of the system that receives control when the operator presses the Attention key. The routine sets up the console for the input of a command, reads the command, and initiates the system service that is requested by the command.

automated system initialization (ASI)

A function that allows control information for system startup to be cataloged for automatic retrieval during system startup.

autostart

A facility that starts VSE/POWER with little or no operator involvement.

auxiliary storage

Addressable storage that is not part of the processor, for example storage on a disk unit. Synonymous with external storage.

B

B-transient

A phase with a name beginning with $$B and running in the Logical Transient Area (LTA). Such a phase is activated by special supervisor calls.

bar

2 GigyByte (GB) line

basic telecommunications access method (BTAM)

An access method that permits read and write communication with remote devices. BTAM is not supported on z/VSE.

BIG-DASD

A subtype of Large DASD that has a capacity of more than 64 K tracks and uses up to 10017 cylinders of the disk.

block

Usually, a block consists of several records of a file that are transmitted as a unit. But if records are very large, a block can also be part of a record only. On an FBA disk, a block is a string of 512 bytes of data. See also a control block.

block group

In VSE/POWER, the basic organizational unit for fixed-block architecture (FBA) devices. Each block group consists of a number of 'units of transfer' or blocks.

C

CA splitting

Is the host part of the VSE JavaBeans, and is started using the job STARTVCS, which is placed in the reader queue during installation of z/VSE. Runs by default in dynamic class R. In VSE/VSAM, to double a control area dynamically and distribute its CIs evenly when the specified minimum of free space get used up by more data.

carriage control character

The fist character of an output record (line) that is to be printed; it determines how many lines should be skipped before the next line is printed.

catalog

A directory of files and libraries, with reference to their locations. A catalog may contain other information such as the types of devices in which the files are stored, passwords, blocking factors. To store a library member such as a phase, module, or book in a sublibrary. See also VSE/VSAM catalog.

cell pool

An area of virtual storage that is obtained by an application program and managed by the callable cell pool services. A cell pool is located in an address space or a data space and contains an anchor, at least one extent, and any number of cells of the same size.

central location

The place at which a computer system's control device, normally the systems console in the computer room, is installed.

chained sublibraries

A facility that allows sublibraries to be chained by specifying the sequence in which they must be searched for a certain library member.

chaining

A logical connection of sublibraries to be searched by the system for members of the same type (phases or object modules, for example).

channel command word (CCW)

A doubleword at the location in main storage that is specified by the channel address word. One or more CCWs make up the channel program that directs data channel operations.

channel program

One or more channel command words that control a sequence of data channel operations. Execution of this sequence is initiated by a start subchannel instruction.

channel scheduler

The part of the supervisor that controls all input/output operations.

channel subsystem

A feature of z/Architecture that provides extensive additional channel (I/O) capabilities to IBM Z.

channel to channel attachment (CTCA)

A function that allows data to be exchanged

1. Under the control of VSE/POWER between two virtual VSE machines running under VM or
2. Under the control of VTAM between two processors.

character-coded request

A request that is encoded and transmitted as a character string. Contrast with *field-formatted request*.

checkpoint

1. A point at which information about the status of a job and the system can be recorded so that the job step can be restarted later.
2. To record such information.

CICS (Customer Information Control System)

An IBM program that controls online communication between terminal users and a database. Transactions that are entered at remote terminals are processed concurrently by user-written application programs. The program includes facilities for building, using, and servicing databases.

CICS ECI

The CICS External Call Interface (ECI) is one possible requester type of the *CICS business logic interface* that is provided by the CICS Transaction Server for z/VSE. It is part of the CICS client and allows workstation programs to CICS function on the z/VSE host.

CICS EXCI

The EXternal CICS Interface (EXCI) is one possible requester type of the *CICS business logic interface* that is provided by the CICS Transaction Server for z/VSE. It allows any BSE batch application to call CICS functions.

CICS system definition data set (CSD)

A VSAM KSDS cluster that contains a resource definition record for every record defined to CICS using resource definition online (RDO).

CICS Transaction Server for z/VSE

A z/VSE base program that controls online communication between terminal users and a database. This is the successor system to CICS/VSE.

CICS TS

CICS Transaction Server

CICS/VSE

Customer Information Control System/VSE. No longer shipped on the Extended Base Tape and no longer supported, cannot run on z/VSE 5.1 or later.

class

In VSE/POWER, a group of jobs that either come from the same input device or go to the same output device.

CMS

Conversational monitor system running on z/VM.

common library

A library that can be interactively accessed by any user of the (sub)system that owns the library.

communication adapter

A circuit card with associated software that enables a processor, controller, or other device to be connected to a network.

communication region

An area of the supervisor that is set aside for transfer of information within and between programs.

component

1. Hardware or software that is part of a computer system.
2. A functional part of a product, which is identified by a component identifier.
3. In z/VSE, a component program such as VSE/POWER or VTAM.
4. In VSE/VSAM, a named, cataloged group of stored records, such as the data component or index component of a key-sequenced file or alternate index.

component identifier

A 12-byte alphanumeric string, uniquely defining a component to MSHP.

conditional job control

The capability of the job control program to process or to skip one or more statements that are based on a condition that is tested by the program.

connect

To authorize library access on the lowest level. A modifier such as "read" or "write" is required for the specified use of a sublibrary.

connection pooling

Introduced with an z/VSE 5.1 update to manage (reuse) connections of the z/VSE database connector in CICS TS.

connector

In the context of z/VSE, a connector provides the middleware to connect two platforms: Web Client and z/VSE host, middle-tier and z/VSE host, or Web Client and middle-tier.

connector (e-business connector)

A piece of software that is provided to connect to heterogeneous environments. Most connectors communicate to non-z/VSE Java-capable platforms.

container

Is part of the JVM of application servers such as the IBM WebSphere Application Server, and facilitates the implementation of servlets, EJBs, and JSPs, by providing resource and transaction management resources. For example, an EJB developer must not code against the JVM of the application server, but instead against the interface that is provided by the container. The main role of a container is to act as an intermediary between EJBs and clients, Is the host part of the VSE JavaBeans, and is started using the job STARTVCS, which is placed in the reader queue during the installation of z/VSE. Runs by default in dynamic class R. and also to manage multiple EJB instances. After EJBs have been written, they must be stored in a container residing on an application server. The container then manages all threading and client-interactions with the EJBs, and co-ordinate connection- and instance pooling.

