IBM

Application Program

GH20-0787-2

CALL-OS

Terminal Operations Manual

Program Number 360A-CX-42X

This manual is addressed to the CALL-OS terminal user and discusses characteristics, operation, and maintenance of terminals supported by the CALL-OS time-sharing system. A functional overview of the CALL-OS terminal command language is followed by a detailed alphabetic presentation of all the commands. System messages are listed and explained. A complete index aids the reader in rapidly locating areas of interest.

CALL-OS is a terminal-oriented, time-sharing system designed to function under the control of the IBM Operating System (OS/360) employing either of two options: Multiprogramming with a Fixed Number of Tasks (MFT), or Multiprogramming with a Variable Number of Tasks (MVT). A large number of users can interact with CALL-OS to enter programs and/or data and receive output at what appears to be the same time.

Condensed programming information for handy reference covering terminal command facilities is given in the CALL-OS Terminal Command Language Reference Card (GX20-1830). The card summarizes pertinent information in this manual for handy reference while the user is signed on to CALL-OS.

Note: The CALL/360-OS system has been renamed the CALL-OS system. Thus, documentation of Version 2 of the CALL/360-OS system refers to the system as CALL-OS.

Terminal Equivalence

Terminals which are equivalent to those explicitly supported may also function satisfactorily. The customer is responsible for establishing equivalency. IBM assumes no responsibility for the impact that any changes to the IBM-supplied products or programs may have on such terminals.

Third Edition (March 1972)

This edition, GH20-0787-2, is a major revision obsoleting GH20-0787-1. It reflects Version 2, Modification Level 0, of the CALL-OS time-sharing system and all subsequent versions and modifications until otherwise indicated in new editions or Technical Newsletters. Modifications have been made to this manual to fully describe CALL-OS Batch Interface (COBI) facilities and extensions to the terminal command language permitting storage of programs in object format and increasing edit capabilities. Steps for online creation of paper tape (contained program statements, terminal commands, and data) formatted to be reenterable to CALL-OS are described. Extensions to utility functions permitting manipulation and maintenance of a CALL-OS data base are discussed. In incorporating new material, major revisions have been made to this manual. Complete sections have been added, and existing material has been reorganized. The manual should be reviewed in its entirety.

Changes are continually being made to the specifications contained herein. Therefore, before using this publication, consult the latest System/360 SRL Newsletter (GN20-0360) for the editions that are applicable and current.

Copies of this and other IBM publications can be obtained through IBM branch offices. A form has been provided at the back of the publication for reader comments. If this form has been removed, address comments to IBM, Technical Publications Department, 1133 Westchester Avenue, White Plains, N. Y. 10604.

PREFACE

Three types of terminal units are supported by the CALL-OS time-sharing system: the IBM 2741 Communications Terminal (Correspondence or EBCD), the Teletype* unit, Type 33, and the Teletype unit, Type 35. This manual is intended for the CALL-OS terminal user and describes characteristics, operations, and maintenance of these terminals, as relevant to CALL-OS users. Subjects discussed include sign-on, sign-off, and typing correction procedures; routine equipment maintenance procedures; and equipment characteristics. CALL-OS system features available to terminal users are described. The CALL-OS terminal command language is explained in detail. System messages are listed and explained alphabetically. Commands and messages of the CALL-OS Batch Interface (COBI) option are also described. Condensed programming information for handy reference covering terminal command facilities is given in the CALL-OS Terminal Command Language Reference Card (GX20-1830).

For a brief introduction to CALL-OS system facilities and capabilities, the reader is referred to the CALL-OS System Description Manual (GH20-0673). Details of available programming languages are presented in the CALL-OS language reference manuals: CALL-OS BASIC Language Reference Manual (GH20-0699), CALL-OS PL/I Language Reference Manual (GH20-0700), and CALL-OS FORTRAN Language Reference Manual (GH20-0710). CALL-OS programming languages are summarized in three language reference cards: CALL-OS PL/I Reference Card (GX20-1810), CALL-OS BASIC Reference Card (GX20-1811), and CALL-OS FORTRAN Reference Card (GK20-1812).

COBI users may wish to refer to IBM System/360 Operating System Messages and Codes (GC28-6631), IBM System/360 Operating System Supervisor and Data Management Macro Instructions (GC28-6647), IBM System/360 Operating System Job Control Language (GC28-6539), and IBM System/360 Operating System Job Control Language User's Guide (GC28-6703).

Details concerning installation of IBM 2741 Communications Terminals are found in IBM Remote Multiplexers and Communications Terminals, Installation Manual - Physical Planning (GA27-3006).

*Trademark of the Teletype Corp., Skokie, Illinois

CONTENTS

FIGURES

TABLES

INTRODUCTION

CALL-OS is a terminal-oriented time-sharing system which combines the vast problem-solving capabilities of a modern computer with the operational simplicity of a typewriter keyboard. The user's terminal, linked online to the computer by telephone lines, is employed in a manner similar to the everyday use of an electric typewriter. Rapid multiplexing from user to user provides each CALL-OS terminal user with the illusion of having the entire resources of the system at his disposal. In addition, since familiarity with internal computer functioning is not required, the user is free to concentrate on the specific problem to be solved.

To interact with the CALL-OS system, the user enters any of a wide variety of terminal commands (described in subsequent sections of this manual). The system performs any required actions and provides an appropriate system response. To meet his problem-solving needs, the user creates programs, using either CALL-OS BASIC, CALL-OS FORTRAN, or CALL-OS PL/I (see CALL-OS language reference manuals). He enters both programs and data from a terminal or requests either or both from his user library or a system library. Output of program execution is available immediately at the terminal--a rapid, timely solution to the user's problem.

Figure 1. IBM 2741 Communications Terminal

SIGN-ON PROCEDURE

To begin work, the user at an IBM 2741 Communications Terminal (see Figure 1) must establish a telephone connection and then identify himself to the computer. The user should:

1. Set the ON/OFF switch (at the right of the keyboard) to ON.
2. Set the COM/LCL (Communicate/Local) switch (housed in the niche in the left side-panel of the terminal) to COM.
3. Press the TALK button on the data set.
4. Lift the data set receiver, wait for a dial tone, and dial the system telephone number (the system answers the call with a

The consoles are assigned to specific logical line numbers with parameters in the EXEC statement of the startup deck (see Appendix C); these logical line numbers are associated with specific terminal types by the DD statements included in the startup deck. Any CALL-OS console may be either a Teletype Unit* (Type 33 or 35) or an IBM 2741 Communications Terminal (Correspondence or EBCD). If IBM 2741 Communications Terminals are used, the correct print element must be used on the terminal to ensure readable output: the print element for the Correspondence terminal is 1167087; the print element for the EBCD terminal is 1167643.

Since the operator for CALL-OS is concerned primarily with the online operations, these operations and the commands used to perform them are described in separate sections.

OFFLINE REQUIREMENTS

The offline operations for CALL-OS involve the execution of the utility programs, and, if COBI is used, the starting of a special COBI writer program under certain conditions. Messages pertaining to offline processing appear on the OS system operator's console and the system printer. The operator intervention required for offline operations is minimal, and is described in the following text.

Executing the Utility Programs

The utilities execute as standard jobs, in a separate task area, under control of OS. Because their use is directed by the installation system programmer, the requirements for executing them and the options available are described in detail in the publication CALL-OS Executive and Utilities Program Description Manual. The operator must monitor the messages and notify the responsible system programmer when an error occurs. Under no circumstances should a utility be restarted without consultation with the programmer.

Starting a COBI Writer

If CALL-OS is no longer operating but jobs submitted through COBI must either still be run or have their output processed, the operator must start the COBI writer program. This program intercepts JCL for later scanning at the terminal and makes SYSOUT data sets not saved for scanning available to the OS output writer. While CALL-OS is operating, the DIBWTR functions are handled by the module M#JCL.

The operator starts the COBI writer by issuing the following command at the OS system operator's console:

```
S DIBWTR
```

When all of the output for the COBI output class has been processed, the operator may terminate the writer by issuing the following command at the OS system operator's console:

```
P DIBWTR
```

Note: The operator does not receive a message instructing him to start the COBI writer. If the writer is not started, the COBI message class associated with the most recent CALL-OS session is not processed; the next time CALL-OS is initialized with COBI, this output class is processed in the normal way.

* Trademark of the TELETYPE Corporation

SYSTEM MESSAGES

Messages from the CALL-OS online system and the offline utility programs may appear on one of four possible devices, as follows:

- OS/360 system operator's console, which receives CALL-OS initialization messages, online ABEND messages, COBI operating messages during online and/or offline operation; and utility ABEND codes and messages during offline operation.
- CALL-OS command console, which receives messages in response to operator commands issued at the console during online operation
- CALL-OS communications console, which only receives messages during online operation; these messages include system messages from the executive, user-related error codes, disk I/O error messages, and compiler-related messages
- OS/360 system printer, which receives messages from the CALL-OS offline utilities

Subsequent sections list the messages which appear at each device. Within each group or subgroup of messages, the messages are listed in alphabetic or numeric order, whichever is appropriate. An explanation of the cause of each message is also given.

procedures are not performed automatically for any data files being created by the interrupted program.

SYSTEM ATTENTION

During the compilation and execution of a user's program, the print element on his terminal will sometimes wiggle back and forth. This movement is generated as a courtesy to the user. It indicates that the system is actively processing. As soon as a required operation has been completed, the user can expect a normal system response (for example, program output or a READY message) to be printed at his terminal.

SIGN-OFF PROCEDURE

To sign off, the user should:

1. Type the word OFF.
2. Press the RETURN key.
3. Wait for the system to reply with the time of day, the processing time, and the terminal time.
4. Set the ON/OFF switch to OFF.

It is not necessary to reset the COM/LCL switch unless the terminal is to be used for ordinary typing. The telephone connection is broken automatically.

If the user does not sign off the system and/or disconnect his data set, but no input/output activity occurs for a terminal within a period exceeding twenty minutes, the user is logged off the system and the terminal disconnected automatically.

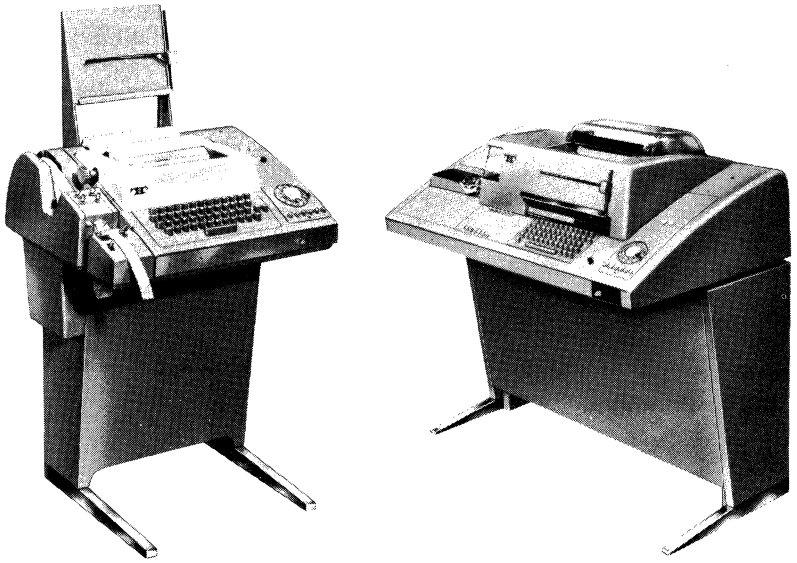

Figure 2. Teletype units, Type 33 ASR (left) and Type 35 ASR

SIGN-ON PROCEDURE

To begin work, the user at a Teletype unit (see Figure 2) must establish a telephone connection and then identify himself to the computer. The user should ensure that the volume control knob (under the key shelf on the right of the 33; between the keyboard and phone dial on the 35) is turned up sufficiently for the dial tone to be audible, and then:

1. Press the ORIG (Originate) button (at the left end of the row of buttons below the telephone dial), and wait for a dial tone.

2. Dial the system phone number (the system answers the call with a high-pitched constant tone). If there is no answer or the line is busy, disconnect (press CLR) and dial again.

At this point, the telephone connection has been made. To identify himself to the computer, the user should be sure that paper has been inserted and then:

1. Wait for the system to print a greeting message and the words: USER NUMBER,PASSWORD--

1a. (Type 35 only) Press the K button (Keyboard, at the left end of the lower row of buttons on the left control panel) to activate the keyboard.

2. Type the user number, a comma (no space after it), and the password.

3. Press the RETURN key.

Each user is assigned a user number consisting of three alphabetic and three numeric characters. The password, selected by the user, consists of up to eight characters in any combination. If the user number were ABC123 and the password were WXYZ90+*, the system's sign-on invitation (up to and including the two hyphens) and the user's typed response would be:

```
USER NUMBER,PASSWORD--ABC123,WXYZ90+*
```

The user types no blank spaces in the sign-on line unless blanks are characters within the password itself. The password may not begin with a blank.

After the identification is accepted, the system replies by printing the word READY, and the user can then begin his problem-solving job. If the user fails to complete sign-on procedures within two minutes after dialing into the system, the terminal is disconnected from the system automatically.

TYPING CORRECTIONS

DELETING INCORRECT CHARACTERS

To delete incorrectly-typed characters before the line has been ended with the RETURN key, the user should:

1. Hold down the SHIFT key.
2. Strike the letter O once for the first incorrect character in the line, and once for each character following it (including any blank spaces).

The back-arrow symbol (←) or the underline symbol is printed for each upshift O. The user then types the correct character and all characters following it. For example, the word PROTECT, misspelled as PRETECT, would be changed by typing:

```
PRETECT ←←←←←OTECT
```

The upshift O symbol erases individual characters from the computer's storage unit. Characters cannot be erased by this procedure after a line has been ended with the RETURN key.

DELETING THE CURRENT LINE

To delete a line before it has been ended with the RETURN key, the user should:

1. Hold down the CTRL (Control) key (at the left of the third keyboard row).
2. Strike the letter X.

The system prints the word DELETED and ends the line automatically.

BREAKING EXECUTION

To interrupt the system while a program is running or being listed, the user should:

1. Press the BREAK key.

2. Press the BRK-RLS button (Break-Release, above the phone dial).

2a. (Type 35 only) Press the K button to unlock the keyboard.

The system stops the execution or listing process. In the first case, it prints:

```
STOP.
RAN xxx SECS.
```

For the second case, it prints:

```
STOP.
READY.
```

When the user interrupts program execution by pressing the BREAK key, he cannot assume that all system processing is performed in the same manner as at normal program termination. In particular, file closing procedures are not performed automatically for any data files being created by the interrupted program.

SYSTEM ATTENTION

During the compilation and execution of a user's program, the type box on his terminal will sometimes move up and down. This movement is generated as a courtesy to the user. It indicates that the system is actively executing his job. As soon as a required operation has been completed, the user can expect a normal system response (for example, program output or a READY message) to be printed at his terminal.

SIGN-OFF PROCEDURE

To sign off, the user should:

1. Type the word OFF.

2. Press the RETURN key.

The system replies with the time of day, the processing time, and the terminal time. The telephone connection is broken automatically.

If the user does not sign off the system and/or disconnect his data set, but no input/output activity occurs for a terminal within a period exceeding twenty minutes, the user is logged off the system and the terminal disconnected automatically.

TELETYPE CHARACTER SET

The character set available to the CALL-OS user from Teletype terminals differs somewhat from the character set available from 2741 Communications Terminals. Certain characters available on the CALL-OS 2741 terminal keyboard are not available on the Teletype unit. These characters are:

≤	° (degree symbol)
≠	¢
≥	

Note that CALL-OS language compilers have multi-character combinations which may be used in place of the ≤, ≠, and ≥. For example, ≠ can be replaced by <> in CALL-OS BASIC, by .NE. in CALL-OS FORTRAN, and by ¬= in CALL-OS PL/I. When a program is entered from a Teletype, these multi-character combinations must be used. There are no substitutes for the ° and ¢ characters available on the Teletype; however, these characters are not part of any CALL-OS language character set. They may be used as data characters in character strings. (The function of the ° character in deleting input lines for the 2741 user is performed for the Teletype user by the CTRL X.) If a program using any of these five characters in program text or in character strings is entered from a 2741, and later is listed or executed on a Teletype terminal (or punched into paper tape), all occurrences of these characters are indicated by non-meaningful characters.

In addition, certain characters which form part of the CALL-OS PL/I language character set have different graphics on the Teletype. These characters and their Teletype representations are shown in Table 1. The user should have no difficulty in using the Teletype to enter programs which employ these characters, or in listing such programs. However, the graphics shown in Table 1 will always replace these characters in Teletype usage. The user is cautioned not to confuse the _ (CALL-OS PL/I break character), which has the Teletype graphic [, with the Teletype back-arrow ( ), which appears as a _ on some Teletype units.

Table 1. 2741/Teletype equivalent graphics

2741 Character	Teletype Character
¬ (NOT)	] (uppercase M)
| (OR)	\ (uppercase L)
_ (BREAK)	[(uppercase K)

CALL-OS SYSTEM FEATURES

CALL-OS USER LIBRARIES

One user library is available to each CALL-OS terminal user. This library contains all source and object programs and data files retained by the user for subsequent processing operations. The name of each program or data file placed in the library is entered automatically in a user catalog associated with the library. It is removed automatically if the user purges the file from the library. The user can enter line-numbered information, either directly from the terminal or by means of a CALL-OS utility, for use as input to an executing program. (See "ENTER" under "CALL-OS Terminal Command Language".) Each user library is a private library; that is, the information in the library is controlled by him and accessible only to him. However, he can make the information available to other system users by means of the CALL-OS shared library facilities if he desires.

CALL-OS SHARED LIBRARIES

CALL-OS provides three types of libraries that can contain program and data files required by multiple CALL-OS users. These are the ***Library, **Library, and *Library. The ***Library is controlled by the computer installation management. Program and data files can be entered in this library only by computer center personnel. The **Library is really a directory containing pointers to program and data files pooled (specified for inclusion in the library) by users of the system. The pooled programs and data files exist only in the private user library of the user who pooled them. Several *Libraries may exist simultaneously in the CALL-OS system. Each is similar to the **Library except that it is accessible to only a particular CALL-OS subscription group (sub group). Each sub group consists of up to 99 users whose six-character user numbers are identical for the first four characters. The purpose of each *Library is to provide shared library facilities among a defined subset of users, such as an organizational unit.

A program or data file is entered in the **Library or a *Library by issuance of a POOL command. It is removed by issuance of a PULL command. These commands cause the program or data filename to be inserted in or removed from the indicated library directory. The user retains control of the actual program or data file.

In some cases, a user may wish to limit the types of actions that other CALL-OS users can perform on a program that he has pooled in a shared library. By using the PROTECT command, he can insure that users accessing the program via the **Library or a *Library cannot gain access to the source code of the program. That is, other users can run the program (via a RUN command) or merge it with other source programs (via a MERGE command), but they cannot list, save, or store it (or the newly formed, merged program). The user can also limit the types of actions performed on a program by storing an object-code version of it in his library (via a STORE command) and then pooling the object-code version of the program.

Data files as well as program files can be pooled in the shared libraries. When data files from shared libraries are opened by other than the users who pooled them, however, only input operations are permitted. The contributor of a pooled data file controls any required updating of the file.

Examples of statements to open pooled data files as expressed in CALL-OS programming languages are given below. Note that, in the CALL-

2. Dial the computer center number. (Response is a steady high-pitched tone.) If there is no answer or the line is busy, disconnect (press CLR) and dial again.

   For Type 35 ASR, press K key (on left control panel) to unlock keyboard.

3. Sign on.

Disabling Telephone Lines

To prevent a user from attempting to establish a connection over an out-of-service line between the telephone company switching station and the computer, the operator needs the ability to prevent (busy out) any particular line from being accessed by the rotary line selector (or an equivalent device used by the telephone company). The manner in which this is done is dependent on the particular type of telephone equipment at this installation. When the operator receives a message to busy out a particular line, he should therefore consult his telephone equipment manuals for details of the specific procedure required. He should also disable that line through the *DISABLE command.

Allocation of Storage

If the amount of core storage available to user programs is limited, the likelihood of running more than one user program at a time is decreased. This means that user terminal response times may suffer during peak usage, resulting in user complaints of service delays. The amount of core available for the execution of user programs is printed on the OS/360 system operator's console after the system is successfully initialized (message DIBIN001). The operator should be aware of the normal new and old job area sizes; he should bring any deviation in these sizes to the attention of the installation system programmer.

COBI OPERATING TECHNIQUES

If COBI is used, the operator may be required to perform additional actions while the system is in operation. These actions include:

- Starting a COBI reader to read submitted jobs into OS/360
- Performing user-requested services

Each of these is described in more detail in the following text.

Starting a COBI Reader

Jobs submitted under CALL-OS with COBI are written into two data sets, known as SYSINA and SYSINB. They are read into OS/360 for execution by the COBI reader cataloged procedures, DIBRDRA or DIBRDRB, respectively. The reading of the data sets may be initiated either automatically by COBI or manually by the operator, depending on the mode of operation in effect at the time a data set becomes available for reading. If the automatic mode is not used and both data sets are full, no more jobs may be submitted to COBI until one of the data sets has been read.

Therefore, during the operation of a CALL-OS system with COBI, the operator may be asked to start a reader if the manual mode of starting a reader is in effect for the session. The operator is instructed when to

start a reader by messages which appear on the OS system operator's console. These messages have the following format:

```
DIBxxxxx  text ... START DIBRDRx
```

where

xxxxx is the message identifier

text is an informative message which indicates the condition causing a reader to be needed

x is either A or B; it indicates which reader procedure is to be started to read the input data set.

The operator must then issue a start reader command on the OS system operator's console. This command is identical to that used for an OS reader except that the operator specifies either DIBRDRA or DIBRDRB. The following example shows a message instructing the operator to start a reader and his response to that message:

```
System:   DIBEX021  20 JOBS ON SYSINB, START DIBRDRB
Operator:  S  DIBRDRB
```

The S DIBRDRB command causes all the jobs on the associated input data set to be read into OS for processing by initiators for the appropriate job classes. However, it may be desirable to start the reader at a particular job rather than at the beginning of the data set. To do this, the operator uses the following format of the start command:

```
S DIBRDRx,,,job-name
```

where

x is either A or B

job-name is the name of the first job to be read

All jobs preceding the specified job-name are skipped, as well as all write-to-operator messages associated with these jobs.

The operator may also use the start command to display the names of the jobs on a COBI input data set. To do this, he uses the following format of the start command:

```
S DIBRDRx,,,$$$$
```

where

x is either A or B

This command causes the names of the jobs on the associated input data set to be displayed on the OS system operator's console. However, the jobs are not read into OS for processing.

In the event the OS reader, IEFIRC, encounters a permanent I/O error while reading the SYSINA or SYSINB data set, the OS operator may physically remount the disk pack to another unit if dynamic device reallocation was specified in the system startup deck. If the error is permanent, IEFIRC issues an IEF419 error message and terminates normally. The OS operator should hold the job queue until the names of the jobs in the queue can be displayed. The names in the job queue should be compared to the list of job names displayed when DIBRDR was started. The first job name that appears in DIBRDR's list but does not appear in the job queue is the job IEFIRC was processing when the I/O

error was encountered. The operator should restart DIBRDR to begin processing with the first job after the job containing the I/O error. The CALL-OS console operator can determine the user number of the user who submitted the job that was lost by entering the command:

```
*COBI DSSTATUS, SYSINx
```

where x is either A or B, as appropriate. The user should then be notified, via the *TELL command, to resubmit his job.

Other operator actions in connection with the COBI reader are identical to those he performs for the OS reader. See the operator's procedures and reference manuals for the IBM Operating System.

Note: The operator may use the *COBI command to specify which mode of reader operation is to be in effect. (See the description of the RESET function of the *COBI command.)

Performing User-Requested Services

The operator may receive messages which originate from a user's terminal. These messages could contain setup instructions for jobs submitted from the terminal or status information regarding the execution of the job. These messages appear either on the OS/360 system operator's console or on the CALL-OS communications console.

Messages intended for the OS/360 system operator's console are identified by the identifiers DIBEX002 or DIBRD003. Messages intended for the CALL-OS communications console do not have an identifier if they appear on the communications console; if, however, the communications console is not active, these messages appear on the OS/360 system operator's console preceded by the identifier DIBEX001.

SHUTDOWN AND RESTART

The operator uses two commands to shut the system down in an orderly manner. The *WARN command transmits a message to all current users that the system is to be shut down; this message should be issued several minutes before the *OFF command to allow users to terminate their work. The *OFF command causes the CALL-OS job to terminate; all CALL-OS close-out procedures are invoked, the OS/360 environment is restored, and the final return is made to OS/360.

These two commands ensure proper shutdown of the system, whether the shutdown is because the normal operating period is at an end or because of some other reason (for example, an OS/360 malfunction). The job termination message appears on the OS/360 system operator's console. This is because CALL-OS terminates approximately one minute after the *OFF command is issued.

The following example shows the sequence of operator commands to be used to shut down the system:

```
Operator:  *WARN
System:    ENTER
Operator:  SYSTEM WILL BE SHUT DOWN AT 16:00
System:    READY
Operator:  *OFF
System:    SYSTEM OFF AT 16:01
```

Note: In the event of a system failure, there are no built-in restart procedures. The center operator must reinitialize the system as explained in "Initializing the System."

OPERATOR COMMAND LANGUAGE

The operator communicates with CALL-OS by means of the operator command language. These commands may be used only by personnel at the central computer installation. With these commands, the operator may:

- Enter the current date and other information into the user's sign-on message - *DATE
- Transmit a message to all users at sign-on time - *MESSAGE
- Enable all or one or more terminal lines - *ENABLE
- Control the percentage of total processing time given to batch processing - *BATCH
- Validate a new user onto the system - *VALIDATE
- Cancel a user from the system - *CANCEL
- Transmit a message, during system operation, to another terminal or console - *TELL
- Print system status information - *REPORT
- Print user status information - *STATUS
- Print the number of users either on the entire system or on a range of lines - *USERS
- Request that the system disable all lines not in use and prevent new users from signing on - *IGNORE
- Request status of COBI jobs and/or data sets, scratch COBI data sets, or override COBI initialization parameters - *COBI
- Disable one or more terminal lines - *DISABLE
- Transmit a warning message to all users - *WARN
- Terminate system operation - *OFF

A brief discussion of the procedures used for entering commands precedes the command formats and descriptions.

ENTERING COMMANDS

The operator commands may be entered only at the command console and are identified by an asterisk, which precedes the command name. Except where noted, each command may be abbreviated to the asterisk plus the first two characters of the command. For example, *BATCH may be abbreviated to *BA. One or more parameters may be required; their formats are given in the following subsection.

The operator must depress the RETURN key after he enters either the command or the parameters, if any are necessary. If the requested operation is completed successfully, the system responds with READY unless additional information is required. In this case, the system responds with ENTER. The operator then enters the additional information; for example, the date or a message. He then depresses the

2741-entered program may not be meaningful when displayed as printed output on the Teletype. This has no effect on the punching operations, by which the output is entered into paper tape. The unprintable characters are punched correctly. The tape can be read as input during subsequent operations.

A discussion of paper-tape input modes is given below. Procedures for punching paper tape (either online or offline) for subsequent reentrance into CALL-OS and for reading paper tape into CALL-OS are also described.

PROGRAM STATEMENT INPUT (TAPE MODE)

The TAPE mode of operation, established by issuance of a TAPE system command, allows only program-statement entry. This is the most rapid method of entering a sequence of previously prepared program statements. Once initiated, reading of input from paper tape in TAPE mode is a continuous operation.

Each statement entered in TAPE mode must be ended with C/R, L/F (Carrier Return, Line Feed), since the CALL-OS system will not automatically respond with an L/F at the completion of each line when operating in TAPE mode. (See Figure 3a.)

CALL-OS terminal commands recognized in the TAPE mode are TAPE ALL, KEY, and OFF. All other commands and unnumbered statements are ignored. The KEY or TAPE ALL command can be used to restore functions which were inhibited by the TAPE command.

The TAPE command can be punched as the initial characters in the paper tape. If this is done, TAPE must be followed by X-OFF and three RUBOUT characters in addition to C/R, L/F (as shown in Figure 3b).

GENERAL PAPER TAPE INPUT (TAPE ALL MODE)

Paper tape input in the TAPE ALL mode provides a method of executing complete programs from previously prepared paper tape without manual intervention. Program statements, data, and terminal commands may be entered from paper tape or using a combination of paper tape and keyboard entry.

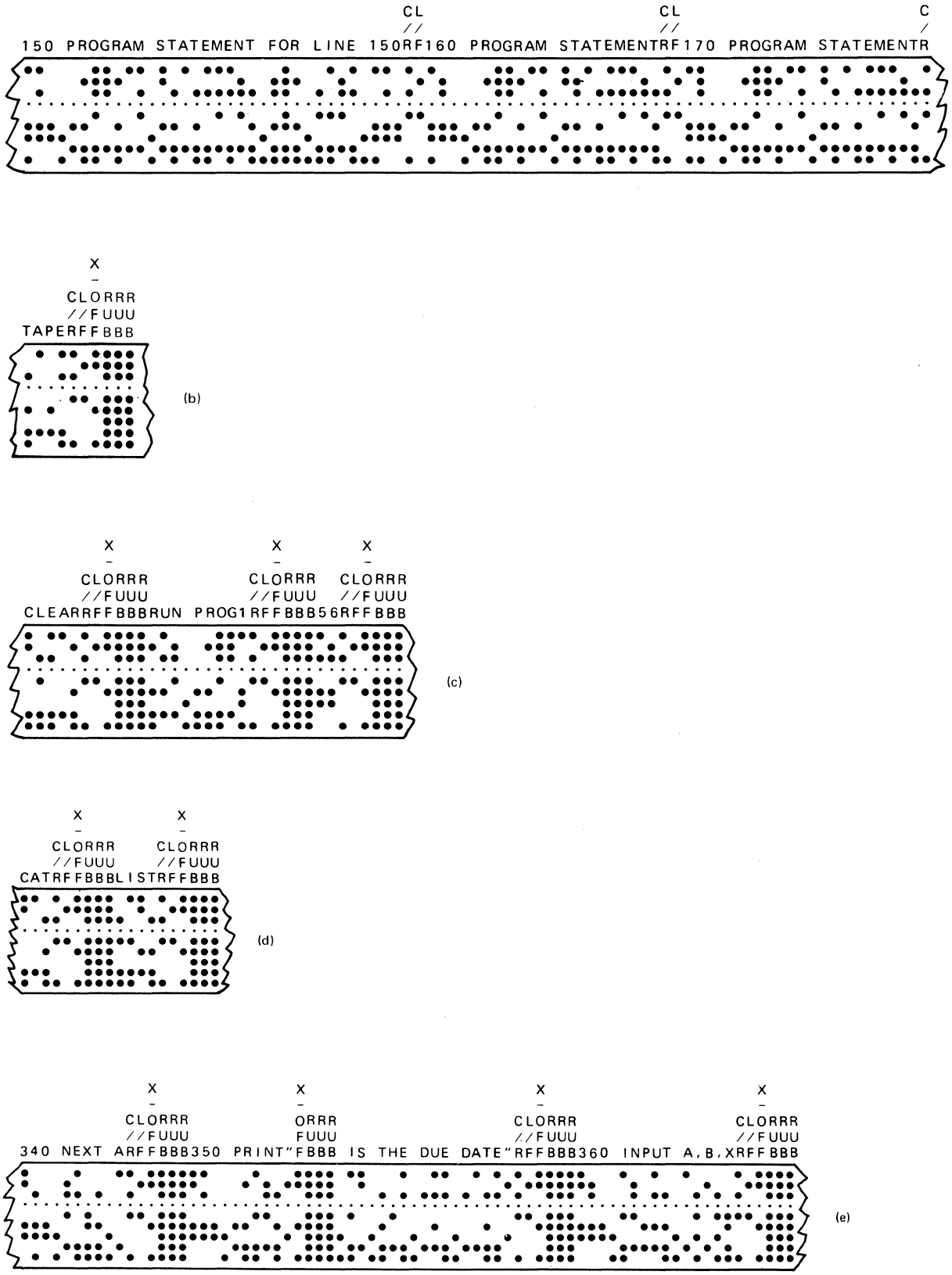

Figure 3. Examples of paper tape input

CALL-OS is placed in TAPE ALL mode by issuance of a TAPE ALL command. The CALL-OS system will not automatically respond with a L/F at the completion of each line when operating in TAPE ALL mode. Therefore, each input line must be followed by C/R, L/F (Carrier Return, Line Feed), to instruct the system that a line of information has been entered and processing may begin. Input enters the CALL-OS system in TAPE ALL mode on a line-by-line basis. Therefore, an X-OFF (reader off) character must follow, to cause the Teletype to stop its paper tape reader. At least three RUBOUT (delete) characters must follow X-OFF to allow time for the paper tape reader to physically halt. The complete line-ending sequence required, then, is C/R, L/F, X-OFF, and three RUBOUT characters (see Figure 3, examples c, d, and e).

If the Teletype terminal is equipped with the Reader Control Arrangement feature, CALL-OS will automatically restart the reader if the previous input line was 1) a source-program statement, 2) a terminal command which normally receives a READY response, or 3) a RUN, LIST, LIST-NO-HEADER, or LIST-TEXT operation that concluded successfully. The tape will also be restarted when a program requests input (that is, when the system prints a question mark at the terminal).

At the completion of certain terminal commands (such as CATALOG) or after use of the ATTN or BREAK key (generating a STOP READY or STOP RAN xxx SECS response), the paper tape will not be automatically restarted. These events usually necessitate an operator decision. Among the choices open to the operator are 1) a manual restart of the paper tape reader (depress paper tape reader switch to START position on Type 33 or press TD ON button on Type 35), or 2) entry of information from keyboard. Note that the system is still in the TAPE ALL mode; the reader will restart if either of the above responses is obtained. The system remains in TAPE ALL mode until a KEY, TAPE, LOGON, or OFF command is given.

Figure 3, examples c, d, and e, show general paper tape input. Figure 3c depicts three typical lines of input. In Figure 3d, information is to be entered manually from the terminal keyboard after the system has responded to the CATALOG command and the tape has come to a halt. The operator will select a program from the catalog of programs listed and load it. The paper tape reader will restart and read in LIST after the system response READY has been given to LOAD.

