GC28-2048-4

IBM

Introducing TSS/360 Printed in U.S.A. GC28-2048-4

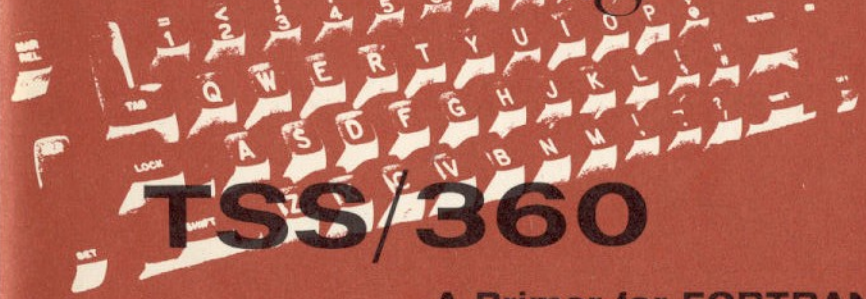
International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
(USA Only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

File No. S360-20
Order No. GC28-2048-4

IBM System/360
Time Sharing System

Introducing...



TSS/360

A Primer for FORTRAN Users

IBM

PREFACE

A subset of Time Sharing System/360 (TSS/360) is presented in this book to allow use of basic system facilities without an extensive knowledge of the command system, by which system functions are invoked. The reader of the book is presumed to have at least a basic knowledge of FORTRAN. In addition, the user profile under which the reader will use TSS/360 should be altered as explained in Appendix B. This can be done by the reader himself, if necessary; preferably, however, it should be done by someone in a supervisory or tutorial relationship to the reader.

The altered user profile can be changed again by the reader if he progresses to using the full command system. The full system is explained in other books of the TSS/360 Systems Reference Library, such as *Command System User's Guide*, Form C28-2001, and *FORTRAN Programmer's Guide*, Form C28-2025.

This book contains basic information on the three kinds of terminals that can be used with TSS/360. If more information is needed, see *Terminal User's Guide*, Form C28-2017.

Fifth Edition (September 1970)

This edition, a revision of C28-2048-3, is current with Version 8, Modification 0, of IBM System/360 Time Sharing System and remains in effect for all subsequent versions unless otherwise indicated. Significant changes will be provided in new editions.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

Comments on this publication may be addressed to IBM Corporation, Time Sharing System/360 Programming Publications, Department 643, Neighborhood Road, Kingston, New York 12401.

IBM

CONTENTS

INTRODUCTION

TSS/360 is a programming system that runs on the IBM System/360 Model 67 to give a number of simultaneous users access to the services of a computer. It is as if each user had access to a separate computer. The work being done by one user cannot interfere with that being done by another.

The Virtual Computer

The illusion that TSS/360 gives each user of having his own computer is called the *virtual machine* or *virtual computer.* TSS/360 provides each user with a virtual computer that is easier to use than the real computer on which the time-sharing system operates.

In addition, each user can address a larger main storage capacity than the real computer has. This is accomplished by sectioning programs and data and keeping the sections that are not currently in use on secondary storage instead of in main storage. But you can write your programs and structure your data without regard for how they are sectioned or where they are actually stored; the sections on secondary storage will be brought into main storage when you need them.

This illusion of main storage capacity larger than that of the real computer is an aspect of the virtual computer that is called *virtual storage.* You can learn more about how it is done from other books of the TSS/360 Systems Reference Library if you want to. But you don't need to know how TSS/360 works in order to use it.

Facilities of TSS/360

TSS/360 features a set of commands by which you can tell the system what you want it to do for you. The full command system is very powerful—but, like any powerful tool, it is hard to learn how to use every feature of it all at once.

This book introduces you to a subset of the command system that still gives you plenty of power for basic uses. It teaches you how to

- gain conversational access to the system from a terminal;
- create, edit and erase data sets;
- compile and execute FORTRAN programs;
- debug your programs dynamically—that is, while executing them.

After mastering the TSS/360 facilities presented in this book, you can go on to other manuals to learn more. But by the time you finish this book, you should be able to use the system successfully; and you may even decide that the command subset presented here fulfills your computing needs.

One: GAINING ACCESS TO TSS/360

To use the facilities of TSS/360, you must first have

- a user identification and password
- a terminal.

The terminal is a typewriter-like device with which you and the system communicate. The three kinds of terminals used with TSS/360 are described in Appendix A. When you find out what kind you will be using, read the part of the appendix that tells you how to use that kind.

The user identification and password will be assigned to you by the person who gives you permission to use TSS/360. The user identification, or user ID, is the name under which the system will keep track of your use of the computer. The password, which can be changed more easily, is provided as a way of making sure that only you can use your user ID to gain access to the system. Keep this information to yourself to prevent others from using the system and charging you for it.

The user ID and the password are operands of the LOGON command. To prevent unauthorized observation, the password may be omitted as the second operand of LOGON, and entered in the blackened space provided by the system, as below. You must go through the LOGON procedure each time you want to use TSS/360.

To start the LOGON procedure, turn on your terminal and connect it to the time-sharing system. (See the instructions on how to do this in Appendix A.) The system will then unlock the terminal keyboard for you to enter the LOGON command and your user ID.

The command and the system messages will look like this (this system types in capital letters, while your entries are in lowercase):

```
logon smith
```

(The system responds with a message indicating the level of the system.)

```
ENTER PASSWD
xxxxxxxx
```

SMITH is the user ID in this example. You enter your password in the blackened space. If your password is valid, the system will respond with a message such as:

```
2D  LOGON AT 12:03 ON 09/15/70
```

In this message, 2D is the task ID number (the system assigns a unique hexadecimal number to each task).

The system will invite you to enter a command. (On an IBM 1052 or 2741 terminal, this invitation will appear as an underscore followed by a backsapce; on the teletypewriter, it will appear as a right bracket and a left arrow. From here on in this book, the IBM terminal will be considered standard; teletypewriter users should use the equivalent characters noted in Appendix A.)

After you have finished using the system, you must disconnect yourself from it. Do not just turn off the terminal. Instead, issue the LOGOFF command so that TSS/360 can disconnect you in an orderly fashion.

```
logoff

LOGOFF ACCEPTED 09/15/70 AT 10:08
```

Notice that the first letter of your LOGOFF command appears directly over the underscore that the system printed to invite you to enter a command. And notice again that your LOGOFF command, like your LOGON command, is typed in lowercase letters while the system's message is in capitals. TSS/360 translates your lowercase letters into capitals; this saves you from constant use of the SHIFT key and provides an easy way to tell your entries from the system's when you read over terminal printout.

After the system accepts your LOGOFF command, turn off the terminal, which will by then be disconnected from the computer.

Review

- To gain access to TSS/360, you need a terminal, a user identification, and a password.
- Each TSS/360 terminal session must begin with the LOGON command. If the user ID is entered as the only operand of LOGON, the system will prompt you to enter your password in a blackened field which it provides.
- To end a terminal session, issue the LOGOFF command. This causes the system to disconnect you in an orderly fashion. *Do not just turn off the terminal.*

Two: ORGANIZING DATA

In TSS/360, data is organized into *data sets*. A data set is a named collection of logically related data records. TSS/360 includes data set organizations that were designed for efficient use with virtual storage. For the purposes of this manual, only two data set organizations will be considered—the *virtual index sequential organization* and the *virtual sequential organization.*

The virtual index sequential organization will be used here for source data sets to be processed by the FORTRAN compiler. Such data sets can be created or modified with the TSS/360 text editing facilities, as explained in Chapter Three. The virtual sequential organization is used with FORTRAN I/O routines.

A virtual index sequential data set consists of records that are indexed by an ascending key that is part of each record. This book will consider only variable-length records in which the key is a seven-digit decimal number at the beginning of each record. Each record is considered to be a line of data, and the numbers are known as line numbers. This type of virtual index sequential data set is called a *line data set.*

Lines in the line data sets used with this book may vary in length up to 132 characters, including the seven-digit line numbers and five characters of system information—leaving 120 characters for data.