control interval (CI)

A fixed-length area of disk storage where VSE/VSAM stores records and distributes free space. It is the unit of information that VSE/VSAM transfers to or from disk storage. For FBA it must be an integral multiple to be defined at cluster definition, of the block size.

control program

A program to schedule and supervise the running of programs in a system.

conversational monitor system (CMS)

A virtual machine operating system that provides general interactive time sharing, problem solving, and program development capabilities and operates under the control of z/VM.

count-key-data (CKD) device

A disk device that store data in the record format: count field, key field, data field. The count field contains, among others, the address of the record in the format: cylinder, head (track), record number, and the length of the data field. The key field, if present, contains the record's key or search argument. CKD disk space is allocated by tracks and cylinders. Contrast with *FBA disk device. See also extended count-key-data device.*

cross-partition communication control

A facility that enables VSE subsystems and user programs to communicate with each other; for example, with VSE/POWER.

cryptographic token

Usually referred to simply as a *token*, this is a device, which provides an interface for performing cryptographic functions like generating digital signatures or encrypting data.

cryptography

1. A method for protecting information by transforming it (encrypting it) into an unreadable format, called ciphertext. Only users who possess a secret key can decipher (or decrypt) the message into plaintext.
2. The transformation of data to conceal its information content and to prevent its unauthorized use or undetected modification .

D

data block group

The smallest unit of space that can be allocated to a VSE/POWER job on the data file. This allocation is independent of any device characteristics.

data conversion descriptor file (DCDF)

With a DCDF, you can convert individual fields within a record during data transfer between a PC and its host. The DCDF defines the record fields of a particular file for both, the PC and the host environment.

data import

The process of reformatting data that was used under one operating system such that it can subsequently be used under a different operating system.

Data Interfile Transfer, Testing, and Operations (DITTO) utility

An IBM program that provides file-to-file services for card I/O, tape, and disk devices. The latest version is called DITTO/ESA for VSE.

Data Language/I (DL/I)

A database access language that is used with CICS.

data link

In SNA, the combination of the link connection and the link stations joining network noes, for example, a z/Architecture channel and its associated protocols. A link is both logical and physical.

data security

The protection of data against unauthorized disclosure, transfer, modification, or destruction, whether accidental or intentional .

data set header record

In VSE/POWER abbreviated as DSHR, alias NDH or DSH. An NJE control record either preceding output data or, in the middle of input data, indicating a change in the data format.

data space

A range of up to 2 gigabytes of contiguous virtual storage addresses that a program can directly manipulate through z/Architecture instructions. Unlike an address space, a data space can hold only user data; it does not contain shared areas, or programs. Instructions do not execute in a data space. Contrast with address space.

data terminal equipment (DTE)

In SNA, the part of a data station that serves a data source, data sink, or both.

database connector

Is a function introduced with z/VSE 5.1.1, which consists of a client and server part. The client provides an API (CBCLI) to be used by applications on z/VSE, the server on any Java capable platform connects a JDBC driver that is provided by the database. Both client and server communicate via TCP/IP.

Database 2 (Db2)

An IBM rational database management system.

Db2-based connector

Is a feature introduced with VSE/ESA 2.5, which includes a customized Db2 version, together with VSAM and DL/I functionality, to provide access to Db2, VSAM, and DL/I data, using Db2 Stored Procedures.

Db2 Runtime only Client edition

The Client Edition for z/VSE comes with some enhanced features and improved performance to integrate z/VSE and Linux on z Systems.

Db2 Stored Procedure

In the context of z/VSE, a Db2 Stored Procedure is a Language Environment (LE) program that accesses Db2 data. However, from VSE/ESA 2.5 onwards you can also access VSAM and DL/I data using a Db2 Stored Procedure. In this way, it is possible to exchange data between VSAM and Db2.

DBLK

Data block.

DCDF

Data conversion descriptor file.

deblocking

The process of making each record of a block available for processing.

dedicated (disk) device

A device that cannot be shared among users.

device address

1. The identification of an input/output device by its device number.
2. In data communication, the identification of any device to which data can be sent or from which data can be received.

device driving system (DDS)

A software system external to VSE/POWER, such as a CICS spooler or PSF, that writes spooled output to a destination device.

Device Support Facilities (DSF)

An IBM supplied system control program for performing operations on disk volumes so that they can be accessed by IBM and user programs. Examples of these operations are initializing a disk volume and assigning an alternative track.

device type code

The four- or five-digit code that is used for defining an I/O device to a computer system. See also ICKDSF

dialog

In an interactive system, a series of related inquiries and responses similar to a conversation between two people. For z/VSE, a set of panels that can be used to complete a specific task; for example, defining a file.

dialog manager

The program component of z/VSE that provides for ease of communication between user and system.

digital signature

In computer security, encrypted data, which is appended to or part of a message, that enables a recipient to prove the identity of the sender.

Digital Signature Algorithm (DSA)

The Digital Signature Algorithm is the US government-defined standard for digital signatures. The DSA digital signature is a pair of large numbers, computed using a set of rules (that is, the DSA) and a set of parameters such that the identity of the signatory and integrity of the data can be verified. The DSA provides the capability to generate and verify signatures.

directory

In z/VSE the index for the program libraries.

direct access

Accessing data on a storage device using their address and not their sequence. This is the typical access on disk devices as opposed to magnetic tapes. Contrast with *sequential access*.

disk operating system residence volume (DOSRES)

The disk volume on which the system sublibrary IJSYSRS.SYSLIB is located including the programs and procedures that are required for system startup.

disk sharing

An option that lets independent computer systems uses common data on shared disk devices.

disposition

A means of indicating to VSE/POWER how a job input or output entry is to be handled: according to its local disposition in the RDR/LST/PUN queue or its transmission disposition when residing in the XMT queue. A job might, for example, be deleted or kept after processing.

distribution tape

A magnetic tape that contains, for example, a preconfigured operating system like z/VSE. This tape is shipped to the customer for program installation.