In the final example, Figure 3e, the tape is halted following the execution of statement 350 to await operator input. After the operator has entered the required input data, he will manually restart the tape for the completion of the line.

PUNCHING PAPER TAPE ONLINE

The CALL-OS system can be used to prepare paper tape online, for subsequent reentrance to CALL-OS in TAPE ALL mode. The PUNCH ON terminal command is issued to set the paper-tape punch mode-switch. While this mode-switch is set, the system automatically appends a six-character line-ending sequence to each line of listed or run-time output. The characters of the sequence are: C/R, L/F, X-OFF, and three RUBOUT characters. They are required at the end of each line to be entered to CALL-OS in TAPE ALL mode. Once the paper-tape punch mode-switch has been set, it remains set until a PUNCH OFF command is given, the user signs off the system, or a new user enters a LOG command to sign-on the system from this terminal without going through the disconnect sequence.

Two other terminal commands often used when punching paper tape are the LIST-NO-HEADER and LIST-TEXT commands. The former causes header-line output to be omitted. The latter causes line numbers to be

omitted; only text is punched. The abbreviated forms of these commands or their combined form (LIST-TN or LIST-NT) can also be used.

Note: The system should not be operating in TAPE ALL mode at the time that the paper tape is being created.

Note that the PUNCH ON and PUNCH OFF terminal commands are used to set a programmed mode-switch. They are not used to physically turn the punch unit off and on. These operations must be performed manually by the terminal user. The complete sequence of operations to be performed by him is illustrated in each of the examples below. The examples demonstrate how paper tape can be punched online for subsequent reentry to CALL-OS.

Example 1:

Assume that a series of CALL-OS terminal commands has been entered into CALL-OS as shown below. Since the commands were not to be treated as commands at the time they were entered, each command was preceded by a line number. The commands have been saved as CMD1.

```
10 TAPE ALL
20 ENTER PL/I
30 WIDTH 58
40 CLEAR
50 NAME ROOTS
```

To create a paper tape without headers and without line numbers, the user must perform the following actions. System responses are also indicated below, to show the complete sequence of system/user communication.

1. Enter PUNCH ON and depress C/R
2. System responds READY
3. Enter LOAD CMD1 and depress C/R
4. System responds READY
5. Enter LIST-NT, depress L/F and C/R
6. Turn paper tape punch on immediately
7. The system punches out a short leader, punches and prints the program CMD1 without the header line and without line numbers, and punches out a short trailer.
8. Turn paper tape punch off

Example 2:

Assume that the CALL-OS PL/I program shown below has been entered and saved as PROG1.

```
10 EXAMPLE:  PROCEDURE;
20 GET LIST (A,B,C);
30 D=B**2-4*A*C;
40 ROOT1=(-B+SQRT(D))/(2*A);
50 ROOT2=(-B-SQRT(D))/(2*A);
60 PUT LIST
70 (A,B,C,ROOT1,ROOT2);
80 END EXAMPLE;
```

To punch this program following the series of commands in Example 1, without a header line but with line numbers, the user must perform the following actions. Since the paper-tape punch mode-switch was set previously (and remains set throughout this terminal session unless reset by a PUNCH OFF or LOG command), it need not be set again before punching this program.

9. Enter LOAD PROG1 and depress C/R
10. System responds READY
11. Enter LIST-N, depress L/F and C/R
12. Turn paper tape punch on immediately
13. The system prints and punches the program PROG1, omitting the header line. A short leader and a short trailer are punched as in Example 1.
14. Turn paper tape punch off

Example 3:

CALL-OS terminal commands and data to be used in response to a request for input by an executing program have been entered and saved as CMD2 as shown below.

```
10 RUN
20 7, 17, 7
30 RUN
40 2, 13, 20
50 OFF
```

These terminal commands and run-requested data are punched following PROG1 (in Example 2) on paper tape as shown below. No more information need be punched into paper tape at this time, so the paper-tape punch mode-switch is turned off when the task is complete.

15. Enter LOAD CMD2 and depress C/R
16. System responds READY
17. Enter LIST-NT, depress L/F and C/R
18. Turn paper tape punch on immediately
19. The system prints and punches output as explained under step 7 in Example 1.
20. Turn paper tape punch off
21. Enter PUNCH OFF

The paper tape produced via this sequence of user/system actions can be reentered and executed in TAPE ALL mode. (See TAPE ALL command.) The sequence of information entered on the tape in examples 1, 2, and 3 constitutes a complete job sequence, beginning with user entry of language name, followed by other terminal commands, the source program, run-requested data provided as input to the executing program, and user sign-off. If more than one job were to be executed, the commands, program, and data for that job could have followed line 40 of example 3 (instead of OFF, as shown above). As noted earlier, the program statements, terminal commands, and run-requested data may have been entered initially into the CALL-OS system using either a 2741 Communications Terminal or a Teletype, or a CALL-OS utility.

COPYING A PROGRAM TO PAPER TAPE ONLINE

This procedure can be used to list (that is, punch out) a program created using the 2741 Communications Terminal or a Teletype unit, Type 33/35. With the terminal signed-on, and the program (to be listed) loaded, the user should:

1. Type LIST or LIST-NO-HEADER.
2. Turn on the paper tape punch.
3. Simultaneously hold down the RUBOUT and REPT keys long enough for a few inches of tape leader to be punched.
4. Press the RETURN key.

5. Observe the program being punched and printed concurrently.

   As noted above, if the program was originally created on the 2741, some of the characters printed at the Teletype will not be meaningful since these characters cannot be created from the Teletype keyboard.

6. After the program has been punched and printed, hold down both the RUBOUT and REPT keys long enough for a few inches of tape trailer to be punched.

7. Remove, identify, and store the program tape.

8. Turn off the punch unit.

Note that in punching paper tape for offline program storage, the WIDTH command must be set equal to or greater than the longest program line prior to listing (punching out) the program. Otherwise, additional C/R, L/F characters will be entered on the tape. Information following these characters may not be interpreted properly when the tape is read back into CALL-OS.

PUNCHING PAPER TAPE OFFLINE

To prepare paper tape offline, the user should:

1. Turn on the Teletype terminal and punch unit.

2. Simultaneously hold down the RUBOUT and REPT keys (located next to the right SHIFT key) long enough for a few inches of tape leader to be punched.

3. a. If the tape is to be read by CALL-OS in TAPE mode, type the program, ending each line with a C/R, L/F (Carrier Return, Line Feed).

   b. If the tape is to be read in TAPE ALL mode, type the program statements, terminal commands, and/or run-requested data. Each line must end with C/R, L/F, X-OFF, and at least three RUBOUT characters.

4. At the end of the program, hold down the RUBOUT and REPT keys long enough for a few inches of tape trailer to be punched.

5. Remove, identify, and store the program tape.

6. Turn off the Teletype terminal and punch unit.

The Teletype unit, Type 33, is turned on and off via the LCL (Local) and CLR (clear) buttons below the phone dial; the punch is turned on and off via the ON and OFF buttons on the punch unit. The Teletype unit, Type 35, and punch unit are turned on and off via the MOTOR ON knob on the left control panel.

CORRECTING PAPER TAPE ERRORS

To correct an incorrectly punched character, the user should:

1. Backspace the tape to the point of the error (using the B.SP. button on the Type 33 punch, or the LOC BSP key next to the keyboard space bar on the Type 35).

The following example shows the use of the ANYBATCH function with a system in the manual mode:

Command Console	OS/360 System Operator's Console
*COBI ANYBATCH	DIBEX021 10 JOBS ON SYSINA,START DIBRDRA S DIBRDRA
READY	

This sequence of operations causes DIBRDRA to begin reading the SYSINA data set. The SYSINB data set is attached to COBI to receive new jobs.

If the ANYBATCH function is used during system initialization and both input data sets are full, the DIBEX021 message is issued with blanks appearing in place of the number of jobs.

DSSTATUS Function

The operator uses the DSSTATUS function of the *COBI command to request the status of data sets associated with COBI jobs. Status may be requested for the data sets associated with all the jobs in the COBI index, for the data sets associated with a particular job or user number, or for the COBI input data sets, known as SYSINA and SYSINB. The valid forms of the command are:

```
*COBI DSSTATUS

*COBI DSSTATUS,userid[,userid]...[,userid]

*COBI DSSTATUS,job-number[,job-number]...[,job-number]

*COBI DSSTATUS,SYSINx
```

where

userid	is a validated user number of the form aaannn
job-number	is a COBI-assigned job number of the form #nnnnn and within the range from 2 to 32767
x	is either A or B

If only DSSTATUS is specified, the status of the data sets associated with all the jobs in the COBI index is printed in job number sequence; if no jobs exist, a message is printed. If one or more user numbers are specified, the status of the data sets associated with all the jobs for the indicated users is printed in the order in which the user numbers appear on the command. If one or more job numbers are specified, the status of the data sets associated with only those jobs is printed in the order in which the job numbers appear on the command.

In all these cases, the output contains the number of the job and its associated data sets. Each data set is identified by type, which may be one of the following:

JCL	Refers to the JCL for the indicated job
nPmm	Refers to a SYSOUT data set which may be scanned and is defined in a cataloged procedure, where n is the number of the procedure and mm is the number of the data set
Unnn	Refers to a user-defined data set which may be scanned, where nnn is the number of the data set

For cataloged procedure and user data sets, the volume serial number of the volume which contains the data set is also printed; for all data sets, the status is printed. The status may be one of the following:

ON-LINE — Indicates that the volume is mounted and the data set is available for scanning

OFF-LINE — Indicates either that the volume is not mounted, or, if the volume is mounted, that the data set is not available for scanning

JCL — For JCL data sets only, indicates that the JCL is available for scanning

A special form of the DSSTATUS function can be used to request the status of a COBI SYSIN data set by specifying either SYSINA or SYSINB. The output in this case consists of a list of all the jobs in the specified SYSIN data set. This list contains the job name and the user number associated with each job. The job names are listed in their order of submission to the SYSIN data set and may be either user-selected or COBI-assigned job names, depending on whether or not ANYJNAME is specified in the system startup deck. If a job name is preceded by an asterisk, the job was cancelled.

The data set status information normally appears on the command console used to issue the request. However, the output may be sent to the high-speed printer by entering *COBI-P with the appropriate DSSTATUS request; in this case, *COBI-P must be entered as shown, with no abbreviation, but DSSTATUS may be abbreviated as DSS. In either case, the system responds with READY at the command console when the output has been completed. When the output is sent to the high-speed printer, the end of the output is indicated by END on the printer.

The following examples show the use of the *COBI command to request data set status. A sample of the appropriate output is also shown.

Example 1:

```
Operator:  *COBI DSSTATUS,#25,#3276,#1085

System:    JOBID:      SYSOUT:     VOL ID:     STATUS:

           #25         1P05        111111      ON-LINE
           #3276       U003        EXEC99      OFF-LINE
                       1P02        RTOSLK      OFF-LINE
           #1085       JCL
                       U001        222222      ON-LINE
                       2P01        111111      ON-LINE
           READY
```

Example 2:

```
Operator:  *CO DSS,IBM406,IBM789

System:    JOB ID:     SYSOUT:     VOL ID:     STATUS:
           IBM406
           #128        2P02        222222      ON-LINE
                       U001        222222      OFF-LINE
           #83         JCL
                       1P03        111111      ON-LINE
           IBM789
           #56
                       1P01        111111      ON-LINE
                       1P02        222222      ON-LINE
           READY
```

Example 3:

```
Operator:  *COBI DSS,SYSINA

When ANYJNAME parameter is specified:

System:    SYSINA      JOBID:      USER ID:
                       PROBN       COB021
                       VARDS       CBA120
                       CBSUBJ      AAB123
                       JOBK        ACX010
                       CALJSB      CAA011
           READY

When ANYJNAME parameter is not specified:

System:    SYSINA      JOBID:      USER ID:
                       COB02101    COB021
                       CBA12002    CBA120
                      *AAB12303    AAB123
                      *ACX01004    ACX010
                       CAA01105    CAA011
           READY
```

JOBSTATUS Function

The operator uses the JOBSTATUS function of the *COBI command to request the status of COBI jobs. Status may be requested for all jobs submitted for processing through COBI, for the jobs of a specific user, or for a specific job. The valid forms of the command are:

```
*COBI JOBSTATUS

*COBI JOBSTATUS,userid[,userid]...[,userid]

*COBI JOBSTATUS,job-number[,job-number]...[,job-number]
```

where

userid — is a validated user number of the form aaannn

job-number — is a COBI-assigned job number of the form #nnnnn and within the range 2 to 32767

If only JOBSTATUS is specified, the status of all the COBI jobs in the COBI index is printed in job number sequence; if no jobs exist, a message is printed. If one or more user numbers are specified, the status of all jobs associated with the specified users is printed in the order in which the user numbers appear on the command. If one or more job numbers are specified, only the status of the indicated jobs is printed in the order in which the job numbers appear on the command; if a job with the number specified does not exist, an error message is printed.

In all cases, the output consists of the job number and the status of the job. The status may be one of the following:

CANCEL RQ — Indicates that the job is in OS batch processing, but a request to cancel the job has been issued

COMPLETED — Indicates that execution of the job has been completed

JCL ERROR — Indicates that an error was detected when processing JCL statements

NOT DONE — Indicates that the job has been sent to OS batch processing but it has not completed execution

One or more completion codes are printed to indicate the cause of each job termination. The codes may be either system or user completion codes. System codes are issued by OS as described in the publication IBM System/360 Operating System: Messages and Codes; user codes, if any, are preceded by the letter U. Finally, the data sets associated with each job are listed by type, as follows:

JCL	Refers to the JCL for the indicated job
nPmm	Refers to a SYSOUT data set defined in a cataloged procedure, where n is the number of the procedure and mm is the number of the data set
Unnn	Refers to a user data set, where nnn is the number of the data set

The job status information normally appears on the command console used to issue the request. However, the output may be sent to the high speed printer by entering *COBI-P with the appropriate JOBSTATUS request; but in this case, *COBI-P must be entered as shown, with no abbreviation, but JOBSTATUS may be abbreviated as JOB. In either case, the system responds with READY at the command console after the output has been completed. When the output is sent to the high-speed printer, the end of the output is indicated by END on the printer.

The following examples show the use of the *COBI command to request job status. A sample of the appropriate output is also shown.

Example 1:

```
Operator:   *COBI JOBSTATUS,CAL428,IBM311,IB406

System:     JOB ID:      STATUS:       CODE:       DATA SET:
            CAL428
            #102         JCL ERROR                 JCL,1P03,2P01
            #506         NOT DONE                      3P04,3P05
            IBM311
            #423         COMPLETED      U0333          4P02,4P03,U001
            #1056        NOT DONE                  JCL,2P01
            #3000        CANCEL RQ
            PARAMETER,IB406    , INVALID ENTRY
            READY
```

Example 2:

```
Operator:   *CO JOB,#63,#145,#7200

System:     JOB ID:      STATUS:       CODE:       DATA SET:

            #63          COMPLETED   0C5,U0777         1P01,U005
            #145         COMPLETED        0C4     JCL,U007,2P01
            JOB ID #7200 HAS NO COBI INDEX RECORD
            READY
```

RESET Function

The operator uses the RESET function of the *COBI command to override initial startup parameters specified when CALL-OS was initialized. The mode of starting a reader may be changed as well as the conditions for switching readers. The valid forms of the command are:

```
*COBI RESET,AUTRDR=xxxxxx

*COBI RESET,RDRQTY=nnn
```

period is assigned to all data sets of the job. If the user scratches a data set, the retention period of any other data sets of the job is not affected. The CALL-OS command console operator can scratch any data sets for which the retention period has expired. The user should scratch specific data sets or complete jobs as soon as possible to ensure that sufficient space is available for other jobs.

If the user specifies an option of the form Unnn for which there is no corresponding DD statement containing &Unnn in the job, the Unnn option is ignored. The same is true for an option of the form nPmm when the job contains less than n invoked procedures. COBI cannot determine the SYSOUT data sets defined in a cataloged procedure. If the user specifies nPmm, and the invoked procedure n does not have a SYSOUT data set defined as Pmm, a JCL error occurs when OS/360 processes the job.

An OS/360 output queue with which a unique OS/360 output class is associated is reserved exclusively for the output of COBI jobs. It is by monitoring this output queue that COBI determines the status of a job and the locations and status of data sets specified by the user in his SUBMIT command for the job. If not otherwise specified at system initialization, the output class is Z.

The CALL-OS system can locate an output data set on the basis of the user number of the terminal user who submitted the job by which the data set was created, the COBI-assigned job number, and the identifier of the particular data set. The COBI option of retaining output data sets for printing (scanning) at the terminal is available only for printable physical sequential output data sets. COBI cannot be used to create partitioned data sets. (Hereafter, both the temporary OS/360 SYSOUT data set and the permanent OS/360 data set with the qualified data set name will be referred to as SYSOUT data sets.)

The CALL-OS terminal user can, within certain restrictions, scan OS/360 data sets other than those he has specified in submitted jobs. Such a data set must contain printable data and the data set name must be a qualified name in which the user number of the terminal user appears as one of the qualifiers. If an installation selects certain options when the CALL-OS system is initialized, the terminal user can also scan any data set whose first qualifier is a name specified by the installation at system initialization.

COBI-ASSIGNED JOB NUMBERS AND JOB NAMES

When a SUBMIT command has been accepted and a job prepared for processing by OS/360, an internal job number of the form #nnnnn is usually assigned to the job. A message is printed, informing the terminal user of this job number, so that he can use it in terminal commands referring to the job. The form of this message is shown below.

```
#nnnnn SUBMITTED AS job-name
```

In this message, #nnnnn is the system-assigned job number and job-name is either of the following:

- a user-entered job name from the OS/360 JOB statement for this job if CALL-OS has been initialized to permit this approach
- a COBI-supplied job name of the form aaannnxx where aaannn is a user number and xx is a unique system-generated identifier if the user has not specified a job name in the JOB statement or if CALL-OS has been initialized to determine the job name

The character set from which two characters may be selected and ordered by the system in creating the xx identifier of a COBI-supplied

job name provides for 1520 unique combinations. It is unlikely that identical identifiers will be assigned concurrently to two or more of a user's jobs. However, the user should be aware that this situation is possible.

Since the COBI-assigned job number is used in terminal commands referring to the job, the user must note and remember the job number. If the user fails to retain the relationship between a job name and its assigned job number, the inquiry functions (JOBSTATUS and DSSTATUS) do not directly enable him to recover it. If the job has completed, he can recover this information by:

1. Using a JOBSTATUS command to determine the job numbers assigned to all his jobs.

2. Using a SCAN command to obtain a printout of the JCL for any specific job for which he specified JCL in the options field of his initial SUBMIT command. The JCL indicates the job name by which OS/360 recognizes the job.

There is a situation where no job number is assigned to a submitted job. This situation occurs when the user does not specify that JCL or SYSOUT data sets be retained for scanning and directs the output of his job by specifying a MSGCLASS other than Z (the unique COBI output class) in the JOB statement for the job (see "JOB Statement"). No job number is needed because neither the terminal user nor COBI will have reason to refer to the job. The JCL and system messages are not retained automatically (and, therefore, are not available in case of job failure) for a job to which no job number is assigned.

CALL-OS TERMINAL COMMAND LANGUAGE

The CALL-OS terminal command language is designed to facilitate communication between the terminal user and the computer. Through a series of terminal commands, the user can access system resources applicable to his particular needs. The language is structured so that the individual user can concentrate on problem solution rather than on system organization or equipment.

USER ENTRY OF TERMINAL COMMANDS

Most terminal commands may be typed in full or shortened to a three-letter abbreviation. Any exceptions are noted in the descriptions which follow. The first parameter of a command must be separated from its command word(s) by one or more blanks. Blanks cannot be embedded within a parameter entry. Blanks before or after succeeding parameter entries are ignored. Commas must separate succeeding parameter entries. Some examples are shown below.

```
CLEAR

CLE

MERGE PROG1,PROG2,40

MER PROG1,PROG2,40
```

The following rules are used in entering the commands:

1. Type the command word(s) and strike the space bar if parameter entries follow.

2. Type any parameter entries, inserting a comma after each entry if more than one entry is typed. Spacing after the comma is optional.

3. Press the RETURN key at the end of each line.

Procedures for making typing corrections when entering either terminal commands or program statements are given under "Typing Corrections" in preceding sections of this manual (for both the 2741 Communications Terminal and the Teletype).

DOCUMENTATION CONVENTIONS

The following conventions are used in this manual to describe command formats:

1. Uppercase letters, asterisks, and commas represent information that must appear exactly as shown.

2. Lowercase letters represent information supplied by the user.

3. A series of three periods indicates that a variable number of entries may be supplied.

4. Brackets surround optional entries that may or may not be specified.

SPECIFIC COMMAND FORMATS

CALL-OS terminal commands are listed and explained in alphabetic sequence in the remainder of this section. Additional terminal commands available to users of the CALL-OS Batch Interface (COBI) option are given in the section that follows.

ADD

General Form (adding to single lines)

```
ADD line-number, line-number,..., 'string'
```

General Form (adding to groups of lines)

```
ADD line-number THRU line-number,..., 'string'
```

General Form (adding to all lines in a program)

```
ADD ALL, 'string'
```

Valid Entries

The string of characters may be any characters allowable in System/360. Line numbers must be positive integers in ascending numeric sequence. When a group of consecutive lines is specified, neither the low nor high line number has to exist (although it may) in the current program; the addition operation is performed on all lines within a specified range, inclusive. An entry of 99999 as a line number in a THRU specification is interpreted as a reference to the last line of the program. Up to nine ranges (single lines, groups of lines, or a combination of both) may be specified in one ADD command. The ALL parameter entry may not be used in conjunction with any other range specification, but either ALL or one or more line numbers must be specified. The abbreviations A for ALL and T for THRU may be used.

Description

This command causes a string of characters to be added to the end of a single line, each of a group of consecutive lines, or every line in a source program. The string of characters must be enclosed in single quotes as the last parameter entry of the ADD command. The quotes are not part of the string and are not included in the addition. If a single quote is to be included in the string, it must be represented by two single quotes. Succeeding parameter entries are separated by commas. If a single line number that does not exist is specified as a parameter entry, the entry has no effect on the current program. A nonexistent line number specified in a THRU specification serves as a boundary indicator, delimiting the range of lines on which the add operation is performed.

Add operations are performed as required on the program in the user work area unless that program is currently protected (flagged as RUN ONLY). If the program is protected, the ADD command is rejected. The following message is printed at the terminal.

```
PROTECTED PROGRAM
```

If the new length of a line exceeds the maximum length allowable (237 characters), an error message is generated. The add operation is terminated, and no lines of the program in the user work area are changed.

Examples

```
ADD 10,30,55,'THIS'
READY

ADD 15,50 T 80,120 T 150,'THIS ISN''T'
READY

ADD ALL,'THIS'
READY
```

ALLOW (see PROTECT)

General Form

ALLOW program or data filename

Valid Entries

The filename must be the name of a program or data file in the user's library.

Description

This command is used to remove protection from a program or data file whose name is in a *Library or the **Library. The user who has pooled and protected the file in the shared library is the person who issues the ALLOW command. A specified program may now be listed, saved, or stored by other users of the library. A specified data file may be opened for input by other users of the library.

If the ALLOW command is given with the filename omitted, the system prints:

FILE NAME--

The user should type the program or data filename.

When the program is accessed by LOAD *, LOAD **, or LOAD ***, the program being loaded is tested for a RUN ONLY condition, which is set using the PROTECT command. If the program is flagged as RUN ONLY, the program may not be listed, saved, or stored. When the program is not RUN ONLY, it may be listed, saved, or stored. When the user retrieves his own program by using LOAD program-name, the RUN ONLY attribute is not attached; he may LIST, SAVE, or STORE his program.

For a description of CALL-OS system libraries, see the CATALOG (shared libraries) command. Note that the ALLOW command may be used to remove protection from a ***Library program or data file, only by central computer installation personnel signed on at the command console.

Examples

```
ALLOW PROG1
READY

ALL PROG2
READY
```

CATALOG (user's library)

General Form

```
CATALOG
```

Description

This command is used to print a list of program and data files stored in the user's library.

Examples

```
CATALOG
18:20    10/15/71    FRIDAY
PROG1      PROG2


CATALOG
14:30    12/20/71    MONDAY
PROG1      MYDATA      PROG2      /PROG2
```

CATALOG (shared libraries)

General Form

```
CATALOG *

CATALOG **

CATALOG ***
```

Description

The CATALOG * command instructs the system to print a list of the program and data filenames in the *Library. The *Library is a special library shared by all users who have the same first four characters in their user numbers. Together, these users constitute a sub group.

The CATALOG ** command instructs the system to print a list of the program and data filenames in the **Library. This library contains program and data files shared by all system users.

The CATALOG *** command instructs the system to print a list of the program and data filenames in the ***Library. The ***Library is a special-purpose library containing program and data files available to all system users. Entry of programs into this library is controlled through the use of a special user number. These programs are entered through the command console at the central computer installation.

Examples

```
CATALOG **
10:05    10/15/71    FRIDAY
PROG1      PROG2       PROG3


CAT **
16:10    12/20/71    MONDAY
DATA1      DATA2       FORT        BUSPROG    /STAT
```

CATALOG ALL

General Form

CATALOG ALL

Valid Entries

The command may be given as shown above, or the abbreviation CAT ALL may be used.

Description

This command instructs the system to print a list of user program and data files, including certain descriptive information. Such information includes program or data filename, type of file, language, number of characters in the source program, number of disk storage units allocated to the file, and days since last accessed. For data files, the language name indicates the language being used when the file was last opened for output.

Example

```
CAT ALL
10:00   10/15/71   FRIDAY
FILE NAME    TYPE   LANGUAGE    PROGRAM SIZE  FILE UNITS  DAYS SINCE USED

LINDA        PROG   BASIC              599                      55
SPREADSH     PROG   FORTRAN           4509                      27
LESSON3      PROG   BASIC  L          2494                       2
FORECAST     DATA   BASIC                          12            2
MYFCST       DATA   BASIC                          10            2
/PROG4       OBJ    FORTRAN                        12           17
PRIII        PROG   BASIC             2277                      84
STOCKPLT     PROG   PL/I               670                     202
DATA1        DATA   PL/I                            2           55
```

Note: The full language name is printed, including the long-form indicator, where applicable.

CLEAR

General Form

```
CLEAR
```

Description

This command instructs the system to clear the user work area. Such a work area is assigned to each user number and password identification code at the successful completion of sign-on. It is that part of the system used for problem solving.

In response to the CLEAR command, all source-program statements and the program name are cleared from the current user work area. The current language name and the print-line-width indication remain unchanged.

Examples

```
CLEAR
READY

CLE
READY
```

General Form (deleting single lines)

```
DELETE line-number, line-number,...
```

General Form (deleting groups of lines)

```
DELETE line-number THRU line-number, line-number THRU line-number,...
```

Valid Entries

Line numbers must be positive integers in ascending numeric sequence. When a group of consecutive lines is specified, neither the low nor high line number has to exist (although it may) in the current program. The delete operation is performed on all lines within a specified range, inclusive. An entry of 99999 as a line number in a THRU specification is interpreted as a reference to the last line of the program. Single lines and groups of lines may be specified in one DELETE command. The abbreviation T for THRU may be used.

Description

This command causes one or more program lines to be deleted. Line numbers of single lines to be deleted are separated by commas. Groups of consecutive lines are specified as shown above. If a single line number that does not exist is specified as a parameter entry, the entry has no effect on the current program. A nonexistent line number specified in a THRU specification serves as a boundary indicator, delimiting the range of lines on which the delete operation is performed. When all statements of a program are deleted, the system response is:

```
NO PROGRAM LEFT
```

Delete operations are performed on the program in the work area. If that program is currently protected (flagged as RUN ONLY), however, the DELETE command is rejected. The following message is printed at the terminal.

```
PROTECTED PROGRAM
```

Note: A single line can also be deleted by entering its line number and striking the carrier return. See "Edit Capabilities" in a preceding section of this manual.

Examples

```
DEL 10,30,65,80
READY

DELETE 10,50 T 75,120 T 140
READY
```

General Form

ECHO any sequence of letters, numerals, and special characters

Valid Entries

Any characters available on the terminal can be specified.

Description

This command instructs the system to check the transmission of characters from the terminal to the computer. Although any number of characters may be keyed, a maximum of 76 characters are checked by the system. (The maximum number of characters that can be keyed may be further limited by the line width of the user terminal; for example, the Teletype maximum line width is 72.) The system responds at the terminal by printing the characters exactly as they were entered.

If the line printed by the system does not match the line typed, transmission is faulty. In this event, the ECHO command should be repeated. If the transmission problem persists, the terminal should be signed off and another sign-on procedure initiated. Transmission checking should be repeated by issuing another ECHO command.

Examples

```
ECHO ABCDEFGHIJKLMNOPQRSTUVWXYZ+-*/          (terminal entry)
ECHO ABCDEFGHIJKLMNOPQRSTUVWXYZ+-*/          (system response)


ECH ABC///+0678PRU                           (terminal entry)
ECH ABC///+0678PRU                           (system response)
```

General Form

ECHOX any sequence of letters, numerals, and special characters

Valid Entries

ECHOX cannot be abbreviated. Any characters available on the terminal can be specified.

Description

This command is identical to the ECHO command except that the system response may be printed repetitively. The response line is reprinted, successively, until the terminal operator presses the ATTN key (2741) or BREAK key (Teletype). Then the system responds as shown by the last two lines of the example below.

Example

```
ECHOX ABCDEFGHIJKLMNOPQRSTUVWXYZ+-*/          (terminal entry)
ECHOX ABCDEFGHIJKLMNOPQRSTUVWXYZ+-*/          (system response)

ECHOX ABCDEFGHIJKLMNOPQRSTUVWXYZ+-*/

ECHOX ABCDEFGHIJKLMNOPQRSTUVWXYZ+-*/
   .                  .
   .                  .
   .                  .
   (user hits ATTN or BREAK key)
STOP.
READY.
```

ENTER

General Form

ENTER language-name

Valid Entries

Valid language names and permissible abbreviations are shown below.

• BASIC (short-form arithmetic)	BASI
• BASICL (long-form arithmetic)	(no abbreviation)
• PL/I	(no abbreviation)
• FORTRAN	FORT
• DATA	(no abbreviation)

Description

This command informs the system of the programming language being used. The command need not be issued before a program is written, but the language name must be declared, retained with the program, and/or used at run time to select the appropriate language processor. BASIC is assumed if no ENTER command is given. An ENTER command remains in effect until a subsequent ENTER, RUN, or LOAD command is typed.

The ENTER DATA facility is provided to enable CALL-OS users to enter line-numbered information without associating that information with a specific CALL-OS language compiler. Information entered under this language-name can be retained in the user's library by means of a SAVE command. The language-name DATA is retained with the information. Thus, for example, the user can avoid the default to BASIC (in cases where neither PL/I, FORTRAN, BASIC, or BASICL is appropriate) and, at the same time, avoid the various editing conventions that might be applied later if the information were involved in edit operations as a CALL-OS BASIC program file. Such a file (called a program-data file) can also be opened for input by an executing program. The line number preceding a line is not considered as part of the input; the first character following one blank (or following the line number if no blank is included) is treated as the first data character of the line.

If the ENTER command is given without the language-name parameter, the user is prompted with:

```
LANGUAGE NAME--
```

The user should type the name of the language.

The specified language name is saved or stored when the program is saved or stored. When a program is loaded using the LOAD command, the language name associated with the program is also loaded.

Examples

```
ENTER BASI
READY

ENTER BASICL
READY

ENT FORT
READY
```

EXTRACT

General Form (extracting single lines)

```
EXTRACT line-number, line-number,...
```

General Form (extracting groups of lines)

```
EXTRACT line-number THRU line-number, line-number THRU line-number,...
```

Valid Entries

Line numbers must be positive integers in ascending numeric sequence. When a group of consecutive lines is specified, neither the low nor high line number has to exist (although it may) in the current program. The extract operation is performed on all lines within a specified range, inclusive. An entry of 99999 as a line number in a THRU specification is interpreted as a reference to the last line of the program. Single lines and groups of consecutive lines may be extracted using one EXTRACT command. The abbreviated form T for THRU may be used.

Description

This command is used to extract one or more lines from a program. Line numbers of single lines to be extracted are separated by commas. Groups of consecutive lines are specified as shown above. All specified lines are extracted from the program and remain in the work area. All other lines in the program are deleted. If a single line number that does not exist is specified as a parameter entry, the entry has no effect on the current program. A nonexistent line number specified in a THRU specification serves as a boundary indicator, delimiting the range of lines on which the extract operation is performed.

If the program in the work area has been protected (flagged as RUN ONLY), the EXTRACT command is rejected. The following message is printed at the terminal.

```
PROTECTED PROGRAM
```

Examples

```
EXTRACT 20,30,40,50,60,70
READY

EXT 65 THRU 140,180,200 T 340
READY
```

FILE

General Form

```
FILE filename[,n]
```

Valid Entries

The filename is the name to be associated with a data file and n is the maximum number of disk storage units to be allowed for the file. The filename must begin with a letter or one of the symbols #, $, or @. It may contain any combination of alphabetic characters (including the characters #, $, and @), numeral Maximum length for the filename is eight characters.

Description

This command is used to name a data file to be used with a program at run time, increase the maximum number of disk storage units that can be allocated to an existing data file, or determine the maximum number of disk storage units specified for an existing data file.

If the command is being used to name a data file, n specifies the maximum number of disk storage units to be allowed for the data file. If n is omitted, a value of 4 is assumed. An entry for n cannot be greater than the maximum number of disk storage units that can be allowed for a data file, as established at system initialization time. The default value for maximum number of units is 100. Each of the allowed disk storage units is 3440 bytes in length. Whether n is specified or not specified for this use of the FILE command, the system responds READY, as shown by the examples below.

The system creates an entry for the file in the user's catalog and sets a limit indicator, which controls the amount of storage that can be used for the file. The filename must be specified in any program that reads or creates data for the file.