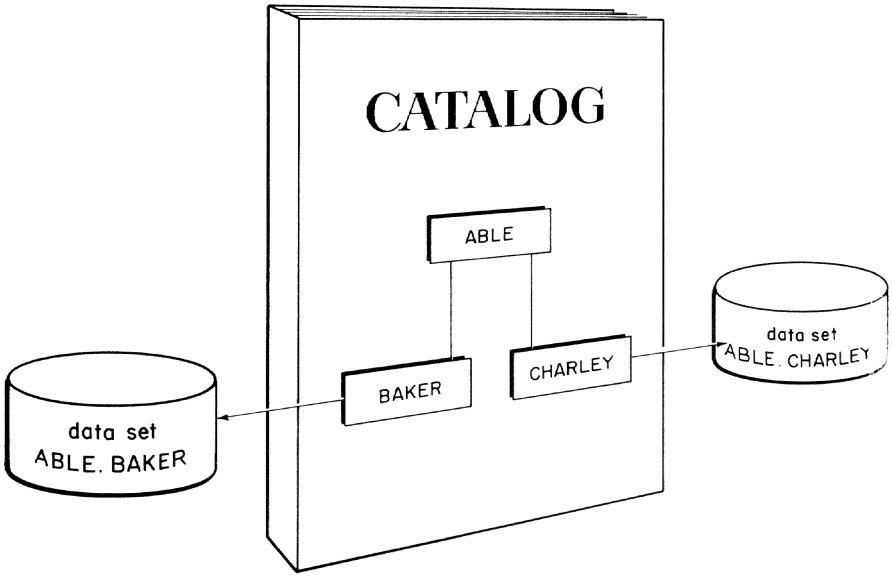


Figure 1. Catalog organization.

The indexing feature of the virtual index sequential organization allows nonsequential processing of the data set records. Line numbers that are supplied by the system are normally produced by adding an increment of 100 to the previous system-supplied line number, thus allowing as many as 99 insertions between lines. System routines can be used to display selected lines and to remove or replace selected lines in an existing data set, and to insert new lines. However, FORTRAN I/O routines cannot be used to process lines nonsequentially; these routines can best be used to process virtual sequential data sets.

Below is an example of a line data set. For illustrative purposes, trivial data is used; each of these lines from Hamlet's soliloquy corresponds to a line of data.

```
0000100 TO BE, OR NOT TO BE, THAT IS THE QUESTION:
0000200 WHETHER 'TIS NOBLER IN THE MIND TO SUFFER
0000300 THE SLINGS AND ARROWS OF OUTRAGEOUS FORTUNE
0000400 OR TO TAKE ARMS AGAINST A SEA OF TROUBLES
0000500 AND BY OPPOSING END THEM.
```

Data Set Names

TSS/360 data sets can be accessed by name alone. This is possible because the system maintains a catalog of data sets; this catalog associates the name of each cataloged data set with the physical address of the data set on secondary storage. The name of each data set is prefixed with the user identification of the user who created it, so that each user's data sets are uniquely identified and cannot be confused with those of another user.

A *simple data set name* consists of one to eight alphanumeric characters, the first of which must be alphabetic. (For instance, DATASET1 would be a valid data set name, but 1DATASET would not.)

Longer data set names can be formed by combining a series of such simple names, called components, so that each component represents a level of qualification. Several levels of qualification are possible, but two levels are sufficient for the applications discussed in this book.

For example, the data set name ABLE.BAKER consists of two components that are delimited by a period to indicate a hierarchy of categories. Starting from the left, each component of the name can be considered a category within which the next component is a unique subcategory.

A *fully qualified* data set name identifies an individual data set and includes all components of that data set's name. A *partially qualified* data set name identifies a group of data sets by omitting one or more of the rightmost

components of their fully qualified names. For instance, a TSS/360 user might have two data sets named ABLE.BAKER and ABLE.CHARLEY. The partially qualified name ABLE would refer to these data sets as a group; the fully qualified names ABLE.BAKER and ABLE.CHARLEY would uniquely identify each data set. (See Figure 1.)

Source Data Sets

The chief application of fully and partially qualified data set names for users of this book is in naming source data sets for processing by the FORTRAN compiler. A source data set contains the source program, or source statements, from which the compiler produces an executable program, called the object program.

A FORTRAN source program may be prestored in the system as a data set or entered from the terminal for line-by-line scanning. In the latter case, the system builds a data set to contain the source statements; this data set may be modified and recompiled later. No matter how the source statements are entered into the source data set—by the user before compilation or by the system during compilation—the source data set has the partially qualified name SOURCE. The fully qualified name consists of the component SOURCE, the delimiting period, and the name of the object program to be produced from that source data set.

For instance, if you had written a FORTRAN source program from which you wanted to produce an object program to be named ADDUP, you could first store the FORTRAN source statements in a data set named SOURCE.ADDUP. You could then call the FORTRAN compiler to process the statements in the data set SOURCE.ADDUP to produce the executable program named ADDUP.

Alternatively, you could call the FORTRAN compiler and enter the source statements one at a time for line-by-line scanning and compilation. In this case the compiler would not only produce the object program ADDUP, it would also construct the data set SOURCE.ADDUP for you.

Program Names

Executable programs are also stored as data sets; however, these data sets do not contain data in the usual sense, but machine instructions and control information that makes it possible to load and execute them as programs.

Each program you compile is made a member of a library of your programs which is names USERLIB. The program can be invoked by using its simple name—such as ADDUP in the example above. To treat it as a data set, however (to get rid of it, for instance), you would refer to it by the name of the library followed by the program name in parentheses. In the case of ADDUP, the fully qualified data set name of the program would be USERLIB(ADDUP).

Although data set names may consist of as many as eight alphanumeric characters, you should restrict your program names to six or fewer characters. This will assure the uniqueness of additional names which the compiler generates based on the program name.

Review

- In TSS/360, data is organized into named collections of logically related records called *data sets.*
- This book illustrates a special form of index sequential data set called a *line data set*—each record is a line indexed by a seven-digit decimal number at the beginning of each line. Lines may vary in length up to 132 characters, including the seven-digit line number and five characters of system information.
- TSS/360 data sets can be accessed by name alone through the system catalog, which associates a physical address with the data set name.
- Simple data set names consist of one to eight alphanumeric characters, the first of which must be alphabetic.
- Longer data set names can be formed by joining a series of simple names with periods so that each represents a level of qualification. A fully qualified name identifies an individual data set; a partially qualified name identifies a group of data sets.
- All source data sets can be identified by the partially qualified name SOURCE. For instance, a source data set for a program named ADDUP would have the fully qualified name SOURCE.ADDUP.
- Executable programs are members of a library of user programs, which is named USERLIB. Programs can be called by their simple names (for instance, ADDUP), but to treat a program as a data set you must address it by its fully qualified name—USERLIB followed by the program name in parentheses. For the program ADDUP, for instance, the fully qualified name would be USERLIB(ADDUP).
- Restrict program names to six or fewer characters to avoid possible duplication of additional names that the compiler generates from the program name.

Three: CREATING AND EDITING DATA SETS

The text editor is a facility that can be used to create line data sets and to alter, or edit, existing line data sets. To invoke the text editor, issue the EDIT command. The operand of this command is the name of the data set to be created or edited. Suppose that you want to create a data set named RHYMES, in which you intend to store nursery rhymes. (As in Chapter 2, we are deliberately using trivial data so that you can concentrate on the method of manipulating data rather than on the data itself.) To create this data set, issue the EDIT command:

```
edit rhymes
```

The system will respond by issuing the first line number of the new data set RHYMES:

```
0000100
```

You enter your first line of data and press the RETURN key:

```
0000100 jack and jill went up the hill
```

The system will then prompt you with the second line number, following which you enter the second line of data, and so on. Your entries and the system-supplied line numbers might look like this:

```
0000100 jack and jill went up the hill
0000200 jack fell done and broak his crown,
0000300 and jill came tumbling after.
0000400
```

The system has prompted you for your fourth line of data when you notice, in reading back over your entries, that you have left out a line of the rhyme and have made some spelling errors in line 200.

You can insert the missing line with the INSERT command. But the system has just prompted you with another line number; if you type in the INSERT command now, the system will interpret it as a line of data, and will enter the line into the data set rather than executing the command.

To tell the system that you want to enter a command now instead of a line of data, you enter the *break character* in the first character position of the line for which the system prompted you—that is, right after the line number. The break character is the underscore on the IBM 1052 and 2741 terminals; on the teletypewriter terminal, it is the right bracket, which is the uppercase character on the "M" key. After receiving the break character, the system will expect a command instead of a data line.

You enter the break character:

```
0000400 _
```

Now you can enter the INSERT command. As its operands, enter the line number of the exiting line that you want your inserted line to follow, and the increment that you want the system to add to this existing line number to produce the line number for your inserted line. You enter the command after the break character:

```
0000400 _insert 100,50
```

This command tells the system that you want to insert one or more lines after line 100, and that the first inserted line should have the line number 150. The system responds by prompting you with the line number 150:

```
0000150
```

. . . and you enter the line to be inserted:

```
0000150 to fetch a pail of water.
```

The system will not prompt you with line number 200, since there is already a line numbered 200 in the data set. Instead, it will prompt you for

another command. Perhaps you'd like to check and see that your insertion was made correctly. Issue the LIST command, giving the line numbers of the first and last lines of the range of lines you want to review:

```
list 100,200
```

The system will then print out on your terminal the data lines from 100 to 200 inclusive:

```
0000100 JACK AND JILL WENT UP THE HILL
0000150 TO FETCH A PAIL OF WATER.
0000200 JACK FELL DONE AND BROAK HIS CROWN,
```

Notice that the insertion was made correctly, and that line 200 was not affected. The INSERT command allows you to make insertions between existing lines—you could have inserted as many as 99 lines between lines 100 and 200 by specifying a line-number increment of 1—but it does not allow you to change existing lines.