DITTO/ESA for VSE

Data Interfile Transfer, Testing, and Operations utility. An IBM program that provides file-to-file services for disk, tape, and card devices.

DSF

Device Support Facilities.

DSH (R)

Data set header record.

dummy device

A device address with no real I/O device behind it. Input and output for that device address are spooled on disk.

duplex

Pertaining to communication in which data can be sent and received at the same time.

DU-AL (dispatchable unit - access list)

The access list that is associated with a z/VSE main task or subtask. A program uses the DU-AL associated with its task and the PASN-AL associated with its partition. See also “PASN-AL (primary address space number - access list)” on page 382.

dynamic class table

Defines the characteristics of dynamic partitions.

dynamic partition

A partition that is created and activated on an 'as needed' basis that does not use fixed static allocations. After processing, the occupied space is released. Dynamic partitions are grouped by class, and jobs are scheduled by class. Contrast with *static partition*.

dynamic space reclamation

A librarian function that provides for space that is freed by the deletion of a library member to become reusable automatically.

E

ECI

See “CICS ECI” on page 365.

emulation

The use of programming techniques and special machine features that permit a computer system to execute programs that are written for another system or for the use of I/O devices different from those that are available.

emulation program (EP)

An IBM control program that allows a channel-attached 3705 or 3725 communication controller to emulate the functions of an IBM 2701 Data Adapter Unit, or an IBM 2703 Transmission Control.

end user

1. A person who makes use of an application program.
2. In SNA, the ultimate source or destination of user data flowing through an SNA network. Might be an application program or a terminal operator.

Enterprise Java Bean

An EJB is a distributed bean. "Distributed" means, that one part of an EJB runs inside the JVM of a web application server, while the other part runs inside the JVM of a web browser. An EJB either represents one data row in a database (entity bean), or a connection to a remote database (session bean). Normally, both types of an EJB work together. This allows to represent and access data in a standardized way in heterogeneous environments with relational and non-relational data. See also *JavaBean*.

entry-sequenced file

A VSE/VSAM file whose records are loaded without respect to their contents and whose relative byte addresses cannot change. Records are retrieved and stored by addressed access, and new records are added to the end of the file.

Environmental Record Editing and Printing (EREP) program

A z/VSE base program that makes the data that is contained in the system record file available for further analysis.

EPI

See *CICS EPI.*

ESCON Channel (Enterprise Systems Connection Channel)

A serial channel, using fiber optic cabling, that provides a high-speed connection between host and control units for I/O devices. It complies with the ESA/390 and IBM Z I/O Interface until z114. The zEC12 processors do not support ESCON channels.

exit routine

1. Either of two types of routines: installation exit routines or user exit routines. Synonymous with exit program.
2. See *user exit routine*.

extended addressability

The ability of a program to use 31 bit or 64 bit virtual storage in its address space or outside the address space.

extended recovery facility (XRF)

In z/VSE, a feature of CICS that provides for enhanced availability of CICS by offering one CICS system as a backup of another.

External Security Manager (ESM)

A priced vendor product that can provide extended functionality and flexibility that is compared to that of the Basic Security Manager (BSM), which is part of z/VSE.

F

FASTCOPY

See "VSE/Fast Copy" on page 393.

fast copy data set program (VSE/Fast Copy)

See "VSE/Fast Copy" on page 393.

fast service upgrade (FSU)

A service function of z/VSE for the installation of a refresh release without regenerating control information such as library control tables.

FAT-DASD

A subtype of Large DASD, it supports a device with more than 4369 cylinders (64 K tracks) up to 64 K cylinders.

FCOPY

See *VSE/Fast Copy.*

fence

A separation of one or more components or elements from the remainder of a processor complex. The separation is by logical boundaries. It allows simultaneous user operations and maintenance procedures.

fetch

1. To locate and load a quantity of data from storage.

2. To bring a program phase into virtual storage from a sublibrary and pass control to this phase.
3. The name of the macro instruction (FETCH) used to accomplish 2. See also *loader.*

Fibre Channel Protocol (FCP)

A combination of hardware and software conforming to the Fibre Channel standards and allowing system and peripheral connections via FICON and FICON Express feature cards on IBM zSeries processors. In z/VSE, zSeries FCP is employed to access industry-standard SCSI disk devices.

fragmentation (of storage)

Inability to allocate unused sections (fragments) of storage in the real or virtual address range of virtual storage.

FSU

Fast service upgrade.

FULIST (FUnction LIST)

A type of selection panel that displays a set of files and/or functions for the choice of the user.

G

generation

See *macro generation.*

generation feature

An IBM licensed program order option that is used to tailer the object code of a program to user requirements.

GETVIS space

Storage space within partition or the shared virtual area, available for dynamic allocation to programs.

guest system

A data processing system that runs under control of another (host) system. On the mainframe z/VSE can run as a guest of z/VM.

H

hard wait

The condition of a processor when all operations are suspended. System recovery from a hard wait is impossible without performing a new system startup.

hash function

A hash function is a transformation that takes a variable-size input and returns a fixed-size string, which is called the hash value. In cryptography, the hash functions should have some additional properties:

- The hash function should be easy to compute.
- The hash function is one way; that is, it is impossible to calculate the 'inverse' function.

- The hash function is collision-free; that is, it is impossible that different input leads to the same hash value.

hash value

The fixed-sized string resulting after applying a *hash function* to a text.

High-Level Assembler for VSE

A programming language providing enhanced assembler programming support. It is a base program of z/VSE.

home interface

Provides the methods to instantiate a new EJB object, introspect an EJB, and remove an EJB instantiation., as for the remote interface is needed because the deployment tool generates the implementation class. Every Session bean's home interface must supply at least one *create()* method.

host mode

In this operating mode, a PC can access a VSE host. For programmable workstation (PWS) functions, the Move Utilities of VSE can be used.

host system

The controlling or highest level system in a data communication configuration.

host transfer file (HTF)

Used by the Workstation File Transfer Support of z/VSE as an intermediate storage area for files that are sent to and from IBM personal computers.