If the FILE command is being used to increase the permissible maximum amount of storage for an existing data file, both filename and n entries must be present. The filename entry must match a filename in the user's catalog. The new value of n replaces the existing value of n for the data file. The new value of n must be greater than the old value. If it is less than the old value, the FILE command is rejected. The system responds with:

```
filename LIMIT = xx UNITS
```

where xx is the previously specified maximum number of disk storage units for the file. The command is also rejected if the filename entry is not the name of a data file.

If the FILE command is being used to determine the maximum number of disk storage units specified for an existing data file, the filename is included, but the n entry is omitted. The system responds with:

```
filename LIMIT = xx UNITS
```

where xx is the maximum number of disk storage units that can be allocated for the file.

Note: The number of disk storage units currently allocated for the file can be determined by use of a CATALOG ALL command.

If the FILE command is given and the filename entry is omitted, the system prompts the user with:

```
FILE NAME--
```

The user should respond with a valid filename entry.

Data files are attached to programs during execution. Use of a filename in an OPEN statement in a program causes that file to be attached to the program.

Examples

```
FILE DATA1
READY

FIL DATA2,9
READY

FILE DATA2
DATA2 LIMIT = 9 UNITS
```

FIND

General Form (searching single lines)

```
FIND line-number, line number,...,{'string'          }[,NOTEXT]
                                  {"qualified-string"}
```

General Form (searching groups of lines)

```
FIND line-number THRU line-number,...,{'string'          }[,NOTEXT]
                                      {"qualified-string"}
```

General Form (searching all lines in a program)

```
FIND ALL,{'string'          }[,NOTEXT]
         {"qualified-string"}
```

Valid Entries

Line numbers must be positive integers in ascending numeric sequence. When a group of consecutive lines is specified, neither the low nor high line number has to exist (although it may) in the working program. All lines within a specified range, inclusive, are searched. An entry of 99999 as a line number in a THRU specification is interpreted as a reference to the last line of the program. Up to nine ranges (single lines, groups of lines, or a combination of both) may be indicated in one FIND command. The ALL parameter entry may not be used in conjunction with any other range specification, but either ALL or one or more line numbers must be specified.

Either type of character string may contain any characters allowable in System/360. In the FIND command, the following abbreviated forms for parameter entry may be used: T for THRU, A for ALL, and N for NOTEXT.

Description

This command causes one or more source-program lines to be searched for the presence of a specified string of characters. A string bounded by single quotes ('string') is located at its first occurrence in any line within the specified range(s). A string bounded by double quotes ("qualified-string") is located (at its first occurrence in a line) only when delimited at each end by a special character serving as a delimiter. The characters recognized as delimiters are:

- one or more blank characters
- a carrier return (at the right end)
- any of the following special characters

```
( ) + - / & * ; : , . ! ? | ' "
> < ≥ ≤ = ≠ ¬
```

The delimiters specified at the ends of a qualified string need not be the same. Neither a string enclosed in single quotes nor a qualified string is detected, however, if it is spread across continuation lines. Since CALL-OS edit routines handle each line as a unit, the complete string must appear in one line.

A single quote within a string must be specified by two successive single quotes. A double quote within a qualified string must be specified by two successive double quotes.

Succeeding parameter entries are separated by commas. If a single line number that does not exist is specified as a parameter entry, the entry has no effect on the current program. A nonexistent line number specified in a THRU specification serves as a boundary indicator, delimiting the range of lines on which the search operation is performed. If the NOTEXT parameter is specified, only the line number of each line containing the specified character string is printed. If NOTEXT is omitted, the line number and the entire text of each line containing the character string are provided as output.

All operations are performed on the program in the work area unless that program is currently protected (flagged as RUN ONLY). If the program is protected, the FIND command is rejected. The following message is printed at the terminal.

```
PROTECTED PROGRAM
```

Examples

```
FIND 10 THRU 30,50 THRU 375,'ABCD',N
10      20      70      . . .
 .
 .
 .
 (listing of all line numbers between 10 and 30 and between 50 and
  375, preceding lines containing the character-string ABCD)
READY


FIN ALL, "X1"
40 X1 = A + B
50 Y1 = X1/R
 .
 .
 .
 (listing of the line numbers and text of all lines in the program
  containing the character-string X1 appropriately delimited)
READY


FIN A,'Q''TY SHIP''D',NOTEXT
100     150     200     210
.
.
.
(listing of the line numbers of all lines in the program
 containing the character-string Q'TY SHIP'D)
READY
```

HELP

General Form

HELP

Description

When the HELP command is issued, a single-line system message is printed out at the terminal. Generally, the user is directed to take installation-determined action. The default content of this message is shown below. The message is put out by the CALL-OS system module M#HELP. Its content can be changed easily by a system programmer at the central computer installation if desired. The programmer need only change the content of the DC labeled MESSG in the M#HELP module. Up to 100 characters can be specified.

Example

```
HELP
PLEASE CONTACT YOUR INSTALLATION MANAGEMENT FOR ASSISTANCE.
```

General Form

```
INSERT line-number, line-number,..., 'text-of-line'
```

Valid Entries

Line numbers must be positive integers in ascending numeric sequence. Each line number may or may not precede a line of the current program. Only single line numbers can be entered (that is, a THRU or ALL range specification is not permitted). Up to nine line numbers can be entered in one INSERT command, and at least one line number is required.

Description

This command is used to insert a specified line in the program currently in the user work area. The text enclosed in single quotes is assigned a line number specified in the INSERT command and inserted in numerical sequence within the current program. If multiple line numbers are specified (separated by commas), the new line is inserted repetitively. Each new line is assigned one of the specified line numbers. If a current line is identified by a line number specified in the INSERT command, that line is replaced by the new line.

The single quotes required in the INSERT command are not included in a line inserted in the working program. A single quote in the line of text must be represented by two single quotes.

All operations are performed on the program in the work area. If that program is currently protected (flagged as RUN ONLY), however, the INSERT command is rejected. The following message is printed at the terminal.

```
PROTECTED PROGRAM
```

Examples

```
INSERT 70,95,'PRINT X1'
READY

INS 100,120,300,'LET X = CAN''T'
READY
```

KEY (see TAPE, TAPE ALL)

General Form

```
KEY
```

Description

This command restores all functions inhibited by a previous TAPE or TAPE ALL command.

If this command is entered from a Teletype, the system response is:

```
READY
```

If entered from any other terminal type, the system response is:

```
WHAT?
```

See also the discussion of paper tape input/output under "Paper Tape Operations".

Example

```
KEY
READY
```

General Form

```
LIST [line-number]
```

Description

This command instructs the system to print all or part of a program in the work area. The line number must consist of from one to five contiguous numeric characters in the range 0-99999. If no line number is specified, the entire program is listed. When a line number is specified, program lines are listed in sequence by line number beginning with the line identified by that number. If a specified line number does not exist in the current program but is not higher than all existing line numbers, program lines are listed in sequence beginning with the first existing line number higher than the one specified. If 99999 or a line number higher than all existing line numbers is specified, only the last line of the program is listed.

Any listing produced in response to a LIST command begins with a heading line that contains the name of the program being listed and the current date. The length of each print line is specified by the user via a WIDTH command or is a default line width of 72 if no WIDTH command is entered. It may be desirable to adjust the current line width specification when printing short-line or column-type output to reduce the total printing time (see WIDTH command). If the user wants to stop the list operation before the program file is exhausted, he can do so by pressing the ATTN key (2741) or the BREAK key (Teletype).

Examples

```
LIST

PROG2        16:19    10/15/71    FRIDAY

10 PLIPRG:  PROCEDURE;
20 DCL LIT CHAR(2);
.
.
.


LIS 300

BASPROG      17:20    12/20/71    MONDAY

300 LET X=10
310 LET Y=30
320 R=X+Y-Z
.
.
.
```

LIST-NO-HEADER

General Form

```
LIST-NO-HEADER [line-number]
```

Valid Entries

The command may be given as shown above, or the abbreviation LIST-N may be used.

Description

This command is identical to LIST except that headers do not print.

Example

```
LIST-N
10 PLIPRG:  PROCEDURE;
20 DCL LIT CHAR(2);
.
.
.
```

Note: The user may combine use of the LIST-N and LIST-T commands by specifying either LIST-TN or LIST-NT. Either causes the complete program in the user work area, or a specified part of it, to be printed without headers and without line numbers.

General Form

```
LIST-TEXT [line-number]
```

Valid Entries

The command may be given as shown above, or the abbreviation LIST-T may be used.

Description

This command is identical to LIST except that line numbers do not print; only text is provided as output.

Example

```
LIST-T

PROG2        16:19    10/15/71    FRIDAY

PLIPRG:  PROCEDURE;
DCL LIT CHAR(2);
.
.
.
```

Note: The user may combine use of the LIST-N and LIST-T commands by specifying either LIST-TN or LIST-NT. Either causes the complete program in the user work area, or a specified part of it, to be printed without headers and without line numbers.

LOAD (user's library)

General Form

```
LOAD program-name
```

Valid Entries

The specified program name must be the name of a source program in the user's library.

Description

This command instructs the system to retrieve a source program from user storage and place it in the user work area. When a program is loaded, it overlays and replaces the previous program, program name, and language name in the work area.

If the program name is not typed with the LOAD command, the system prints:

```
FILE NAME--
```

The user should type the name of the program.

Note: Loading a data file or an object program is not allowed.

Examples

```
LOAD PROG2
READY

LOA INVPRG
READY
```

LOAD (shared libraries)

General Form

```
LOAD *program-name

LOAD **program-name

LOAD ***program-name
```

Valid Entries

The specified program name must be the name of a source program pooled into the indicated library.

Description

This command instructs the system to locate a source program name in the shared library, designated by asterisks, and place the program in the work area. The program being loaded is tested for a RUN ONLY condition, which is set using the PROTECT command. If the program is flagged as RUN ONLY, the program may not be listed, saved, or stored. When the program is not RUN ONLY, it may be listed, saved, or stored (see ALLOW command). When the user retrieves his own program by using LOAD program-name, the RUN ONLY attribute is not attached; he may LIST, SAVE, or STORE his program. When the program is loaded, it overlays and replaces the previous program, program name, and language name in the work area.

If the program name is not typed with the LOAD command, the system prints:

```
FILE NAME--
```

The user should type the appropriate number of asterisks and the name of the program.

Note: Loading a data file or an object program is not allowed.

Examples

```
LOAD **PROG2
READY

LOA *PAYPROG
READY
```

LOCK (see UNLOCK)

General Form

```
LOCK filename
```

Valid Entries

The filename must be the name of a program or data file in the user's library.

Description

Execution of this command prevents a program or data file in the user's library from being accidentally destroyed. The program or data file is locked in the user's library. The program or data file cannot be deleted from the library, nor may another program or data file with the same name be saved or stored in the library, without first removing the LOCK protection. If a data file is locked, it cannot be opened for output.

If, instead of READY, the system responds:

```
FILE NAME--
```

the user should type the program or data filename.

Examples

```
LOCK PROG1
READY

LOC PAYPROG
READY
```

General Form

LOGON

Description

This command is used for either of two purposes:

1. To sign-on with a different user number at a 2741 Communications Terminal or Teletype without disconnecting the terminal from the system, or

2. To sign-on at a 2741 Communications Terminal after successfully dialing in the system.

In the former case, a user types LOGON or LOG without disconnecting the line from the system. The system responds by printing the off message, time, and sign-on message on the terminal. This procedure is employed by a second user at a terminal who wishes to sign-on the system from the terminal without going through the disconnect sequence.

In the latter case, the system determines the type of 2741 Communications Terminal (Correspondence or EBCD) by analysis of the transmission codes generated by the input command LOG or LOGON. It prints a sign-on message in correspondence or EBCD code on the terminal accordingly. If the input data is only a carrier return or data other than the LOGON command, the system defaults to the terminal type defined in the system startup deck. The time and sign-on message are printed on the terminal. Since this case is an initial sign-on, the off message is not provided.

In either case, the user then enters a new user number and password. The system responds with READY if all is in order; otherwise, the user is again asked for his user number and password. The user must respond with correct indentifiers within two minutes or his terminal is disconnected. After the LOGON command has been successfully completed, the user has a cleared work area. The BASIC language processor with short-form arithmetic is assumed. The terminal line width is assumed to be a length of 72.

Example

```
LOG
OFF AT 8:52
PROC. TIME...    0 SEC.
TERM. TIME...    3 MIN.


ON AT 8:52  01/15/71 S.J. LINE 14
WELCOME TO CALL-OS.......UP UNTIL 10:30
USER NUMBER,PASSWORD--HAL020,ADHDQ
READY
```

DIBIN429 I/O ERROR WRITING COBI VOLID TABLE

A permanent I/O error occurred while attempting to update the COBI volume identification table. The ABEND dump produced should be brought to the attention of the installation system programmer immediately. To initialize the CALL-OS system, specify the NOCOBI parameter in the startup deck and restart the job.

DIBIN430 MAXDCB VALUE EXCEEDS NUMBER OF LINES

The value specified for the MAXDCB parameter exceeds the number of lines defined for this session. Initialization continues. The value used is the number of lines.

DIBIN431 INSUFFICIENT MAIN STORAGE ALLOCATED FOR DEB'S.

MFT uses main storage in the CALL-OS partition for data extent blocks. The actual amount used depends on the number of users to be permitted to scan data sets concurrently (MAXDCB parameter). Increase the amount of storage in the CALL-OS partition accordingly and restart the job. To initialize CALL-OS immediately without COBI, specify the NOCOBI parameter and restart the job.

DIBIN432 INVALID SYSJOBQ DATA SET

The data set defined by the SYSJOBQ DD statement is not a valid OS/360 system job queue data set. Correct the DD statement and restart the job. To initialize the system immediately without COBI, specify the NOCOBI parameter and restart the job.

DIBIN433 INVALID CBSYSIN DATA SET. DDNAME-dddddddd.

The data set described by the specified DD statement is not a valid formatted COBI input data set. Ensure that the DD statement is correct; if the DD statement is correct, the data set must be reformatted. The ABEND dump produced should be brought to the attention of the installation system programmer immediately. To initialize CALL-OS without COBI, specify the NOCOBI parameter and restart the job.

DIBIN434 IDENTICAL COBI SYSIN DATA SETS

The same data set is described by both the CBSYSINA and CBSYSINB DD statement. Correct the DD statements and restart the job. To initialize CALL-OS without COBI, specify the NOCOBI parameter and restart the job.

DIBIN436 DISK ERROR READING MASTER QCR CONTROL RECORD

A permanent I/O error has occurred while reading the master queue control record from SYS1.SYSJOBQE. The installation system programmer should be contacted.

DIBIN501 SYNTAX ERROR IN RESMODS LIST.

The RESMODS list has an invalid format. Check the RESMODS DD card to ensure that it is pointing to the desired member of SYS1.PROCLIB. Each card image in the member pointed to must contain a list of module names, starting in column 1 of each card image, separated by commas, with no intervening blanks, and not extending beyond column 71. No continuation indicator is required in column 72.

To bring the system up immediately, remove the RESMODS DD card. If a completely resident system is desired, also remove the OVLY DD card. If a completely nonresident system is desired, leave the OVLY DD card in the startup deck.

DIBIN502 ERROR IN RESMODS HIERARCHY PARAMETER.

A parenthesized expression other than (0) or (1) followed a module name in the RESMODS list. Correct the list and restart the job. To bring the system up immediately, remove the RESMODS DD card. If a completely resident system is desired, also remove the OVLY DD card. If a completely nonresident system is desired, leave the OVLY DD card in the startup deck.

DIBIN503 MODULE IN RESMODS LIST NAMED mmmmmmmm IGNORED. UNKNOWN TO SYSTEM.

The module name mmmmmmmm was included in the RESMODS list but cannot be found in the system. Initialization will continue, but a check should be made to ensure that this message does not indicate a misspelled name in the RESMODS list.

DIBIN504 MODULE NAMED mmmmmmmm IGNORED, NOT IN SPECIFIED JOBLIB.

Initialization will continue, but this message may indicate a serious system problem. If the module named is necessary for successful system operation (that is, not a temporary testing or development module) the situation should be corrected and the system restarted.

DIBIN505 INSUFFICIENT CORE STORAGE ALLOCATED FOR RESIDENT MODULES.

Check to ensure that the CALL-OS job has been initiated in the desired task area. For MVT, ensure that the desired task area size was specified correctly. Increase the task area specifications and restart the job.

If Hierarchy 1 storage is being used, check the heirarchy specifications in the RESMODS list to determine in which hierarchy the shortage occurred.

Note: The terminal translate tables are loaded the same as potentially resident modules. However, the default hierarchy for these tables is the hierarchy specified for pots (24-byte buffers).

DIBIN506 NO SPACE TO WRITE MODULE mmmmmm IN OVLY DATA SET. REST OF MODULES MADE RESIDENT.

Not enough space was available in the OVLY data set. Since one cylinder is sufficient for this data set, the occurrence of this message may mean serious system problems.

Initialization will continue with the remaining modules being made resident, but to obtain the exact resident configuration desired, the OVLY data set should be reallocated.

DIBIN507 I/O ERROR WRITING MODULE mmmmmmmm IN OVLY DATA SET. REST OF MODULES MADE RESIDENT.

An irrecoverable I/O error occurred while writing nonresident modules to the CALL-OS OVLY data set. Initialization will continue with the remaining modules being made resident, but to obtain the exact resident configuration desired, the OVLY data set should be reallocated.

General Form (moving a single line)

MOVE destin-line-number, line-number[, increment]

General Form (moving a group of lines)

MOVE destin-line-number, line-number THRU line-number[, increment]

Valid Entries

Line numbers must be positive integers. The destination line number must not be the same as the number of the line to be moved or within the range of a group of lines to be moved. It may be either lower or higher and need not actually exist in the current program. When a group of consecutive lines is specified, neither the low nor high line number has to exist (although it may) in the current program. All lines within a specified range, inclusive, are moved. The lower of the specified limits must be indicated first. The abbreviated form T for THRU may be used. An entry of 99999 as a destination line number or in a THRU specification is interpreted as a reference to the last line of the program. The increment must be a positive integer.

Description

This command extracts the single line or group of lines indicated by the second parameter of the command from their current location in the working program. It moves them to the position immediately following the position indicated by the destination line number. If an increment is specified, the moved line or lines are renumbered using this increment. Existing lines following the moved lines are renumbered using the same increment but only as necessary to prevent overlapping of line numbers. If no increment is specified, a default of 1 is assumed.

If the renumbering operation causes one or more main-program lines not within a range specified to be moved to be renumbered, the following message is printed:

MAIN PROGRAM RENUMBERED AT nnnnn

where nnnnn is the original line number of the first unmoved main-program line to be renumbered.

If the move operation and subsequent renumbering (by the specified increment or by 1) causes a line number greater than 99999 to be generated, the move operation is not performed. The following message is printed at the user's terminal.

GENERATED LINE NUMBER EXCEEDS 99999

If a single line number that does not exist is specified to identify a line to be moved, or if no lines exist within a specified range, no change is made to the program. The following message is printed at the user's terminal.

NO LINES MOVED

The move operation is performed on the program in the work area. If that program is currently protected (flagged as RUN ONLY), however, the

MOVE command is rejected. The following message is printed at the terminal.

```
PROTECTED PROGRAM
```

A program written in CALL-OS BASIC may contain statements with line-number references. The text of such statements is modified to reflect any new line numbers created during the move operation. A line number is reserved after the moved portion of the program as a reference point for any statements that originally referenced nonexistent lines.

Note: If original BASIC statements contain invalid BASIC syntax, causing line-number references to be unclear, unpredictable results may occur.

Examples

```
MOVE 15,50
READY

MOV 10,300 T 400,10
READY
```

General Form

NAME program-name

Valid Entries

The specified program name must begin with a letter (any character from the extended alphabet, which includes the characters #, $, and @). It may contain any combination of letters (including #, $, and @), numerals, and break characters (_). Maximum length for a program name is eight characters.

Description

This command is used to assign a program name to a program in the work area. Should the program in the work area already have a program name, the new name replaces the old one. The specified name appears in the heading line printed out as a result of a subsequent LIST or RUN command, and is saved or stored as the name of the program if a SAVE or STORE command is used.

If, instead of READY, the system prints:

```
FILE NAME--
```

The user should type the program name after the hyphens.

Examples

```
NAME PROG2
READY

NAM MULTPROG
READY
```

General Form

```
OFF
```

Description

This command is issued by the user to sign off, that is, to deactivate his terminal. The system prints the amount of processing time used and the length of time the terminal was connected to the computer. Then it disconnects the terminal from the system.

Example

```
OFF
OFF AT 14:36
PROC. TIME...    1 SEC.
TERM. TIME...    4 MIN.
```

PASSWORD

General Form

```
PASSWORD password
```

Valid Entries

The password may contain up to eight characters and may consist of any combination of letters, numerals, and special characters. The first character may not be a blank, but blanks may be present in any other positions. Blanks that precede the first letter, numeral, or special character are ignored. The password begins with the first significant (nonblank) character typed.

Description

Execution of this command establishes a password for a new user (permitting him to select other than a system-assigned default password) or replaces the password of an existing user. The new password replaces the password by which the user signed on the system. The PASSWORD command can be used to change the password only after the user is successfully connected to the system. The new password must be used in all future sign-ons.

If, instead of READY, the system prints:

```
PASSWORD--
```

the user should type the new password.

If the carrier is returned without typing a password, the old password is retained.

Examples

```
PAS R68$4
READY

PASSWORD XYZ123##
READY
```

POOL (see PULL)

General Form

```
POOL *filename

POOL **filename
```

Valid Entries

Any name (up to a maximum of eight alphameric characters) may, in general, be pooled into the *Library or **Library. The program or data file to be shared should have been saved or stored in the pooling user's storage.

Description

This command instructs the system to place a user's program or data filename in a *Library or the **Library so that the program or data file may be shared by other users. The name of the program or data file is placed in the library identified by the asterisks preceding the filename. A program or data file whose name is in the *Library may be used by all users with the same first four characters of the user number. These user numbers make up a sub group, and the library serves, for example, as a department library, available only on special request. A program or data file whose name is in the **Library may be used by all users of the system, provided that the user group which pooled the program or data file is active. A user group comprises all users whose user numbers are within a specified range established by the installation. For example, all users having user numbers within the range AAA000-MZZ999 might constitute one user group; all users having user numbers within the range NAA000-ZZZ999 might constitute another.

A shared data file can be opened for input by all users of the library in which the file is pooled. It can be opened for output by only the user who pooled it. Two types of operations are not permitted: (1) opening a file for output when it has previously been opened for input by other system users, or (2) opening a file for input when it has previously been opened for output by the original pooling user. In either of these cases, a message is printed, notifying the user that he cannot have immediate access to the desired file.

If the addressed library already contains the specified program or data filename, the POOL command is rejected. The system responds:

```
FILENAME PREVIOUSLY USED
```

If, instead of READY, the system responds:

```
FILE NAME--
```

the user should type * or ** and the program or data filename.

Examples

```
POOL *PROG1
READY

POO **ACCTPRG
READY
```

PROTECT (see ALLOW)

General Form

PROTECT program or data filename

Valid Entries

The filename must be the name of a program or data file in the user's library.

Description

This command is used to provide protection to a program or data file whose name is in a *Library or the **Library. It is issued by the user who pooled the program or data file. A protected program is identified as RUN ONLY. It cannot be listed, saved, or stored by other users of the library. A protected data file may not be opened by other users of the library; such protection is particularly desirable prior to, or until verifying the successful completion of, a required update process.

If the PROTECT command is given with the filename omitted, the system prints:

```
FILE NAME--
```

The user should type the program or data filename.

Note that the PROTECT command may be used to provide protection to a ***Library program or data file, only by central computer installation personnel signed on at the command console.

Examples

```
PRO NEWPROG
READY

PROTECT PROG1
READY
```

PULL (see PURGE, POOL)

General Form

PULL *filename

PULL **filename

Valid Entries

The filename must be a valid program or data filename.

Description

This command instructs the system to remove the specified program or data filename from the shared library identified by asterisks. The command is rejected if the user attempting to pull a program or data file is not the user who pooled it. The system prints:

ILLEGAL USER NUMBER

If the PULL command is given with the filename omitted, the system prints:

FILE NAME--

The user should type * or ** and the program or data filename.

Examples

```
PULL **PROG2
READY

PUL *PROG4
READY
```

attaches the OS reader program to read the appropriate data set. For some reason, the OS reader program cannot continue processing and it terminates abnormally. This message is followed by a list of the names of all the jobs in the data set being read. The list is followed by message DIBRD015, which instructs the operator to restart the reader.

DIBRD013 UNABLE TO READ JOBNAMES,DIBRDRx TERMINATED

Execution of the COBI reader program either has been initiated with the parameter $$$$ (to display job names) or was initiated normally but the OS reader terminated abnormally. There are some jobs in the input data set, but the job names cannot be read. Generally, an OS or hardware-related problem has occurred. The COBI reader program terminates. In this message, the x is replaced by either A or B.

DIBRD014 UNABLE TO COMPLETE JOBNAMES

Execution of the COBI reader program either has been initiated with the parameter $$$$ (to display job names) or was initiated normally but the OS reader terminated abnormally. One or more jobs were read, but the reader could not finish reading the names of the other jobs in the data set. Generally, an OS or hardware-related problem has occurred. The COBI reader program terminates.

DIBRD015 COMPARE Q AND JOBNAME LIST, RESTART DIBRDRx,,,(NEXT JOBNAME)

The message follows the list of job names printed after message DIBRD012. The operator should compare the names of the jobs in the job queue against the list to determine the next COBI job to be processed. He then should start the specified reader at the next job. In this message, the x is replaced by either A or B.

DIBRD016 DID NOT LOCATE FILE, RETRY

For some reason, the COBI reader could not read the first record of an input data set. The operator must reissue the previous start command, specifying the appropriate reader (DIBRDRA or DIBRDRB). If this message appears again, it indicates a possible hardware error and the installation system programmer should be notified.

DIBRD017 FILE PREVIOUSLY READ. REPLY 'CANCEL' OR 'CONT'.

The COBI reader detects that the input data set has already been read. The operator may either cancel reader operation or continue processing by rereading the data set. If he decides to continue, message DIBRD018 is issued.

DIBRD018 REPLY 'YES' TO INCLUDE WTO MESSAGES; ELSE 'NO'.

This message appears after the operator has replied 'CONT' to message DIBRD017. The replies have the following effect:

1. A reply of 'YES' causes the jobs to be reread and all WTO messages associated with those jobs are reissued.

2. A reply of 'NO' causes the jobs to be reread but the WTO messages associated with the jobs are ignored.

DIBRD019 EOF PRIOR TO LAST JOBNAME

This message is issued to inform the OS console operator that the DIBRDR job name display was terminated early due to reading an unexpected end of file. In this case, it is possible that more jobs than are identified by the DIBRDR job name display will be read from the SYSIN file. The system programmer should be warned of a potential programming error.

DIBWT041 COBI CLASS NOT IN PARM OF EXEC CARD

The CBCLASS parameter was omitted from the EXEC statement in the DIBWTR procedure. This parameter specifies the COBI output class; this class is intercepted by DIBWTR for processing. The procedure must be corrected before DIBWTR can be executed.

DIBWT042 OS CLASS NOT IN PARM OF EXEC CARD

The OSCLASS parameter was omitted from the EXEC statement in the DIBWTR procedure. This parameter specifies the OS/360 output class to which the COBI class is transferred after processing by DIBWTR; this OS/360 class is printed by an output writer assigned to the class. The procedure must be corrected before DIBWTR can be executed.

DIBWT043 DISK ERROR READING VOLID TABLE

A permanent I/O error occurred while trying to read the volume identification table from the COBI JCL data set. This condition could indicate a hardware problem; contact the installation system programmer.

DIBWT044 VOLID TABLE IDENTIFIER MISSING

The volume identification table was read successfully from the COBI JCL data set; however, the expected identifier CBVD was not found in the first record. Contact the installation system programmer.

DIBWT045 DISK ERROR READING MASTER QCR CONTROL RECORD

A permanent I/O error occurred while reading the master queue control record from SYS1.SYSJOBQE. Contact the installation system programmer.

DIBWT046 DISK ERROR READING COBI CLASS QCR RECORD

A permanent I/O error occurred while reading the queue control record for the COBI output class. Contact the installation system programmer.

DIBWT047 DISK ERROR READING LOGICAL TRACK HEADER

A permanent I/O error occurred while reading a logical track header from the COBI output class. Contact the installation system programmer.

DIBWT048 DISK ERROR READING LOGICAL TRACK

A permanent I/O error occurred while reading a logical track from the COBI output class. Contact the installation system programmer.

DIBWT049 **** RESET 1ST JOB IN COBI SYSOUT CLASS TO OS CLASS

For some reason, DIBWTR cannot process the first job in the COBI output class. The operator should display the names of the jobs in the queue for the class and reset the first job to the OS/360 output class.

DIBWT050 TIME INTERVAL IN PARM NOT VALID

Either the time interval was not specified in the ITIME parameter for DIBWTR or, if present, the interval is either zero, or greater than 9999, or contains other than numeric characters. Correct the parameter field and restart the job.

DIBWT051 DISK ERROR READING COBI INDEX RECORD

A permanent I/O error occurred while reading the COBI index record. Contact the system installation programmer.

DIBWT052 NO MORE COBI JCL SPACE AVAILABLE

The COBI writer has attempted to write the JCL for a job in the COBI JCL data set but there is no more space available in the data set. The job is reassigned to the OS message class for processing by an output writer.

DIBWT053 DISK ERROR SEARCHING FOR AVAILABLE JCL SPACE

A permanent I/O error occurred while the COBI writer was searching for space in which to write the JCL for a COBI job. The job is reassigned to the OS message class for processing by an output writer. This error may indicate a hardware problem, and the installation system programmer should be contacted.

DIBWT054 FAILURE TO OPEN DIBWTR DATA SETS

All or one of the following data sets cannot be opened: CBJCL, CBNDX, or SYS1.SYSJOBQE. Since these data sets are defined in the DIBWTR cataloged procedure, the installation system programmer should be contacted.

DIBWT055 DISK ERROR WRITING COBI INDEX RECORD

A permanent I/O error occurred while writing the COBI index record. The job is reassigned to the OS message class for processing by an output writer. This error may indicate a hardware problem, and the installation system programmer should be contacted.

DIBWT056 VOL SER JCL MESSAGE NOT FOUND FOR DSN=DIB...

Ordinarily, one JCL allocation message appears for each COBI scannable data set. This allocation message indicates the volume serial number of the volume on which the data set was written. The preceding error message indicates that the allocation message did not appear, which indicates an error in the OS job queue. The installation system programmer should be contacted.

OS/360 System Console COBI
Messages and Explanations

DIBWT057 DISK ERROR WRITING COBI JCL RECORD

A permanent I/O error occurred while writing the COBI JCL record for a job. The job is reassigned to the OS message class for processing. This error may indicate a hardware problem, and the installation system programmer should be contacted.

DIBWT058 UNABLE TO FIND DIBWTR CSCB

The DIBWTR task, started at the OS system operator's console, cannot continue processing because the command scheduling control block (CSCB) could not be found. This makes it impossible to terminate the DIBWTR with the STOP command. Contact the installation system programmer.

DIBWT059 COBI JCL DATA SET IS ENQUEUED. CANNOT CONTINUE

When CALL-OS is initialized, an OS ENQ is made using the name 'DIBWTR '. When the DIBWTR is initialized, the same OS ENQ is made. This logic inhibits the simultaneous execution of CALL-OS with the DIBWTR or another CALL-OS, or of two DIBWTR's.

DIBWTR ABEND CODES

When COBI is used, the COBI writer (DIBWTR) is executed when CALL-OS is no longer in operation. During its execution, DIBWTR may issue ABEND codes when it can no longer continue. OS/360 issues the final termination message in the following format:

IEF450I DIBWTR ABEND Uddd

where

ddd is the ABEND code issued by DIBWTR

The message itself is explained in greater detail in the publication IBM System/360 Operating System: Messages and Codes. The possible ABEND codes are given in numerical order along with an explanation.

OS/360 System Console DIBWTR ABENDs

Code	Explanation
001	An error has been detected in TTR translation.
003	One of the data sets required for DIBWTR operation cannot be opened; these are the COBI index, JCL, and OS/360 job quque data sets.
321	An incorrectly compressed job queue record has been detected.

RENUMBER

General Form

```
RENUMBER [new-line-number][, old-line-number][, increment]
```

Valid Entries

New-line-number is the line number to be assigned to the first line to be renumbered. Old-line-number is the line number currently assigned to the first line to be renumbered. The increment must be a positive integer not greater than 99999.

Description

This command is used to renumber program lines. The new line number is assigned to the line previously identified by the old line number. All subsequent lines are renumbered using the increment specified. If the old line number does not exist in the current program but is not higher than all existing line numbers, the next higher line number is the first line number to be changed. If the old line number is higher than all existing line numbers, a message is printed at the user's terminal and no change is made to the program. The new line number must be higher than any line numbers prior to the selected renumbering point; that is, renumbering cannot cause any lines not included in the renumbering operation to be overlapped.

One, two, or all of the parameters may be omitted. Defaults of 100, 0, and 10 are assumed for the new line number, old line number, and increment, respectively. A comma must be used to indicate an omitted parameter if any specified parameters follow.

When a CALL-OS BASIC program is renumbered, the text of source-program statements containing line-number references for branching or control is modified to use newly assigned line numbers. When a CALL-OS FORTRAN or CALL-OS PL/I program is renumbered, the text of source-program statements is not changed.

If the program in the work area is currently protected (flagged as RUN ONLY), the RENUMBER command is rejected. The following message is printed at the terminal.

```
PROTECTED PROGRAM
```

A program written in CALL-OS BASIC may contain statements with line-number references. The text of such statements is modified to reflect any new line numbers created during the renumber operation. A line number is reserved preceding the last statement of the renumbered program as a reference point for any statements that originally referenced nonexistent lines.

Note: If original BASIC statements contain invalid BASIC syntax, causing line-number references to be unclear, unpredictable results may occur.