Notice also that the system has translated the lowercase letters which you typed when entering the data lines into capital letters. As mentioned, the system performs this translation to save you from having to use the SHIFT key frequently.

To correct the spelling errors in line 200, use the REVISE command. The REVISE command removes existing lines from a data set and allows you to replace them. If you want, you can replace a single line with several new lines.

To remove line 200 and replace it with two lines, you issue the REVISE command:

```
revise 200,incr=50
```

The INCR operand tells the system to increment the line number 200 by 50 when prompting you for additional lines. It responds by prompting you for data, which you enter:

```
0000200 jack fell down
0000250 and broke his crown
```

The REVISE command, like the INSERT command, terminates execution when the increment specified produces a line number equal to or greater than that of a line already in the data set.

When you are prompted for another command, you enter an INSERT command to let you add lines to the end of the data set:

```
insert last
```

Here the operand LAST specifies the highest-numbered line now in the data set. This form of relative addressing can be used with the INSERT, LIST, and REVISE commands. Using it with the INSERT command allows you to add lines at the end of a data set.

Since no increment is specified in your INSERT command, the default increment of 100 is used, and the system prompts you with the line number 100 more than the line number of the last line now in the data set:

```
0000400
```

You can now enter another rhyme:

```
0000400 mary, mary, quite contrary,
0000500 how does your garden go?
0000600 snips and snails and puppy-dog tails
0000700 and pretty maids all in a row.
0000800
```

Only after entering these lines do you notice that your memory has played tricks on you and you have inserted a line from another poem into this one. Also, you have entered "how does your garden go?" instead of "how does your garden grow?"

To correct this, enter a break character and then the REVISE command, specifying that you want to remove all of the lines from 500 to 600 and replace them. You default the increment this time, since you plan to replace two lines with two new lines:

```
0000800 _revise 500,600
```

The system prompts for the replacement lines and you enter them:

```
0000500 how does your garden grow?
0000600 silver bells and cockle shells
```

As before, the system terminates execution of the command when line numbers match.

To review your work so far, issue the break character followed by the LIST command. You have seen that the LIST command can specify a single line number, a range of line numbers, or a relative line (such as LAST). If you enter no operands, the LIST command causes the entire data set to be displayed:

```
list
0000100 JACK AND JILL WENT UP THE HILL
0000150 TO FETCH A PAIL OF WATER.
0000200 JACK FELL DOWN
0000250 AND BROKE HIS CROWN
0000300 AND JILL CAME TUMBLING AFTER.
0000400 MARY, MARY, QUITE CONTRARY,
0000500 HOW DOES YOUR GARDEN GROW?
0000600 SILVER BELLS AND COCKLE SHELLS
0000700 AND PRETTY MAIDS ALL IN A ROW.
```

You decide to stop editing this data set until you find another good poem. When the system prompts with an underscore, you issue the END command:

```
end
```

This terminates text editing. You could also have issued another EDIT command, with a different data set name; this would have terminated editing of the first data set and allowed you to start editing the next one.

Later you return to the data set you have named RHYMES. You issue the EDIT command:

```
edit rhymes
```

. . . and the system responds by prompting with the next available line number:

```
0000800
```

You insert a line giving the title of the first poem:

```
0000800 _insert 0,50
0000050 jack and jill
0000100
```

Next you remove all the lines of the second poem:

```
0000100 _revise 400,last
0000400 _end
```

When no data is entered following the REVISE command, the specified lines are removed without being replaced. The END command concludes your editing of this data set.

Creating Virtual Sequential Data Sets

You may wish to create a virtual sequential data set that is to be used with FORTRAN I/O routines. In this example, each line will consist of a number in F6.2 format. You enter a DATA command with a name for the data set:

```
data numbers
```

The system prompts you to enter a line by issuing a pound sign followed by a space. You enter the line and are prompted again:

```
# 240.00
#
```

When you have entered all the lines you wish, you indicate the end of your data set by responding to the prompt with %e:

```
# 125.50
# %e
```

The system then prompts you with an underscore for your next command.

If at a later time you wish to add some more lines to the data set, you simply issue another DATA numbers command with the same data set name as operand, and the system will position you to the end of the data set and prompt you for additional lines. Note that the DATA command is used in creating virtual sequential data sets—the text editor cannot be used to modify such data sets once they have been created.

Review

- The text editor is invoked with the EDIT command.
- Text editing is terminated with the END command or with another EDIT command.
- The text editor creates and alters virtual index sequential data sets. For the purposes of this book, these are restricted to data sets consisting of lines indexed by seven-digit line numbers in the first position of each line.
- The break character tells the system to expect a command rather than a line of data.
- The INSERT command inserts one or more data lines between existing lines in a data set.
- The REVISE command removes a specified line or range of lines, which can then be replaced with other lines.
- The LIST command allows you to review a line, a range of lines, or an entire line data set.
- The DATA command allows you to create or add to a virtual sequential data set.

Four: PRINTING AND ERASING DATA SETS

In Chapter 3 you learned that you can cuase an entire data set to be printed out on your terminal by issuing the LIST command with all operands defaulted. However, this is a slow process compared to having the data set printed on the high-speed printer at the central computer installation. Use of the LIST command to print out an entire data set at the terminal should be restricted to short data sets.

To print a data set on the installation's printer, issue the PRINT command:

```
print rhymes
```

If you will no longer need the data set after having it printed, follow the data set name with the ERASE operand:

```
print rhymes,erase=y
```

The ERASE option will cause the data set to be erased after it has been printed. If ERASE=N is entered, or if the option is omitted, the data set will not be erased.

The system will respond to the PRINT command with a message like this:

```
PRINT BSN 0285
```

"BSN 0285" in this sample message stands for "Batch Sequence Number 0285." The batch sequence number, which the system assigns, will help you find your printout when you go to the computer room for it.

The system may not execute your PRINT command immediately; another user may be using the printer. The command will be executed as soon as a printer is available. After issuing a PRINT command, you must not use that data set until it has been printed.

Erasing Data Sets

You can erase a data set after printing by using the ERASE option of the PRINT command. You can also use the ERASE command. To erase the data set RHYMES without printing it, you could enter this command:

```
erase rhymes
```

Entering the ERASE command with no operand results in a review of the names of all of your data sets; as each name is presented, you can enter E (for erase) or R (for retain) to erase or retain that data set. Entering an A (for all) erases all cataloged data sets for which you have not specified R.

Entering the ERASE command with a partially qualified data set name as the operand results in a review of all of the data sets with that partially qualified name. Suppose you had created several source data sets from which you had compiled executable programs; you want to erase the source data sets you no longer need, and retain those which you may need to modify and recompile. Issue the ERASE command with the partially qualified name SOURCE:

```
erase source
```

The system will then present the fully qualified name of each of your source data sets and invite you to enter E, R, or A for each.

To erase an object program, use as the operand of the ERASE command the name of the library in which the program is stored, USERLIB, followed by the program name in parentheses. For instance, to erase an object program named ADDUP, you would issue this command:

```
erase userlib(addup)
```

This would erase only the object program ADDUP. If you had not previously erased the source program SOURCE.ADDUP, from which you compiled the object program, it would remain in the system after execution of this ERASE command. You could erase SOURCE.ADDUP separately, or modify it and recompile to produce a modified object program with the name ADDUP.

Do *not* issue the ERASE command with only USERLIB as an operand. This would erase your user library and all the programs in it, as well as control information that the system needs to provide your operating environment, as described in Appendix B.

Review

- The PRINT command can be used to cause a data set to be printed on the high-speed printer at the computer installation. The data set may be erased after printing by using the ERASE option of the PRINT command.
- The ERASE command can be used to erase a data set. Used with a partially qualified data set name or with no operand, it allows you to review data sets and decide whether to erase or retain each.
- Do *not* erase USERLIB. This would destroy not only all of your programs stored there, but other information the system needs to provide your operating environment.