HTTP Session

In the context of z/VSE, identifies the web-browser client that calls a servlet (in other words, identifies the connection between the client and the middle-tier platform).

I

ICCF

See *VSE/ICCF.*

ICKDSF (Device Support Facilities)

A z/VSE base program that supports the installation, use, and maintenance of IBM disk devices.

include function

Retrieves a library member for inclusion in program input.

index

1. A table that is used to locate records in an indexed sequential data set or on indexed file.
2. In, an ordered collection of pairs, each consisting of a key and a pointer, used by to sequence and locate the records of a key-sequenced data set or file; it is organized in levels of index records. See also *alternate index.*

input/output control system (IOCS)

A group of IBM supplied routines that handle the transfer of data between main storage and auxiliary storage devices.

integrated communication adapter (ICA)

The part of a processor where multiple lines can be connected.

integrated console

In z/VSE, the service processor console available on IBM Z that operates as the z/VSE system console. The integrated console is typically used during IPL and for recovery purposes when no other console is available.

Interactive Computing and Control Facility (ICCF)

An IBM licensed program that serves as interface, on a time-slice basis, to authorized users of terminals that are linked to the system's processor.

interactive partition

An area of virtual storage for the purpose of processing a job that was submitted interactively via VSE/ICCF.

Interactive User Communication Vehicle (IUCV)

Programming support available in a VSE supervisor for operation under z/VM. The support allows users to communicate with other users or with CP in the same way they would with a non-preferred guest.

intermediate storage

Any storage device that is used to hold data temporarily before it is processed.

IOCS

Input/output control system.

IPL

Initial program load.

irrecoverable error

An error for which recovery is impossible without the use of recovery techniques external to the computer program or run.

IUCV

Interactive User Communication Vehicle.

J

JAR

Is a platform-independent file format that aggregates many files into one. Multiple applets and their requisite components (.class files, images, and sounds) can be bundled in a JAR file, and then downloaded to a web browser using a single HTTP transaction (much improving the download speed). The JAR format also supports compression, which reduces the files size (and further improves the

download speed). The compression algorithm that is used is fully compatible with the ZIP algorithm. The owner of an applet can also digitally sign individual entries in a JAR file to authenticate their origin.

Java application

A Java program that runs inside the JVM of your web browser. The program's code resides on a local hard disk or on the LAN. Java applications might be large programs using graphical interfaces. Java applications have unlimited access to all your local resources.

Java bytecode

Bytecode is created when a file containing Java source language statements is compiled. The compiled Java code or "bytecode" is similar to any program module or file that is ready to be executed (run on a computer so that instructions are performed one at a time). However, the instructions in the bytecode are really instructions to the *Java Virtual Machine*. Instead of being interpreted one instruction at a time, bytecode is instead recompiled for each operating-system platform using a just-in-time (JIT) compiler. Usually, this enables the Java program to run faster. Bytecode is contained in binary files that have the suffix**.CLASS**

Java servlet

See *servlet.*

JHR

Job header record.

job accounting interface

A function that accumulates accounting information for each job step, to be used for charging the users of the system, for planning new applications, and for supervising system operation more efficiently.

job accounting table

An area in the supervisor where accounting information is accumulated for the user.

job catalog

A catalog made available for a job by means of the file name IJSYSUC in the respective DLBL statement.

job entry control language (JECL)

A control language that allows the programmer to specify how VSE/POWER should handle a job.

job step

In 1 of a group of related programs complete with the JCL statements necessary for a particular run. Every job step is identified in the job stream by an EXEC statement under one JOB statement for the whole job.

job trailer record (JTR)

As VSE/POWER parameter JTR, alias NJT. An NJE control record terminating a job entry in the input or output queue and providing accounting information.

K

key

In VSE/VSAM, one or several characters that are taken from a certain field (key field) in data records for identification and sequence of index entries or of the records themselves.

key sequence

The collating sequence either of records themselves or of their keys in the index or both. The key sequence is alphanumeric.

key-sequenced file

A VSE/VSAM file whose records are loaded in key sequence and controlled by an index. Records are retrieved and stored by keyed access or by addressed access, and new records are inserted in the file in key sequence.

KSDS

Key-sequenced data sets. See *key-sequenced file.*

L

label

1. An identification record for a tape, disk, or diskette volume or for a file on such a volume.
2. In assembly language programming, a named instruction that is generally used for branching.

label information area

An area on a disk to store label information that is read from job control statements or commands. Synonymous with *label area*.

Language Environment for z/VSE

An IBM software product that is the implementation of Language Environment on the VSE platform.

language translator

A general term for any assembler, compiler, or other routine that accepts statements in one language and produces equivalent statements in another language.

Large DASD

A DASD device that

1. Has a capacity exceeding 64 K tracks and
2. Does not have VSAM space created prior to VSE/ESA 2.6 that is owned by a catalog.

LE/VSE

Short form of Language Environment for z/VSE.

librarian

The set of programs that maintains, services, and organizes the system and private libraries.

library block

A block of data that is stored in a sublibrary.

library directory

The index that enables the system to locate a certain sublibrary of the accessed library.

library member

The smallest unit of a data that can be stored in and retrieved from a sublibrary.

line commands

In VSE/ICCF, special commands to change the declaration of individual lines on your screen. You can copy, move, or delete a line declaration, for example.

linkage editor

A program that is used to create a phase (executable code) from one or more independently translated object modules, from one or more existing phases, or from both. In creating the phase, the linkage editor resolves cross-references among the modules and phases available as input. The program can catalog the newly built phases.

linkage stack

An area of protected storage that the system gives to a program to save status information for a branch and stack or a stacking program call.

link station

In SNA, the combination of hardware and software that allows a node to attach to and provide control for a link.

loader

A routine, commonly a computer program, that reads data or a program into processor storage. See also *relocating loader*.

local shared resources (LSR)

A VSE/VSAM option that is activated by three extra macros to share control blocks among files.

lock file

In a shared disk environment under VSE, a system file on disk that is used by the sharing systems to control their access to shared data.