Examples

```
REN 200,10,20
READY

RENUMBER 60,,5
READY
```

General Form (single lines)

```
REPLACE line-number, line-number,..., 'string1'[,'string2']
```

General Form (groups of lines)

```
REPLACE line-number THRU line-number,..., 'string1'[,'string2']
```

General Form (all lines in a program)

```
REPLACE ALL, 'string1'[,'string2']
```

Valid Entries

A string of characters may be any characters allowable in System/360. Line numbers must be positive integers in ascending numeric sequence. When a group of consecutive lines is specified, neither the low nor high line number has to exist (although it may) in the current program. The replace operation is performed on all lines within a specified range, inclusive. An entry of 99999 as a line number in a THRU specification is interpreted as a reference to the last line of the program. Up to nine ranges (single lines, groups of lines, or a combination of both) may be specified in one REPLACE command. The ALL parameter entry may not be used in conjunction with any other range specification, but either ALL or one or more line numbers must be specified. The abbreviated forms A for ALL and T for THRU may be used.

Description

This command causes the first character string to be replaced by the second character string at each occurrence in the specified range or ranges. If the second string is omitted or specified by two adjacent single quotes, the first string is deleted at each occurrence in the specified range or ranges. Specified strings must be enclosed in single quotes. A single quote in either string must be represented by two single quotes. Succeeding parameter entries are separated by commas. If a single line number that does not exist is specified as a parameter entry, the entry has no effect on the current program. A nonexistent line number specified in a THRU specification serves as a boundary indicator, delimiting the range of lines on which the replace operation is performed.

When all replacements or deletions have been made, the following message is printed at the terminal.

```
REPLACED nnnnn
```

where nnnnn is the number of times that the specified character string has been replaced or deleted.

All operations are performed on the program in the work area unless that program is currently protected (flagged as RUN ONLY). If the program is protected, the REPLACE command is rejected. The following message is printed at the terminal.

```
PROTECTED PROGRAM
```

If the new length of a line exceeds the maximum length allowable (237 characters), an error message is generated. The replace operation is terminated, and no lines of the program in the user work area are changed.

Notes: 1. Since the specified strings are treated as groups of characters, the user must ensure that the strings are specified uniquely. Otherwise, unanticipated deletions or replacements may occur. For example, a single letter, say A, specified for replacement, would be replaced every time it was encountered, even embedded within a source-statement word.

2. A character to be replaced can be part of only one string. For example, a replace specification of ' 0 ', if matched against ' 0 0 ', will delete the first three characters of the string, but will not delete the final '0 '.

Examples

```
REPLACE 100 THRU 400,'THIS','THIS IS IT'

REPLACED 0
READY

REP 200,'ATT'

REPLACED 8
READY

REP 200,'ATT',''

REPLACED 8
READY

REPLACE ALL,'FILE1','FILE2'

REPLACED 2
READY
```

RUN (work area)

General Form

```
RUN
```

Description

This command instructs the system to compile and execute the source program in the user work area. The program should have been entered from the terminal or loaded from one of the libraries. If no source program is present in the user work area, but the command immediately preceding this command caused the running of an object program, that object program is rerun.

If no source program resides in the user work area and an object program has not just been run, the system prints:

```
NO PROGRAM PRESENT
```

When serious errors are detected in program statements during compilation, error messages are printed and the program is not executed. Statements may then be corrected and another RUN request entered.

Example

```
RUN

PROG1        16:18    10/12/71    TUESDAY

THIS IS THE PROGRAM OUTPUT.
TIME    2 SECS.
```

RUN (user's library)

General Form

```
RUN program-name
```

Valid Entries

The named program must be a valid source or object program name in the user's library; it cannot be the name of a data file.

Description

This command instructs the system to clear the user work area and locate the program in the user's library. If the program is in source form, it is loaded, compiled, and executed. Programs in object form are executed directly from the library. Either source or object programs can be rerun immediately without reentering the program name.

When the named program is not contained in the user's library, the system prints:

```
program-name NOT FOUND
```

Example

```
RUN PROG2

PROG2        16:18    10/12/71    TUESDAY

THIS IS THE PROGRAM OUTPUT.
TIME    2 SECS.
```

RUN (shared libraries)

General Form

```
RUN *program-name

RUN **program-name

RUN ***program-name
```

Valid Entries

The named program must be a valid source or object program name in the library identified by asterisks.

Description

This command instructs the system to clear the user work area and locate the program name in the designated library. If the program is in source form, it is loaded, compiled, and executed. Programs in object form are executed directly from the library. Either source or object programs can be rerun immediately without reentering the program name.

When the specified program name is not contained in the designated library, the system prints:

```
program-name NOT POOLED
```

When the specified program name appears in the designated library, but no corresponding program exists in the pooling user's library, the system prints:

```
program-name NOT FOUND
```

Example

```
RUN *PROG1

PROG1        15:27   10/12/71   TUESDAY

THIS IS THE PROGRAM OUTPUT.
TIME   2 SECS.
```

SAVE

General Form

```
SAVE [program-name]
```

Description

This command instructs the system to write a source program from the work area into user storage. A program previously saved with the same name is destroyed unless it has been locked, in which case the current SAVE command is rejected. The program-name parameter is not required if the program in the work area already has a program name. A request to save a program under a name currently assigned to an object program or data file in the user's library is rejected.

If the program in the work area does not have a name associated with it, and none is specified in the SAVE command, the system prints:

```
FILE NAME--
```

After the program name is typed, the system saves both the name and the program.

If the program in the work area has a name associated with it, but a different name is specified in the SAVE command, the specified name replaces the current name and is saved with the program.

Examples

```
SAVE
READY

SAV
READY
```

SECURE (see RELEASE)

General Form

SECURE filename

Valid Entries

The filename must be the name of a program or data file in the user's library.

Description

This command causes the SECURE attribute to be set for the program or data file named in the SECURE command. This prevents any user other than the creator of a file from copying the file into another library by means of the INSERT/REPLACE function of the CALL-OS Data Base Utility.

For all program and data files saved or stored through the terminal or entered into the CALL-OS system via a CALL-OS utility, the SECURE attribute is set automatically. To allow another utility user to copy the program or data file into his library, a user may set the RELEASE attribute for the file by issuing a RELEASE command. When the INSERT/REPLACE function of the CALL-OS Data Base Utility accesses the file, it resets the SECURE attribute automatically. Alternatively, the user who created the file may reset the attribute by issuing a SECURE command.

Examples

```
SEC LASTPROG
READY

SECURE PROG1
READY
```

STATUS

General Form

STATUS

Description

This command instructs the system to print information about the program currently in the work area. The printed information includes terminal number, user number, program name (if any), program length, and language. The character and line count values are incremented only after new source lines have been sorted together with existing source-program lines. Therefore, the program length is accurate only after a command which causes a sort to take place, such as RUN, LIST, LIST-N, LIST-T, RENUMBER, SAVE, STORE, or WEAVE. The amounts of terminal and processor time used for the current session are also printed.

Example

```
STA
TERMINAL NO..14
USER NO...IBM030
PGM NAME..EXAMPLE
PGM LENGTH..4823 CHARS.,191 LINES
LANGUAGE..BASIC
TERMINAL TIME..2 MIN
PROCESSOR TIME..0 SEC
```

General Form (to compile and store program)

```
STORE [program-name]
```

General Form (to determine disk storage units required for object program)

```
STORE ?
```

Description

This command, in the first form shown above, instructs the system to compile the program currently in the user work area and store the resulting object program in the user's library. If a program name is specified, the program is stored under that name. If no program name is specified, the program is stored under a name consisting of the slash character and up to seven characters of the current program name (for example, /PROG).

If the current program name already contains eight characters, the store operation is not performed. The following message is printed at the terminal:

```
SOURCE NAME HAS 8 CHARACTERS
```

A program previously stored with the name specified for a current STORE command is destroyed unless the program has been locked, in which case the STORE command is rejected. A request to store a program under a name currently assigned to a source program or data file in the user's library is also rejected.

The STORE command has no effect on the saved version of a source program. If the user has modified the source program since loading it into his work area, he must also save the program if he wishes to include those modifications in the source program version in his library.

Since object program compatibility from one version of CALL-OS to the next is not guaranteed, it is recommended that the user retain a copy of the source program version of any program stored in his library.

If source-statement errors that would prevent execution of the program under a RUN command are detected, diagnostic messages are printed at the terminal and the object program is not stored.

When the user enters the command word STORE followed by a question mark, the system prints the number of disk storage units (each 3440 bytes in length) that would be required to store the program currently in the work area into the user's library in object form. This form of the STORE command does not cause the program to be stored. The format of the system-provided message is:

```
xx STORAGE UNITS REQUIRED TO STORE program-name
```

where xx is the number of disk storage units that would be required to store the specified program.

The library space required to store object programs is significantly greater than that required for source programs. Therefore, the STORE

command should be used only for frequently used programs having long compile times. When an object program is stored, the system prints:

```
OBJ PROG STORED - xx STORAGE UNITS REQUIRED
```

where xx is the number of disk storage units required to store the program.

Examples

```
LOAD PROGX
READY

STORE

/PROGX      14:20   10/12/71   TUESDAY

OBJ PROG STORED - 14 STORAGE UNITS REQUIRED

TIME   6 SECS.


STO OBJPROGY

OBJPROGY    14:27   10/12/71   TUESDAY

OBJ PROG STORED - 10 STORAGE UNITS REQUIRED

TIME   2 SECS.
```

TAPE (see TAPE ALL)

General Form

```
TAPE
```

Description

This command conditions the system to allow the entry of source-program statements from paper tape (Teletype only). It inhibits output line feeds after carrier returns, ignores edited lines which contain only a carrier return, and prevents messages from being printed out during normal statement entry.

If this command is entered from a Teletype, the system response is:

```
READY
```

If entered from any other terminal type, the system response is:

```
WHAT?
```

Only the OFF, KEY, and TAPE ALL commands are accepted by the system when it is operating in TAPE mode. All other commands and unnumbered statements are ignored.

Note that if any character is lost during paper tape input, the system responds to a subsequent KEY command with the following error message:

```
LOST INPUT LINES, LIST YOUR PROGRAM
```

See also the discussion of paper tape input/output under "Paper Tape Operations".

Examples

```
TAPE
READY

TAP
READY
```

SYSTEM MESSAGES ISSUED BY THE EXECUTIVE

Messages are issued by the executive to communicate system status by identifying problems as they occur. Operator invervention will be noted when necessary.

Communications Console System Messages and Explanations	Initiating Module

AT tt:tt, LINE nn — M#ERMSG
COMMUNICATIONS CONSOLE IS ACTIVE.
SYSTEM ENABLED AT hh:mm.

This message is transmitted to the communications console whenever it is brought up and connected to the system either for the first time or from any on-hook condition.

- tt:tt — is the time of day when the communications console is connected to the system.
- nn — is the line number of the communications console.
- hh:mm — is the time of day when one or more command consoles are enabled and the system is initialized.

CANNOT LOAD COMPILER cccccccc — S#LDSP

An irrecoverable disk error occurred while reading compiler cccccccc. May be due to system build procedure error.

CENTRAL CYLINDER DISK ERROR ON SOURCE READ — M#RDSO

An irrecoverable disk error occurred while reading the user's work area.

CENTRAL CYLINDER DISK ERROR ON SOURCE WRITE — M#WRSO

An irrecoverable disk error occurred while writing the user's current sorted source program to his work area.

CENTRAL CYLINDER DISK ERROR ON SWAP READ — M#ISWP

An irrecoverable disk I/O error was detected while trying to read the user's swap area.

CENTRAL CYLINDER DISK ERROR ON SWAP WRITE — M#OSWP

An irrecoverable disk error has occurred while trying to swap out a user program.

CHAN OR 2702/3 BUSY FOR 20 SIOS. USER DISABLED — I#NEAPP

A condition code from a start I/O to a terminal I/O indicates one of the following:

1. The terminal is not available.
2. The 2702/3 TCU shows a busy status.

3. A busy channel status is given in the 20 start I/O attempts.

The terminal is disabled.

COMMAND AND COMMUNICATIONS CONSOLES ENABLED
YEAR = xxxx, DAY = yyy — M#CCDI

CALL-OS has been activated and both command and communications consoles are enabled by the system. If a communications console is unavailable at this time, the OS/360 system operator's console will receive this message, and messages addressed to the communications console will continue to be sent to the OS/360 system operator's console unless a communications console is made available.

xxxx = year (for example, 1970)
yyy = day (for example, 365 for December 31st)

DISK ERROR READING COBI CLASS QCR RECORD — M#JCL

A permanent I/O error has occurred while reading the queue control record (QCR) for the COBI job class. The installation system programmer should be contacted.

DISK ERROR READING COBI INDEX RECORD — M#JCL

A permanent I/O error has occurred while reading the COBI index record. The installation system programmer should be contacted.

DISK ERROR READING LOGICAL TRACK — M#JCL

A permanent I/O error has occurred while reading a logical track from the COBI output class. The installation system programmer should be contacted.

DISK ERROR READING LOGICAL TRACK HEADER — M#JCL

A permanent I/O error has occurred while reading a logical track header from the COBI output class. The installation system programmer should be contacted.

DISK ERROR WRITING COBI INDEX RECORD — M#JCL

A permanent I/O error has occurred while writing the COBI index record. The job is reassigned to the OS message class for processing by an output writer. This error may indicate a hardware problem, and the installation system programmer should be contacted.

DISK ERROR WRITING COBI JCL RECORD — M#JCL

A permanent I/O error has occurred while writing the COBI JCL record for a job. The job is reassigned to the OS message class for processing. This error may indicate a hardware problem, and the installation system programmer should be contacted.

Communications Console System Messages and Explanations	Initiating Module
DISK READ ERROR FOR SAVED PROGRAM	M#LDRD
An irrecoverable disk I/O error was detected while trying to read the user's program from save storage.	
EQUIVALENCY CLASS DISK ERROR AT SIGN-ON TIME	M#LOG
An irrecoverable disk error has occurred while trying to read the equivalency class file.	
EQUIVALENCY CLASS DISK ERROR DURING SIGN-OFF	M#LOG
An irrecoverable disk error has occurred while trying to read or write the user's equivalency class file.	
ERROR MESSAGE QUEUE EXHAUSTED. MESSAGE(S) LOST	S#QEMSG
The number of messages being routed to the communications console has exceeded the capacity of the queue.	
ERROR RATIO EXCEEDED	I#NEAPP
Ten errors have been detected in the last 500 I/O operations on this line. The user is disconnected but the line is not taken out of service. An error recording has been made on SYS1.LOGREC. The operator may want to take the line out of service with a *DISABLE command if the error persists.	
ERROR THRESHOLD EXCEEDED	I#NEAPP
Excessive consecutive error conditions have been detected on this line. An error recording has been made on SYS1.LOGREC. The user is disconnected but the line is not taken out of service. The operator may want to take the line out of service with a *DISABLE command if the error persists.	
xxxx HAS ATTEMPTED MOD CALL WITHOUT RELEASING OVLY BUFFER	M#OVLY
The module represented by xxxx has failed to relinquish the overlay buffer. This is a CALL-OS system problem.	
HEX xx, ERR RECORDED, LINE LEFT IN SERVICE, BUSY TEL LINE IF ERROR PERSISTS	I#NEAPP
An error was recorded for the logical line (nnn, given in message header) associated with the indicated device address (xx). The current user has been successfully dropped by the system and the specific error has been recorded in SYS1.LOGREC. The statistical data recorded for the line may contain valuable information, so this information should be consulted on the error record printout. The error record is identified by time stamp	

(tt:tt, given in message header) and device address (xx).

This message usually precedes one of the following two messages:

ERROR RATIO EXCEEDED
ERROR THRESHOLD EXCEEDED

HEX xx, LINE OUT OF SERVICE, ERROR RECORDED, ***BUSY OUT TELEPHONE*** — I#NEAPP

The logical line (nnn, given in message header) associated with the indicated device address (xx) is taken out of service due to a hardware or software system problem; an error recording has been made on SYS1.LOGREC. The operator must either busy-out the telephone associated with this line or reenable the line with the *ENABLE command. To reenable, he uses the logical line number specified in the message header.

NO MORE COBI JCL SPACE AVAILABLE — M#JCL

The COBI writer has attempted to write the JCL for a job in the COBI JCL data set, but there is no more space available in the data set. The job is reassigned to the OS message class for processing by an output writer.

NO TRANSLATION TABLE. DISABLED. — I#NEAPP

This message is printed on the communications console when a user terminal or command console receives an INVALID TYPE DISABLED message due to lack of translation table. A header line precedes this message giving the logical line number of the affected terminal.

A 2741 terminal on the indicated line is disabled because the translation table for this terminal type is not core resident.

The major causes for this are the following:

1. The 2741 correspondence terminal is not specified in any of the terminal DD cards in the system startup deck.

2. The correspondence translation table is not specified in the RESMODS entry.

SYSTEM ERROR, *** BUSY OUT DATA SET ***
ERROR IN MODULE M#CCDI, CODE 2 — M#CCDI

When the system is in the process of disabling or aborting a terminal, an *ENABLE command is received. The above error message is output and the terminal is disabled.

Communications Console System Messages and Explanations	Initiating Module

TYPE LOG OR RETURN KEY/(garbled data) — I#NEAPP
(garbled data)/TYPE LOG OR RETURN KEY

One of the above two messages is printed if the 2741 is used as the communications console and its operator fails, in the course of the sign-on procedure, to input either:

1. LOGON/LOG and RETURN, or

2. RETURN for a default 2741 type.

If the terminal is a 2741 correspondence, the first message is printed; if it is 2741 EBCD, the second message is printed.

Unlike a user terminal and command console, the communications console is expected to transmit either LOGON/LOG or RETURN to the system before it enters the RECEIVE mode.

UNABLE TO READ OVERLAY MODULE -- L#STAT=xx — M#OVLY

A disk error has occurred while reading a module in the xx sequence; see Appendix A.

USER aaannn x-RD, x-WR, x-IR, x-BUS, x-EQ, x-OR, x-LD, x-TO — I#NEAPP

This message identifies the user number (aaannn) and displays the IOS error counter contents for the line, where x is the number of times one of the following conditions occurred:

RD	Read error
WR	Write error
IR	Intervention required
BUS	Bus out check
EQ	Equipment check
OR	Overrun
LD	Lost data
TO	Time out

This information appears immediately after any of the following messages:

ERROR RATIO EXCEEDED
ERROR THRESHOLD EXCEEDED
HEX xx, ERROR RECORDED, LINE LEFT IN SERVICE ...
HEX xx, LINE OUT OF SERVICE ...

VOL SER JCL MESSAGE NOT FOUND FOR DSN=DIB... — M#JCL

Ordinarily, one JCL allocation message appears for each COBI scannable data set. This allocation message indicates the volume serial number of the volume on which the data set was written. The preceding error message indicates that the allocation message did not appear, signalling an error in the OS job queue. The installation system programmer should be contacted.

Communications Console System Messages and Explanations	Initiating Module
****** RESET 1ST JOB IN COBI SYSOUT CLASS TO OS CLASS**	M#JCL

DIBWTR cannot process the first job in the COBI output class. The operator should display the names of the jobs in the job queue for the class and reset the first job to the OS output class.

WIDTH

General Form

WIDTH number-of-print-positions

Valid Entries

The entry for number of print positions may be any number from 18 through 255.

Description

This command sets the width of a print line to be used for printing information at the terminal. It is used only if the line width desired for listed or runtime output is other than 72 print positions, which is the default assumed by the CALL-OS system at sign-on time. A line width specified by the WIDTH command remains in effect throughout the current session of CALL-OS unless another WIDTH command is given. The WIDTH attribute is effective for the LIST and RUN commands. It also governs printing of output in response to a SCAN command when the CALL-OS Batch Interface (COBI) option is utilized.

The system automatically eliminates printing of all blank characters between the last nonblank character on a line and the final print position of the line (as specified by the WIDTH command or by default). If output lines contain many blank characters between columns, for example, the user can set the width of the print line at the last print position preceding the second column. If he does so, a carrier return is made immediately following the last nonblank character preceding the specified width; intervening blanks are not printed; and the characters beyond the width specification are printed at the beginning of the next line. Since printing of intervening blanks (trailing blanks, because of the width specification) is eliminated, the total printing time is reduced.

If the user has entered the WIDTH command and the parameter supplied is nonnumeric, contains more than three digits, or is less than 18 or greater than 255, the system prints:

INVALID PAGE WIDTH

If the WIDTH command is given without the width parameter, the system prints:

PAGE WIDTH--

The user should type the desired page width.

Note: Although a line width as long as 255 characters can be specified in the WIDTH command, it is not possible to print out lines of this length on a 2741 Communications Terminal or Teletype, Model 33 or 35. The actual, effective maximum line widths for these terminals are shown below:

2741	130 positions with 10 positions per inch
Teletype	72 positions

If, for example, a current page width greater than 72 is in effect, and output is printed on the Teletype, all characters in

excess of 72 are overprinted in the last position of the 72-position line.

Examples

```
WIDTH 130
READY

WID 60
READY
```

COBI TERMINAL COMMANDS AND CONTROL STATEMENTS

The CALL-OS Batch Interface (COBI) option permits a CALL-OS terminal user to prepare jobs for OS/360 batch processing and submit those jobs to the OS/360 batch-processing environment. When the user submits a job, he can specify that JCL (job control language statements and system messages), SYSOUT data sets defined in cataloged procedures, and user-defined SYSOUT data sets of the job be saved for subsequent scanning (printing) at the terminal. (See "SUBMIT".)

A job number of the form #nnnnn is assigned to most submitted jobs during SUBMIT processing. This job number must be specified in subsequent terminal commands whereby the user inquires about the status of the job or its SYSOUT data sets, cancels the job before it completes OS/360 execution, scans output data sets, or scratches all data sets or specific data sets or JCL of the job.

The functional capabilities available to COBI users are explained in detail below. First, the formats of CALL-OS terminal commands that may be issued by COBI users are presented and explained. Then, the formats of COBI control statements which the user may include in his input job stream are described. The interrelationships between COBI and OS/360 job control language (JCL) statements are clarified. Typical job sequences using the COBI option are given at the end of the section.

Each terminal command word may be entered in full, or only the first three characters of the command word may be used. Documentation conventions used in this manual to describe command and statement formats are explained at the beginning of the preceding section, "CALL-OS Terminal Command Language". For an overview of the COBI option, see "CALL-OS Batch Interface (COBI) Option". Additional information pertaining to JCL statements is available in the OS/360 publication IBM System/360 Operating System Job Control Language.

COBI TERMINAL COMMANDS

The following CALL-OS terminal commands, in addition to those described in the preceding section, are available to COBI users: CANCEL, DSSTATUS, JOBSTATUS, NOTIFY, SCAN, SCRATCH, and SUBMIT. The formats and uses of these commands are described below.

CANCEL

General Form

CANCEL job-number[,SCRATCH]

Valid Entries

A specified job number must be an entry of the form #nnnnn, where #nnnnn is the internal job number assigned to the job (see "COBI-Assigned Job Numbers and Job Names").

Description

This command is used to cancel a job submitted via CALL-OS for OS/360 batch processing. The CANCEL command is accepted as valid if the current user signed on under the same user number as the submitter of the job. Otherwise, the CANCEL command is rejected. Omission of the SCRATCH parameter indicates that the user would like to cancel the job he has submitted but would like to retain any JCL and SYSOUT data sets already created. The user should include the SCRATCH parameter when he wants to cancel the job and does not want to save any JCL or SYSOUT data sets.

Since it is possible that execution of the job is underway or completed before the user issues the CANCEL command, use of the SCRATCH parameter is usually desirable. If this parameter is specified, the COBI index record and any JCL and SYSOUT data sets created for the job are scratched by the system. If this parameter is not specified, and execution of the job is underway or completed prior to user issuance of the CANCEL command, the user is responsible for scratching any unwanted data sets.

If the job has not been sent to OS/360 batch processing before the CANCEL command is entered, the system responds:

JOB CANCELLED

If the job has been sent to OS/360 batch processing but has not been completed, OS/360 is requested to cancel the job. The system prints the following message.

JOB CANCELLATION REQUEST INITIATED

In this case, if the SCRATCH parameter is not specified in the CANCEL command, the user should check his job the next time he signs on the system from a terminal. As noted above, he is responsible for scratching any unwanted data sets.

If the job has been sent to OS/360 batch processing and has already been completed, subsequent action depends on whether or not the SCRATCH parameter is specified in the CANCEL command. If it is, any JCL and SYSOUT data sets created for the job are scratched, the record for the

Communications Module	Console Code	User-Related Error Codes Explanation
M#SCAN	1	TTR conversion error.
	2	An I/O error has occurred while attempting to read a record of the user's data set being scanned.
	3	An I/O error has occurred while attempting to read the minor or major records of the variable table on the work/swap data set. Such an error can occur only during the scan of a variable data set.
M#SCR	1	Data set not dequeued. The user has requested that a data set be scratched. The data set has been closed and scratched, but an error was encountered while dequeing the data set.
	2	The volume identification in the COBI index record cannot be found in the volume identification table; actual volume identification cannot be made and the user's scratch request is denied.
	3	An I/O error has occurred while attempting to write either an updated COBI index record back on the COBI index data set or an updated JCL record back on the JCL data set.
	4	An I/O error has occurred while attempting to either read a catalog record or write an updated catalog record back on the user group data set.
M#SORT	1	The last character of the old source is not an EOF character.
	2	The last character of the current line is not a new line character and the old source is being sorted.
	3	The last character of the current line is not a new line character and the old source is not being sorted.
	4	The sort input and output areas overlap.
M#SSVC	1	Next compiler code not found in the language table. Language table or UTT is destroyed. Error is irrecoverable.
M#SUB	1	The last character of the current line is not a new-line character
M#TTIMR	1	A zero queue link was encountered. The link should have contained a pointer to the next UTT or a full word containing hexadecimal 'FFFFFFFF'.
	2	Accounting data was not processed for one time interval.

Communications Module	Console Code	User-Related Error Codes Explanation
M#WRSO	1	A program error detected when trying to write the user's current sorted source program to disk.
	2	M#WRSO was entered by a process that was not initiated by RUN, LIST, LOAD, SAVE or any edit command.
S#INDXQ	1	Exceeded table size, too many concurrent users.
	2	A request was received to dequeue a COBI index record that had not been enqueued.
	3	An exclusive dequeue request was issued for a UTT different from the original enqueuing UTT.
	4	A shared dequeue request was issued for an index record that was exclusively enqueued.
S#OUTR	1	An attempt was made to release the 256-byte buffer when it was not assigned.
S#RERDP	1	Either the link field of the first buffer assigned to the UTT was 0, or there was no beginning-of-line buffer address.
S#SRTR	1	An attempt was made to release the sort buffer when it was not assigned.
	2	The sort buffer was released by a UTT other than the one to which it was assigned.
	3	In the process of manipulating the sort buffer queue, a link was found that was zero.
S#SYSR	1	An attempt was made to release the system buffer when it was not assigned.
	2	The system buffer was released by a UTT other than the one to which it was assigned.
	3	In the process of manipulating the system buffer queue, a link was found that was zero.

General Form

JOBSTATUS [job-number][,job-number,...]

Valid Entries

A specified job number must be an entry of the form #nnnnn, where #nnnnn is the internal job number assigned to the job (see "COBI-Assigned Job Numbers and Job Names").

Description

This command is used to request the status of all jobs the user has submitted for OS/360 batch processing, of a specified job, or of specified jobs. If job numbers are specified in the JOBSTATUS command, the status of those jobs is printed. The jobs appear on the printout in the order in which they are specified. If no entry follows the command word JOBSTATUS, the status of all of the user's jobs is indicated.

The system prints the number and status of each job. There are four possible status conditions, indicated as follows:

Status	Meaning
JCL ERROR	A JCL error was detected when processing job-control statements.
CANCEL RQ	The job is in OS/360 batch processing, but a request to cancel the job has been issued.
COMPLETED	Execution of the job has been completed.
NOT DONE	The job has been submitted to COBI but execution of the job has not been completed.

One or more completion codes may be printed to indicate the cause of each job termination. These codes are issued by OS/360 as described in IBM System/360 Operating System Messages and Codes. User ABEND codes may also be printed and are preceded by the letter U.

Whether JCL of the job was requested for scanning, and the identifiers of any data sets specified in the options field of the SUBMIT command for the job, are also indicated on the printout. User-defined SYSOUT data set identifiers are preceded by U. A data set identifier of the form nPmm refers to a SYSOUT data set defined in a cataloged procedure. n is the procedure number, and mm is the number of the data set. (See "Scannable Data Sets".) Note that the response to a JOBSTATUS command does not indicate the status of any data sets; it only lists them as initially specified by the user in his SUBMIT command for the job. To obtain data set status information, the user must enter a DSSTATUS command.

Example

```
JOBSTATUS

JOB ID:  STATUS:     CODE:     DATA SET:

#49     COMPLETED       0C4   JCL,1P01,1P02
#52     CANCEL RQ       222   JCL,U001,U002
#63     COMPLETED OC5,U0777      1P01,U005
#145    JCL ERROR             JCL,U007,1P01
  .        .           .          .
  .        .           .          .
  .        .           .          .
```

General Forms

```
NOTIFY [{CALL,          }]   'message'
       [{OS,            }]
       [{nn,[nn,nn,...]}]
```

Valid Entries

Any of three forms of entry are valid, as indicated above. In the third form, nn is an OS/360 routing code within the range 1 through 16, inclusive, as described in IBM System/360 Operating System Supervisor and Data Management Macro Instructions. Such entries are meaningful only when the Multiple Console Support (MCS) feature is included at system generation. The message may contain any characters allowable in OS/360 write-to-operator messages.

Description

This command enables the CALL-OS terminal user to send a message to either the CALL-OS communications console or one or more OS/360 consoles. The CALL entry sends a message to the CALL-OS communications console. The OS entry sends a message to the OS/360 operator console (equivalent to routing code 2). Each nn entry identifies a specific OS/360 console. The CALL entry may not be used with any other destination entry. Multiple nn entries may be specified in one NOTIFY command. If both OS and one or more nn's are specified in a single command, the OS parameter entry is ignored. If no message destination is specified, CALL is assumed.

A message sent to the communications console must be less than or equal to 64 characters in length. A message sent to an OS/360 console must be less than or equal to 100 characters in length. If either limit is exceeded, excess characters are truncated. The message is terminated by either the rightmost single quote or a carrier return. Single quotes within the message will be treated like any other data characters.

Messages sent to the CALL-OS communications console appear as:

```
tt:tt LINE nnn C/R
userid: message
```

where tt:tt is the time of day, nnn is the user's line number, C/R is a carrier return, userid is the user number of the user sending the message, and message is the user-supplied message. If the communications console is not active when a message is sent to it, the first line is prefixed by DIBEX001. This causes the message to be routed to the OS/360 operator console.

Messages sent to one or more OS/360 consoles appear as:

```
DIBEX002  userid: message
```

where userid is the user number of the user sending the message and message is the user-supplied message.

A NOTIFY command containing a CALL operand can be issued by any CALL-OS terminal user. This form of the NOTIFY command can be used in a system that does not include the COBI option, or in a system in which COBI modules are available--irrespective of whether COBI has been

activated or deactivated (at system initialization time) for the current session of CALL-OS. If a NOTIFY command containing OS or one or more OS/360 routing codes is entered, but the system has not been built for COBI, the NOTIFY command is rejected. In this situation, messages can be sent to only the CALL-OS communications console.

Note: A user of a CALL-OS system that includes the COBI option can also communicate with the OS/360 computer operator by including comment statements in his input job stream. See "Comment Statement" later in this section for a discussion of how this statement can be used.

Examples

```
NOTIFY CALL,'MOUNT TAPE 2'
READY

NOT 2,6,12,'LAST REPORT'
READY
```

General Forms

```
SCAN job-number,data-set[,option]

SCAN job-number,JCL[,option]

SCAN DSN=xxx.yyy.zzz[,VOL=volid][,MEM=memname][,option]
```

Valid Entries

A specified job number must be an entry of the form #nnnnn, where #nnnnn is the internal job number assigned to the job (see "COBI-Assigned Job Numbers and Job Names"). In the first form shown above, data-set must be the identifier of a SYSOUT data set specified to be saved for scanning, by the user, when he issued the SUBMIT command. Likewise, JCL must have been specified in the SUBMIT command (or the job must have been terminated by a JCL error) if job control language statements and system messages are to be scanned. A user SYSOUT data set is referred to by an entry of the form Unnn, where nnn is the data set number. An entry of the form nPmm refers to a SYSOUT data set defined in a cataloged procedure. n is the procedure number, and mm is the number of the data set. If n is 1, it may be omitted; that is, Pmm is equivalent to 1Pmm. (See "Scannable Data Sets".)

In the third form shown above, xxx.yyy.zzz is the qualified data set name of an OS/360 data set. The data set name may contain one or more levels of qualification. Either (1) one of the qualifiers must be the user number of the user issuing the SCAN command, or (2) the leftmost qualifier must be one of two installation-specified high-level qualifier names. The user should contact the central computer installation to learn these qualifier names. OS/360 data sets that are password-protected to prevent reading cannot be scanned.

The volid entry is the volume identification to be used in locating a data set. If the data set is not an OS/360 cataloged data set, this parameter is required. If the data set name refers to a partitioned data set, the member name parameter (memname) should be specified to request scanning of a particular member of the partitioned data set.

In any of the forms shown above, the entire data set or member is printed unless an option entry is specified to indicate which logical records (lines) of the data set are to be printed, as explained below. All entries other than the option entry may be specified in any order. If an option is specified, it must be the last entry in the SCAN command.

Description

This command enables the user to have the content of a specific data set printed at the terminal. Data sets to be scanned (printed) must reside on a 2314 or 2319 disk storage unit and may be any of the following unkeyed record formats: variable unblocked (V), variable blocked (VB), fixed unblocked (F), fixed blocked (FB), or fixed blocked standard (FBS). The block size specified for a blocked file must be such that not more than 127 physical records reside on one track. Data sets having an undefined record format cannot be scanned. In addition, user-defined and procedure-defined SYSOUT data sets to be scanned must be sequentially organized; data sets scanned by specifying a DSN parameter entry may be either sequential or partitioned data sets. All carriage control characters will be ignored and all lines will be single spaced.