Five: COMPILING FORTRAN PROGRAMS

TSS/360 includes a compiler to process source programs written in the IBM FORTRAN IV language and produce an object program that can be loaded and executed. Let's use the text editor to create a source data set for a simple FORTRAN program. Issue the EDIT command:

```
edit source.powers
```

SOURCE.POWERS is the name of the source data set to be created; POWERS will be the name of the executable program to be compiled from this source data set. The system prompts with line numbers and you enter your source statements as lines of data:

```
0000100      m=1
0000200      n=5
0000300 2    do 1 i=m,n
0000400      j = i**2
0000500      k = j*i
0000600      kk = k*i
0000700 1    print 3,i,j,k,kk
0000800 3    format (1x, i3, i4, i5, i6)
0000900      stop
0001000      end
0001100 _end
```

Line 1000 contains the END statement that will tell the FORTRAN compiler that it has reached the end of your source program. When the system

prompted for line 1100, you entered the break character and the text-editor END command, which terminates text editing. Don't confuse the END statement of FORTRAN with the text editor END command.

If you make mistakes in entering your source statements, correct the erroneous lines using the text-editor INSERT and REVISE commands as illustrated in Chapter 3.

Having created your source data set, you now want to compile the source statements. The system will prompt for another command following your END command; after being prompted, enter the FTN command to invoke the FORTRAN compiler:

```
ftn powers,y
```

As shown here, the FTN command has two operands—the name of the program to be created, and an indication of whether the source statements for that program have been stored in the system as a data set.

POWERS is the name of the program to be created. Notice that you do not give the name of the data set you have just created. SOURCE.POWERS is the name of the source data set for your program; the name of the executable program will be POWERS. It's important to understand that these are two separate data sets—one containing FORTRAN source statements, the other containing executable machine instructions.

The second operand of this FTN command, Y, stands for "Yes" and tells the system that your source statements are already stored as a data set. The system will now look for the data set named SOURCE.POWERS and start processing the source statements in that data set to create the executable program POWERS.

If the compiler detects an error in the source statements, it issues a diagnostic message at the terminal, prints out the line in which the error was found, and invites you to correct the error. The source statements in SOURCE.POWERS have been entered correctly so that you can copy them easily if you want to. But suppose that the statement number had been omitted from the FORMAT statement. When the compiler reached the FORMAT statement, it would halt its scan, issue a diagnostic message giving the line number and contents of the erroneous line, and invite you to enter a correction:

```
0000800    E *** FORMAT STATEMENT DOES NOT HAVE
           STATEMENT NUMBER
0000800    FORMAT (1X, I3, I4, I5, I6)
#
```

The pound sign (#) invites your correction. Enter the line number, a comma, and the corrected line:

```
#800,3 format (1x, i3, i4, i5, i6)
#
```

After you enter your corrected line, the system prompts with another pound sign in case you need to enter more than one line. To tell the system that you have finished making corrections, press the RETURN key to enter a null line. The compiler will then resume processing your source statements.

Correction lines entered after the pound-sign prompt may be either replacement or insertion lines. You can also delete a line or a range of lines. To do this, enter a D immediately after the pound sign, followed by a comma, followed by the line number of the line you want to delete. To delete line 600, for instance, you would enter:

```
#d,600
```

To delete a range of lines, enter the D, the comma, the line number of the first line in the range, another comma, and the line number of the last line in the range. For instance:

```
#d,500,700
```

If the compiler has detected errors while scanning your source statements, it gives you a chance to enter corrections:

```
MODIFICATIONS?
```

If you enter Y for "Yes," you will be prompted for corrections with the pound sign as described above. If you enter N for "No," processing will continue.

If the compiler finds no more errors, you will be returned to command mode. The executable program POWERS is now stored in your library of user programs, ready to be executed.

Line-by-Line Compilation

Instead of creating a source data set and then compiling the source statements in it, you can also invoke the FORTRAN compiler and enter your source statements for line-by-line scanning.

To do this, enter the FTN command in this form:

```
ftn inout,n
```

The N as the second operand of this command indicates that your source program is NOT already stored in the system as a data set. When there is no second operand of the FTN command, the system assumes the value N, so you could also have entered the FTN command in this form:

```
ftn inout
```

When N is specified as the second operand of the FTN command, or when the program name is the only operand entered, the system will prompt you with line numbers for the lines of your FORTRAN source program. Then, as you enter the source statements, the system will both create the source data set and compile the source statements into an executable program.

The system prompts with line numbers and you enter your source statements. The statements may be entered free form. (Note that the FORTRAN compiler does not provide a blank space between the line numbers and the first column you enter, as the text editor does.)

```
0000100    print  2
00002002   format  (1x, 'program started')
00003003   read    (5,4, end=8, err=5) figure
0000400    format  (1x, f6.2)
```

You have provided no statement number in entering the FORMAT statement. The system issues a diagnostic message and invites a correction immediately, instead of prompting you for the next line. You enter the corrected line:

```
0000400  E *** FORMAT STATEMENT DOES NOT HAVE
           STATEMENT NUMBER
#400,4     format (1x, f6.2)
#
```

After the second pound sign, you signal the end of your modifications by pressing the RETURN key to enter a null line. The system then prompts you with the next line number and you enter source statements as before:

```
00005005   write  (6,7) figure
00006007   format (1x, f10.2)
0000700    go to 3
00008008   stop
0000900    end
```

The END statement indicates the last line of your source program. After encountering it, the system will stop prompting you with line numbers and finish scanning the source program for syntax. It will ask for modifications and return you to command mode when it has completed an error-free compilation. The executable program INOUT is now stored in your library of user programs, ready to be run.

Source Listings

Now that you have successfully compiled the programs POWERS and INPUT, you may want to print the source data sets to provide you with a record of the source programs. Issue these PRINT commands:

```
print source.powers

print source.inout
```

The system will assign a batch sequence number to each PRINT command; these numbers will help you find the printout at the central computer installation.

If you are sure your programs are correct, you may want to add the operand ERASE=Y after the data set name in each of these PRINT commands, causing the system to erase your source data sets after printing them.

On the other hand, you may want to wait until you have run the programs and made sure that they work correctly. If one of them gives incorrect results, you may want to modify the source data set and recompile.

If you decide not to use the ERASE option of the PRINT command, you can erase the source data sets when you no longer need them with the ERASE command.

Review

- Source statements can be compiled after being stored in the system as a data set, or entered line by line from the terminal for compilation.
- The operands of the FTN command, which invokes the FORTRAN compiler, give the name of the executable program to be created and indicate whether the source statements are stored as a data set or will be entered from the terminal.
- The compiler issues a diagnostic message when it discovers an error, and then invites correction.
- You can use the PRINT command to obtain a listing of your source data set.

Six: EXECUTING A PROGRAM

Now let's execute one of the two programs that you compiled in the previous chapter. To invoke a program, issue the CALL command with the program name as the operand:

```
call powers
```

The system will load the program POWERS and give it control. POWERS loops through a sequence of instructions that squares, cubes, and raises to the fourth power a number I, the value of which is incremented from 1 to 5. (See Figure 2.)

```
      M=1
      N=5
2     DO 1 I=M,N
      J = I**2
      K = J*I
      KK = K*I
1     PRINT 3,I,J,K,KK
3     FORMAT (1X, I3, I4, I5, I6)
      STOP
      END
```

Figure 2. Source statements for programs POWERS.

The program prints the results at your terminal:

```
1     1     1     1
2     4     8     16
3     9     27    81
4     16    64    256
5     25    125   625
TERMINATED: STOP
```

The last line of the printout, TERMINATED: STOP, indicates that the program has reached its end, corresponding to the STOP statement in your source data set.

Obviously, you will get the same results every time you run this program unless you can change the values of the variables M and N. With TSS/360 you can do so, using program-control commands.

First, issue the QUALIFY command:

```
qualify powers
```

This command tells the system that you are going to refer to internal symbols in the program POWERS.

You can use the SET command to change the values of the variables M and N. But if you do this now, and then execute the program, the variables will be restored to their original values by the instructions that correspond to the statements M=1 and N=5 in your source program.

What you need is a way of altering the values of these variables just before the program executes the DO statement. With TSS/360 you can do this with a *dynamic command statement.*

A dynamic command statement starts with the AT command, which indicates at what point in your program the command statement is to be executed. The commands following the AT command at the command statement are delimited by semicolons.

You want to use the SET command to alter the values of the variables M and N just before the DO statement in your program is executed. The DO statement has a statement number of 2. Enter this dynamic command statement:

```
at 2; set  m=5,n=7
00001
```

The system will respond by assigning a reference number (00001) to your dynamic command statement. When the command statement is executed, the system prints out this number; so if there are several command statements implanted in a program, the reference number helps you tell which one has been executed.

Now issue the CALL command as before:

```
call powers
```

Execution is initiated. When the program reaches the DO statement, the dynamic command statement is executed, and the system informs you by printing the reference number of your statement:

```
00001
```

The program then completes execution with the values M=5 and N=7. Its output looks like this:

```
5   25  125  625
6   36  216 1296
7   49  343 2401
TERMINATED: STOP
```

The dynamic command statement implanted in your program will be executed every time the program is executed, until the program is unloaded; then the command statement will be removed.