logical partition

In LPAR mode, a subset of the server unit hardware that is defined to support the operation of a system control program.

logical record

A user record, normally pertaining to a single subject and processed by data management as a unit. Contrast with *physical* record, which may be larger or smaller.

logical unit (LU)

1. A name that is used in programming to represent an I/O device address. *physical unit (PU), system services control point (SSCP), primary logical unit (PLU), and secondary logical unit (SLU).*
2. In SNA, a port through which a user accesses the SNA network,
   a. To communicate with another user and
   b. To access the functions of the SSCP. An LU can support at least two sessions. One with an SSCP and one with another LU and might be capable of supporting many sessions with other LUs.

logical unit name

In programming, a name that is used to represent the address of an input/output unit.

logical unit 6.2

A SNA/SDLC protocol for communication between programs in a distributed processing environment. LU 6.2 is characterized by

1. A peer relationship between session partners,
2. Efficient utilization of a session for multiple transactions,
3. Comprehensive end-to-end error processing, and
4. A generic Application Programming Interface (API) consisting of structured verbs that are mapped into a product implementation.

logons interpret interpret routine

In VTAM, an installation exit routine, which is associated with an interpret table entry, that translates logon information. It also verifies the logon.

LPAR mode

Logically partitioned mode. The CP mode that is available on the Configuration (CONFIG) frame when the PR/SM feature is installed. LPAR mode allows the operator to allocate the hardware resources of the processor unit among several logical partitions.

M

macro definition

A set of statements and instructions that defines the name of, format of, and conditions for generating a sequence of assembler statements and machine instructions from a single source statement.

macro expansion

See *macro generation*

macro generation

An assembler operation by which a macro instruction gets replaced in the program by the statements of its definition. It takes place before assembly. Synonymous with *macro expansion*.

macro (instruction)

1. In assembler programming, a user-invented assembler statement that causes the assembler to process a set of statements that are defined previously in the macro definition.

2. A sequence of VSE/ICCF commands that are defined to cause a sequence of certain actions to be performed in response to one request.

maintain system history program (MSHP)

A program that is used for automating and controlling various installation, tailoring, and service activities for a VSE system.

main task

The main program within a partition in a multiprogramming environment.

master console

In z/VSE, one or more consoles that receive all system messages, except for those that are directed to one particular console. Contrast this with the *user* console, which receives only those messages that are specifically directed to it, for example messages that are issued from a job that was submitted with the request to echo its messages to that console. The operator of a master console can reply to all outstanding messages and enter all system commands.

maximum (max) CA

A unit of allocation equivalent to the maximum control area size on a count-key-data or fixed-block device. On a CKD device, the max CA is equal to one cylinder.

memory object

Chunk of virtual storage that is allocated above the bar (2 GB) to be created with the IARV64 macro.

message

In VSE, a communication that is sent from a program to the operator or user. It can appear on a console, a display terminal or on a printout.

MSHP

See maintain system history program.

multitasking

Concurrent running of one main task and one or several subtasks in the same partition.

MVS

Multiple Virtual Storage. Implies MVS/390, MVS/XA, MVS/ESA, and the MVS element of the z/OS (OS/390) operating system.

N

NetView

A z/VSE optional program that is used to monitor a network, manage it, and diagnose its problems.

network address

In SNA, an address, consisting of subarea and element fields, that identifies a link, link station, or NAU. Subarea nodes use network addresses; peripheral nodes use local addresses. The boundary function in the subarea node to which a peripheral node is attached transforms local addresses to network addresses and vice versa. See also *network name*.

network addressable unit (NAU)

In SNA, a logical unit, a physical unit, or a system services control point. It is the origin or the destination of information that is transmitted by the path control network. Each NAU has a network address that represents it to the path control network. See also *network name, network address*.

Network Control Program (NCP)

An IBM licensed program that provides communication controller support for single-domain, multiple-domain, and interconnected network capability. Its full name is ACF/NCP.

network definition table (NDT)

In VSE/POWER networking, the table where every node in the network is listed.

network name

1. In SNA, the symbolic identifier by which users refer to a NAU, link, or link station. See also *network address*.
2. In a multiple-domain network, the name of the APPL statement defining a VTAM application program. This is its network name, which must be unique across domains.

node

1. In SNA, an end point of a link or junction common to several links in a network. Nodes can be distributed to host processors, communication controllers, cluster controllers, or terminals. Nodes can vary in routing and other functional capabilities.
2. In VTAM, a point in a network that is defined by a symbolic name. Synonymous with *network node*. See *major node and minor node*.

node type

In SNA, a designation of a node according to the protocols it supports and the network addressable units (NAUs) it can contain.

O

object module (program)

A program unit that is the output of an assembler or compiler and is input to a linkage editor.

online application program

An interactive program that is used at display stations. When active, it waits for data. Once input arrives, it processes it and send a response to the display station or to another device.

operator command

A statement to a control program, issued via a console or terminal. It causes the control program to provide requested information, alter normal operations, initiate new operations, or end existing operations.

optional licensed program

An IBM licensed program that a user can install on VSE by way of available installation-assist support.

output parameter text block (OPTB)

in VSE/POWER's spool-access support, information that is contained in an output queue record if a * $$ LST or * $$ PUN statement includes any user-defined keywords that have been defined for autostart.

P

page data set (PDS)

One or more extents of disk storage in which pages are stored when they are not needed in processor storage.

page fixing

Marking a page so that it is held in processor storage until explicitly released. Until then, it cannot be paged out.

page I/O

Page-in and page-out operations.

page pool

The set of page frames available for paging virtual-mode programs.

panel

The complete set of information that is shown in a single display on terminal screen. Scrolling back and forth through panels like turning manual pages. See also *selection panel*.

partition balancing

A z/VSE facility that allows the user to specify that two or more or all partitions of the system should receive about the same amount of time on the processor.

PASN-AL (primary address space number - access list)

The access list that is associated with a partition. A program uses the PASN-AL associated with its partition and the DU-AL associated with its task (work unit). See also *DU-AL*.