Communications Console Compiler-Related Message (Reasons Only) and Explanations	Initiating Module
SVC 3/4 ERROR	M#SSVC
User's communications region is unacceptable. In particular, FILENBR is outside the range 1-4, FILEPTR has a displacement value beyond this user's program area, or an attempt was made to read or write past the end of the file.	
SVC 5 ERROR	M#SSVC
User is returning too much core.	
SVC 7/8 ERROR	M#SSVC
The user has requested an exit from the arithmetic trap routine to an address beyond his limits. (PSW2SV is out of bounds.)	
SVC 11 ERROR	M#SSVC
Compiler generated a program that issued an SVC 11.	
SVC 16 ERROR	M#SSVC
Invalid address passed during a chaining operation.	
SVC 17 ERROR	M#SSVC
An SVC 17 has occurred when phase 1 of a compilation is not in process.	
SVC 21 ERROR	M#SSVC
User's communications region is unsuitable for performance of the open function. Any one of the following may be true: • Open request was received for a file previously opened but never closed. • Open bit set but neither as an input file nor as an output file. • Data file table extends beyond the user's program area. • No file found with open bit set. • The file table pointer table has no end-of-table marker.	

OS SYSTEM PRINTER UTILITY MESSAGES

During execution of the CALL-OS utilities, messages are printed at the OS system printer. This printer is the device defined by either a SYSPRINT or PRINTA DD statement when the utility is run. The messages are in alphabetical order either according to an identifier of the form DIBxxxxx or, if no identifier is present, according to the first word of message text.

An identifier indicates the utility program which issued the message, and in some cases, which module within the utility. The following identifiers are used:

DIBDBynn — Indicates that the message was issued by the data base utility, DIBCADBU. In these messages, the ynn is the message number and y identifies the module which issued the message as follows:

Value of y	Module(s) and Corresponding Function
0	DIBINIT, DIBINTER, DIBOPEN, DIBTERM, DIBSUB, DIBDISK, DIBGDSK, DIBMESG, DIBTTOT
1	DIBREORG - REORGANIZE function
2	DIBVALDT - VALIDATE function
3	DIBDELET - DELETE function
4	DIBWRITE - WRITE function
5	DIBTAPE - TAPE function
6	DIBINSRE - INSERT/REPLACE function Note: When INSERT/REPLACE is used with OPTIONS=VAL, the VALIDATE function is used before INSERT/REPLACE; therefore, DIBDB2nn messages may precede DIBDB6nn messages in the output.
7	DIBRECON - RECONSTRUCT function
8	DIBACCNT - ACCOUNT function
9	DIBJOBFD - JOBFIND function

DIBUT2nn — Indicates that the message was issued by U#UTIL2 during the system build process

DIBUT5nn — Indicates that the message was issued by U#5COBI while processing COBI data sets

DIBUT6nn — Indicates that the message was issued by DIBCONPR, the COBI utility which converts cataloged procedures to a form that can be used to scan COBI output data sets

next line (in the same direction as the last scan). If the scan was terminated by pressing the ATTN or BREAK key, and no option is specified, the scan begins with the line following the last line of the 256-byte buffer area that was being printed when the scan was terminated. (Thus, assume that the current buffer contains lines 15, 16, and 17 and that the ATTN key is pressed during the printing of line 16. Further assume that the RETURN key is pressed to request subsequent scanning. If a forward scan was in progress, line 18 will be the first line printed. If a backward scan was being made, line 14 will be the first line.)

After a scan operation has been completed or the ATTN or BREAK key has been pressed, the user is still in SCAN mode. The system prints a question mark at the terminal. When a data set of a current job is being scanned, any of the options explained above can be used. The primary purpose of the question mark, however, is to remind the user that he has control and is responsible for disposition of any JCL and SYSOUT data sets attached to the job. Usually, a JCL or SYSOUT data set that has been scanned should be scratched. The user must, in response to the question mark, enter another complete SCAN command pertaining to the same job and including one of the following responses, or enter an abbreviated form of the SCAN command by specifying only one of the following abbreviated responses:

- option or SCAN option
  indicates that the scan of the current data set is to continue

- data-set,option or SCAN data-set,option
  indicates that a data set of the same job is to be scanned

- JCL,option or SCAN JCL,option
  indicates that JCL of the same job is to be scanned

- KEEP
  indicates that all remaining SYSOUT data sets and JCL for the job being scanned are to be retained

- SCRATCH
  indicates that the data set or JCL being scanned is to be scratched

- SCRATCH data-set
  indicates that a data set of the same job is to be scratched

- SCRATCH JCL
  indicates that JCL of the same job is to be scratched

- SCRATCH job-number
  indicates that all SYSOUT data sets and JCL for the current job are to be scratched, and the entry for the job is to be removed from the COBI index

If the user enters SCAN job-number or SCRATCH job-number, and the specified job number is not the number of the current job, the SCAN or SCRATCH command is rejected. Any user entry other than one of any of the forms listed above, pertaining to the current job, is unacceptable. A message is printed, and the question mark is repeated. The user cannot relinquish control by pressing the ATTN or BREAK key. The SCAN mode is removed (and the system responds with READY) only when the user has exercised his option to (1) SCRATCH all SYSOUT data sets and JCL for the current job; (2) SCRATCH certain SYSOUT data sets and/or the JCL for the job, and KEEP remaining data sets and/or the JCL; or (3) KEEP all remaining SYSOUT data sets and JCL for the job. By specifying KEEP, the user ensures that SYSOUT data sets and JCL for the current job are retained in the system for at least seven days unless he causes them to be scratched. Data sets upon which no action is performed for a period

exceeding seven days may be deleted from the system by special operator action at the CALL-OS command console.

When a data set identified by a DSN parameter is being scanned, the user may enter scan requests by line number as described above. He must, in response to the SCAN mode question mark, enter another complete SCAN command pertaining to the same data set or one of the following abbreviated responses:

- option or SCAN option
  indicates that the scan of the data set is to continue

- MEM=memname[,option]
  indicates that the data set being scanned is a partitioned data set and the user wishes to scan another member of that data set; if no option is specified, the scan begins at the beginning of the member

- KEEP
  indicates that the scanned data set is to be retained

Note, however, that none of the other responses listed as acceptable when a data set of a current job is being scanned are acceptable when a data set identified by a DSN parameter is being scanned. The user cannot SCRATCH the data set. When the user has completed all desired scanning, he must KEEP the data set.

Note: The system will recognize one other user response when the user is in SCAN mode: entry of a WIDTH terminal command to adjust the current width of the print line. If output lines contain many blank characters between two columns, for example, the user can set the width of the print line at the last print position preceding the second column. If he does so, a carrier return will be made immediately following the last nonblank character preceding the specified width; intervening blanks will not be printed; and the characters beyond the width specification will be printed at the beginning of the next line. Because printing of blanks is eliminated, the total printing task can be completed more rapidly.

Examples

```
SCAN #123,2P3,1 THRU 10
(list of logical records 1 through 10 of SYSOUT data set 3
 of the second procedure of job #123)
   .
   .
   .
?
20
(logical record 20 of the same data set)
?
C/R
(list of logical record 21 and succeeding records of the same
 data set)
   .
   .
   .
ATTN
?
1000 T 1010
(list of logical records 1000 through 1010 of the same data set)
   .
   .
   .
```

DIBDB008 ERROR - CARD SHOULD HAVE ./ IN COLUMNS 1 AND 2.

A control statement card was expected containing ./ in the first two columns. Correct control statement card and rerun. The control statement on which the error occurred is not processed, and processing continues with the next control statement. Condition code = 8.

DIBDB009 ERROR - INVALID PARAMETER FIELD ON ABOVE xxxxxxxxx.

When xxxxxxxxx is CARD, the message indicates a syntactical error on the card printed immediately above the message. This error could be an invalid keyword parameter, parameter value, or delimiter, or the parameters exceed column 71. Correct error on card and rerun. When xxxxxxxxx is STATEMENT, the message indicates a parameter inconsistency on the control statement. Check the valid parameter combinations for the function requested and correct the control statement. In either case, the control statement on which the error occurred is not processed, and processing continues with the next control statement. Condition code = 8.

DIBDB010 NO PARAMETERS ON CONTROL STATEMENT.

A ./ card has been read that contains no parameters. The card is ignored. Condition code = 4.

DIBDB011 ERROR - INVALID OR NO FUNCTION SPECIFIED.

The first field following the ./ of the control statement does not contain a valid data base utility function name. Correct the control statement and rerun. The control statement on which the error occurred is not processed, and processing continues with the next control statement. Condition code = 8.

DIBDB012 ERROR - INVALID OR NO xxxxxxxx PARAMETER SPECIFIED.

For a function requiring the named parameter, either the parameter was not given on the control statement or the parameter was given but did not specify a valid parameter value. Correct the control statement and rerun. The control statement on which the error occurred is not processed, and processing continues with the next control statement. Condition code = 8.

DIBDB013 ERROR - REQUIRED DATA SETS NOT INCLUDED IN JCL.

No ddcard was found in the job control language for those data sets required by the control statement. Include ddcards in JCL and rerun. The control statement on which the error occurred is not processed, and processing continues with the next control statement. Condition code = 8.

DIBDB014 ERROR - NO INDEX ENTRY FOUND FOR TO GROUP.

The group indicated by either the USER parameter or the USRGROUP parameter is not specified by any of the CALL-OS index entries. Ensure that the USER parameter specifies a user number within a CALL-OS user or system group or that the USRGROUP parameter exactly names a CALL-OS user or system group. The control statement on which the error occurred is not processed, and processing continues with the next control statement. Condition code = 8.

DIBDB015 ERROR - NO INDEX ENTRY FOUND FOR FROM GROUP.

The group indicated by either the FROMUSER parameter or the FRMGROUP parameter is not specified by any of the CALL-OS Index entries. Ensure that the FROMUSER parameter specifies a user number within a CALL-OS user or system group or that the FRMGROUP parameter exactly names a CALL-OS user or system group. The control statement on which the error occurred is not processed, and processing continues with the next control statement. Condition code = 8.

DIBDB016 HIGHEST CONDITION CODE WAS nn.

The listed code is the highest generated in the job step.
- 00 - no errors encountered.
- 04 - an error was encountered that did not terminate processing of a control statement.
- 08 - an error was encountered that did terminate processing of a control statement.

DIBDB017 END OF CALL-OS DATA BASE UTILITY RUN.

The data base utility program has completed processing.

DIBDB030 ERROR - UNABLE TO OPEN DATA SET; DDNAME=dddddddd.

Unable to open the data set specified by the named ddcard. Condition code = 8.

DIBDB031 ERROR - BAD DEB; DDNAME=dddddddd.

After a successful open, the DCB pointer to the data extent block data extent block (DEB) points to an area that does not have hexadecimal 'F' in the low-order four bytes of the DEB identifier (DEB+24). Condition code = 8.

DIBDB032 ERROR RETURN FROM OS CONVERT ROUTINE; DDNAME=dddddddd.

Use of the OS/360 routine to convert a relative track address to absolute (pointed to by the Communication Vector Tables plus 28) resulted in an error return code. Routine was unable to convert the relative disk address to an absolute address. Condition code = 8.

DIBDB033 USER GROUP DATA SET DEPLETED; DDNAME=dddddddd.

There is no more disk space available for allocation in this data set. Condition code = 4.

DIBDB034 ERROR - INDEX ENTRIES OUT OF SEQUENCE; GROUP=aaabbb.

Resequence CALL-OS Index entries according to their type (user group, system group, compiler, etc.), then according to group names, and finally according to the relative data set number within the group. Condition code = 8.

DIBDB035 NO GROUP BEING OPENED.

No group was opened during this pass through DIBOPEN. This error should not occur since earlier checks in DIBINIT should discover the error. Condition code = 4.

General Form (Direct Submit)

```
SUBMIT program-name1[,program-name2,...][,(options)]
```

General Form (Indirect Submit)

```
SUBMIT ↑program-name[,(options)]
```

Valid Entries

A specified program name must be the name of a source-program file in the user's library. For a direct submit, a maximum of twelve program names can be specified. For an indirect submit, only one program name can be specified. This name identifies a source-program file that contains, as its first statement, a $$SUBMIT control statement. The $$SUBMIT control statement contains from one to twelve program names assigned to source-program files of the job. In either case, the combination of program files, in the order specified, must constitute one complete OS/360 job.

The options that may be specified for either form of the SUBMIT command are noted below and explained in detail under "Scannable Data Sets" in a previous section of this manual. No comments or other information should follow the last entry in the SUBMIT command or the $$SUBMIT control statement. Inclusion of such information will cause a syntax error to occur.

Description

This command is used to format and copy a requested job from CALL-OS program files to a sequential data set for input to OS/360 batch processing. If the SUBMIT command contains only one program name, that program file contains the complete OS/360 job. If the SUBMIT command contains more than one program name, the contents of the program files are concatenated in the order specified to form the OS/360 job. If more than twelve program files are to be included in a job, the user must merge some of the program files and save them in merged form so that he can operate within the limit of twelve.

The user specifies options to cause data sets to be retained for scanning (printing) at his terminal. The JCL and up to eight user-defined and procedure-defined SYSOUT data sets can be retained. Numbers of user-defined SYSOUT data sets must be preceded by U and may range from 1 through 127. An entry of the form nPmm refers to a SYSOUT data set defined in a cataloged procedure. n is the procedure number and may range from 1 through 7; mm is the number of the data set and may range from 1 through 15. If n is 1, it may be omitted, because 1 will be assumed. Data sets to be retained may be named in any order. Each entry except the last must be followed by a comma, and the complete list must be enclosed in parentheses. If the user does not desire to retain data sets for scanning, the parameter entry NONE or its abbreviated form, N, should be enclosed in parentheses as the options entry. If one option is specified repeatedly (for example, if U2 is specified twice), only one copy of the data set is retained, but each specification is counted as one toward the limit of eight.

If no entry is specified in the options field of a direct SUBMIT command, the user is prompted with the following message:

```
ENTER (OPTION) OR CR
```

The user may respond by entering options or pressing the RETURN key. The latter response indicates that no data sets are to be retained for scanning (that is, it is equivalent to the parameter entry NONE).

An up-arrow preceding a program name identifies an indirect SUBMIT command. It indicates that the program file referred to in the SUBMIT command contains, as its first statement, a $$SUBMIT control statement. The $$SUBMIT control statement provides the information that must otherwise be specified in full in the SUBMIT command for the job. This indirect form of the SUBMIT command is provided to make it easier for the terminal user to submit a job again, once it has been set up with its options for OS/360 batch processing. (For a complete discussion of $$SUBMIT, see "$$SUBMIT", which follows.)

The JCL, source-program statements, and data for a submitted job must be available in the user's library at the time the job is submitted. It may have been entered at the terminal previously and saved (via a SAVE command) just as though it were a CALL-OS BASIC, CALL-OS FORTRAN, or CALL-OS PL/I source program; loaded into the user work area from a CALL-OS shared library and saved in the user's library; or copied into the user's library via a CALL-OS utility. Each line of each source-program file in a submitted job is expanded to represent an 80-character card-image as it is copied to the OS/360 batch processing input data set. (For an example of a user's input job stream and of COBI handling of that input, see the CALL-OS Executive and Utilities Program Description Manual.)

A check is made to determine whether each program name specified in a SUBMIT command refers to a source-program file in the CALL-OS user's data base. If no entry for the file exists in the user's catalog, or if the specified name does not refer to a source-program file, the system prints:

```
PROGRAM NOT IN CATALOG, OR NOT A PROGRAM FILE - program-name
```

where program-name identifies the file that cannot be submitted. Since program files cannot be submitted directly from CALL-OS shared libraries, the user must copy any such file into his user library before including it in an OS/360 job.

Examples

```
SUBMIT JOB1,(2P3,U1)
#1234 SUBMITTED AS JOB1

SUB ↑JOB2
#5678 SUBMITTED AS IBM02015
```

COBI CONTROL STATEMENTS

$$SUBMIT and $$TABS are COBI control statements the user may include in his input job stream to simplify the formatting and submitting of an OS/360 job. Since these statements are COBI control statements, they are acted upon and deleted automatically during SUBMIT processing. The user should not include other statements in which $$SUBMIT or $$TABS is entered beginning in column 1 in his input job stream. (A single exception to this guideline is given under "DD Statement".)

$$SUBMIT

General Form

```
nnn $$SUBMIT program-name1[,program-name2,...][,(options)]
```

Valid Entries

nnn is the line number preceding the $$SUBMIT control statement, as entered by the user at his terminal. $$SUBMIT may be abbreviated to $$SUB. At least one blank must separate $$SUBMIT (or $$SUB) from the first program name. All program names must be assigned to source-program files saved in the terminal user's library. A maximum of twelve program names can be included in one $$SUBMIT control statement. The options that may be specified are explained in detail under "Scannable Data Sets" and "SUBMIT". No comments or other information should follow the last entry in the SUBMIT command specified in a $$SUBMIT control statement. Inclusion of such information will cause a syntax error to occur.

Description

If a submission procedure is to be used repeatedly or if the text of the SUBMIT command is long, the user may wish to use the indirect form of the SUBMIT command. This form of the command is identified by an up-arrow preceding a single program name which points to a source-program file containing a $$SUBMIT control statement. The $$SUBMIT control statement must be the first statement in this file. It provides the information that the user must otherwise enter each time he requests the job from the terminal. The $$SUBMIT control statement cannot, itself, be another indirect submit. If options are specified in the indirect SUBMIT command entered at the terminal, they override any options specified in the $$SUBMIT control statement. If no options are specified in the SUBMIT command or the $$SUBMIT control statement, the user is not prompted to enter options. The system assumes that no options are desired.

The length of a $$SUBMIT control statement is not restricted to 80 characters, as is the length of JCL, source-program, and data lines. The user may enter a $$SUBMIT control statement that exceeds the length of one terminal line. If so, consecutive lines can be used. Each line must begin with a line number and $$SUBMIT, followed by entries specified as though the complete statement were on one line. An example is shown below.

```
100  $$SUBMIT A,B,C,D,E,
110  $$SUBMIT F, G, H, I, (JCL, P1, P2, U1, U2, U3,
120  $$SUBMIT 2P1, 2P2)
```

All lines of the $$SUBMIT control statement must be placed, consecutively, in their logical order, at the beginning of the first

program file named in the indirect SUBMIT command. The total length of the $$SUBMIT control statement from the beginning of the first $$SUBMIT entry, and including the succeeding line numbers, blanks, $$SUBMIT entries, and two additional characters for each continuation line, cannot exceed 244 characters. $$SUBMIT statements encountered subsequent to the $$SUBMIT statement beginning on the first line of the program file named in the indirect SUBMIT command are not taken to be control statements. Any such lines are deleted during SUBMIT processing.

Assume that the terminal user enters

```
SUBMIT ↑PROG1
```

and the first line in the source-program file PROG1 is

```
100 $$SUBMIT PROG1,PROG2,PROG3,(JCL,P1)
```

The indirect SUBMIT command entered by the user causes COBI to look for a $$SUBMIT control statement as the first line in PROG1. Since PROG1 is also the first program file named in the $$SUBMIT statement, the contents of the lines following the first line in PROG1 are combined with the contents of program files PROG2 and PROG3 to form an OS/360 job.

Now assume that the terminal user enters the same indirect SUBMIT command but the first line in the source-program file PROG1 is

```
100 $$SUBMIT PROG2,PROG3,(JCL,P1)
```

Any lines following the $$SUBMIT control statement in PROG1 are ignored. The contents of program files PROG2 and PROG3 constitute the OS/360 job.

$$TABS

General Form

```
nnn $$TABS n1,n2,n3,...,s1-s2
```

Valid Entries

nnn is the line number preceding the $$TABS control statement, as entered by the user at his terminal. n1, n2, and so on, are numerals, entered in ascending sequence and corresponding to specific columns of an 80-column punched card. Up to a maximum of 20 tab positions can be specified in one $$TABS control statement. s1 and s2 represent the sequence number field to be carried in every expanded, 80-character card-image record prepared for OS/360 processing. The sequence number field may be from one to eight characters in length. The line number preceding the source-program line from which the card image is created is inserted in these positions of the record. If necessary, the line number is preceded on the left by high-order zeros. At least one blank must follow the keyword $$TABS, and the length of the $$TABS statement cannot exceed 80 characters.

Description

When input is entered from a 2741 Communications Terminal or Teletype, the tabulation function available with the terminal can be used to simplify the input of fixed-format information, that is, to simulate the program drum card of a keypunch. This function is invoked by means of a $$TABS control statement formatted as shown by the example below.

```
110 $$TABS   1,10,16,36,72,73-80
```

The entries 1, 10, 16, 36, and 72 in the above statement correspond to the label, operation, operand, comments, and continuation fields established for a fixed-form source-program (OS/360 Assembler Language) statement line. The entry 73-80 causes each CALL-OS program-file line number to be converted to an eight-digit sequence number (with leading zeros) and placed in columns 73-80 of the card-image record corresponding to that line. Thus, the record is formatted to be compatible with IEBUPDTE and other OS/360 utilities.

For the $$TABS control statement shown above, the left margin on the 2741 Communications Terminal should be set at 0. Tabs should be set at 11, 20, 26, 46, and 82. Thus, position 11 on the 2741 corresponds to column 1 of an input line, position 20 to column 10, and so on.

Only one blank or one tab character can separate the line number of a $$TABS control statement from the keyword $$TABS. Only one sequence-number-field entry is permissible in one $$TABS statment. If an input line contains other data in the portion of a line specified by this entry, that data will be overlaid by the sequence number.

If the tabulation function is being used, a blank or tab character immediately following the line number of a line in the input job stream is interpreted as a tab character. The first character following that character is placed in the column specified by n1. If a blank or tab does not follow the line number, a tab is forced, and the character immediately following the line number is placed in column n1. The second tab character in the input line causes the next data character to be placed in the column specified by n2, the next positions succeeding data in the column specified by n3, and so on. If, for example, the

column specified by n2 has been passed by the time the second tab character is encountered, that tab character is assumed to correspond to the n3 specification.

Use of the tabulation function permits efficient skipping of columns which are to be blank during initial keying of input from a 2741 Communications Terminal. Tab characters are placed in the source-program line, compressing the file to an optimum reduced format. Disk storage requirements are lessened accordingly. The same file compression is achieved when the tabulation function is employed by a Teletype user. In this case, however, neither printing nor spacing occurs at the terminal when the user enters the tab character (control I) from the keyboard.

When the tabulation function is employed on input, the readability of OS/360 printout of that data (say, a source-program listing) is enhanced. Spacing is as specified by tab characters of the input line and corresponding $$TABS specifications. Tab characters do not cause similar spacing if the data is listed by the terminal user at his terminal. Generally, one blank is substituted for each tab character on terminal output.

Backspace characters can be included in lines of a source-program file and used in conjunction with the tabulation function to format lines entered at the terminal for OS/360 processing. If one or more backspace characters are entered immediately following a tab, and the tabulation function is being used, characters of the card-image record are positioned accordingly. Backspaces cannot be used to indicate a position to the left of the first column (n1) defined in the $$TABS control statement. If excess backspaces are entered, column n1 is assumed.

As an example, assume that a CALL-OS FORTRAN user enters the following $$TABS statement in his input job stream:

```
120 $$TABS 1,7
```

Two tab characters and a backspace entered in a line of the source-program file, say, a line corresponding to a FORTRAN continuation card, would position the first data character in column 6 of the corresponding card-image record.

Any backspace characters entered other than immediately following tab characters, or other than when the tabulation function is being used, are taken as data.

Use of more than one $$TABS control statement in an input job stream is permissible. For example, one $$TABS statement can be used to indicate the form of the JCL, a second to indicate the form of source-program statements, and a third to indicate the form of input data. Each $$TABS statement must precede the input lines to which it applies. Tabs on the terminal should be adjusted accordingly.

A $$TABS statement containing only a sequence-number-field entry can be included in the input job stream. In this case, the user does not wish to use tabs or does not have tabulation at his terminal. The $$TABS statement causes line numbers to be positioned as sequence numbers, as described above.

COBI-RELATED OS/360 CONTROL STATEMENTS

The JOB, EXEC, DD, and comment statements of OS/360 JCL may involve special considerations when a job is submitted via COBI for OS/360 batch processing. Specific details are given below. Additional information concerning these statements is given in IBM System/360 Operating System Job Control Language.

COMMENT STATEMENT

General Form

```
nnn //* [cc,][cc,...]'message'
```

Valid Entries

nnn is the line number preceding the comment statement, as entered by the user at his terminal. If the Multiple Console Support (MCS) feature is included at system generation, entries of the form cc can be used to send messages to specific OS/360 consoles. The cc entries are OS/360 routing codes, as described in IBM System/360 Operating System Supervisor and Data Management Macro Instructions. The message may contain any characters allowable in OS/360 write-to-operator messages.

Description

In as much as OS/360 allows JCL comment statements to be placed anywhere within a job except before the JOB statement, COBI is designed to recognize any JCL comment statements before the JOB statement as text to be sent as write-to-operator (WTO) messages to one or more OS/360 consoles. Such statements are not considered to be part of the job.

If a job requires special action by the computer operator, a comment statement can be included to cause text enclosed in single quotes in the comment statement to be printed on the OS/360 operator console. In effect, a write-to-operator (WTO) message is created. Single quotes in the message will be handled like any other data characters.

If the Multiple Console Support (MCS) feature is available, OS/360 routing codes can be specified in a comment statement to cause a message to be printed on one or more specific OS/360 consoles. A comment statement cannot be used to send a message to the CALL-OS communications console.

Up to four WTO messages can be entered for one job. Each message must begin on a separate JCL comment statement and may extend to one continuation line if necessary. The first line of a JCL comment statement is flagged as a continued line by placing a nonblank character in column 72 of the line. Comment statements in excess of four messages are ignored. Any comment statements appearing after the JOB statement are treated as normal JCL statements of the user's job.

Messages sent to OS/360 via comment statements appear as:

```
DIBRD003 userid job-name message
```

where userid is the user number of the user sending the message, job-name is the name by which OS/360 recognizes the job, and message is the user-supplied message. The message may be up to 95 characters in length. The characters DIBRD003, user number, and job-name are appended to the message by system programs.

Notes: 1. If the terminal user does not include comment statements in his input job stream but wishes to send a message to the computer operator at the OS/360 operator console, he can do so by issuing a NOTIFY command at his terminal. This command can also be used to send a message to the CALL-OS communications console or, if the Multiple Console Support (MCS) feature is included, to specific OS/360 consoles.

2. Write-to-operator messages created by comment statements will be printed as indicated, even if a job is cancelled by user issuance of a CANCEL command for the job.

DIBDB354 ERROR - FOR PULL, FILE MAY NOT = JOB OR EVERY.

COBI job entries may not be pooled. Therefore, FILE=JOB or EVERY (which includes JOB) may not be specified for PULL. Condition code = 8.

DIBDB355 ERROR - IF DATE IS INPUT, FILE MAY NOT = JOB OR EVERY.

FILE=JOB or EVERY (which includes JOB) refer to COBI jobs; however, these jobs do not have a last-used date in their catalog entries. Condition code = 8.

DIBDB356 ERROR - FOR PURGE, FROMUSER MAY NOT = xxxxxx.

xxxxxx may be **LIB or aaan00, referring to a directory; directory entries are pulled, not purged. Condition code = 8.

DIBDB357 WARNING - OPTION INVALID FOR COMMAND.

Certain OPTIONS parameters may be specified only with certain commands. For example, OPTIONS=NOTAPE or UNLOCK may be specified only with the PURGE command; OPTIONS=* or ** may be specified only with the PULL command. Condition code = 4.

DIBDB360 ERROR - OPEN FAIL OF COBI xxxxx DATA SET.

Either the COBI index or COBI JCL data set could not be opened. Ensure that the DD statement is present and specified correctly. Condition code = 8.

DIBDB361 ERROR - OPEN FAIL OF SYSPUNCH DATA SET.

The card punch data set could not be opened. Ensure that the DD statement is present and specified correctly. Condition code = 8.

DIBDB362 ERROR - CATALOG ENTRY AND COBI INDEX ENTRY DO NOT MATCH FOR USER aaannn ENTRY xxxxxxxx.

The COBI index record referred to by the job number in the catalog entry was either not active, belonged to a different user, or had a different COBI job name. The JOBFIND function should be executed to update the catalog. Condition code = 4.

DIBDB401 ERROR - BAD OPEN OF dddddddd DATA SET [; WRITE TERMINATED.]

The data set defined by the specified DD statement could not be opened and the output specified by OUTPUT=CARD or OUTPUT=OSDS could not be produced; in this case, condition code = 4. If dddddddd is either SYSPUNCH or LIBRARY, the WRITE function is terminated; in this case, condition code = 8. In all cases, check the JCL to ensure that the required DD statement is present and specified correctly. Correct the JCL and resubmit the function request.

DIBDB402 ERROR - DATA SET SPECIFIED BY dddddddd HAS INVALID CHARACTERISTICS [; WRITE TERMINATED.]

The data set defined by the specified DD statement was opened, but the DCB parameters of the data set did not satisfy the WRITE function requirements; in this case, condition code = 4. If dddddddd is LIBRARY, the WRITE function is terminated; in this case, condition code = 8. See the publication CALL-OS Executive and Utilities Program Description Manual for a discussion of the

required DCB attributes. Correct the JCL and resubmit the function request.

DIBDB411 ERROR - PARTITION/REGION NOT LARGE ENOUGH FOR WRITE; FUNCTION TERMINATED.

The WRITE function could not be executed due to insufficient main storage. Specify a larger partition or region for the data base utility and resubmit the function. Condition code = 8.

DIBDB421 ERROR - REQUIRED xxxx PARAMETER NOT SPECIFIED FOR WRITE; FUNCTION TERMINATED.

The user has failed to specify a control statement parameter which is required by the WRITE function. Consult the publication CALL-OS Executive and Utilities Program Description Manual for a discussion of required parameters. Some parameters are required in all cases, while others may be required only in conjunction with certain parameters. Condition code = 8.

DIBDB431 ERROR - INCORRECT xxxxxxx PARAMETER SPECIFIED FOR WRITE; FUNCTION TERMINATED.

Either a control statement parameter has been specified with an unallowable value, or an invalid combination of parameters has been given. See the publication CALL-OS Executive and Utilities Program Description Manual for allowable parameters and parameter combinations. Condition code = 8.

DIBDB441 ERROR - FROMUSER SPECIFIED COULD NOT BE FOUND; WRITE TERMINATED.

The FROMUSER parameter on the function control statement either specified a user number in a user group which was not available in the specified cluster, or, if the user group was available, the specified user number could not be found in the equivalency file for the group. Condition code = 8.

DIBDB442 THE ** LIBRARY HAS NO EQUIVALENCY ENTRY. THE FIRST DIRECTORY LINK IS AT 00 0002 02 IN SYSGRP.

The user has specified parameters which indicate that an equivalency entry is to be printed for the **Library. This message indicates that the system is designed so that the first directory link for the **Library is always located at DTTR 00 0002 02 in the system group data set, and that there is no equivalency entry in the CALL-OS system for the **Library. Condition code = 0.

DIBDB445 ERROR - REQUIRED PASSWORD INCORRECT OR NOT SPECIFIED; WRITE TERMINATED.

The PASSWORD parameter is required on the control statement in order to perform the requested WRITE function; either the PASSWORD parameter was not specified, or the specified password did not match the corresponding password in the equivalency file, or the SYSLIB password is required and the system group in the specified cluster was not available (no SYSGRPnn DD statements were provided). Condition code = 8.

DD STATEMENT

General Form

```
nnn //ddname  DD  [operand1][,operand2,...] [comments]
```

Valid Entries

nnn is the line number preceding the DD statement, as entered by the user at his terminal. Operands can be specified as described below.

Description

Data definition (DD) statements are included in the user's input job stream to describe data sets of the job and request allocation of I/O resources. (See IBM System/360 Operating System Job Control Language User's Guide.)

If the user specifies that user-defined SYSOUT data sets are to be kept (that is, enters options of the form Unnn in the SUBMIT command for the job), DD statements with corresponding &Unnn entries as first parameters must be included in his input job stream. The user may include SPACE parameters or allow COBI to insert them. DISP and UNIT parameters are built by COBI and should not be specified by the user. If a SPACE parameter is included in a DD statement, it must immediately follow the &Unnn parameter. Any other parameters acceptable to OS/360 can be entered in DD statements.

The last operand of a DD statement must end prior to or at position 70 of the last line of the statement. In a statement containing comments, at least two blanks should separate the last operand of the statement from the comments. If the user leaves only one blank rather than two, the initial character of the comment is overlaid with a blank. One blank is required.

The user should not include &U type functions in his input job stream. Any &U designation must be followed by a number, and that number must be followed by a comma or a blank. Thus, for example, entries such as &USPACE and &U34A on user-entered DD statements are invalid.

Assume that U20 is specified as an option in the SUBMIT command for a job. The examples below show DD statements, any of which could be included in the input job stream to identify U20 as PRTOUT and cause I/O resources to be allocated for it.

```
200 //PRTOUT   DD  &U20
210 //PRTOUT   DD  &U20,SPACE=(TRK,(10,10))
220 //PRTOUT   DD  &U20,DCB=BLKSIZE=400
```

The expansion that would be performed by COBI for each of these statements, respectively, is shown below. (#NNNNN is the COBI-assigned job number.)

```
(1)  //PRTOUT  DD  DSN=DIB.USERID.#NNNNN.U020,
     //            SPACE=(TRK,(010,010),RLSE),DISP=(NEW,KEEP),
     //            UNIT=SYSDA

(2)  //PRTOUT  DD  DSN=DIB.USERID.#NNNNN.U020,
     //            DISP=(NEW,KEEP),UNIT=SYSDA,
     //            SPACE=(TRK,(10,10))
```

```
(3)  //PRTOUT  DD  DSN=DIB.USERID.#NNNNN.U020,
     //            SPACE=(TRK,(010,010),RLSE),DISP=(NEW,KEEP),
     //            UNIT=SYSDA,DCB=BLKSIZE=400
```

If DD statements containing &Unnn parameters for which no corresponding Unnn options are specified in the SUBMIT command are included in the input job stream, COBI replaces the &Unnn parameters with a SYSOUT=Z parameter. Any other information included on the DD statement is moved to a continuation line.

If the processing steps described above are not acceptable to the terminal user, he can override them by entering DD statements formatted as shown by the example below. Similar DD statements can be used to override DD statements in cataloged procedures.

```
//PRTOUT   DD  DSN=USERID.ANYNAME,UNIT=2314,DISP=(NEW,KEEP),
//             VOL=SER=222222,SPACE=(CYL,(020,020))
```

The user can scan this data set, provided that a SCANxx DD statement defining volume 222222 is included in the system startup deck. (See the CALL-OS Executive and Utilities Program Description Manual for a description of the job control statements that may be required when initiating CALL-OS.)