You have seen an example of how program-control commands can be used. The full use of all the program-control commands is described in *Command*

System User's Guide, Form C28-2001. These commands can be used to debug programs dynamically through the dynamic command statement as illustrated above. With them you can display and alter the contents of variables and data fields and analyze the results; and you can then correct your source program based on your observations and compile a corrected object program.

However, you can use TSS/360 without employing the program-control commands; the only one that you need to know is the CALL command to invoke your programs.

(To use the program-control commands as shown here, you must have an Internal Symbol Dictionary produced when the program is compiled, as noted in Appendix B.)

Review

- The CALL command, with the name of a program as operand, invokes that program.
- The QUALIFY command tells the system you will be referring to internal symbols in the program named as the operand.
- The SET command can be used to alter the contents of variables or data fields.
- The AT command can be used to form a dynamic command statement; the commands following the AT command will not be executed until the instruction location named in the AT command is reached.
- Semicolons delimit commands in a command statement.

Seven: PROGRAM INPUT/OUTPUT

In TSS/360, the data set reference number in a FORTRAN I/O statement does not identify the data set directly. It links the program to the data set through a control block which the system builds based on information supplied by the user in a DDEF command. This method of linking program to data sets allows flexibility; one program can be linked to any available data set by supplying the correct linkage with the DDEF command.

DDEF stands for "data definition." The DDEF command defines to the system the characteristics of a data set that you want to create or process. For the purposes of this book, the DDEF command has been simplified so that you need to specify only the name of the data set and two other operands—the data definition name (DDNAME) and the data set organization.

The DDNAME is the link between the program and the control block that the system builds from the information in your DDEF command. You can think of this control block as containing the DDNAME and the data set name. The program is linked to the control block through the DDNAME, and the control block is linked to the data set through the data set name. See Figure 3.

The DDNAME is formed from the data set reference number in the FORTRAN I/O statement. The data set reference number is a two-digit decimal number from 01 to 99. For a data set reference number of 55, for instance, the DDNAME would usually be FT55F001; for a data set reference number of 99, the DDNAME would be FT99F001.

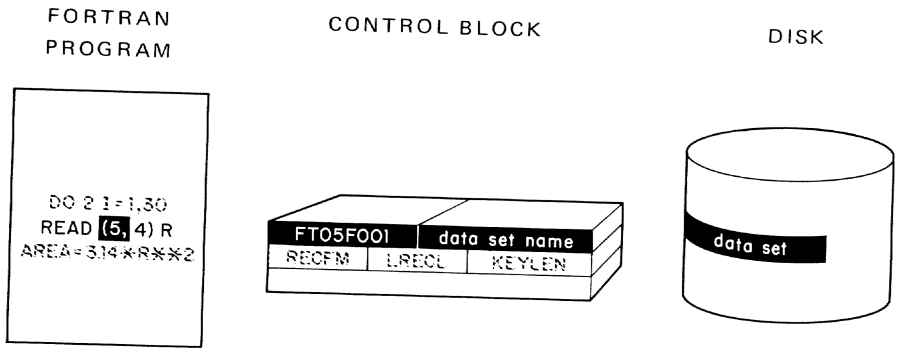


Figure 3. The program is linked to the data set through the control block that the system builds from information in the DDEF command.

In Chapter 5 you compiled a brief FORTRAN program that reads numbers from one data set and writes these numbers into another data set. The data set reference number in the READ statement was 5. (See Figure 4.) Before running that program, issue a DDEF command to link that READ statement to a data set named NUMBERS, a virtual sequential data set that you created with a DATA command (as explained in Chapter Three) or with another FORTRAN program. The DDEF command is:

```
ddef ft05f001,vs,numbers
```

(where *ft05f001* is the DDNAME, *vs* stands for virtual sequential organization, and *numbers* is the data set name).

The system will then build the control block linking the program to the data set. If the data set NUMBERS has already been referenced by commands during this terminal session—if you have just created it during your current task, for instance—there may be a control block already built for it. In this case, the system will issue you a message telling you that the system-supplied DDNAME has been replaced by the DDNAME in your DDEF command, FT05F001.

If you execute it now, your FORTRAN program will read from the data set NUMBERS. Issue the CALL command:

```
call inout
```

```
      PRINT 2
2     FORMAT (1X, 'PROGRAM STARTED')
3     READ   (5,4, END=8, ERR=5) FIGURE
4     FORMAT (1X, F6.2)
5     WRITE  (6,7) FIGURE
7     FORMAT (1X, F10.2)
      GO TO 3
8     STOP
      END
```

Figure 4. Source statements for program INOUT.

The system will load INOUT from your library of programs and initiate execution. The first output will be the message PROGRAM STARTED. Then the program will read input records from the data set. (The *1x* in statement 4 causes the READ command to skip over a control character.)

But you have not defined a data set for the WRITE statement in the program to link to. The system will therefore assume that you want the records written out on your terminal. When no DDEF command is issued to provide a DDNAME that includes a data set reference number that is used in a program, the system assumes that the records are to be read from or written to the terminal. Your program's WRITE statement is linked to your terminal and the numbers from the data set NUMBERS are printed on the terminal.

Now that you are sure your program runs correctly, you can use it to process other data sets. First, however, you must cancel the effect of your previous DDEF command. To do this, issue the RELEASE command:

```
release ft05f001
```

This eliminates the system control block containing the DDNAME of FT05F001. Now your program can be linked to another data set. To link it to a data set named FIGURES, issue another DDEF command, and then issue a CALL command to run your program:

```
ddef ft05f001,vs,figures; call inout
```

The program will print out the message PROGRAM STARTED and then read numbers from the records of the data set FIGURES.

If no DDEF command with the DDNAME of FT05F001 is issued, the system will link your program to your terminal and the program will read input records from the terminal. You may want to try this. Remember that the FORMAT statement in your program causes the first byte of the input records to be skipped.

When you have finished entering data from your terminal, enter %END the next time the system expects an input record. This will cause the system to recognize an end-of-data-set condition, and execution of your program will terminate.

With the proper FORMAT statement, a FORTRAN WRITE statement can be used to output records to a new data set defined by a DDEF command with the corresponding DDNAME. (For a data set reference number of 6, the DDNAME would be FT06F001.) For information on writing FORMAT statements, see *IBM FORTRAN IV*, Form C28-2007.

Now you can see the flexibility that the DDEF command provides. You can link your FORTRAN program to any existing data set, or use it to create a new data set, with the proper DDEF commands; or, by defaulting the DDEF command, you can link either of the FORTRAN I/O statements in your program to your terminal for input or output.

Review

- A program is linked to a data set through a control block which the system builds from information that is supplied in the DDEF command.
- The data definition name, or DDNAME, is the link between the program and the control block.
- The DDNAME is formed from the data set reference number in the FORTRAN I/O statement; for instance, the reference number 5 would result in the formation of the data definition name FT05F001.
- When no DDEF command is issued with a DDNAME corresponding to a data set reference number, the I/O statement containing that reference number is linked to the terminal for input or output.

Eight: INTERRUPTING THE SYSTEM

At times you may find it necessary to interrupt the system while it is executing a command, running a program, or printing out a message. For instance, suppose you have entered the FTN command, intending to compile from lines entered at the terminal. But you absent-mindedly entered the second operand as Y, indicating that the source program is already stored in the system as a data set:

```
ftn prog2,y
```

The system will search for a data set named SOURCE.PROG2. When it does not find one, it will issue a message at your terminal:

```
SOURCE.PROG2 DOES NOT EXIST. REENTER
```

Now you want to re-enter your FTN command correctly, but if you do so, the system will interpret it as a program name instead of a command.

To get out of this loop, generate an *attention interrupt* with the attention key. (On the IBM 2741 terminal, this key is at the upper right-hand corner of the keyboard, marked ATTN. On the IBM 1052, it is at the lower left-hand corner, marked either ATTENTION or RESET LINE. On the teletypewriter, it is to the left of the keyboard, marked BREAK; on this terminal, you must also press the BRK-RLS key to unlock the keyboard after an attention interrupt.)

The attention interrupt will halt the system's execution of your FTN command. (It can also be used to halt execution of the FORTRAN compiler, of a user program, or of a message being printed on the terminal.) The system will prompt you for another command by printing a logical-NOT sign (¬), an exclamation mark (!), or an underscore and backspace (or the equivalents on the teletypewriter). The prompt character will vary depending on what has been interrupted. (See Table 1.)