Each partition has its own unique PASN-AL. All programs running in this partition can access data spaces through the PASN-AL. Thus a program can create a data space, add an entry for it in the PASN-AL, and obtain the ALET that indexes the entry. By passing the ALET to other programs in the partition, the program can share the data space with other programs running in the same partition.

PDS

Page data sets.

phase

The smallest complete unit of executable code that can be loaded into virtual storage.

physical record

The amount of data that is transferred to or from auxiliary storage. Synonymous with *block*.

PNET

Programming support available with VSE/POWER; it provides for the transmission of selected jobs, operator commands, messages, and program output between the nodes of a network.

POWER

See *VSE/POWER*.

pregenerated operating system

An operating system such as z/VSE that is shipped by IBM mainly in object code. IBM defines such key characteristics as the size of the main control program, the organization, and size of libraries, and required system areas on disk. The customer does not have to generate an operating system.

preventive service

The installation of one or more PTFs on a VSE system to avoid the occurrence of anticipated problems.

primary address space

In z/VSE, the address space where a partition is executed. A program in primary mode fetches data from the primary address space.

primary library

A VSE library owned and directly accessible by a certain terminal user.

printer/keyboard mode

Refers to 1050 or 3215 console mode (device dependent).

Print Services Facility (PSF)/VSE

An access method that provides support for the advanced function printers.

private area

The virtual space between the shared area (24 bit) and shared area (31 bit), where (private) partitions are allocated. Its maximum size can be defined during IPL. See also *shared area*.

private memory object

Memory object (chunk of virtual storage) that is allocated above the 2 GB line (bar) only accessible by the partition that created it.

private partition

Any of the system's partitions that are not defined as shared. See also *shared partition*.

production library

1. In a pre-generated operating system (or product), the program library that contains the object code for this system (or product).
2. A library that contains data that is needed for normal processing. Contrast with *test library*.

programmer logical unit

A logical unit available primarily for user-written programs. See also *logical unit name*.

program temporary fix (PTF)

A solution or by-pass of one or more problems that are documented in APARs. PTFs are distributed to IBM customers for preventive service to a current release of a program.

PSF/VSE

Print Services Facility/VSE.

PTF

See *Program temporary fix*.

Q

Queue Control Area (QCA)

In VSE/POWER, an area of the data file, which might contain:

- Extended checkpoint information
- Control information for a shared environment.

queue file

A direct-access file that is maintained by VSE/POWER that holds control information for the spooling of job input and job output.

R

random processing

The treatment of data without respect to its location on disk storage, and in an arbitrary sequence that is governed by the input against which it is to be processed.

real address area

In z/VSE, processor storage to be accessed with dynamic address translation (DAT) off

real address space

The address space whose addresses map one-to-one to the addresses in processor storage.

real mode

In VSE, a processing mode in which a program might not be paged. Contrast with *virtual mode*.

recovery management support (RMS)

System routines that gather information about hardware failures and that initiate a retry of an operation that failed because of processor, I/O device, or channel errors.

refresh release

An upgraded VSE system with the latest level of maintenance for a release.

relative-record file

A VSE/VSAM file whose records are loaded into fixed-length slots and accessed by the relative-record numbers of these slots.

release upgrade

Use of the FSU functions to install a new release of z/VSE.

relocatable module

A library member of the type object. It consists of one or more control sections cataloged as one member.

relocating loader

A function that modifies addresses of a phase, if necessary, and loads the phase for running into the partition that is selected by the user.

remote interface

In the context of z/VSE, the remote interface allows a client to make method calls to an EJB although the EJB is on a remote z/VSE host. The container uses the remote interface to create client-side stubs and server-side proxy objects to handle incoming method calls from a client to an EJB.

remote procedure call (RPC)

1. A facility that a client uses to request the execution of a procedure call from a server. This facility includes a library of procedures and an external data representation.
2. A client request to service provider in another node.

residency mode (RMODE)

A program attribute that refers to the location where a program is expected to reside in virtual storage. RMODE 24 indicates that the program must reside in the 24-bit addressable area (below 16 megabytes), RMODE ANY indicates that the program can reside anywhere in 31-bit addressable storage (above or below 16 megabytes).

REXX/VSE

A general-purpose programming language, which is particularly suitable for command procedures, rapid batch program development, prototyping, and personal utilities.

RMS

Recovery management support.

RPG II

A commercially oriented programming language that is specifically designed for writing application programs that are intended for business data processing.

S

SAM ESDS file

A SAM file that is managed in VSE/VSAM space, so it can be accessed by both SAM and VSE/VSAM macros.

SCP

System control programming.

SDL

System directory list.

search chain

The order in which chained sublibraries are searched for the retrieval of a certain library member of a specified type.

second-level directory

A table in the SVA containing the highest phase names that are found on the directory tracks of the system sublibrary.

Secure Sockets Layer (SSL)

A security protocol that allows the client to authenticate the server and all data and requests to be encrypted. SSL was developed by Netscape Communications Corp. and RSA Data Security, Inc..

segmentation

In VSE/POWER, a facility that breaks list or punch output of a program into segments so that printing or punching can start before this program has finished generating such output.

selection panel

A displayed list of items from which a user can make a selection. Synonymous with *menu*.

sense

Determine, on request or automatically, the status or the characteristics of a certain I/O or communication device.

sequential access method (SAM)

A data access method that writes to and reads from an I/O device record after record (or block after block). On request, the support performs device control operations such as line spacing or page ejects on a printer or skip some tape marks on a tape drive.

service node

Within the VSE unattended node support, a processor that is used to install and test a master VSE system, which is copied for distribution to the unattended nodes. Also, program fixes are first applied at the service node and then sent to the unattended nodes.

service program

A computer program that performs function in support of the system. See with *utility program*.

service refresh

A form of service containing the current version of all software. Also referred to as a *system refresh*.

service unit

One or more PTFs on disk or tape (cartridge).

shared area

In z/VSE, shared areas (24 bit) contain the Supervisor areas and SVA (24 bit) and shared areas (31 bit) the SVA (31 bit). Shared areas (24 bit) are at the beginning of the address space (below 16 MB), shared area (31 bit) at the end (below 2 GB).

shared disk option

An option that lets independent computer systems use common data on shared disk devices.

shared memory objects

Chunks of virtual storage allocated above the 2 GB line (bar), that can be shared among partitions.

shared partition

In z/VSE, a partition that is allocated for a program (VSE/POWER, for example) that provides services and communicates with programs in other partitions of the system's virtual address spaces. In most cases shared partitions are no longer required.

shared spooling

A function that permits the VSE/POWER account file, data file, and queue file to be shared among several computer systems with VSE/POWER.

shared virtual area (SVA)

In z/VSE, a high address area that contains a list system directory list (SDL) of frequently used phases, resident programs that are shared between partitions, and an area for system support.