A special form of the DD statement is recognized by OS/360 and may be included in a submitted job. This form is

```
nnn //ddname  DD   DATA
```

When OS/360 encounters a DD statement of this form, it assumes that the input following this statement and preceding a delimiter statement (having /* in columns 1 and 2) is to be entered through the input stream for use by a processing program. This input commonly includes JCL statements (having // in columns 1 and 2) but OS/360 does not interpret the statements at this time. Likewise, statements within such a grouping are not interpreted and acted upon by COBI during SUBMIT processing. COBI expands such statements to 80-character card-image records, but neither OS/360 JCL statements nor COBI control statements following a DD statement are acted upon.

EXAMPLES

COBI is an extremely useful and flexible tool for the CALL-OS terminal user. Terminal listings are reproduced below, with comments, as documentation of three typical job sequences using the COBI option.

SUBMITTING A FORTRAN JOB

In the following example, a FORTRAN program, together with the JCL needed to execute it, is loaded from the user's library, listed, and then submitted via COBI to OS/360. It is assigned job number 10 during the submit process.

User entry of both JOBSTATUS and DSSTATUS commands, and system responses to these commands, follow. The user requests scanning of SYSOUT data set P1, interrupts scanning of the data set by pressing the ATTN key, and (in response to the question mark) scratches the data set. He scans a portion of the JCL and then scratches that information. Since only JCL and P1 were retained for scanning and both are now scratched, all references to the job are removed from the system. The user can continue to other jobs.

```
LOAD FORTA
READY
LIST

FORTA       10:02   10/29/71    MVTDEV

100 //FORTRAN  JOB  (816,6673),'NAME, I.',MSGLEVEL=1
110 //FORT     EXEC FORTHCLG
120 //SYSIN    DD   *
130         DIMENSION A(10)
140 55      DO 66 I = 1,10
150 66      A(I) = I
160         DO 77 J = 1,10
170 77      WRITE (6,*) A(J)
180         END
190 /*

SUB FORTA,(JCL,P1)
#10    SUBMITTED AS FORTRAN

JOBSTATUS #10
JOB ID:  STATUS:   CODE:   DATA SETS:
#10    NOT DONE          JCL,1P01
READY
```

Time passes, and the job is completed in OS.

```
JOBSTATUS #10
JOB ID:  STATUS:   CODE:   DATA SETS:
#10   COMPLETED           JCL,1P01
READY

DSSTATUS #10
JOB ID:     SYSOUT:     VOL ID:     STATUS

#10         JCL
            1P01        111111      ON-LINE
READY

SCAN #10,P1
LEVEL 20.1 (MAY 71)                          OS/360 FORTRAN H
                    DATE    71.300/11.00.56

       COMPILER OPTIONS - NAME= MAIN,OPT=02,LINECNT=50,SIZE=0000K,
                          SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,
NOEDIT,NOID,NOXREF
   ISN 0002            DIMENSION A(10)
   ISN 0003    55      DO 66 I = 1,10
   ISN 0004    66      A(I) = I
   ISN 0005            DO 77 J = 1,10
   ISN 0006    77      WRITE (6,*) A(J)
ERROR DETECTED - SCAN POINTER = 4
   ISN 0007            END
                                          FORTRAN H ERROR MESSAGES

                    ERROR NO   LEVEL     ERROR MESSAGE

       ISN   0006      IEK160I    8      THE I/O STATEMENT HAS AN INVALID
DELIMITER IN THE PARAMETERS.
       ISN   0007      IEK509I    4      THE PROGRAM DOES NOT END WITH A
STOP,RETURN,OR GO TO.
```

```
*OPTIONS IN EFFECT*     NAME= MAIN,OPT=02,LINECNT=50,SIZE=0000K,
?SCRATCH                                    ----------------------------
?SCAN JCL                                   User presses ATTN key.
                                            ----------------------------

//FORTRAN JOB (816,6673),'NAME, I.',MSGLEVEL=1,
//MSGCLASS=Z
*** #00010-- --FORTRAN -1003  ----- H          H
//FORT EXEC FORTHCLG
XXFORH   PROC  P1='SYSOUT=Z',     *** FORT SYSPRINT
XX             P2='SYSOUT=Z',     *** LKED SYSPRINT
XX             P3='SYSOUT=Z'      *** GO FT06001
XXFORT   EXEC  PGM=IEKAA00,REGION=228K
XXSYSPRINT DD &P1,SPACE=(TRK,(10,10),RLSE)
IEF6531 SUBSTITUTION JCL - DIB.AAA222.#00010.P101,DISP=(NEW,KEEP),
UNIT=235,SPACE=(TRK,(010,010),RLSE)
XXSYSPUNCH DD  SYSOUT=B
XXSYSLN    DD  DSNAME=&LOADSET,UNIT=SYSSQ,DISP=(MOD,PASS),
?SCRATCH                                    ----------------------------
READY                                       User presses ATTN key.
                                            ----------------------------
```

PRINTING A SYSOUT DATA SET

The CALL-OS terminal user may elect to print a SYSOUT data set that he specified be retained for scanning when he submitted a COBI job. The following example is a list of the JCL used to print the first procedure-defined SYSOUT data set of the first procedure of job #230, submitted by a user signed on with user number AAA222.

```
100 //PRINT    JOB  (816,6661),'NAME, I.',MSGLEVEL=1,CLASS=I
110 //XX       EXEC PGM=IEBGENER
120 //SYSPRINT DD   SYSOUT=Z
130 //SYSUT1   DD   DSN=DIB.AAA222.#00230.P101,UNIT=2314,
140 //              VOL=SER=222222,DISP=SHR
150 //SYSUT2   DD   SYSOUT=Z,SPACE=(CYL,(3,10)),
160 //              DCB=(BLKSIZE=1330,LRECL=133,RECFM=FBM)
170 /*
```

The SYSOUT parameter (Z) should be the same as the CBCLASS parameter specified in the system startup deck if the JCL and the data set are to be printed as one job.

The user enters the following SUBMIT command to submit the printing task as a COBI job.

```
SUBMIT PRINT
```

The JCL and the content of the data set specified in the JCL will be printed as output.

PUNCHING DATA FROM A CALL-OS PROGRAM FILE

The terminal user may wish to punch the contents of a CALL-OS program file. To do so, he enters JCL such as shown by the example below in a CALL-OS program file.

```
100 //PUNCH    JOB  (887,4521),'NAME, I.',MSGLEVEL=1,CLASS=I
110 //XX       EXEC PGM=IEBPTPCH
120 //SYSPRINT DD   SYSOUT=Z
130 //SYSUT2   DD   SYSOUT=B
```

```
140 //SYSIN     DD   *
150             PUNCH TYPORG=PS
160 /*
170 //SYSUT1    DD   *
```

If both the JCL above and the data from the CALL-OS program file PGMNAME are to be punched as output, the SYSOUT parameter (Z) should be the same as the CBCLASS parameter specified in the system startup deck. The program file must not contain any lines which begin with // or /*. If the program file contains such lines, then line 170 of the JCL above should be changed to

```
170 //SYSUT1    DD   DATA
```

The user enters the following SUBMIT command to submit the punching task as a COBI job.

```
SUBMIT PUNCH,PGMNAME
```

The JCL above and the data from the CALL-OS program file PGMNAME will be punched as output.

System messages are generated by the CALL-OS system and printed at the terminal for the benefit of the terminal user. These messages, together with indications of causes, are listed alphabetically below. Some messages are generated when error conditions occur during processing. In most cases, if one or more serious errors exist, detection of the first error causes a message to be printed, and processing is terminated. Other CALL-OS system messages are not related to specific error conditions. Such messages provide for system/user communication during normal system operations.

Other messages to the terminal user may be generated at the central computer installation. These messages are entered by the CALL-OS operator at the command console and routed to all online CALL-OS terminal users or to a particular user. They are issued primarily to warn that something unexpected has happened or is about to happen in the system. Examples are:

SYSTEM WILL BE SHUT DOWN AT 4:30 P.M.

PLEASE CALL CENTRAL COMPUTER INSTALLATION (322-3671)

The text of these messages is determined by the computer operator. Their timing is controlled. No user will be interrupted in the middle of an input line or output block. Thus, no user input or processing information can be lost. A warning message is transmitted at the first logical breakpoint in a user's terminal input/output.

Message: ASTERISK(S) MUST PREFIX FILE NAME

Cause: An attempt has been made to POOL or PULL a program without specifying leading asterisks.

Message: ASTERISKS SHOULD NOT PREFIX FILE NAME

Cause: One of the following commands has been entered with an asterisk preceding the program or data file name: LOCK, UNLOCK, PROTECT, ALLOW, SECURE, RELEASE, PURGE. These commands are valid only when operating on entries in the user's catalog although the specified action does affect the use of a named program or data file via a shared library.

Message: CATALOG EMPTY

Cause: The user has entered the CATALOG command (for the user's library or ***Library) and there are no program or data filenames in the catalog.

Message: COMPILATION INVALID WHEN LANGUAGE = 'DATA'

Cause: The user has entered a RUN or STORE command for a file with which DATA is associated as a language name. The user should enter the appropriate language (via an ENTER command) and then reenter his RUN or STORE command.

1. For either program input or data file input without the FORMDATA option:

   RECFM=F,BLKSIZE=80 or

   RECFM=FB,LRECL=80

2. For data file input with the FORMDATA option:

   RECFM=F,BLKSIZE less than or equal to 225, or

   RECFM=FB,LRECL less than or equal to 255, or

   RECFM=V or VB,LRECL less than or equal to 259

3. For data file input, the data set organization must be sequential.

4. For program input, the data set organization must be partitioned.

For any case, condition code = 8.

DIBDB631 ALL FILES PROCESSED FOR THIS CONTROL STATEMENT.

Indicates end of processing for the current control statement. Condition code = 0.

DIBDB632 I/O ERROR READING aaannn EQUIVALENCY RECORD.

An I/O error occurred while attempting to read the equivalency record for the specified user. Processing of the control statement is terminated. Condition code = 8.

DIBDB633 MARGIN PARAMETER MAY NOT BE OVERRIDDEN.

The MARG parameter was specified when standard margins must be used for the input. The MARG parameter is ignored. Condition code = 4.

DIBDB634 ERROR - UNABLE TO FIND OSDS MEMBER xxxxxxxx.

The named member was not found in the partitioned data set defined by the LIBRARY DD statement. For a single file, condition code = 8; for multiple files, condition code = 4.

DIBDB635 ABOVE RECORD HAS INVALID LINE NUMBER.

For INPUT=CARD or OSDS and LANG=BASIC, a line number was found to be either not all numeric characters or not sequential. The record is ignored. Condition code = 4.

DIBDB636 ABOVE RECORD NOT IN SEQUENCE.

OPTIONS=SEQ was specified on the control statement and the record printed was not in sequence. The record is ignored. Condition code = 4.

DIBDB637 ERROR - INPUT DATA COUNT DOES NOT EQUAL COUNT INDICATED ON /$ RECORD.

For INPUT=CARD or OSDS and FILE=DATA without the FORMDATA option, an input record was found to have a data count that did not equal the amount of data actually present for the record. Condition code = 8.

DIBDB638 ERROR - NO FILE WAS FOUND WHICH MET CONTROL STATEMENT REQUIREMENTS.

For INPUT=OSDS, TAPE, or DISK, no file was found which exactly met those requirements specified on the control statement. Condition code = 4.

DIBDB639 FILE=xxxxxxxx NOT POOLED.

The named file was to be transferred from a shared library; the file could not be found. For multiple files, condition code = 4; for a single file, condition code = 8.

DIBDB640 FUNCTION TERMINATED.

The INSERT/REPLACE operation requested has been terminated. The condition code is determined by the severity of the error which caused termination.

DIBDB641 FILE xxxxxxxx TO BE INSERTED ALREADY IN aaannn LIBRARY.

On a search of the specified user's catalog for an INSERT function, the named file was found to already exist in the catalog. The input file was not inserted. Condition code = 4 if multiple files are being inserted; condition code = 8 if a single file is being inserted.

DIBDB642 FILE xxxxxxxx TO BE REPLACED IN aaannn LIBRARY NOT SAME FILE TYPE AS INPUT FILE.

The named file to be replaced was found to be a different file type from the input file specified by the FILE parameter. The existing file was not replaced by the input file. Either ensure that the FILE parameter is correct or purge the active file in the catalog and insert the new file. Condition code = 4 if multiple files are being transferred; condition code = 8 if a single file is being entered.

DIBDB643 FILE xxxxxxxx TO BE REPLACED IN aaannn LIBRARY IS LOCKED.

The named file to be replaced was found to have the LOCK attribute and OPTIONS=(UNLOCK) was not specified. The existing file was not replaced by the input file. Condition code = 4 if multiple files are being transferred; condition code = 8 if a single file is being processed.

DIBDB644 NO DISK SPACE AVAILABLE FOR CATALOG/DIRECTORY LINK FOR xxxxxx LIBRARY.

A request was issued for a half-track record needed for a catalog/directory link in the specified library; the response indicated no more disk space was available in the group data set(s). The group should be reorganized to return purged space to a usable state. Condition code = 8.

DIBDB645 NO DISK SPACE AVAILABLE FOR FILE xxxxxxxx IN aaannn LIBRARY.

A request for disk space needed for writing the named file indicated none was available in the group data set(s). The group should be reorganized to return purged space to a usable state. Condition code = 8.

Message: INPUT PROGRAMS EXCEED ALLOWABLE SIZE

Cause: The user has attempted to weave programs whose total input size exceeds twice the maximum single program size (2 x 28848, or 57696 bytes).

Note: If more than two programs were specified in the WEAVE command, it may be possible to perform the desired weaving of programs. The user can enter successive WEAVE commands, each of which specifies some of the programs named in the WEAVE command causing this error message to be generated.

Message: INVALID CHARACTER(S) IN FILE NAME

Cause: The filename supplied with a NAME, SAVE, STORE, or FILE command contains one or more invalid character(s).

Message: INVALID EDIT OPERATION AT LINE nnnnn

Cause: As a result of an edit operation being performed on line nnnnn, one of the following has occurred.

1. The original line is preceded by a five-digit line number, with a numeric character immediately following. This line is to be renumbered with less than a five-digit line number. (This can occur for MERGE, MOVE, or RENUMBER.)

2. The original line is preceded by a line number of less than five digits. An attempt has been made to place one or more numeric characters immediately following this line number. (This can occur for REPLACE.)

Message: INVALID LANGUAGE NAME

Cause: A language-processor name specified in an ENTER command is invalid.

Message: INVALID PAGE WIDTH

Cause: The width parameter specified in a WIDTH command to control printing of listed or runtime output is less than 18, greater than 255, contains more than three digits, or contains nonnumeric characters.

Message: INVALID RANGE SPECIFICATION

Cause: A parameter entry of the form line-number THRU line-number is not specified correctly in an ADD, FIND, INSERT, MOVE, or REPLACE command.

Message: LANGUAGE NAME--

Cause: The user has typed the ENTER command and has failed to supply a valid language name.

Message: LINE NOISE, RETYPE

Cause: The system has detected a parity error in the data stream from the terminal. The input line is ignored.

Message: LINE TOO LONG, RETYPE

Cause: An input line exceeds the maximum length (237 characters if 2741; 238 characters if Teletype). The line is ignored.

Message: LINE nnnnn IS LONGER THAN MAX LINE LENGTH

Cause: A source-program line generated in response to an ADD, REPLACE, MOVE, MERGE, or RENUMBER command exceeds the permissible maximum line length for these commands (237 characters).

Message: LOST INPUT LINE(S), LIST YOUR PROGRAM

Cause: The CALL-OS system has detected loss of data as input is being entered from the terminal.

Message: MAIN PROGRAM RENUMBERED AT nnnnn

Cause: This message may be generated in either of two situations:

1. Renumbering of main-program lines is necessary during a merge operation, and none of the programs involved in the merge is protected. (If one or more of the programs is protected, this message is not provided.)

2. Renumbering of unmoved main-program lines is necessary during a move operation.

nnnnn is the line number of the main-program line at which renumbering begins.

Message: MORE THAN NINE PROGRAMS SPECIFIED

Cause: More than nine programs have been specified for a merge or weave operation.

Message: MOVE TO AND FROM AREAS OVERLAP

Cause: A MOVE command has requested the moving of a range of statements to a location within the specified range.

Message: NAME MATCHES SOURCE OR DATA FILE

Cause: The name of a program specified in a STORE command matches either a source-program name or a data filename in the user's catalog. The user should select a different name for the current program and reenter the STORE command.

Message: NEW LINE NUMBER TOO LOW

Cause: The user has entered a RENUMBER command in which the new line number (specified or by default) is so low that renumbering

would cause existing line numbers prior to the renumbering point to be overlapped.

Message: NO LINES MOVED

Cause: A MOVE command has been entered, but the move operation cannot be completed for either of two reasons.

- A single line number is specified to be moved, but that line does not exist in the program.
- A range of lines is indicated in the MOVE command, but no lines within the range exist in the program.

Message: NO LINES RENUMBERED

Cause: The old line number specified in a RENUMBER command is higher than all existing lines in the program.

Message: NO NAME SUPPLIED

Cause: A STORE command has been entered with no program-name parameter, and no source-program name is available to use under the default naming convention. The user should issue a NAME command to provide a source-program name or include a name in the STORE command to be assigned to the object program.

Message: NO PROGRAM LEFT

Cause: One or more source-program lines or a complete source program was loaded into the user work area or entered into it from a terminal. However, all lines have been deleted using a line-by-line delete. This message is printed when a terminal command such as RUN, LIST, or SAVE is entered in an attempt to perform some operation on the deleted program.

Message: NO PROGRAM PRESENT

Cause: No source-program lines have been entered or loaded into the user work area; that is, the area is clear. This message is printed when the user attempts to perform some operation such as RUN, LIST, or SAVE on a program in the user work area but no program is there.

Message: NO ROOM IN POOL STORAGE

Cause: The user has entered a POOL command and there is no room for a new entry in the required directory. The user should contact installation management to request more space or reorganization of existing space on the disk.

Message: NO ROOM IN SAVE STORAGE

Cause: An attempt has been made to save a file or create an output file when there is no room on the disk. The user should contact installation management to request more space or reorganization of the existing space on the disk.

Message: OBJ PROG STORED - xx STORAGE UNITS REQUIRED

Cause: A store operation has been completed, and xx disk storage units (each 3440 bytes in length) were required to store the object program.

Message:
```
OFF AT xxxx
PROC. TIME..  xxxx  SEC.
TERM. TIME..  xxxx  MIN.
```

Cause: A current user is being logged off system due to the entering of an OFF command or of a LOGON command by another user.

Message:
```
ON AT xxxx date system LINE xx
installation-determined message
USER NUMBER,PASSWORD--
```

Cause: A new user is being logged onto the system. This is the signal for him to supply his user number and password. The current date and system identification are printed out. Content of the installation-determined message is established by the command console operator for each execution of CALL-OS.

Message: PAGE WIDTH--

Cause: The user has entered the WIDTH command and has not provided the width parameter.

Message: PASSWORD--

Cause: The user has entered the PASSWORD command without supplying a password. (A second refusal to supply a password is interpreted as a desire to retain the old password and the system responds READY.)

Message: PLEASE CONTACT YOUR INSTALLATION MANAGEMENT FOR ASSISTANCE.

Cause: The user has entered a HELP command. This is the default CALL-OS system response to the command. The response can be changed by central computer installation personnel. (See the discussion of the HELP command.)

Message: POOLING DATA SET(S) NOT ACTIVE

Cause: The user has entered a LOAD **program-name command, and the data set(s) of the user who pooled the file named in the command have not been made available for the current execution of CALL-OS.

Message:
```
PROCESSOR TIME MONTH-TO-DATE..xxx  SEC.
TERMINAL TIME MONTH-TO-DATE..xxx  MIN.
DISK STORAGE MONTH-TO-DATE..xxx  UNIT(S)
```

Cause: This is the normal system response to entry of the TIME command.

Message: PROGRAM EXCEEDS AVAILABLE MEMORY

Cause: A program requires more than the maximum number of storage locations available for an object program, as specified by the user installation at initialization time for the current session of CALL-OS. This installation-specified maximum is less than the system maximum, which is 114,688 bytes.

Message: PROGRAM EXCEEDS MAXIMUM MEMORY

Cause: A program requires more than the system-defined maximum number of storage locations that can be allocated (total program area) for an object program. This maximum is 114,688 bytes.

Message: PROGRAM EXCEEDS 800 LINES OR 28848 BYTES

Cause: This message may be generated in any of several situations.

1. The source program resulting from one of the following operations is too large: ADD, INSERT, MERGE, MOVE, RENUMBER, REPLACE, or WEAVE. (The message may be generated immediately or when a subsequent operation is to be performed.)

2. The user has caused the maximum source-program size to be exceeded by typing line-numbered entries (source-program lines) at his terminal.

Message: PROGRAM LINES LOST, LIST YOUR PROGRAM

Cause: Certain CALL-OS terminal commands, such as a LIST command entered after source-program lines have been inserted, cause a sort to take place. This message is printed to indicate that one or more source-program lines were deleted during the sort process. The user should list his program.

Message: PROGRAMS NOT ALL SAME LANGUAGE TYPE

Cause: The user has entered a MERGE or WEAVE command in which the named programs are written in different source languages.

Message: PROTECTED PROGRAM

Cause: A requested function cannot be performed because one or more of the programs involved is currently protected (that is, assigned RUN ONLY status).

Message: READY

Cause: This is the normal reply to most terminal commands. It indicates that the command process has been completed.

Message: REPLACED nnnnn

Cause: This message (followed by READY) is the normal response to the REPLACE command. The field nnnnn provides a count of the number of successful replacements or deletions of the character string specified in the command.

Message: SOURCE NAME HAS 8 CHARACTERS

Cause: A STORE command has been entered with no program-name parameter, and a slash cannot be appended to the current program name without causing truncation. The user should issue a NAME command to provide a new source-program name or include a name in the STORE command to be assigned to the object program.

Message: SPACE NOT AVAILABLE TO STORE PROGRAM

Cause: An attempt has been made to store an object program, but insufficient space exists in the user-group data set. The user should contact installation management to request more space or reorganization of the existing space on the disk.

Message: STOP.
RAN xxx SECS.

Cause: The user has interrupted execution of his program by pressing the ATTN key (2741) or BREAK key (Teletype).

Message: STOP.
READY.

Cause: The user has interrupted system operation by pressing the ATTN key (2741) or BREAK key (Teletype) in a situation other than during execution of his program.

Message: SYSTEM PROBLEM, RETRY

Cause: The system has encountered a problem during processing. The user should retry his process. If the problem persists, he should contact installation management.

Message: SYSTEM PROBLEM
USER NUMBER,PASSWORD--

Cause: The system has encountered a problem during the sign-on process. The user should reenter his user number and password.

Message: TEMPORARY MALFUNCTION - TRY AGAIN

Cause: An irrecoverable error has occurred while the compiler or user program was being executed. If the error persists, the user should contact installation management.

Message: TERMINAL NO..xx
USER NO..xxxxxx
PGM NAME..xxxxxxxx
PGM LENGTH..xxxx CHARS.,xxx LINES
LANGUAGE..xxxxxxx
TERMINAL TIME..xx MIN
PROCESSOR TIME..xx SEC

Cause: This is the normal response to a STATUS command.

DIBDB909 ERROR - NO EQUIVALENCY ENTRY FOR USER aaannn.

The user specified in the COBI index record is not a validated user in its user group equivalency file. Condition code = 4.

DIBUT202 "ERRNO" NOT BETWEEN 220 AND 229.

The ERRNO parameter supplied is invalid. The value must be between 220 and 229. Correct the parameter and restart the job.

DIBUT203 "SVC" NUMBER NOT BETWEEN 200 AND 225.

The SVC number supplied is invalid. The value must be between 200 and 225. Correct the parameter and restart the job.

DIBUT204 "NUC" FIELD ERROR.

The NUC parameter is invalid. Both the from and to nucleus values must be between 1 and 9. Correct the parameter and restart the job.

DIBUT205 "TYPE" FIELD ERROR.

The TYPE parameter is invalid. The value must be either MFT or MVT only. Correct the parameter and restart the job.

DIBUT206 "COBI" FIELD ERROR.

The COBI parameter is invalid. The value must be either YES or NO. Correct the parameter and restart the job.

DIBUT500 MORE THAN ONE EXTENT IS NOT ALLOWED FOR DDNAME dddddddd

An old data set was defined on the specified DD statement (dddddddd) and this data set contained more than one extent. A user ABEND 507 is issued and processing terminates. Allocate sufficient space in one extent for the data set.

DIBUT501 SPLIT CYLINDER ALLOCATION SHOULD NOT BE SPECIFIED FOR DDNAME dddddddd

The data set specified by the DD statement with the ddname dddddddd was allocated by use of the SPLIT parameter. Split cylinder data sets cannot be used for the COBI index, JCL data set, or SYSIN data sets. A user ABEND 507 is issued and processing terminates. Allocate these data sets by use of the SPACE parameter or SUBALLOC parameter.

DIBUT502 SECONDARY ALLOCATION SHOULD NOT BE SPECIFIED FOR DDNAME dddddddd

The specified DD statement (dddddddd) contains a secondary allocation in the SPACE parameter. A user ABEND 507 is issued and processing terminates. Correct the SPACE parameter and resubmit the job.

DIBUT503 DATA SET FOR DDNAME dddddddd FOUND TO BE NOT VACUOUS

The data set defined by the specified DD statement (dddddddd) contains one or more records but the utility only formats data sets which are completely empty. A user ABEND 507 is issued and processing terminates. Scratch the data set and ensure that it is empty; resubmit the job.

DIBUT504 BAD OPEN FOR DATA SET, DDNAME ddddddd

It was impossible to open ddddddd. A user ABEND 507 is issued and processing terminates. Either supply the missing DD statement or correct the existing one; resubmit the job. If the DD statement is correct, notify the installation system programmer immediately.

DIBUT507 CYLINDER ALLOCATION NOT SPECIFIED FOR DDNAME CBSYSINx

The DD statement for the specified COBI input data set (where x is either A or B) did not contain CYL specification in the SPACE parameter. A user ABEND 507 is issued and processing terminates. Correct the SPACE parameter and resubmit the job.

DIBUT508 UNEQUAL SPACE ALLOCATION FOR CBSYSINA AND CBSYSINB

The two COBI input data sets must be identical. A user ABEND 507 is issued and processing terminates. Check the DD statements for both data sets to ensure that CYL and CONTIG are specified in the SPACE parameter for both and that the number of cylinders specified is the same for both.

DIBUT509 THE U#5xxxxx UTILITY FUNCTION HAS COMPLETED SUCCESSFULLY

This is an informative message, indicating that the specified utility function (where xxxxx is INIT, CBXPN, RINIT, or PURGE) has been completed successfully. Condition code = 0 unless message DIBUT529 is issued during execution of U#5PURGE; in this case, condition code = 8.

DIBUT517 INVALID PARAMETER SPECIFIED ON THE EXEC STATEMENT

A parameter on the EXEC statement could not be recognized. Ensure that all parameters are spelled correctly. A user ABEND 507 is issued and processing terminates.

DIBUT518 NUMERIC VALUE SPECIFIED FOR A PARAMETER IS TOO LONG

The value specified for a parameter has more than four numeric digits. A user ABEND 507 is issued and processing terminates.

DIBUT519 INVALID CHARACTER WITHIN OR ENDING A NUMERIC VALUE

The value specified for a parameter either contains a nonnumeric character within it or, for all but the last one in the parameter list, ends with a character other than a comma. A user ABEND 507 is issued and processing terminates.

DIBUT520 DUPLICATE PARAMETERS SPECIFIED ON EXEC STATEMENT

The EXEC statement contains two or more duplicate parameters. A user ABEND 507 is issued and processing terminates.

DIBUT522 ZERO VALUE FOUND FOR ONE OR MORE SPECIFIED PARAMETERS

A zero was found when a numeric parameter value was expected. A user ABEND 507 is issued and processing terminates.

COBI-GENERATED SYSTEM MESSAGES

Communication between CALL-OS terminal users and the CALL-OS Batch Interface (COBI) option is achieved by means of the terminal commands and control statements described under "COBI Terminal Commands and Control Statements". Communication from COBI to terminal users occurs as system messages, which may be generated at any of several points during system processing. Some messages are generated because of unusual or current, temporary conditions within the system (for example, background using data set). Others are due to user errors. In most cases, detection of a special condition or of the first serious user error causes a message to be printed, and processing of the current job to be terminated. Other messages provide for system/user communication during normal system operation.

The text of all COBI-provided messages, together with explanations of them, are listed in alphabetic order below. The user may also receive system messages not unique to COBI. Such messages are documented under "System Messages", the preceding section of this manual.

Message: BACKGROUND USING DATA SET

Cause: An attempt has been made to access a data set currently being used in an exclusive mode (for example, DISP=OLD has been entered in the corresponding DD statement) by an OS background job. The data set cannot be accessed until the background job has terminated.

Message: CLEAR OR SAVE BEFORE VAR RECORD SCAN

Cause: The user has requested scanning of a data set containing variable-length records. The COBI scan routine requires more space than is currently available in the user disk work/swap area. The user must save the program in that area (if he wishes to do so) and then clear the area. Then he can reenter his SCAN request.

Message: DATA SET CLOSED DUE TO SYSTEM PROBLEM

Cause: The user has entered a request to scan a data set, but the system has encountered a problem when attempting to respond to his request. The data set cannot be scanned. If the problem persists, the user should contact installation management. It is possible that the data set characteristics in the data set control block do not reflect the actual characteristics of the data set to be scanned.

Message: DATA SET EMPTY OR LACKS END OF FILE

Cause: The COBI system has detected that a data set to be scanned contains no data or has no end-of-file marker. The data set is not scannable. The lack of an end of file in a data set is determined by a test for zero in the last block pointer of the data set control block (DSCB) for the data set.

Message: DATA SET NOT CATALOGED

Cause: An attempt has been made to scan a data set by specifying its data set name, without indicating the volume on which the data set can be found. COBI has searched for the data set in the OS catalog, but the data set name was not found. The user should reenter the SCAN command, including the VOL=volid parameter entry.

Message: DATA SET NOT READABLE BY USER

Cause: The user has entered a request to scan a particular data set, but the data set name does not contain, as one of its qualifiers, either the user number by which this user signed on the system or an installation-specified high-level qualifier.

Message: DATA SET NUMBER NOT IN COBI INDEX RECORD

Cause: The user has requested scanning of a data set for which no pointer exists in the COBI index record for the job. Possible causes are:

1. The user did not specify this data set identifier as an option in the SUBMIT command for the job.
2. The data set has been scratched previously.
3. The data set was not written or kept by an executing program of the job.

Message: ENTER (OPTION) OR CR

Cause: This message is printed to point out to the terminal user that he has not specified options in his SUBMIT command. The user may respond by entering options or pressing the RETURN key. The latter response indicates that no options are to be specified. If the user does not want to retain any data sets but wants to avoid printout of this message, he can specify NONE or its abbreviated form N as an options entry in his SUBMIT command.

Message: FIELD EXCEEDS 8 CHARACTERS

Cause: An operand of a SUBMIT command contains more than eight characters. Either a program name has been specified incorrectly or one of the options is in error.

Message: ILLEGAL SYNTAX FOR WTO MESSAGE - LINE nnnnn

Cause: A comment statement containing a write-to-operator (WTO) message is not formatted correctly. nnnnn is the line number preceding the statement, as originally entered by the user.

Message: ILLEGAL $$TABS CONTROL STATEMENT - LINE nnnnn - PROGRAM pppppppp

Cause: An incorrectly formatted $$TABS control statement has been encountered in the input job stream. nnnnn is the line number preceding the statement, as originally entered by the user. The user should load the program file in error

(identified by the program name pppppppp), correct the $$TABS statement, save the program, and reenter the SUBMIT command causing the file to be read.

Message: ILLEGAL TAB CHARACTER - LINE nnnnn - PROGRAM pppppppp

Cause: A program statement (preceded by line number nnnnn in program pppppppp) is being expanded under tab control. In this process, a tab has been encountered beyond the last tab position specified in the $$TABS control statement controlling tabulation. The user should either correct the tab setting or change the $$TABS statement in the program file.

Message: INCOMPATIBLE PARAMETERS

Cause: Parameters specified in an initial SCAN command, or in an option pertaining to a data set currently being scanned are in conflict. For example, the user has specified both job-number and MEM=memname parameter entries in an initial SCAN command, or he has specified a MEM=memname parameter entry when the data set being scanned is a sequential data set.

Message: INDEX RECORD IN USE, RETRY

Cause: The user has entered a SCAN, SCRATCH, or CANCEL command, but the requested operation cannot be completed immediately for either of the following reasons:

1. Another user, signed on with the same user number, has caused access to the COBI index record for the specified job. (Thus, for example, a job cannot be cancelled while the system is responding to a prior SCAN, SCRATCH, or CANCEL command for that job.)

2. The storage space available for enqueuing of COBI-submitted job names has been exceeded.

In either case, the user should reenter his command because the condition is temporary and will soon be eradicated.

Message: INDIRECT SUBMIT FOUND IN INDIRECT SUBMIT

Cause: The user has entered a SUBMIT command containing an up-arrow, thus signifying an indirect submit. The $$SUBMIT control statement in the referenced program file is also an indirect submit. This is not permissible. The user should load the program file in error, correct the $$SUBMIT control statement, save the program, and reenter the SUBMIT command.

Message: INSUFFICIENT PARAMETERS

Cause: The user has failed to specify all necessary parameters in a SCAN command. For example, he has entered a job number and specified an option, but he has failed to identify the data set to be scanned.

Message: INVALID DATA SET ID

Cause: A data set identifier specified in a SCAN or SCRATCH command is not acceptable. This message may be printed for any of the following reasons.

1. A user SYSOUT data set number is not within the range from 1 through 127.
2. A procedure-defined SYSOUT data set number is not within the range from 1 through 15.
3. A procedure number specified in a procedure-defined SYSOUT data set identifier (of the form nPmm where n is the procedure number and mm is the data set number) is not within the range from 1 through 7.