Review

- You can interrupt the system with the attention key.
- The attention key is labeled ATTN on the 2741 terminal, ATTENTION or RESET LINE on the 1052, and BREAK on the teletypewriter. On the teletypewriter, you must press the BRK-RLS key after an attention interrupt.
- After the interrupt, the system will respond with a prompt for a command. The prompt character will vary depending on what has been interrupted.

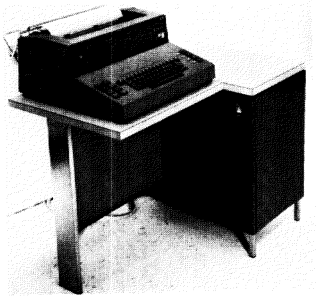
Table 1. Effect of attention interrupt.

Situation When Interrupt Occurs	Effect
A line is being entered; the RETURN key has not been pressed.	All characters in the line as entered so far are ignored. You are prompted with a logical-NOT sign (¬) and may enter any command.
The system has just issued a message asking for information to be entered.	The command in operation—that is, the last one transmitted to the system—is canceled, and you are prompted with a logical-NOT sign (¬). You may enter any command.
A command is being executed.	The command is completed before the attention interruption is recognized, unless the system issues a message saying that a command has been canceled.
A message is being printed out.	The remainder of the message is canceled. You are prompted with a logical-NOT sign (¬) and may enter any command.
The FORTRAN compiler is in operation.	Language processing stops and the system prompts with either an exclamation mark (!) or a logical-NOT sign (¬). You can enter any command. You may resume language processing from the point of interruption by issuing the GO command.
A user program is in operation.	You are prompted with an exclamation mark (!) and may enter any command. To resume the program from the point of interruption, issue the GO command.

Appendix A: HOW TO USE TSS/360 TERMINALS

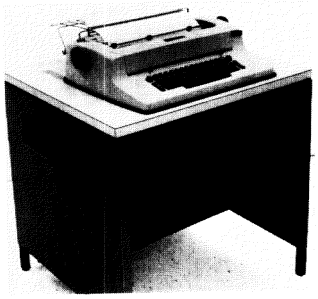
Basic information on the three types of terminal used with TSS/360 is given in this appendix. If more information is needed, see *IBM System/360 Time Sharing System: Terminal User's Guide,* Form C28-2017.

If your terminal looks like this,



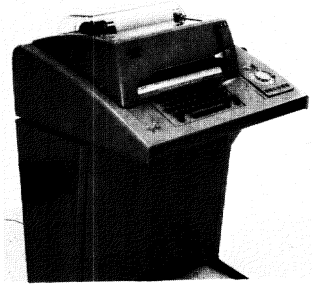
turn to page 38

If your terminal looks like this,



turn to page 42

If your terminal looks like this,



turn to page 45

IBM 1050 DATA COMMUNICATIONS SYSTEM

The IBM 1050 Data Communications System as used with TSS/360 includes an IBM 1051 Control Unit, a 1052 Printer-Keyboard, and, optionally, a telephone-like modulator-demodulator, or *modem*. Figure A-1 illustrates the printer-keyboard mounted on the control unit. The modem is used to dial up the time-sharing system.

Figure A-1. 1051 Control Unit and 1052 Printer-Keyboard.

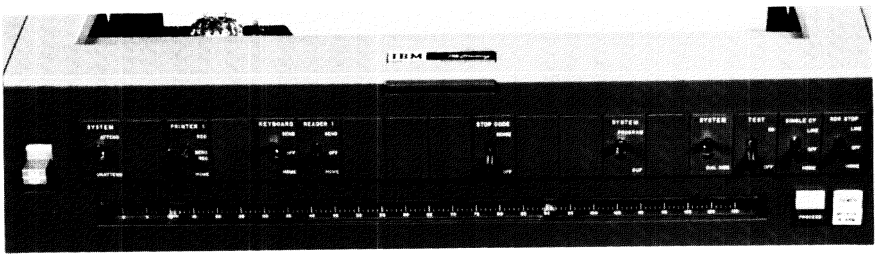


Figure A-2. 1052 switch panel.

Initiation Procedure—To ready the IBM 1050 for use with TSS/360, proceed as follows:

1. Set the panel switches on the IBM 1052 Printer-Keyboard as directed in Table A-1. (A typical panel is shown in Figure A-2.) If the 1052 has additional switches, set them to the OFF or HOME position. Do not change the switch positions while using the terminal.
2. Turn on the main-line switch indicated in Figure A-3. The POWER light should come on. If the DATA CHECK light is on, turn it off by pressing the DATA CHECK key. (See Figure A-3.)

3a. If the terminal is directly connected to the computer, initiate the LOGON procedure by pressing the ATTENTION/RESET LINE key.

3b. If the terminal has a modem that resembles a telephone, press the TALK button on the modem, dial the TSS/360 number and, when a continuous high-pitched tone is heard, press the DATA button on the modem. The terminal is now connected to the time-sharing system. The receiver of the modem can be replaced in its cradle.

Table A-1. 1052 switch settings.

Switch	Setting	Toggle Position
SYSTEM	ATTEND	up
PRINTER 1	SEND REC	middle
KEYBOARD	SEND/ON	up
READER 1	ON/SEND	up
STOP CODE	OFF	down
SYSTEM	PROGRAM	up
SYSTEM	—	up
TEST	OFF	down
SINGLE CY	OFF	middle
RDR STOP	OFF	middle
Note: Set all other panel switches to OFF or HOME position.		

Figure A-3. 1052 keyboard.

Keyboard Operation——The keys and controls on the keyboard are illustrated in Figure A-3. The numeric and special-character keys, the space bar, and the SHIFT, LOCK, and TAB keys operate like their counterparts on standard typewriters. Note, however, that no distinction is made between capital and lowercase alphabetic characters (A through Z); they are all interpreted by the system as capital letters.

PROCEED Light——When the green PROCEED light is on (see Figure A-2), the keyboard is unlocked and data or commands can be entered. As soon as a line has been entered, the keyboard is locked; the PROCEED light turns off shortly afterwards. All keys except the ATTENTION/RESET LINE key are locked out while the PROCEED light is out.

Attention Key——The ATTENTION/RESET LINE key in the lower left-hand corner of the keyboard cannot be locked out. It generates an attention interruption. The attention interruption is explained in Chapter 8. (The ATTENTION key may also be used, as explained above, in initiating the LOGON procedure from a directly connected terminal.)

RETURN Key——Pressing the RETURN key causes a line feed and carrier return at the terminal printer and transmits an end-of-block character to the system. After the RETURN key is pressed, the keyboard is locked out (except for the ATTENTION key) and control passes to the system.

Continuation Lines——When the hyphen is entered as the last character in a line, the system recognizes the next line as a continuation. The hyphen is not entered as part of the line.

Canceling Lines——When a pound sign (#) is entered as the last character before the RETURN key is pressed, the entire line is canceled. The system will then expect the corrected line to be entered without additional prompting. The pound sign is defined as the *line-kill character.*

A line may also be canceled with the CANCEL key. To do this, hold down the ALTN CODING key at the upper left-hand side of the keyboard and press the zero key.

Correcting Lines--A line that you have started to enter incorrectly can be corrected by backspacing to the first incorrect character with the BACKSPACE key and re-entering the line from that point on.

DATA CHECK and RESEND--The DATA CHECK light may come on after the terminal is first turned on; it can be extinguished with the DATA CHECK key. The RESEND light will come on briefly after the RETURN key is pressed; it should turn off when the line has been accepted by the system. If the DATA CHECK and RESEND lights are on together, an error is indicated. While the RESEND light is on, the system will not accept input from the terminal keyboard. Press the DATA CHECK and RESEND keys to turn off the lights and re-enter the line.

Termination Procedure--After the LOGOFF command has been accepted by the system, close down the terminal by setting the main-line switch to POWER OFF.

IBM 2741 COMMUNICATIONS TERMINAL

The IBM 2741 consists of an IBM SELECTRIC typewriter mounted on a stand that includes the electronic controls needed for communication with TSS/360. Its keyboard may be compatible with that of the Selectric typewriter, or it may be as shown in Figure A-5.

If the terminal is directly connected to the system, merely turning on the terminal results in connection with the system. If not, it can be connected to the system through a modulator-demodulator, or *modem,* that resembles a telephone.