SIT (System Initialization Table)

A table in CICS that contains data used the system initialization process. In particular, the SIT can identify (by suffix characters) the version of CICS system control programs and CICS tables that you have specified and that are to be loaded.

skeleton

A set of control statements, instructions, or both, that requires user-specific information to be inserted before it can be submitted for processing.

socksified

See *socks-enabled*.

Socks-enabled

Pertaining to TCP/IP software, or to a specific TCP/IP application, that understands the *socks protocol*. "Socksified" is a slang term for socks-enabled.

socks protocol

A protocol that enables an application in a secure network to communicate through a firewall via a *socks server*.

socks server

A circuit-level gateway that provides a secure one-way connection through a firewall to server applications in a nonsecure network.

source member

A library member containing source statements in any of the programming languages that are supported by VSE.

split

To double a specific unit of storage space (CI or CA) dynamically when the specified minimum of free space gets used up by new records.

spooling

The use of disk storage as buffer storage to reduce processing delays when transferring data between peripheral equipment and the processor of a computer. In z/VSE, this is done under the control of VSE/POWER.

Spool Access Protection

An optional feature of VSE/POWER that restricts individual spool file entry access to user IDs that have been authenticated by having performed a security logon.

spool file

1. A file that contains output data that is saved for later processing.
2. One of three VSE/POWER files on disk: queue file, data file, and account file.

SSL

See Secure Sockets Layer.

stacked tape

An IBM supplied product-shipment tape containing the code of several licensed programs.

standard label

A fixed-format record that identifies a volume of data such as a tape reel or a file that is part of a volume of data.

stand-alone program

A program that runs independently of (not controlled by) the VSE system.

startup

The process of performing IPL of the operating system and of getting all subsystems and applications programs ready for operation.

start option

In VTAM, a user-specified or IBM specified option that determines conditions for the time a VTAM system is operating. Start options can be predefined or specified when VTAM is started.

static partition

A partition, which is defined at IPL time and occupying a defined amount of virtual storage that remains constant. See also *dynamic partition*.

storage director

An independent component of a storage control unit; it performs all of the functions of a storage control unit and thus provides one access path to the disk devices that are attached to it. A storage control unit has two storage directors.

storage fragmentation

Inability to allocate unused sections (fragments) of storage in the real or virtual address range of virtual storage.

suballocated file

A VSE/VSAM file that occupies a portion of an already defined data space. The data space might contain other files. See also *unique file*.

sublibrary

In VSE, a subdivision of a library. Members can only be accessed in a sublibrary.

sublibrary directory

An index for the system to locate a member in the accessed sublibrary.

submit

A VSE/POWER function that passes a job to the system for processing.

SVA

See shared virtual area.

Synchronous DataLink Control (SDLC)

A discipline for managing synchronous, code-transparent, serial-by-bit information transfer over a link connection. Transmission exchanges might be duplex or half-duplex over switched or non-switched links. The configuration of the link connection might be point-to-point, multipoint, or loop.

SYSRES

See system residence volume.

system control programming (SCP)

IBM supplied, non-licensed program fundamental to the operation of a system or to its service or both.

system directory list (SDL)

A list containing directory entries of frequently used phases and of all phases resident in the SVA. The list resides in the SVA.

system file

In z/VSE, a file that is used by the operating system, for example, the hardcopy file, the recorder file, the page data set.

System Initialization Table (SIT)

A table in CICS that contains data that is used by the system initialization process. In particular, the SIT can identify (by suffix characters) the version of CICS system control programs and CICS tables that you have specified and that are to be loaded.

system recorder file

The file that is used to record hardware reliability data. Synonymous with *recorder file*.

system refresh

See *service refresh*.

system refresh release

See *refresh release*.

system residence file (SYSRES)

The z/VSE system sublibrary IJSYSRS.SYSLIB that contains the operating system. It is stored on the system residence volume DORSES.

system residence volume (SYSRES)

The disk volume on which the system sublibrary is stored and from which the hardware retrieves the initial program load routine for system startup.

system sublibrary

The sublibrary that contains the operating system. It is stored on the system residence volume (SYSRES).

T

task management

The functions of a control program that control the use, by tasks, of the processor and other resources (except for input/output devices).

time event scheduling support

In VSE/POWER, the time event scheduling support offers the possibility to schedule jobs for processing in a partition at a predefined time once repetitively. The time event scheduling operands of the * $$ JOB statement are used to specify the wanted scheduling time.

TLS

See Transport Layer Security.

track group

In VSE/POWER, the basic organizational unit of a file for CKD devices.

track hold

A function that protects a track that is being updated by one program from being accessed by another program.

transaction

1. In a batch or remote batch entry, a job or job step. 2. In CICS TS, one or more application programs that can be used by a display station operator. A given transaction can be used concurrently from one or more display stations. The execution of a transaction for a certain operator is also referred to as a task.
2. A given task can relate only to one operator.

transient area

An area within the control program that is used to provide high-priority system services on demand.

Transport Layer Security

The newest SSL cryptographic protocol. It provides additional strength to privacy and data integrity.

Turbo Dispatcher

A facility of z/VSE that allows to use multiprocessor systems (also called CEC: Central Electronic Complexes). Each CPU within such a CEC has accesses to be shared virtual areas of z/VSE: supervisor, shared areas (24 bit), and shared areas (31 bit). The CPUs have equal rights, which means that any CPU might receive interrupts and work units are not dedicated to any specific CPU.

U

UCB

Universal character set buffer.

universal character set buffer (UCB)

A buffer to hold UCS information.