Message: INVALID JOB NUMBER

Cause:
1. The user has entered a job number of the form #nnnnn, but that number is not within the range of job numbers established as valid for this execution of CALL-OS.
2. The user has attempted to cancel or scratch a job or to scan the JCL or a SYSOUT data set of a job, but the specified job number is not assigned to any of this user's jobs.

Message: INVALID SCAN ENTRY

Cause:
1. A syntax error has been detected in processing of a SCAN command.
2. Only certain requests are acceptable when the user is in SCAN mode. This message is generated when the user enters a request for an operation that cannot be performed until the scan of the current data set has been completed and the SCAN mode removed. (See "SCAN".)

Message: INVALID SCRATCH REQUEST

Cause: A SCRATCH command has been issued following a scan operation but (1) the data set identified in the SCRATCH command is not a data set of the current job, or (2) no parameter is specified in a SCRATCH command after the data set being scanned has already been kept or scratched. The user must exercise his option to keep or scratch data sets of the current job before he can deal with other data sets.

Message: INVALID USERID

Cause: The user has requested that a data set be scratched. However, the job to which the data set is attached was not originally submitted to OS batch processing under the user number by which this user signed on the system. Only the user who enters a job or the command console operator can cause data sets retained for scanning to be scratched.

Following the error message, the contents of the stack which caused the message is printed. Conversion of this procedure terminates; processing continues with the next procedure. Condition code = 4.

DIBUT617 INVALID PARAMETER SPECIFIED ON THE EXEC STATEMENT.

The parameter field of the EXEC statement for DIBCONPR can contain only the SPACE, OSCLASS, and CBCLASS parameters. No other parameters may appear. Either a parameter is misspelled or a parameter other than these three is specified. Execution terminates. Correct the parameter field and restart the job. Condition code = 8.

DIBUT619 PARAMETER VALUE SPECIFIED ON THE EXEC STATEMENT IS INVALID OR TOO LONG.

The OSCLASS and CBCLASS parameters must specify valid OS/360 output classes. The SPACE parameter must specify a valid space allocation value in 50 characters or less. Execution terminates. Correct the parameter in error and restart the job. Condition code = 8.

DIBUT620 DUPLICATE PARAMETERS SPECIFIED ON EXEC STATEMENT.

The parameter field of the EXEC statement for DIBCONPR contains two identical parameters. Execution terminates. Correct the parameter field and restart the job. Condition code = 8.

DIBUT634 PROCEDURE xxxxxxxx REQUIRED NO MODIFICATION

The procedure xxxxxxxx contained no DD statement with the 'SYSOUT=a' parameter, where a is the output class specified in the OSCLASS parameter of the DIBCONPR EXEC statement. Since no conversion is required for this procedure, this message is printed and the procedure is not output. Condition code = 0.

DIBUT635 SYSOUT UNIT EXCEEDED IN ABOVE PROCEDURE

The procedure printed above this message contained more than 15 DD statements with the 'SYSOUT=a' parameter, where a is the output class specified in the OSCLASS parameter of the DIBCONPR EXEC statement. Only the first 15 such DD statements are modified in the output procedure. Condition code = 0.

DIBUT650 HIGHEST CONDITION CODE WAS c.

This is an informative message issued before DIBCONPR terminates execution. It indicates the highest condition code encountered during execution. A condition code of 0 indicates successful execution.

DIBUT651 END OF CALL-OS BATCH INTERFACE UTILITY RUN.

This is an informative message issued when DIBCONPR terminates execution.

*** DSNAME MATCHES DSNAME IN INDEX ***

This message is issued by U#UTIL1; it indicates that the dsname specified on the ddcard matches a dsname on an entry in the index. This diagnostic does not appear on a COMPILER run if the index entry with the matching dsname is a compiler entry with a ddname which matches the ddcard's ddname.

*** FIELD n INVALID - NO UPDATE ATTEMPTED ***

This message is issued by UTILX; it means that the indicated field on the previous detail card has a syntax error, or contains information which is not of the correct form, or is inconsistent with the other card fields or with the present index entries.

*** FORMATTING - BAD OPEN ***

This message is issued by U#UTIL1 during a SYSGROUP or USRGROUP run; the message indicates that OS/360 was unable to open for output the specified data set in order to format it.

*** FORMATTING - DATA SET TOO SMALL ***

This message is issued by U#UTIL1; it can occur on a SYSGROUP or USRGROUP run if the data set specified is the first data set in its group, and it does not have at least three or two tracks, respectively.

*** FORMATTING - I/O ERROR IN WRITING TRACKS ***

This message is issued by U#UTIL1; it indicates that a permanent I/O error occurred while attempting to format a system group or user group data set. The disk pack is possibly bad.

*** FORM OF DDNAME IS NOT CORRECT ***

This message is issued by U#UTIL1. During a WORKSWAP run, this indicates that the first four characters of the ddname on the ddcard being processed are not SWAP, or that the fifth and sixth characters are not decimal digits, or that the ddname is longer than six nonblank characters. During a SYSGROUP run, it indicates that the first six characters are not SYSGRP or the last two characters are not decimal digits. During a USRGROUP run, it indicates that the first six characters are not alphabetic or the last two are not decimal digits. During an OVERLAY run, it indicates that the ddname is not OVLY.

*** INDEX FULL ***

This message is issued by U#UTIL1; it indicates that the CALL-OS index data set is full and hence the data set whose ddname appears above this message cannot be added to the CALL-OS data base. UTILX may be used to delete one or more entries from the index in order to make room for new entries.

*** INVALID USER GROUP - FIRST 3 > SECOND 3 ***

This message is issued by U#UTIL1 and may occur during a USRGROUP run. It indicates that the ddname (on the ddcard being processed) does not represent a valid user group, since the first three characters are greater (in collating sequence) than the second three.

*** INVALID USER GROUP - OVERLAPS USER GROUP IN INDEX ***

This message is issued by U#UTIL1 and may occur during a USRGROUP run. It indicates that the ddname on the ddcard being processed represents an invalid user group, since its range of user ID characters overlaps the range of a user group in the same cluster in

Message: MEMBER NAME--

Cause: The user has entered a SCAN command containing a DSN parameter entry to request scanning of a partitioned data set, but he has not identified the member of the data set to be scanned. This message is printed to prompt the user to enter a member name. Only the member name should be entered in response to the prompt. All other parameters of the SCAN command apply as formerly specified.

Message: MEMBER NOT FOUND, MEMBER--

Cause: This message may be generated in either of two cases.

- The directory of a partitioned data set specified in a SCAN command does not contain a requested member name.
- The user has entered information other than, or in addition to, member name in response to the prompt message MEMBER NAME--.

the index. The index entry information for the conflicting entry is printed above this diagnostic.

*** MATCHING DDNAME IN INDEX OR ON PREVIOUS DDCARD ***

This message is issued by U#UTIL1. If index entry information appears with this message, it indicates that the ddname on the ddcard being processed matches the ddname in the index entry. This form of the message occurs only if the run is not of the COMPILER or OVERLAY type and if the index entry with the matching ddname is of the type (System Group, User Group) which is being processed by the run. If no index entry information appears with this message, it indicates that the ddname on the ddcard being processed matches the ddname on a ddcard processed previously in the same run.

** NO ADD OR DEL FUNCTION SPECIFIED ***

This message is issued by UTILX; it indicates that no function control card (which must specify either ADD or DEL) has yet been encountered.

*** NOT ENOUGH FIELDS FOR FUNCTION SPECIFIED ***

This message is issued by UTILX. If the DEL function has been specified, this message indicates that this function could not be performed for the previous detail card because the first and fourth fields were not given. If the ADD function has been specified, this message appears if any of the six fields have been omitted.

*** NOT NEXT RELATIVE DSNUMBER ***

This message is issued by U#UTIL1 and may occur during a WORK/SWAP, SYSGROUP, or USRGROUP run. It indicates that the data set number (specified by the last two characters in the ddname on the ddcard being processed) is more than one greater than the highest data set number recorded in the index for that particular work/swap, system group, or user group (in the appropriate cluster). Since data sets can only be added in sequence with each group and cluster, UTILX should be used to obtain a listing of the index in order to determine which data set number may be added next.

*** RELATIVE DSNUMBER OVER IMPLEMENTATION LIMITS ***

This message is issued by U#UTIL1. During a WORK/SWAP run, this message indicates that the data set number (specified by the last two characters in the ddname on the ddcard being processed) is 20 or more. During a SYSGROUP or USRGROUP run, it indicates that the data set number is 80 or more. Only data set numbers of 0 to 19 are allowed for work/swap data sets, and system group and user group data sets may use only data set numbers 0 to 39 for the primary cluster and 40 to 79 for the alternate cluster.

*** THERE WAS AN ERROR IN THE PARM FIELD ON THE EXEC CARD. SEE THE
CONSOLE LISTING FOR CORRECTION BY THE OPERATOR. ***

This message is issued by U#UTIL1; it indicates that the PARM field on the EXEC card was not correct; that is, was not COMPILER, SYSGROUP, USRGROUP, WORK/SWAP, or OVERLAY. An error message to this effect is also printed on the operator's console, and the operator is given a chance to enter a correct parameter.

*** UPDATE IMPOSSIBLE - INDEX FULL OR ENTRY NOT FOUND ***

This message is issued by UTILX. If the ADD function has been specified, this message indicates that this function could not be performed for the previous detail card because the index is full. If the DEL function has been specified, this message indicates that this function could not be performed for the previous detail card because there was no corresponding entry in the index.

*** VALIDATING - BAD I/O READING HOME ADDRESSES ***

This message is issued by U#UTIL1. During a COMPILER or WORKSWAP run, this message indicates that an I/O error condition occurred while attempting to read the home addresses on tracks to see if alternate tracks have been assigned.

*** VALIDATING - BAD OPEN ***

This message is issued by U#UTIL1. During a COMPILER or WORKSWAP run, this message indicates that OS/360 was unable to open (for input) the specified data set in order to check for flagged tracks and multiple extents.

*** VALIDATING - DATA SET HAS IMPROPER BOUNDARIES ***

This message is issued by U#UTIL1. On a COMPILER run, this message means that the data set (specified by the DD card being processed) crosses a cylinder boundary. On a WORKSWAP run, this message means that the data set does not begin and end at cylinder boundaries.

*** VALIDATING - FLAGGED TRKS FOUND ***

This message is issued by U#UTIL1. During a COMPILER or WORKSWAP run, this message indicates that bad tracks (for which alternate tracks have been assigned) have been found in the space allocated to the data set being processed. Since compiler and work/swap data sets cannot use alternate tracks, a different space, free from defective tracks, must be allocated for the data set.

*** VALIDATING - MULTIPLE EXTENTS FOUND ***

This message is issued by U#UTIL1. During a COMPILER or WORKSWAP run, this message indicates that the data set specified has had more than one extent allocated. Compiler and work/swap data sets cannot have multiple extents. To avoid this situation, the DD card used to create the data set should specify the CONTIG subparameter of the SPACE parameter. The RLSE subparameter is also recommended.

*** VALIDATING - MULTIPLE VOLUME DATA SET FOUND ***

This message is issued by U#UTIL1 and can occur on any type of run. It indicates that the data set specified does not reside on a single volume.

valid only when the user is in SCAN mode, the KEEP request is rejected.

Message: NOT SCRATCHED: xxxx c, xxxx c,...

Cause: The user has requested the scratching of one or all data sets for a job, but his request cannot be satisfied because of conditions encountered during processing. Each xxxx is an entry of the form nPmm or Unnn identifying a data set that cannot be scratched. Each c is a code indicating why a specified data set cannot be scratched. Possible code values and their meanings are:

1 - Background using data set
2 - System problem
3 - Data set not on volume indicated
4 - Data set not in COBI index
5 - Volume not in volume table
6 - System problem of an I/O nature
7 - Volume not available

The user should verify that he has entered his information correctly. If he has, but the error persists, he should contact installation management.

Message: OS OPEN FAILURE

Cause: The user has requested scanning of an OS/360 data set, but the scan operation cannot be completed because of conditions such as the following:

- The required data set is not on the volume indicated.
- The OS/360 data set name is invalid.
- One of several OS/360 miscellaneous situations has occurred.

If the error persists, the user should contact installation management.

Message: PARAMETER, xxxxxx.., INVALID ENTRY

Cause: A job-number entry of a DSSTATUS or JOBSTATUS command cannot be processed successfully. It contains other than numeric characters or no characters other than #. If subsequent job numbers are specified in the command, processing continues with those jobs.

Message: PROCEDURE SYSOUT NUMBER GREATER THAN 15

Cause: A procedure-defined SYSOUT data set number (mm in a parameter entry of the form nPmm) greater than 15 has been specified in a SUBMIT command. Since a valid procedure-defined SYSOUT data set number cannot exceed 15, the user must reenter the SUBMIT command correctly.

Message: PROGRAM NOT IN CATALOG, OR NOT A PROGRAM FILE - program-name

Cause: An entry specified as a program name in a SUBMIT command cannot be found in the user's catalog or does not refer to a source-program file. The name is printed as part of the message to identify the file that cannot be submitted. None of the files named in the SUBMIT command are routed to OS/360; that is, the complete job is rejected. The user must ensure that all required program files are saved in his library and reenter the SUBMIT command.

Note: Programs cannot be submitted from CALL-OS shared libraries. To submit a program in one of these libraries, the user can load the program, save it in his library, and then submit it. After the submit process has been completed, he can purge the program from his library.

Message: SCAN MODE REMOVED, BUT INDEX NOT UPDATED

Cause: The user has entered a request to KEEP or SCRATCH one or all data sets of the job currently being scanned. The system is unable to complete the operation, and the COBI index record for the job is not changed. This message is printed to alert the user to this condition. The SCAN mode is removed, and he can request other operations if he desires.

Message: SUBMIT COMMAND DISABLED

Cause: The user has entered a SUBMIT command, but no job can be submitted to OS/360 at this time. The submit process has been inhibited by operator action at the CALL-OS command console. The user should reenter his SUBMIT command at a later time.

Message: SUBMITTED AS job-name

Cause: A SUBMIT command has been accepted as valid, and the job is being submitted to OS/360. However, no job number need be assigned to this job, because the user has specified a message class other than the one assigned to COBI jobs (at system initialization time) in his JCL for the job and has not specified any options in the SUBMIT command for the job. The name by which OS/360 recognizes the job (job-name) may be either of the following:

- the user's job name from his JOB statement if the ANYJNAME parameter is specified in the system startup deck
- aaannnxx where aaannn is the user number and xx is a system-generated identifier if ANYJNAME is not specified in the system startup deck

Message: SUBMITTED JOB DOES NOT BEGIN WITH REQUIRED JCL

Cause: The initial JCL for a job to be submitted to OS/360 includes job control statements other than those shown below.

```
$$SUBMIT          (required if indirect submit)
$$TAB             (required if tab characters are used and
                  must precede all lines to which it applies
//*               (up to four will be routed by COBI)
  .                          .
  .                          .
//          JOB   (one is required for each job)
  .                          .
  .                          .
```

Message: SYNTAX ERROR

Cause: The user has entered a CANCEL, DSSTATUS, JOBSTATUS, NOTIFY, SCRATCH, or SUBMIT command containing a syntax error. He should reenter the command correctly.

Message: SYSOUT NUMBER LESS THAN 1

Cause: A SYSOUT data set number less than 1 has been specified in a SUBMIT command. Since a valid data set number cannot be less than 1, the user must reenter the SUBMIT command correctly.

Message: TERMINATE PREVIOUS SCAN DSN= REQUEST

Cause: The user has scanned an OS/360 data set, which he identified by a SCAN command parameter entry of the form DSN=xxx.yyy.zzz. He must enter the KEEP option in response to the system-generated question mark printed out at the terminal after his scanning requirements for this data set have been satisfied.

Message: TOTAL LENGTH OF INDIRECT SUBMIT EXCEEDS 244 CHARACTERS

Cause: The length of a $$SUBMIT control statement in the user's input job stream exceeds the maximum number of characters permissible in a $$SUBMIT control statement.

Message: UNSUPPORTED DATA SET

Cause: An attempt has been made to scan a data set that:

- is not defined as either a sequential or a partitioned data set
- is not formatted as fixed, fixed blocked, fixed blocked standard, variable, or variable blocked
- is currently protected under an OS/360 password
- has a specified block size such that more than 127 physical blocks would reside on one track
- is keyed or has split cylinder allocation

Message: USER ID aaannn HAS NO COBI INDEX RECORD

Cause:

1. The user has entered a DSSTATUS command, but either he has no current COBI jobs, or none of the jobs which he has submitted have been completed.

2. The user has entered a JOBSTATUS command, but he has no current COBI jobs.

Message: USER SYSOUT NUMBER GREATER THAN 127

Cause: A user-defined SYSOUT data set number (nnn in a parameter entry of the form Unnn) greater than 127 has been specified in a SUBMIT command. Since a valid user-defined SYSOUT data set number cannot exceed 127, the user must reenter the SUBMIT command correctly.

Message: VOLUME NOT AVAILABLE

Cause: The user has entered a SCAN command referencing a data set on a volume which is not currently available. Among possible causes are failure to mount the required volume or to include a DD statement for the volume in the CALL-OS system startup deck. The user should contact installation management.

Message: 2ND OS/360 JOB CARD ENCOUNTERED

Cause: The user has entered more than one JOB statement in input processed as a result of one SUBMIT command. Since only one JOB statement is allowed per submittal, the input job stream is in error. The user must correct the input and resubmit his job.

Message: &U STATEMENT HAS ERRONEOUS DATA - LINE nnnnn

Cause: The DD statement for a user-defined SYSOUT data set contains a syntax error. For example, information does not end prior to or at position 70, or the &U parameter entry is not followed by a valid integer.

Message: #nnnnn SUBMITTED AS job-name

Cause: A SUBMIT command has been accepted as valid, and the job is being submitted to OS/360. This message is printed at the terminal to inform the user of the job number assigned to his job and the name by which OS/360 recognizes the job. The latter (job-name) may be either of the following:

- the user's job name from his JOB statement if the ANYJNAME parameter is specified in the system startup deck
- aaannnxx where aaannn is user number and xx is a system-generated identifier if ANYJNAME is not specified in the system startup deck

Option	Explanation
ACTIME=nnn	Specifies the number of minutes that is to elapse between accounting checkpoints. The default is 30 minutes.
ANYJNAME	Specifies that user-supplied job names are to be used for all COBI jobs. If COBI is not used, this option is ignored if present; if COBI is used and the option is omitted, COBI assigns a name to every job submitted.
AUTRDR	Specifies that COBI is to start a reader for the COBI input data sets when necessary. If COBI is not used, this option is ignored if present; if COBI is used and the option is omitted, the operator must start a reader when instructed by COBI. (See "Starting a Reader" in the chapter "Online Operations".) The operator may override the mode of starting a reader with the *COBI RESET, AUTRDR command as described in the chapter "Operator Command Language".
CBCLASS=z	Specifies the output class to be used for COBI and must be a valid output class (0 through 9 or A through Z). If CALL-OS is not executing and COBI jobs must still be processed, the operator starts DIBWTR, which processes this class. (See "Starting a Writer" in the Introduction). If COBI is not used, this option is ignored if present; if COBI is used and the option is omitted, the default is output class Z. If the operator displays the job queue, COBI jobs appear in this class. Information in this class is switched to another class (specified by the OSCLASS parameter) for processing by an OS output writer. If the operator attempts to change the COBI class to the OS class or starts an OS writer for the COBI class, the output will be lost.
COMCON=nnn	Specifies the logical line number of the communications console for CALL-OS. The default is logical line number 2. If COMCON=0 is specified, all messages intended for the communications console appear on the OS system operator's console preceded by the identifier DIBEX001.
COMTSL=(nnn,name)	Specifies the time slice in seconds to be allotted to the indicated compiler or compiler phase. The minimum is one second and the maximum is ten seconds. The default is one second for BASIC, and three seconds each for FORTRAN, PL/I, and PL/2. Note: These times are for the IBM System/360, Model 50.

DFLINK=nnn

Specifies the maximum number of half tracks to be allocated to each data file; the number must be within the range 4 through 100. The default is 100 half tracks.

DSPACE=(ppp,sss)

Specifies the primary and secondary allocation in tracks for the scannable output data sets created by COBI jobs. If COBI is not used, this option is ignored if present; if COBI is used and the option is omitted or a zero value is specified, the default is ten tracks for the primary and ten for the secondary.

IPBUFS=nnn

Specifies the number of 24-byte buffers (pots) to be allocated per line. The default is four pots per line with a minimum total of 60 pots in the system. The maximum is 15 pots per line, or the default total of 60 if the number of lines is less than or equal to four.

LCSRES=aaaaaaaa

Specifies which portions of CALL-OS system are to be loaded into Hierarchy 1 storage. Up to nine different portions may be indicated by letter; the allowable letters are:

- B - 256-byte buffers
- C - compiler area
- D - data control blocks (DCB)
- G - COBI tables and bit strings
- J - new and old job areas
- O - overlay buffer
- P - pots (24-byte buffers) and terminal translate tables
- S - sort buffer
- U - user terminal tables (UTT)

MAXDCB=nn

Specifies the maximum number of users permitted to scan data sets at the same time when COBI is used. If COBI is not used, this option is ignored if present; if COBI is used and the option is omitted, the default is one user for every ten lines with a minimum of two users.

NOCOBI

Specifies that COBI is not to be active for this session of CALL-OS.

NOSORT

Specifies that the sort buffer is not to be present in the system.

NOTRACE

Specifies that the trace table, if one is present in the OS/360 system, is not to contain entries for CALL-OS.

the second position for one original and three carbon copies, and to the third position for one original and five or more copies. (See also the discussion of the impression-control lever under "Ribbon/Print Element Carrier".)

The paper bail is the rod that holds the paper against the platen. It must be moved away from the platen while changing forms and may be set in the forward position to avoid carbon streaks when multiple-part forms are being used.

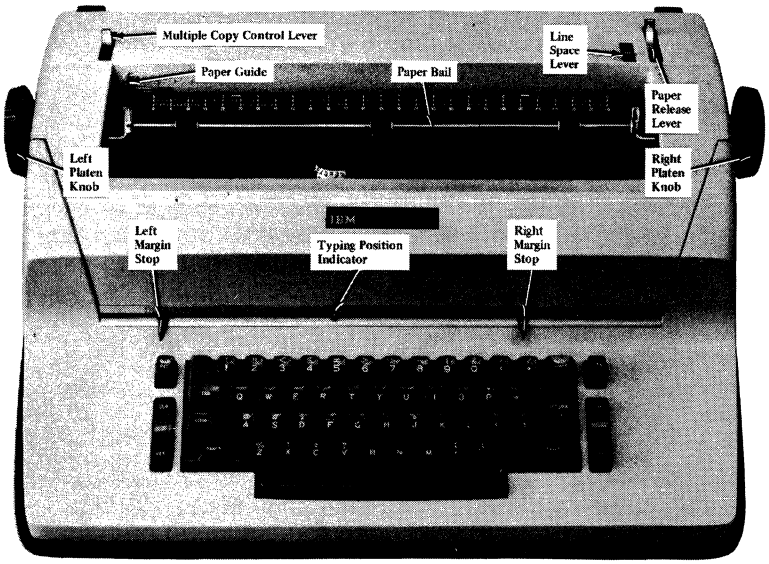

Figure 5. 2741 paper insertion and movement controls

The paper guide is an aid to paper alignment and may be freely moved to either the right or left.

The paper-release lever is set forward (in the released position so that paper slides freely) when paper is changed. It is kept in the released position when a pin-feed platen is installed.

The line-space lever may be set for either single spacing or double spacing.

A margin stop is positioned by pushing it in and sliding it to the desired setting. When changing margin settings, it may be necessary to space or backspace to move the carrier, since the margin stop does not slide past it.

The typing-position indicator marks the position of the print element. A warning bell rings when the carrier nears the right margin.

PIN-FEED PLATEN

As an optional feature, the standard friction-feed platen may be replaced by a pin-feed platen for use with forms designed with sprocket holes. Pin-feed platens are available in several widths, from 5 1/4 inches to 13 1/8 inches (based on the hole-to-hole measurement of the paper). Further information on pin-feed platens may be obtained from an IBM representative.

RIBBON/PRINT ELEMENT CARRIER

The carrier, illustrated in Figure 6, holds the ribbon cartridge and print element.

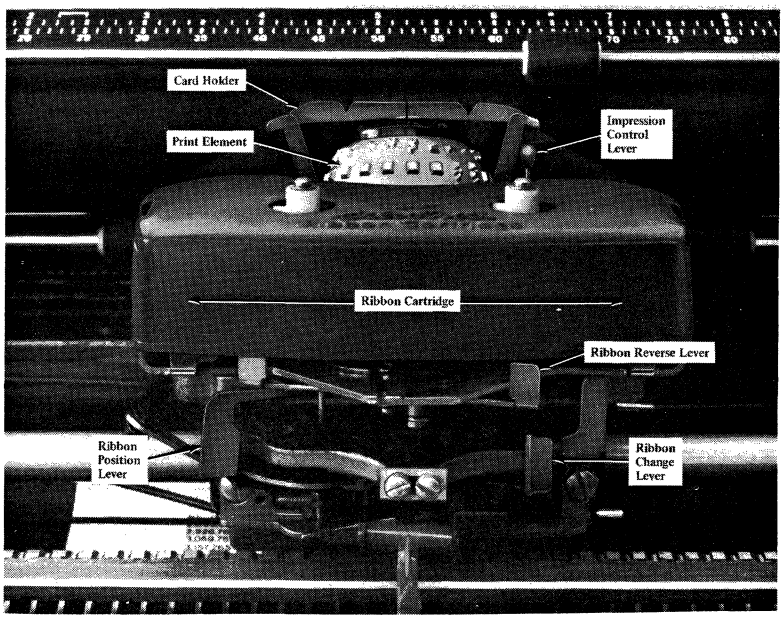

Figure 6. 2741 ribbon/print element carrier

The clear plastic card holder, which rides between the print element and platen, is imprinted with a centered vertical line and a horizontal line, which serve as alignment aids for Local typing.

The print element is a MANIFOLD 72 element, IBM Part No. 1167087 for correspondence type and IBM Part No. 1167643 for EBCD type. It is identified by the pitch number 10 under the arrowhead on the plastic cap, and by a code number, 087 for correspondence type and 643 for EBCD type, located to the left of the snap lock.

The ribbon-reverse lever reverses the direction of the ribbon movement. Ribbon movement automatically reverses when the end of the ribbon is reached.

The ribbon-position lever has four settings (left to right), which permit the ribbon to be positioned so that either the top, middle, or bottom section of the ribbon is used. The rightmost position is used for stencil operation. Periodic repositioning of this lever extends the life of the ribbon and permits the used portion of the ribbon fabric to be re-inked.

The ribbon-change lever, set to the left for typing, is moved to the extreme right to lift the ribbon guides for ribbon changing.

The red-knobbed impression-control lever adjusts the striking force of the print element. It is set to 1 for light impression, 3 for medium impression (one to five copies), and 5 for heavy impression. (See also the discussion of the multiple-copy control lever under "Paper Insertion and Movement Controls".)

The removable plastic ribbon cartridge carries an identifying name and reorder number on its underside.

RIBBON CHANGING

The steps for changing the ribbon are listed below. The letters A, B, and C refer to Figure 7.

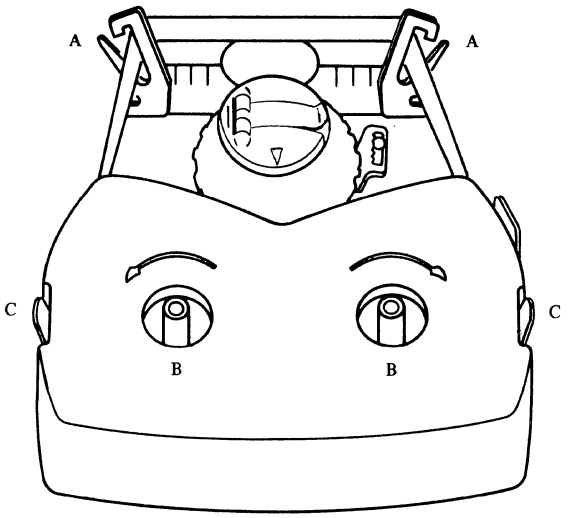

Figure 7. Ribbon changing

To remove the ribbon cartridge, the user should:

1. Center the print element and ribbon carrier.
2. Set the ON/OFF switch to OFF.
3. Raise the cover of the terminal.
4. Move the ribbon-change lever to the far right to raise the ribbon lifts (A).
5. Remove the ribbon from the ribbon lifts (A).
6. Lift the cartridge off the spindles (B), freeing it from the retaining clips (C).
7. Rewind the excess ribbon by inserting a pencil in either of the top cartridge holes and turning it in the direction of the arrow.

To install a new cartridge, the user should:

1. Balance the cartridge on the spindles (B).
2. Pull out several inches of ribbon and thread it through the ribbon lifts (A).
3. Snap the cartridge down onto the spindles.
4. Move the ribbon-change lever back to the left, making sure that the ribbon is lowered between the print element and the plastic card holder.
5. Rewind any excess ribbon by turning either spindle (B), with thumb and forefinger, in the direction of the arrow.
6. Close the cover; turn on power.

CHANGING PRINT ELEMENTS

The print element is shown in Figure 8. When it is changed, the terminal should be in the lower shift position and turned off. When an element is replaced or removed, it should be snapped directly into and out of position and not forcibly tilted or rotated on its post.

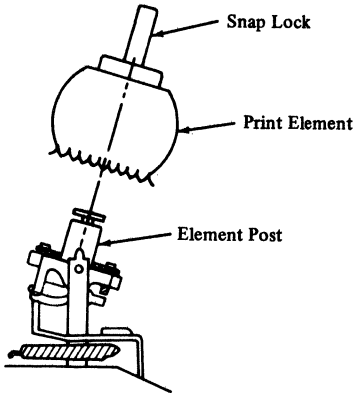

Figure 8. Position of print element while being removed or replaced

To remove an element, the user should:

1. Make sure the terminal is in lower-shift position (that is, the arrowhead on the element cap is pointed toward the platen).
2. Turn off the terminal and raise the cover.
3. Lift the snap lock in the center of the cap and use it to lift the element from the post.

To replace an element, the user should:

1. Lift the snap lock and place the element on the post with the arrowhead on the cap pointed toward the platen.
2. Gently press the element down on the post until it clicks into place.
3. Push the snap lock on the element cap down into position.

TEST PROCEDURES

CHECK-LOOP TEST

The check-loop test checks the electronic circuitry of the terminal and is performed with the Communicate/Local switch in the Local position. While the ATTN key is held down, a single operation of any one of the character keys on the keyboard causes that character to print repeatedly. This indicates that the electronic circuitry is functioning properly and that the character would be transmitted if the terminal were operating in Communicate mode. Releasing the ATTN key stops the repetitive printing.

If the character does not print repeatedly, the electronic circuitry of the terminal is not functioning properly and an IBM customer engineer should be notified.

ECHO TEST

The echo test is provided by the CALL-OS system as a method of checking the communications line and is performed with the terminal signed on the system. The user should type the terminal command ECHO or ECHOX, consisting of the command word, a space, and any sequence of alphabetic, numeric, and special characters available on the terminal. The computer responds to the ECHO command by duplicating, on the next line, all characters that were transmitted, up to a limit of 76 print positions. In response to the ECHOX command, the characters in the line are printed repetitively until the user presses the ATTN key.

If the line entered by the user and the system response do not match, the user should disconnect, reestablish the phone connection, and try the echo test again. If trouble persists, an IBM customer engineer should be notified.

INSTALLATION REQUIREMENTS

To assist users in the planning of a terminal installation, some of the physical, electrical, and environmental considerations of the 2741 Communications Terminal are given in the remaining portion of this appendix.

Physical considerations are illustrated in Figure 9.

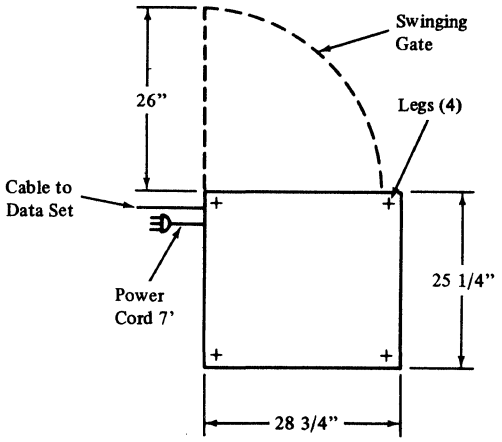

Figure 9. 2741 physical considerations

Dimensions: Width 28 3/4 in. Depth 25 1/4 in.

Height (Overall) 36 1/2 in.

Height (Desk-Top) 29 in.

Maximum Weight: 194 lbs.

Service Clearances: Front - 30 in. Rear - 42 in.
Right - 18 in. Left - 18 in.

Electrical Requirements:

Voltage	115 v ac ± 10%	Service:	15 amps
Frequency	60 Hz ± 1/2 Hz	Plug:	(see notes
Phase	Single	Receptacle:	(see notes)
KVA	0.15	Connector:	(see notes)

Environmental Specifications:

Heat Dissipation: 400 BTU/Hr.

	Temp.	Rel. Hum.	W. B. Temp.
Operating:	50-110°F	10-80%	85°F max.
Non-Operating:	50-110°F	10-80%	85°F max.

The terminal point-to-point configuration is illustrated in Figure 10.

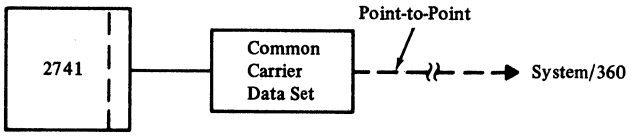

Figure 10. 2741 terminal attachment

Notes:

1. Plug types: Hubbell or Pass & Seymour 5267 (115 volts, nonlock). Customer provides matching receptacle: Hubbell or Pass & Seymour 5262 receptacle for 5267 plug.

2. Circuit should be separate three-wire single-phase branch circuit from power distribution panel, with green wire connected to ground, not current neutral.