Initiation Procedure—To ready the 2741 for use with TSS/360, proceed as follows:

1. Check that the terminal mode switch on the left side of the stand (see Figure A-4) is set to COM.

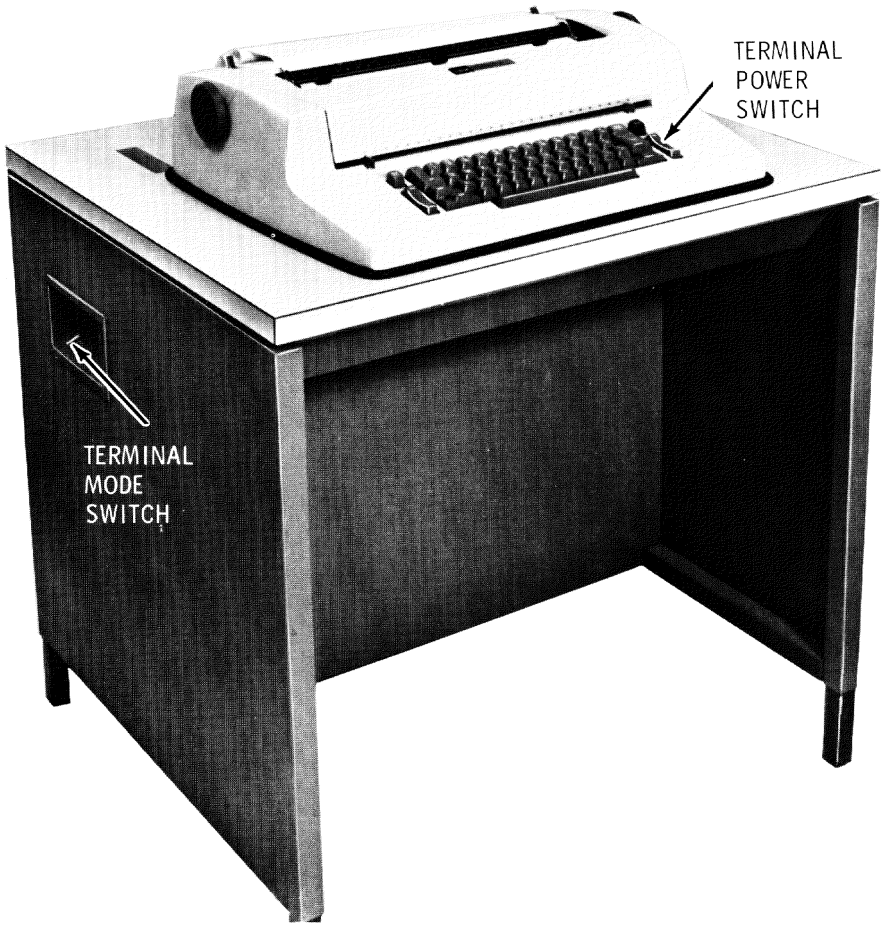


Figure A-4. 2741 Communications Terminal.

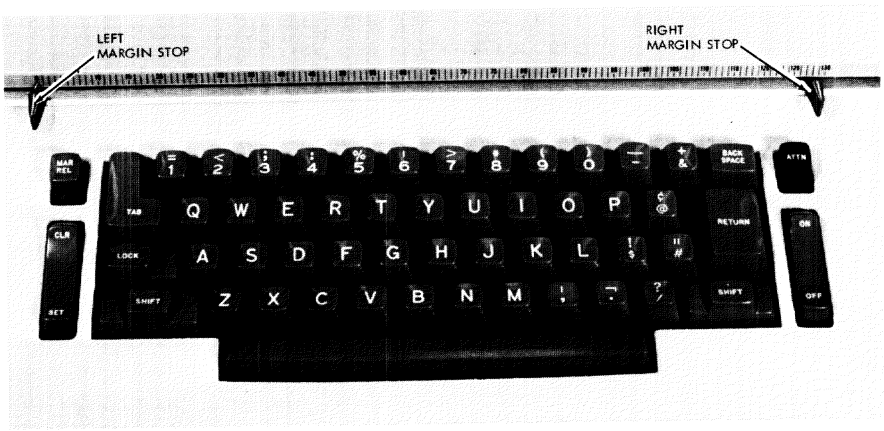


Figure A-5. 2741 keyboard.

2. Press on the ON portion of the terminal power switch (see Figure A-4).

3a. If the terminal is directly connected to the computer, initiate the LOGON procedure by pressing the ATTN key at the upper right-hand corner of the terminal keyboard.

3b. If the terminal has a telephone-like modem, press the TALK button, lift the receiver, dial the TSS/360 number, and, when you hear a continuous high-pitched tone, press the DATA button. The terminal is now connected to the system. The receiver of the modem can now be placed in its cradle.

Keyboard Operation—The terminal keyboard works like an IBM Selectric typewriter except for the ATTN key, which is used to generate attention interrupts as explained in Chapter 8. (It may also be used in initiating the LOGON procedure from directly connected terminals, as explained above.) The system unlocks the keyboard when it is expecting input; at other times, the keyboard is locked. The ATTN key is the only key that cannot be locked out.

Note that the system recognizes no distinction between capital and lowercase letters; they are all interpreted as capital letters. This saves the user from having to use the SHIFT key in entering commands, which consist only of capital letters.

Equivalencies—The 2741 terminal with the keyboard compatible with the Selectric typewriter (known as the 2741 correspondence terminal) lacks several characters used in TSS/360. The character translation tables used by the system provide these equivalencies:

- The plus-or-minus sign (±) or the left bracket ([ ), whichever is present, is translated into the logical OR sign (|).
- The right bracket ( ]), if present, is translated into the numeral one.
- The degree sign (°), for which there is no internal code, is translated into the less-than sign (<).

RETURN Key—Pressing the RETURN key causes a line feed and carrier return at the terminal and transmits an end-of-transmission character to the system. The RETURN key must be pressed to end every line of input from the keyboard. After the RETURN key has been pressed, the keyboard is locked out (except for the ATTN key) and control passes to the system.

Continuation Lines—When the hyphen is entered as the last character in a line, the system recognizes the next line as a continuation. The hyphen is not entered as part of the line.

Canceling Lines—When a pound sign (#) is entered as the last character before the RETURN key is pressed, the entire line is canceled. The system will then expect the corrected line to be entered without additional prompting. The pound sign is defined as the *line-kill character.*

Correcting Lines—A line that you have started to enter incorrectly can be corrected before the RETURN key is pressed by backspacing to the first incorrect character with the BACKSPACE key and re-entering the line from that point on.

Termination Procedure—After the LOGOFF command has been accepted by the system, close down the terminal by setting the terminal power switch to OFF.

TELETYPE MODEL 33/35 KSR

The Teletype* Model 33 or 35 KSR (Keyboard Send-Receive) consists of a printer, a four-row keyboard, and a control unit, all mounted in a special cabinet. (See Figure A-6).

Initiation Procedure—To ready the teletypewriter for use with TTS/360, proceed as follows:

1. Press the ORIG button. The lamp should light under the button, and the teletypewriter will be on.
2. Dial the TSS/360 number with the telephone dial. A continuous tone will be heard momentarily as the connection is made. The LOGON procedure will then begin.

Keyboard Operation—The alphanumeric and special-character keys, the space bar, and the SHIFT key all work like their counterparts on conventional typewriters (except that the SHIFT key does not lock in the down position). Only capital letters are provided; lowercase letters are not. the BREAK key is used to generate an attention interrupt, as explained in Chapter 8. After using the BREAK key, you must press the BRK-RLS key above the telephone dial to unlock the keyboard.

Do not use the keyboard when the system is not expecting input. The keyboard is not locked when the system is not expecting input, and pressing a key at such a time will cause the equivalent of an attention interrupt.

*A trademark of the Teletype Corporation.

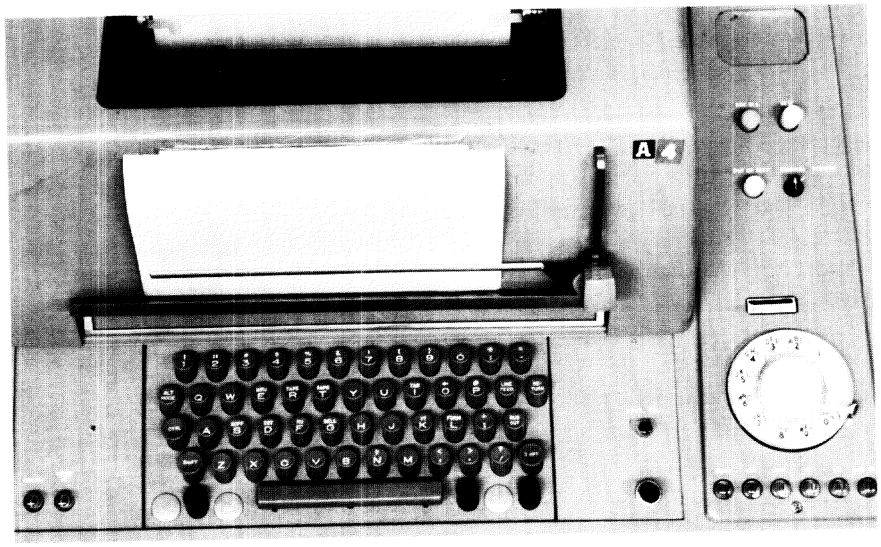


Figure A-6. Teletypewriter control panel and keyboard.