UCS

Universal character set.

user console

In z/VSE, a console that receives only those system messages that are specifically directed to it. These are, for example, messages that are issued from a job that was submitted with the request to echo its messages to that console. Contrast with *master console*.

user exit

A programming service that is provided by an IBM software product that can be requested during the execution of an application program for the service of transferring control back to the application program upon the later occurrence of a user-specified event.

V

variable-length relative-record data set (VRDS)

A relative-record data set with variable-length records. See also *relative-record data set*.

variable-length relative-record file

A VSE/VSAM relative-record file with variable-length records. See also *relative-record file*.

VIO

See virtual I/O area.

virtual address

An address that refers to a location in virtual storage. It is translated by the system to a processor storage address when the information stored at the virtual address is to be used.

virtual addressability extension (VAE)

A storage management support that allows to use multiple virtual address spaces.

virtual address space

A subdivision of the virtual address area (virtual storage) available to the user for the allocation of private, nonshared partitions.

virtual disk

A range of up to 2 gigabytes of contiguous virtual storage addresses that a program can use as workspace. Although the virtual disk exists in storage, it appears as a real FBA disk device to the user program. All I/O operations that are directed to a virtual disk are intercepted and the data to be written to, or read from, the disk is moved to or from a data space.

Like a data space, a virtual disk can hold only user data; it does not contain shared areas, system data, or programs. Unlike an address space or a data space, data is not directly addressable on a virtual disk. To manipulate data on a virtual disk, the program must perform I/O operations.

Starting with z/VSE 5.2, a virtual disk may be defined in a shared memory object.

virtual I/O area (VIO)

An extension of the page data set; used by the system as intermediate storage, primarily for control data.

virtual mode

The operating mode of a program, where the virtual storage of the program can be paged, if not enough processor (real) storage is available to back the virtual storage.

virtual partition

In VSE, a division of the dynamic area of virtual storage.

virtual storage

Addressable space image for the user from which instructions and data are mapped into processor storage locations.

virtual tape

In z/VSE, a virtual tape is a file (or data set) containing a tape image. You can read from or write to a virtual tape in the same way as if it were a physical tape. A virtual tape can be:

- A VSE/VSAM ESDS file on the z/VSE local system.
- A remote file on the server side; for example, a Linux, UNIX, or Windows file. To access such a remote virtual tape, a TCP/IP connection is required between z/VSE and the remote system.

volume ID

The volume serial number, which is a number in a volume label that is assigned when a volume is prepared for use by the system.

VRDS

Variable-length relative-record data sets. See *variable-length relative record file*.

VSAM

See *VSE/VSAM*.

VSE (Virtual Storage Extended)

A system that consists of a basic operating system and any IBM supplied and user-written programs that are required to meet the data processing needs of a user. VSE and hardware it controls form a complete computing system. Its current version is called z/VSE.

VSE/Advanced Functions

A program that provides basic system control and includes the supervisor and system programs such as the Librarian and the Linkage Editor.

VSE Connector Server

Is the host part of the VSE JavaBeans, and is started using the job STARTVCS, which is placed in the reader queue during installation of z/VSE. Runs by default in dynamic class R.

VSE/DITTO (VSE/Data Interfile Transfer, Testing, and Operations Utility)

An IBM licensed program that provides file-to-file services for disk, tape, and card devices.

VSE/ESA (Virtual Storage Extended/Enterprise Systems Architecture)

The predecessor system of z/VSE.

VSE/Fast Copy

A utility program for fast copy data operations from disk to disk and dump/restore operations via an intermediate dump file on magnetic tape or disk.

VSE/FCOPY (VSE/Fast Copy Data Set program)

An IBM licensed program for fast copy data operations from disk to disk and dump/restore operations via an intermediate dump file on magnetic tape or disk. There is also a stand-alone version: the FASTCOPY utility.

VSE/ICCF (VSE/Interactive Computing and Control Facility)

An IBM licensed program that serves as interface, on a time-slice basis, to authorized users of terminals that are linked to the system's processor.

VSE/ICCF library

A file that is composed of smaller files (libraries) including system and user data, which can be accessed under the control of VSE/ICCF.

VSE JavaBeans

Are JavaBeans that allow access to all VSE-based file systems (VSE/VSAM, Librarian, and VSE/ICCF), submit jobs, and access the z/VSE operator console. The class library is contained in the *VSEConnector.jar* archive. See also *JavaBeans*.

VSE library

A collection of programs in various forms and storage dumps stored on disk. The form of a program is indicated by its member type such as source code, object module, phase, or procedure. A VSE library consists of at least one sublibrary, which can contain any type of member.

VSE/POWER

An IBM licensed program that is primarily used to spool input and output. The program's networking functions enable a VSE system to exchange files with or run jobs on another remote processor.

VSE/VSAM (VSE/Virtual Storage Access Method)

An IBM access method for direct or sequential processing of fixed and variable length records on disk devices.

VSE/VSAM catalog

A file containing extensive file and volume information that VSE/VSAM requires to locate files, to allocate and deallocate storage space, to verify the authorization of a program or an operator to gain access to a file, and to accumulate use statistics for files.

VSE/VSAM managed space

A user-defined space on disk that is placed under the control of VSE/VSAM.

W

wait for run subqueue

In VSE/POWER, a subqueue of the reader queue with dispatchable jobs ordered in execution start time sequence.

wait state

The condition of a processor when all operations are suspended. System recovery from a hard wait is impossible without performing a new system startup. See *hard wait*.

Workstation File Transfer Support

Enables the exchange of data between IBM Personal Computers (PCs) linked to a z/VSE host system where the data is kept in intermediate storage. PC users can retrieve that data and work with it independently of z/VSE.

work file

A file that is used for temporary storage of data being processed.

Numerics

24-bit addressing

Provides addressability for address spaces up to 16 megabytes.

31-bit addressing

Provides addressability for address spaces up to 2 gigabytes.

64-bit addressing

Provides addressability for address spaces up to 2 gigabytes and above.

Index

Special Characters

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z

Product Number: 5686-066

SC33-6739-02