3. For shipping, temperature limits are -40°F and +150°F.

4. The 2741 can communicate with CALL-OS over common-carrier switched telephone networks equipped with a Western Electric Data Set 103A (or its equivalent).

5. For further details, see IBM Remote Multiplexers and Communications Terminals, Installation Manual - Physical Planning.

APPENDIX B: EQUIPMENT CHARACTERISTICS - TELETYPE UNIT, TYPE 33

The Type 33 Teletype unit provides access to the CALL-OS system. It is available in two models: the Automatic Send/Receive (ASR) with a paper tape punch and reader, and the Keyboard Send/Receive (KSR). Both models are shown in Figure 11.

Although most Type 33 units have the characteristics described on the following pages, a specific unit may have slight variations.

If automatic restart capabilities are to be used with the expanded paper tape (TAPE ALL) facility, the Type 33 ASR must be equipped with the Reader Control Arrangement feature.

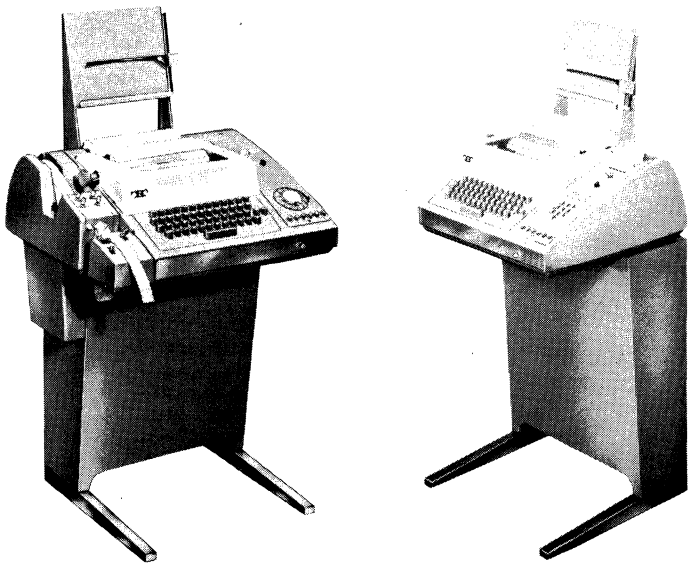

Figure 11. Teletype units, Type 33 ASR (left) and KSR

KEYBOARD AND CONTROLS

Figure 12 shows the keyboard and control panels of the Type 33 ASR. It contains all standard characters in the conventional arrangement as well as special characters. All alphabetic characters are capitals. The SHIFT key is used only for typing the "upshift" special characters.

The CTRL (Control) key (above the left SHIFT), like the SHIFT, is used in conjunction with other keys to perform special functions. Neither the SHIFT nor CTRL key is self-locking; each must be held down when used.

In addition to the standard keys, the keyboard contains several nonprinting keys with specialized functions. These are discussed below.

The LINE FEED key (near the right end of the second row) moves the paper up one line without moving the ribbon carrier and printing mechanism. When the terminal is used offline, the LINE FEED should be used after each line of typing to avoid overprinting of the next line.

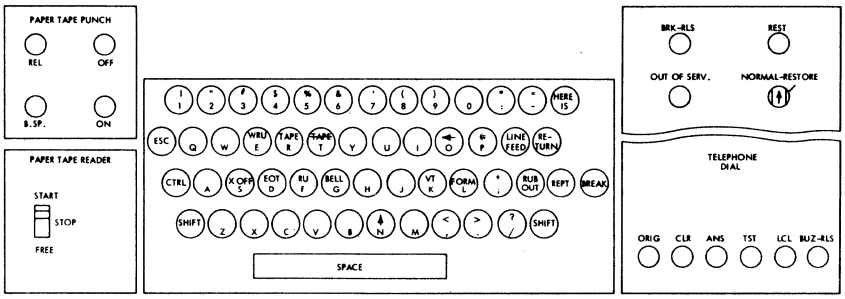

Figure 12. Type 33 ASR keyboard

The RUBOUT key (situated beneath the LINE FEED key) punches a full row of holes in the paper tape; this character is ignored when read by the computer. This key is used to punch tape leaders and trailers, and, when the tape has been backspaced, to overpunch and thus delete a character previously punched.

Holding down the REPT (Repeat) key (situated adjacent to RUBOUT) causes repeated action of any other key pressed.

The BREAK key (next to REPT) is used to interrupt program execution or listing. After breaking execution, the user must press the BRK-RLS button to activate the keyboard.

The HERE IS and ESC (ALT MODE on some units) keys are not used by the CALL-OS system.

The CTRL (Control) key (above the left SHIFT) is used in conjunction with the letter X to delete a typed line.

The WRU (letter E), EOT (letter D), and RU (letter F) functions of the CTRL key are not meaningful to the CALL-OS system. Use of them may cause incorrect system action. Therefore, they should not be employed by CALL-OS terminal users.

Most units have a loudspeaker and a volume control knob (VOL) located under the keyboard shelf. The knob is turned clockwise to increase volume. On some units, a telephone handset may be provided instead.

RIGHT CONTROL PANEL

The right-hand control panel, also called the Attendant's Control Unit, has six buttons below the telephone dial; above the dial it has two lights, a button, and the NORMAL-RESTORE knob.

The ORIG (Originate) button (at the left of the lower row) is pressed to obtain a dial tone before dialing. The volume control on the loudspeaker (under the keyboard shelf to the right) should be turned until the dial tone is audible. After connection has been made, volume can be turned down.

The CLR (Clear) button is pressed to turn off the Teletype unit.

The ANS (Answer) button is not used by the CALL-OS system.

The TST (Test) button is used for testing purposes and may be ignored.

The LCL (Local) button is pressed to turn on the Teletype unit when it is to be used offline (not connected to the system) for typing or

punching tape, or for changing tape or ribbon. It is recommended that the Teletype unit be turned off when it is not in use.

The BUZ-RLS (Buzzer-Release) button is used to shut off the buzzer that sounds to warn of low paper supply. The light in the BUZ-RLS button remains on until paper has been replenished.

The NORMAL-RESTORE knob (above the telephone dial) is set to NORMAL except to change ribbon or tape, in which case it is turned so that the arrow points to the OUT-OF-SERV light. The knob is momentarily set to RESTORE and then returned to the NORMAL position when the operation is completed.

The OUT-OF-SERV (Out-of-Service) light goes on when the NORMAL-RESTORE knob is pointed to it for ribbon or tape changing.

The BRK-RLS (Break-Release) button is used to unlock the keyboard after program execution has been interrupted with the BREAK key.

The REST light is not used by the CALL-OS system.

PAPER TAPE PUNCH AND READER

The paper tape punch and paper tape reader are housed in a unit situated to the left of the keyboard of the Type 33 ASR. The punch generates a row of holes in the tape to represent each character (including nonprinting functions).

The control panel contains four buttons and a switch:

The ON and OFF buttons are used to turn the unit on and off.

The B.SP. (Backspace) button moves the tape backward one character each time it is pressed, so that incorrect characters can be deleted by overpunching with the RUBOUT character.

The REL (Release) button frees the tape so that it can be pulled through the punch. Tape that has been released from the punch contains no holes and cannot be fed through the reader.

The paper tape reader feed-wheel switch controls the movement of the paper tape reel in the reader. This switch is set to the START position when tape is to be read. It is set to FREE when tape is to be positioned or released. When the switch is set to the STOP position, tape movement is halted.

The chad box, under the punch, collects the chad (bits of paper) and must be emptied periodically.

READER CONTROL ARRANGEMENT (X-ON, X-OFF) FEATURE

Stopping and starting of paper tape can be controlled by the computer when the optional Reader Control Arrangement feature is provided. When the letter S is struck while the CTRL (Control) key is held down, the X-OFF character is punched. When the Type 33 ASR reads the X-OFF character in the tape, it will turn off the paper tape reader. It is recommended that the X-OFF character be followed by at least three RUBOUT (delete) characters.

The tape reader can later be turned on with the X-ON character by the computer.

REPLACING PAPER TAPE

To place a new roll of paper tape in the punch, the user should:

1. Turn the NORMAL-RESTORE knob (above the telephone dial) counterclockwise so that its arrow points to the OUT-OF-SERV light.
2. Press the LCL button (below the dial).
3. Press the ON button on the punch unit.
4. Press the RUBOUT key (above the right SHIFT key on the keyboard) a sufficient number of times to feed out any remaining tape.
5. Remove the used roll of tape from the holder.
6. Place the new roll of tape in the holder so that tape feeds from the top of the roll.
7. Ease the tape through the loading plate, striking the RUBOUT key until the tape passes through and is visible at the punch head.
8. Press the OFF button on the punch unit.
9. Press the CLR button (below the telephone dial).
10. Turn the NORMAL-RESTORE knob to RESTORE so that a dial tone may be heard; then return it to NORMAL.

PAPER INSERTION

A buzzer sounds when the supply of paper is low. The buzzer may be silenced by pressing the BUZ-RLS button (Buzzer-Release, under the telephone dial). The light in the BUZ-RLS button remains on until the paper has been replenished.

To add paper, the user should:

1. Turn the NORMAL-RESTORE knob (above the telephone dial) counterclockwise so that its arrow points to the OUT-OF-SERV light.
2. Raise the Teletype unit cover.
3. Tilt the paper release lever forward (Figure 13), raise the paper guide, and pull the remaining paper out from under the platen.
4. Lower the paper guide.
5. Lower the Teletype unit cover.
6. Remove the used roll of paper.
7. Take the spindle from the used roll and insert it into the new roll.
8. Position the new roll so that the paper feeds out from under the roll toward the platen.
9. Raise the cover of the Teletype unit.
10. Feed the paper over the straightener rod and under the platen.

11. Raise the paper guide, and feed the paper between it and the platen.

12. Pull the paper up a few inches and straighten it.

13. Push back the paper guide so that it rests on the paper.

14. Tilt the paper release lever back into its original position.

15. Lower the Teletype unit cover (making sure the paper feeds out through the top).

16. Press the CLR (Clear) button (below the telephone dial).

17. Turn the NORMAL-RESTORE knob to RESTORE so that a dial tone may be heard; then return it to NORMAL.

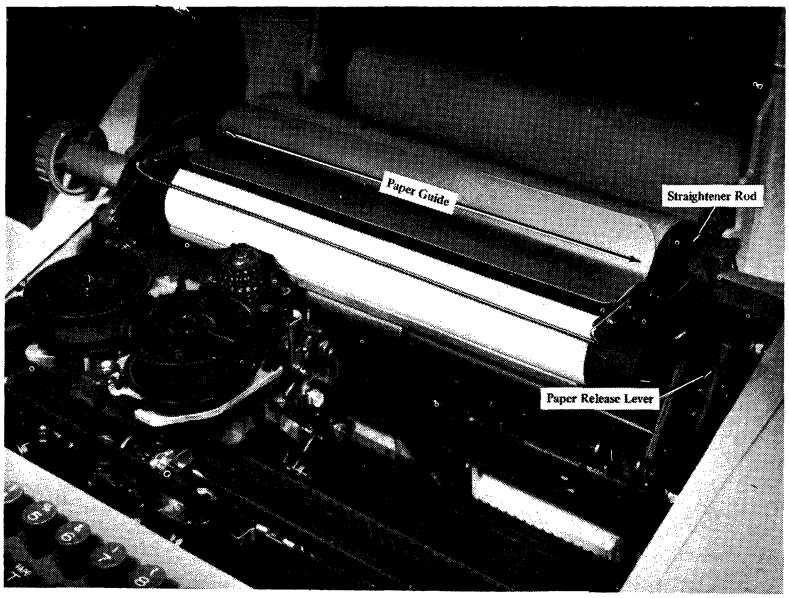

Figure 13. Type 33 ASR paper insertion and movement controls

RIBBON CHANGING

The Teletype unit uses a specially made ribbon. To change the ribbon, the user should:

1. Turn the NORMAL-RESTORE knob (above the telephone dial) counterclockwise so that its arrow points to the OUT-OF-SERV light.

2. Raise the Teletype unit cover.

3. Lift both ribbon spools (Figure 14) from their shafts.

4. Remove the ribbon from the ribbon guides and the ribbon reverse levers.

5. Remove the old ribbon from one spool.

6. Hook the end of the new ribbon to the hub of the empty spool (on spools having a barb, use it to pierce the ribbon) and wind the ribbon until the eyelet is on the spool.

7. Replace the spools on their shafts (making sure the spools go all the way down on their shafts and that ribbon feeds from outside the spools).

8. Thread the ribbon around the ribbon rollers, through the slots on the reverse levers, around the ribbon guides, and then through the slots on both the right and left side of the ribbon guides.

9. Turn the free spool to take up slack (the ribbon is properly threaded if the eyelet is between the spool and the reverse lever).

10. Lower the Teletype unit cover (making sure the paper feeds out through the top).

11. Turn the NORMAL-RESTORE knob to RESTORE so that a dial tone may be heard; then return it to NORMAL.

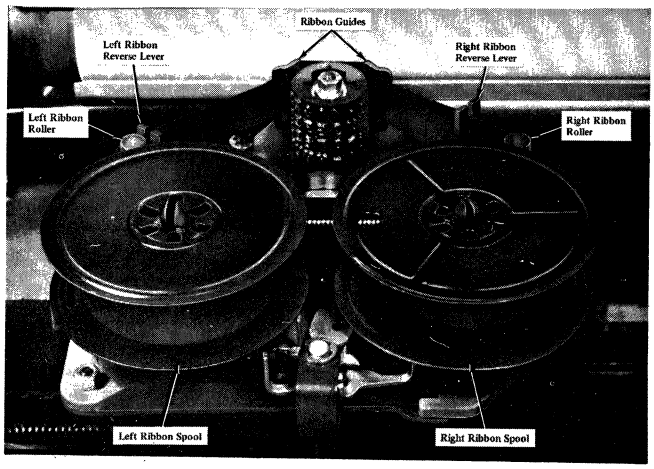

Figure 14. Type 33 ASR ribbon controls

ECHO TEST

The echo test is provided by the CALL-OS system as a method of checking the communications line and is performed with the terminal signed on the system. The user should type the terminal command ECHO or ECHOX, consisting of the command word, a space, and any sequence of alphabetic, numeric, and special characters available on the terminal. The computer responds to the ECHO command by duplicating, on the next line, all characters that were transmitted, up to a total of 72 print positions. In response to the ECHOX command, the characters in the line are printed repetitively until the user presses the BREAK key.

If the line entered by the user and the system response do not match, the user should disconnect, reestablish the phone connection, and try the echo test again. If trouble persists, installation management should be notified.

APPENDIX C: EQUIPMENT CHARACTERISTICS - TELETYPE UNIT, TYPE 35

The Teletype unit, Type 35, provides access to the CALL-OS system. It is available in two models: the Automatic Send/Receive (ASR), with a paper tape punch and reader, and the Keyboard Send/Receive (KSR). Both models are shown in Figure 15.

Although most Type 35 units have the characteristics described on the following pages, a specific unit may have slight variations.

If automatic restart capabilities are to be used with the expanded paper tape facility (TAPE ALL), the Type 35 must be equipped with the Reader Control Arrangement feature.

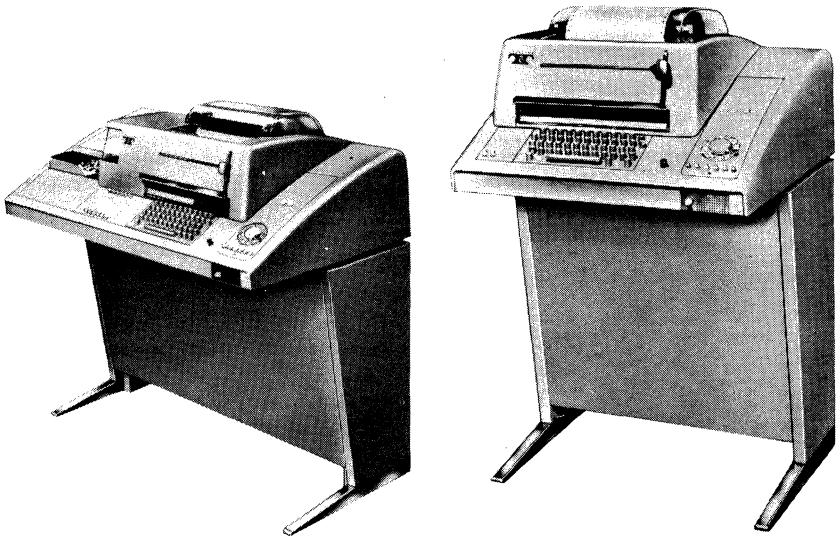

Figure 15. Teletype units, Type 35 ASR (left) and KSR

KEYBOARD AND CONTROLS

Figure 16 shows the keyboard and control panels of the Type 35 ASR. The keyboard contains all standard characters in the conventional arrangement, as well as a number of special characters. All alphabetic characters are capitals. The SHIFT key is used only for typing the "upshift" special characters.

The CTRL (Control) key (above the left SHIFT), like the SHIFT key, is used in conjunction with other keys to perform special functions. Neither the SHIFT nor CTRL key is self-locking; each must be held down when used.

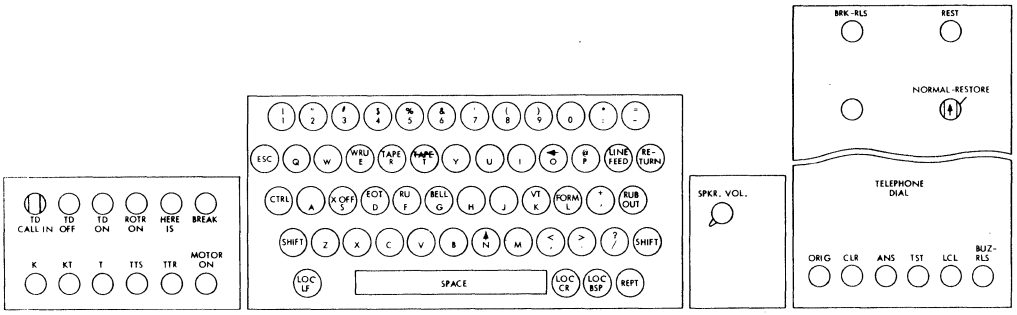

Figure 16. Type 35 ASR keyboard and control panels

In addition to the standard keys, the keyboard also contains nonprinting keys with specialized functions.

The LINE FEED key (near the right end of the second row) moves the paper up one line without moving the ribbon carrier and printing mechanism. When the terminal is used offline, the line feed must be used after each line of typing to avoid overprinting of the next line.

The RUBOUT key (situated beneath the LINE FEED key) punches a full row of holes in the paper tape; this character is ignored when read by the computer. This key is used to punch tape leaders and trailers, and, when the tape has been backspaced, to overpunch and thus delete a character previously punched.

Holding down the REPT (Repeat) key (situated below the right SHIFT) causes repeated action of any other key pressed.

The LOC BSP (Local/Backspace) key (situated adjacent to REPT) is used to backspace paper tape for the correction of punching errors.

The LOC CR (Local/Carrier Return) key (next to LOC BSP) may be used to return the carrier without generating a carrier return (end-of-line) character. Ordinarily, it is used only when a program statement containing a lengthy expression requires more than one line of typing. No statement may contain more than 238 characters.

The LOC LF key (Local/Line Feed, below the left SHIFT) may be used as a line feed without generating a line-feed character. It is used in conjunction with the LOC CR.

The ESC key (ALT MODE on some units) is not used by the CALL-OS system.

The CTRL (Control) key (above the left SHIFT) is used in conjunction with the letter X to delete a typed line.

The WRU (letter E), EOT (letter D), and RU (letter F) functions of the CTRL key are not meaningful to the CALL-OS system. Use of them may cause incorrect system action. Therefore, they should not be employed by CALL-OS terminal users.

Most units have a volume control knob (SPKR VOL) for the loudspeaker located to the right of the keyboard. This knob is turned clockwise to increase volume. On some units, a telephone handset may be provided instead.

A column indicator at the upper right side of the keyboard indicates the column that has just been printed. When the LOC CR key is used, no carrier return is recorded and the column indicator does not return.

A red light to the right of the column indicator goes on to warn that the carrier is approaching the right margin.

RIGHT CONTROL PANEL

The right-hand control panel, also called the Attendant's Control Unit, has six buttons below the telephone dial; above the dial it has two lights, a button, and the NORMAL-RESTORE knob.

The ORIG (Originate) button (at the left of the lower row) is pressed to obtain a dial tone before dialing. The volume control on the loudspeaker (under the keyboard shelf to the right) should be turned until the dial tone is audible. After connection is made, volume can be turned down.

The CLR (Clear) button is pressed to turn off the Teletype unit.

The ANS (Answer) button is not used by the CALL-OS system.

The TST (Test) button is used for testing purposes and may be ignored.

The LCL (Local) button is pressed to turn on the Teletype unit when it is to be used offline (not connected to the system) for typing or punching tape, or for changing tape or ribbon.

The BUZ-RLS (Buzzer-Release) button is used to shut off the buzzer which sounds to warn of low paper supply. The light in the BUZ-RLS button remains on until paper has been replenished.

The NORMAL-RESTORE knob (above the telephone dial) is set to NORMAL except to change ribbon or tape, in which case it is turned so that the arrow points to the OUT-OF-SERV light. The knob is momentarily set to RESTORE and then returned to the NORMAL position when the operation is completed.

The OUT-OF-SERV (Out-of-Service) light goes on when the NORMAL-RESTORE knob is pointed to it for ribbon or tape changing.

The BRK-RLS (Break-Release) button is used to enable the terminal to transmit after program execution has been interrupted with the BREAK key. The BRK-RLS button does not unlock the keyboard; the K button must also be pressed to allow keyboard transmission.

The REST light is not used by the CALL-OS system.

LEFT CONTROL PANEL

The panel to the left of the keyboard contains two rows of six buttons each. Their functions are described below:

The TD CALL IN knob (at the far left of the top row) must be on when prepunched tape is being transmitted to the computer.

The TD OFF button (next to TD CALL IN) turns off the paper tape reader.

The TD ON button (next to TD OFF) turns on the paper tape reader.

The ROTR ON and HERE IS buttons are not used by the CALL-OS system.

The BREAK button (next to HERE IS) is used to interrupt program execution or listing. After breaking execution, the user must press the BRK-RLS button and then the K button.

The K button (Keyboard, at the far left of the lower row) is used to unlock the keyboard. It sets the unit for page copy only (not tape).

The KT button (Keyboard/Tape, next to K) sets the unit for simultaneous printing and tape punching.

The T button (Tape, next to KT) sets the unit for tape punching only (not printing) or for reading and transmitting tape that has already been punched. When prepunched tape is read, the characters transmitted are also printed at the terminal.

The TTS button (Tape-to-Tape Send, next to T) sets the unit for transmitting prepunched tape. The characters read and transmitted are not printed.

The TTR button (Tape-to-Tape Receiver, next to TTS) sets the unit for punching characters sent from the computer. The characters punched are not printed.

The MOTOR ON knob (next to TTR) allows the user to punch tape offline without placing the terminal in Local mode.

PAPER TAPE PUNCH AND READER

The paper tape punch and reader are housed in a unit situated to the left of the keyboard of the Type 35 ASR. The punch generates a row of holes in the tape to represent each character (including nonprinting functions).

The punch is controlled by the buttons on the lower row of the left control panel. The LOC BSP key, next to the space bar on the keyboard, controls backspacing in the punch.

The chad box, under the punch, collects the chad (bits of paper) and must be emptied periodically.

The paper tape reader is controlled by the feed-wheel (RUN/FREE) switch and the TD ON and TD OFF buttons on the left control panel. The TD ON and TD OFF buttons turn the reader on and off. The switch is set to FREE when tape is to be moved freely through the reader; it is set to RUN when tape is being read and when the reader is not in use.

READER CONTROL ARRANGEMENT (X-ON, X-OFF) FEATURE

Stopping and starting of paper tape can be controlled by the computer when the optional Reader Control Arrangement feature is provided. When the letter S is struck while the CTRL (Control) key is held down, the X-OFF character is punched. When the Type 35 ASR reads the X-OFF character in the tape, it will turn off the paper tape reader. It is recommended that the X-OFF character be followed by at least three RUBOUT (delete) characters.

The tape reader can later be turned on with the X-ON character (through the CTRL key function of the letter Q) by the computer.

REPLACING PAPER TAPE

To place a new roll of tape in the punch, the user should:

1. Turn the NORMAL-RESTORE knob (above the telphone dial) counter-clockwise so that its arrow points to the OUT-OF-SERV light.

2. Raise the Teletype unit cover (release it by pushing in the two buttons on the sides of the cover).

3. Turn on the MOTOR ON knob on the left control panel.

4. Tear off any remaining old tape and then strike the RUBOUT key (above the right SHIFT key on the keyboard) enough times to feed out any remaining tape.

5. Remove the used roll of tape from the holder.

6. Place the new roll of tape in the holder so that tape feeds from the top of the roll.

7. Ease the tape through the tape guide arm and down into the chute, striking the RUBOUT key until the tape passes through the punch head and is visible.

8. Lower the Teletype unit cover.

9. Turn off the MOTOR ON knob.

10. Turn the NORMAL-RESTORE knob to RESTORE so that a dial tone may be heard; then return it to NORMAL.

PAPER INSERTION

A buzzer sounds when the supply of paper is low. The buzzer may be silenced by pressing the BUZ-RLS button (Buzzer-Release, under the telephone dial). The light in the BUZ-RLS button remains on until paper has been replenished.

To add paper, the user should:

1. Turn the NORMAL-RESTORE knob (above the telephone dial) counterclockwise so that its arrow points to the OUT-OF-SERV light.

2. Raise the Teletype unit cover.

3. Push the paper release lever back (Figure 17), raise the paper fingers, and pull the remaining paper out from under the platen.

4. Remove the used roll of paper.

5. Take the spindle from the used roll and insert it into the new roll.

6. Position the new roll with the spindle in the spindle grooves (paper should feed from under the roll toward platen).

7. Feed the paper over the straightener rod, under the platen, and up between the platen and paper fingers.

8. Pull the paper up a few inches and straighten it.

9. Push back the paper fingers so that they rest on the paper.

10. Pull the paper release lever forward into its original position.

11. Lower the Teletype unit cover (making sure the paper feeds out through the top).

12. Press the CLR (Clear) button (below the telephone dial).

13. Turn the NORMAL-RESTORE knob to RESTORE so that a dial tone may be heard; then return it to NORMAL.

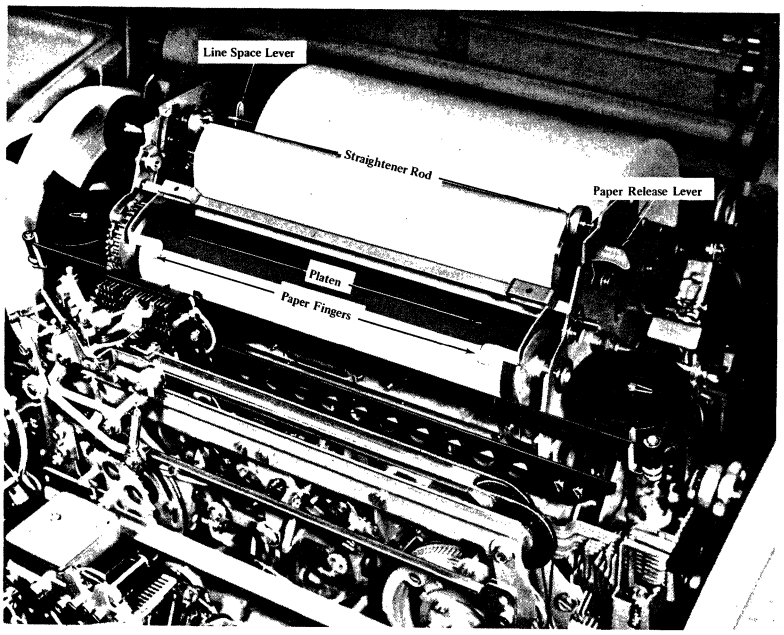

Figure 17. Type 35 ASR paper insertion and movement controls

RIBBON CHANGING

The Teletype unit uses a specially made ribbon. To change it, the user should:

1. Turn the NORMAL-RESTORE knob (above the telephone dial) counterclockwise so that its arrow points to the OUT-OF-SERV light.

2. Raise the Teletype unit cover.

3. Snap up the ribbon-spool locks (Figure 18) and lift both spools from their shafts.

4. Remove the ribbon from the ribbon rollers, the reverse levers, and the ribbon guides.

5. Remove the old ribbon from one spool.

6. Hook the end of the new ribbon to the hub of the empty spool (on spools having a barb, use it to pierce the ribbon) and wind the ribbon until the eyelet is on the spool.

7. Replace the spools on their shafts (making sure the spools go all the way down on their shafts and that ribbon feeds from outside the spools).

8. Snap the ribbon-spool locks down into position.

9. Thread the ribbon around the ribbon rollers, through the slots on the reverse levers, and around the ribbon guides.

10. Turn the free spool to take up slack (the ribbon is properly threaded if the eyelet is between the spool and the reverse lever).

11. Lower the Teletype unit cover (making sure the paper feeds out through the top).

12. Turn the NORMAL-RESTORE knob to RESTORE so that a dial tone may be heard; then return it to NORMAL.

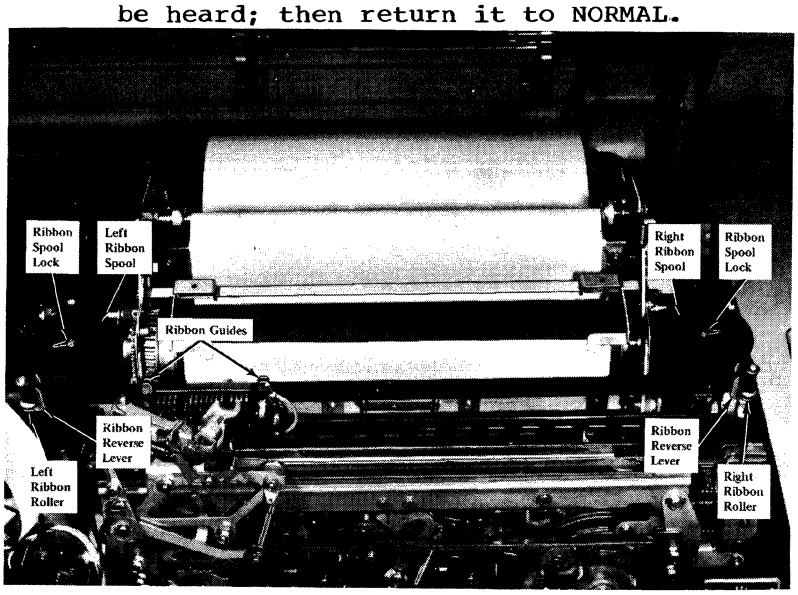

Figure 18. Type 35 ASR ribbon controls

ECHO TEST

The echo test is provided by the CALL-OS system as a method of checking the communications line and is performed with the terminal signed on the system. The user should type the terminal command ECHO or ECHOX, consisting of the command word, a space, and any sequence of alphabetic, numeric, and special characters available on the terminal. The computer responds to the ECHO command by duplicating, on the next line, all characters that were transmitted up to a total of 72 print positions. In response to the ECHOX command, the characters in the line are printed repetitively until the user presses the BREAK key.

If the line entered by the user and the system response lines do not match, the user should disconnect, reestablish the phone connection, and try the echo test again. If trouble persists, intallation management should be notified.

INDEX

GH20-0787-2

CALL–OS V 2 TOM Printed in U.S.A. GH20-0787-2

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

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

READER'S COMMENT FORM

CALL–OS V 2

GH20-0787-2

Terminal Operations Manual

Please comment on the usefulness and readability of this publication, suggest additions and deletions, and list specific errors and omissions (give page numbers). All comments and suggestions become the property of IBM. If you wish a reply, be sure to include your name and address.

COMMENTS

fold fold

fold fold

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A. FOLD ON TWO LINES, STAPLE AND MAIL.

GH20-0787-2

YOUR COMMENTS PLEASE...

Your comments on the other side of this form will help us improve future editions of this publication. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material.

Please note that requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or the IBM branch office serving your locality.

fold fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY...

IBM Corporation
1133 Westchester Avenue
White Plains, N.Y. 10604

Attention: Technical Publications

fold fold

CALL–OS V 2 TOM Printed in U.S.A. GH20-0787-2

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

READER'S COMMENT FORM

CALL–OS V 2

Terminal Operations Manual

GH20-0787-2

Please comment on the usefulness and readability of this publication, suggest additions and deletions, and list specific errors and omissions (give page numbers). All comments and suggestions become the property of IBM. If you wish a reply, be sure to include your name and address.

COMMENTS

fold

fold

fold

fold

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A. FOLD ON TWO LINES, STAPLE AND MAIL.

GH20-0787-2

YOUR COMMENTS PLEASE...

Your comments on the other side of this form will help us improve future editions of this publication. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material.

Please note that requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or the IBM branch office serving your locality.

fold fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY...

IBM Corporation
1133 Westchester Avenue
White Plains, N.Y. 10604

Attention: Technical Publications

fold fold

CALL—OS V 2 TOM Printed in U.S.A. GH20-0787-2

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

IBM Technical Newsletter

Base Publ. No. GH20-0787-2

This Newsletter No. GN20-2782

Date July 31, 1972

Previous Newsletter Nos. None

CALL-OS
Terminal Operations Manual
Program Number 360A-CX-42X

This Technical Newsletter is intended only for those who wish to make the base publication apply to Version 1, Modification Level 2. It provides replacement pages for the subject manual. These replacement pages remain in effect for subsequent versions and modifications unless specifically altered. Pages to be inserted and/or removed are listed below.

9–12
17–20
39–40
71–72
89–90
95–96
109–110
119–120
123–132
132.1–132.2 (pages added)
137–138

A vertical rule in the left margin indicates a change. Absence of a vertical rule on a page bearing a "revised" notice means only that existing copy has been moved or that a minor typographical error has been corrected.

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

Note: The IBM Operating System is commonly referred to as OS. For consistency, the term OS/360 has been replaced by the term OS on all changed pages of this manual. Users of the manual should regard OS and OS/360 as synonymous.

IBM Corporation, Technical Publications Dept., 1133 Westchester Avenue, White Plains, N. Y. 10604

Printed in U.S.A.