Since the teletypewriter lacks keys for the backspace, underscore, and logical-NOT sign, you must use substitutes. The usual TSS/360 prompt of underscore and backspace, for instance, will be represented on the teletypewriter as a right bracket and a left arrow:

] ⇐

Since the right bracket is the equivalent of the underscore, it is used as the break character for the text editor (see Chapter 2). The right bracket is obtained by holding down the SHIFT key and pressing the "M" key.

In addition, the backwards slash (\) is the teletypewriter equivalent for the logical-NOT sign (¬). The backwards slash is obtained by holding the SHIFT key down and pressing the "L" key. The left bracket is obtained by holding the SHIFT key down and pressing the "K" key.

End-of-Line Sequence—To signal the end of an input line, hold down the CTRL (control) key while pressing the key marked X OFF. After the end-of-line signal, control passes to the system. Do not use the keyboard again until prompted for further input, except to generate an attention interrupt. When the system prompts for additional input, it will issue a carrier return and line feed.

Continuation Lines—When the hyphen is entered as the last character before the end-of-line sequence, the system recognizes the next line as a continuation. The hyphen is not entered as part of the line.

Canceling Lines—When a pound sign (#) is entered as the last character before the end-of-line sequence, the entire line is canceled. They system will then expect the corrected line to be entered without further prompting. The pound sign is defined as the *line-kill character.*

Correcting Lines—To correct a line that you have started to enter incorrectly, enter the left arrow (which is obtained by holding down the SHIFT key and pressing the "O" key) *once* for each character to be eliminated. In other words, use the left arrow as if it were the BACKSPACE key on a typewriter. For instance, if you have typed ERESE and want to change it to ERASE, your correction would look like this:

ERESE⇐⇐⇐ASE

The left arrows will not be entered as part of the line; TS/360 treats the teletypewriter left arrow as the equivalent of the backspace, as mentioned above.

Termination Procedure—After the LOGOFF command has been accepted by the system, close down the teletypewriter by pressing the CLEAR button on the control unit.

Appendix B: HOW TO READY THE SYSTEM

The instructions given in this book for using TSS/360 are based on the supposition that the user profile has been permanently altered with DEFAULT and PROFILE commands as explained in this appendix. The DEFAULT command supplies default values for command operands to meet the needs of the reader of this book and make it unnecessary for him to enter, or even be aware of, these operands. The PROFILE command makes these defaults a part of the permanent user profile; the default values will be in force every time the user logs on, unless and until he changes them with further DEFAULT commands.

The DEFAULT and PROFILE commands can be entered either by the user of this book himself, if necessary, or, preferably, by someone in a supervisory or tutorial relationship to the user.

Ideally, the user's supervisor or instructor should log on with the user identification assigned to the user, enter the commands as given here, and log off. The user profile for that user identification will then be ready for the user the first time he logs on. If necessary, however, the user can enter the commands.

Entering the DEFAULT and PROFILE Commands

The DEFAULT and PROFILE commands should be entered precisely as shown in Figure B-1 or B-2. These commands should be entered as shown in Figure B-1 unless the user does not want to employ the facilities of the program-control commands for dynamic debugging, in which case the commands should be entered as shown in Figure B-2. Simply log on with the user identification and password to be used by the user, enter the commands as shown, and issue the LOGOFF command.

If the user himself is entering these commands to prepare the system for use with this book, he can enter the commands immediately after logging on for the first time. (Chapter 1 explains how to log on.) He can then proceed, during that terminal session and all subsequent ones, to use the system as described in this book.

What the DEFAULT and PROFILE Commands Do

The DEFAULT command specifies default values other than those supplied by the system; the PROFILE command records these defaults permanently in the user profile, so that they will be in force during that user's terminal sessions from then on, unless altered by further DEFAULT commands.

```
default slist=n,isd=y,trantab=n; profile
```

Figure B-1. DEFAULT and PROFILE **commands for user who will employ program-control commands.**

```
default slist=n,trantab=n,isd=n; profile
```

Figure B-2. DEFAULT and PROFILE **commands for user who will not employ program-control commands.**

The SLIST=N operand of the DEFAULT command causes the FORTRAN compiler to produce no source listing unless SLIST=Y is specified when the FTN command is entered. Since the system-supplied default for all other listings that may be specified with the FTN command is N, no listings will be produced by the compiler when these operands are not entered. The user can print his source data set to obtain the equivalent of a source listing.

The TRANTAB=N operand of the DEFAULT command causes the transaction table of the text editor to be inoperative. The facilities provided by this table are not described in this book, and the text editor will operate more quickly if it does not have to maintain the transaction table.

The ISD=N operand of the DEFAULT command shown in Figure B-2 causes the FORTRAN compiler to produce no Internal Symbol Dictionary (ISD). The ISD is essential for easy use of the program-control commands for dynamic debugging; if the user does not intend to use these facilities, however, the generation of the ISD should be inhibited, since the compiler can produce more efficient code without the ISD.

After This Book

The subset of the command system that is presented in this book may satisfy the computing needs of the user; however, it may also be used as an introduction to TSS/360 by a user who will go on to learn more about the system after mastering the material here.

In the latter case, the user may set his defaults back to the system-supplied values by entering the DEFAULT command as shown in Fibure B-1 or B-2 but with Y instead of N on the right side of the equal sign in each operand; he should then enter the PROFILE command to make these values apply until he changes them again.

After finishing with this book, the user who wants more information about TSS/360 may turn to these books of the TSS/360 Systems Reference Library:

- *Concepts and Facilities*, Form C28-2003, for a description of the full facilities of the system and a conceptual treatment of how these facilities are implemented.
- *Command System User's Guide*, Form C28-2001, for both instructional and reference material on the full command system.
- *FORTRAN Programmer's Guide*, Form C28-2025, for instructional material slanted to the FORTRAN-oriented user of TSS/360. This book also contains illustrations of the uses of most user commands and discussions of TSS/360 concepts, as well as reference material.

Appendix C: COMMAND DESCRIPTIONS

This appendix describes the commands as illustrated in Chapters 1 through 8, except for the program-control commands. Only the operands used in this book are given in the command descriptions; operands for which a default value has been established that makes it unnecessary for the user of this book to enter the operand are omitted. Complete command descriptions may be found in *Command System User's Guide*, Form C28-2001.

Command Format and Notation

The basic format of a command description is:

Command	Operand	Function
command name	one or more operands, limited by commas; field may be blank	description of function

The operand field may be blank or may contain several operands. Multiple operands must be separated by commas. Blank and/or tab characters may also be used between operands, but they are ignored by the system. The operand field is separated from the operation field by either a tab character or one or more blanks.

To facilitate the representation of the statements in the format illustrations, four metasymbols will be used.

Name	Symbol	Use
braces	{ }	to enclose and thus delimit operands; alternatives are stacked within braces.
brackets	[ ]	to enclose and thus delimit optional operands.
ellipsis	. . .	to indicate that the preceding syntactical unit may be repeated one or more times.
underscore	—	indicates the default value that the system will assume if an operand is omitted.

Table C-1. Basic command descriptions.

Command	Operands	Function
CALL	program name	Invokes user program and gives it control.
DATA	data set name	Creates virtual sequential data set.
DDEF	data definition name, organization, data set name	Links data set to user program through DDNAME.
EDIT	data set name	Invokes text editor to create or edit data set.
END		Terminates text-editing.
ERASE	[partially or fully qualified data set name]	Erases data set; used with no operands or with partially qualified data set name, provides review of data set names with option of erasing or retaining each data set whose name is presented.
FTN	program name [,{Y / N}]	Invokes FORTRAN compiler. Y indicates source statements are stored as a data set; N, that source statements will be entered for line-by-line scanning and compilation.
INSERT	{line number / LAST}[,increment]	Inserts a line or lines between existing lines or at the beginning or end of a data set.
LIST	[{line number / LAST}] [,{second line number / LAST}]	Causes line or range of lines specified to be printed on terminal. If no operands are entered, entire data set is printed out.
LOGOFF		Ends terminal session.
LOGON	user identification, password	Identifies user to system.
PRINT	data set name [,ERASE={Y / N}]	Causes data set to be printed at computer installation and optionally erased afterwards.
RELEASE	data definition name	Cancels previous DDEF command.
REVISE	{line number / LAST} [,{second line number / LAST}] [,INCR=increment]	Causes line or range of lines specified to be removed from data set; system prompts for replacement lines; if none is entered, no replacement is made; increment for replacement lines is 100 unless specified otherwise.