IBM

Application Program

GH20-0673-5

CALL-OS Version 2

System Description Manual

Program Number 360A-CX-42X

This manual is intended primarily to define the scope of applicability of CALL-OS for customer executives, system analysts, and programmers. It describes the facilities of CALL-OS and includes a general discussion of the CALL-OS system configuration, terminal processing, system structure and control, and system support and maintenance facilities, and an introduction to the language processors supported. The CALL-OS terminal command language is summarized for the user.

CALL-OS is a high-performance, terminal-oriented, time-sharing system designed to operate as a problem program under the IBM Operating System (OS) Multiprogramming with a Fixed Number of Tasks (MFT) or Multiprogramming with a Variable Number of Tasks (MVT) control program. The Model 65 Multiprocessing System (M65MP) is not supported.

CALL-OS provides a personal computing facility designed to handle a high volume of traffic in a problem-solving environment. The system is designed to satisfy the needs of the professional programmer as well as of the individual with no programming experience. It provides remote terminal services for CALL-OS BASIC, CALL-OS FORTRAN, and CALL-OS PL/I users.

In addition, CALL-OS permits the terminal user to create OS jobs and submit those jobs via the CALL-OS Batch Interface (COBI) option for OS batch processing. All or a portion of the output of submitted jobs can be listed at the terminal.

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.

Sixth Edition (June 1972)

This edition, GH20-0673-5, is a major revision obsoleting GH20-0673-4. It reflects Version 2, Modification Level 1, of the CALL-OS time-sharing system and all subsequent versions and modifications until otherwise indicated in new editions or Technical Newsletters. All changes to text and illustrations are indicated by vertical lines in the left margin.

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 this 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

CALL-OS is a terminal-oriented, time-sharing system designed to operate as a problem program under the IBM Operating System (OS) Multiprogramming with a Fixed Number of Tasks (MFT) or Multiprogramming with a Variable Number of Tasks (MVT) control program. It provides a personal computing facility for both the professional programmer and the individual with no programming experience. A user interacts with the system using the CALL-OS terminal command language and CALL-OS BASIC, CALL-OS FORTRAN, or CALL-OS PL/I. Programs can be entered at a terminal and executed immediately and/or placed in either source or object form in system libraries for subsequent, immediate use. CALL-OS Batch Interface (COBI) facilities permit the user to create OS jobs, submit those jobs to the OS batch-processing environment, and request that all or a portion of the output from submitted jobs be listed at the terminal.

This publication is intended primarily to define the scope of applicability of CALL-OS for customer executives, system analysts, and programmers. The manual includes a general discussion of the CALL-OS system configuration, terminal processing, system structure and control, and system support and maintenance facilities, and an introduction to the language processors supported. The terminal command language is summarized in Appendices A and B.

For additional details concerning the CALL-OS system, the following CALL-OS publications should be referred to:

CALL-OS BASIC Language Reference Manual (GH20-0699)

CALL-OS PL/I Language Reference Manual (GH20-0700)

CALL-OS FORTRAN Language Reference Manual (GH20-0710)

CALL-OS Executive and Utilities Program Description Manual (GH20-0786)

CALL-OS Terminal Operations Manual (GH20-0787)

CALL-OS Operator's Manual (GH20-0788)

CALL-OS PL/I Reference Card (GX20-1810)

CALL-OS BASIC Reference Card (GX20-1811)

CALL-OS FORTRAN Reference Card (GX20-1812)

CALL-OS Terminal Command Language Reference Card (GX20-1830)

CONTENTS

FIGURES

INTRODUCTION

CALL-OS is a terminal-oriented, interrupt-driven, time-sharing system which provides an individualized computing capability to a variety of users. It is designed to handle a high volume of traffic in a problem-solving environment, and to satisfy the needs of the experienced professional programmer as well as the uninitiated computer user.

SYSTEM OVERVIEW

CALL-OS operates as a task under the control of the IBM Operating System, Multiprogramming with a Fixed Number of Tasks (MFT) or Multiprogramming with a Variable Number of Tasks (MVT) -- hereafter referred to in this manual as OS. It may reside totally or in part in the IBM 2361 Large Capacity Storage (LCS) under OS Hierarchy Memory Support. Background jobs may execute concurrently with CALL-OS to capitalize on remaining central processing unit (CPU) time and core storage capacity. Thus the multiprogramming functions of the standard operating system plus a time-sharing capability are provided. In this environment, each CALL-OS terminal user operates independently of every other user, and is usually unaware of other terminal users or other OS activity.

The CALL-OS system includes the CALL-OS Executive, which interfaces to OS, language processors which operate under control of the CALL-OS Executive, and utilities designed to be run under OS when the CALL-OS time-sharing system is not active. The system is designed as a modular, multiprogramming system. Program modules are written in System/360 Assembler Language.

CALL-OS Batch Interface (COBI) facilities permit CALL-OS terminal users to create OS jobs, submit those jobs to the OS batch-processing environment, and list all or a portion of the output from submitted jobs. COBI is an optional component of the CALL-OS system. If it is to be used, it must be included in the system at system build time. If included at system build, it may or may not be activated for a particular session of CALL-OS.

ADVANTAGES

Some of the highlights of the CALL-OS system follow:

- Highly responsive personal computing system under OS
- Concurrent batch-processing capability under OS
- Extensive terminal command language--directed towards both experienced and inexperienced users
- System structure designed for problem solving
- Multiprogramming within a single task area
- Dynamic assignment of dispatching priorities to provide efficient use of CPU time
- Fast, load-and-go compilers
- Compiler-generated dynamically relocatable code

- Extensive edit capabilities for modification of source programs
- Optional storing of programs in object format
- Terminal checkout of user programs
- Operator control of system resources
- Facility for entry of source programs, data, and terminal commands via paper tape
- Facility for terminal entry of OS jobs, submittal of those jobs to OS batch processing, and retrieval of output from submitted jobs via the CALL-OS Batch Interface (COBI) option
- Extensive CALL-OS data-base manipulation and maintenance capabilities available via offline (OS batch-oriented) CALL-OS utilities
- Support of 2314, 2319, and 3330 disk storage devices; CALL-OS will run with all of one device type, or a combination of supported devices

SYSTEM FEATURES

LANGUAGES

CALL-OS supports the following languages:

- CALL-OS BASIC, an enhanced version of the BASIC language originally developed at Dartmouth College, Hanover, New Hampshire
- CALL-OS PL/I, a powerful general-purpose language
- CALL-OS FORTRAN, a widely used scientific and engineering language

Details concerning language capabilities are provided in three CALL-OS language reference manuals: CALL-OS BASIC Language Reference Manual, CALL-OS PL/I Language Reference Manual, and CALL-OS FORTRAN Language Reference Manual. The language capabilities are summarized on three corresponding reference cards.

USER LIBRARIES

CALL-OS supports multiple levels of stored libraries that can be used to retain either program or data files. The libraries are:

- A library accessible to an individual user (user library)
- A library accessible to only a specified group of users as determined by the user installation (*Library)
- Two libraries accessible to all users (the **Library, into which files can be pooled by terminal users, and the ***Library, whose contents are controlled by central computer installation personnel)

Thus, the *Library, **Library, and ***Library are shared libraries. When the term "system libraries" appears in this manual, these libraries are inferred. Program files can be shared by multiple users of CALL-OS. They can be pooled into system libraries in either source or object form. Data files can also be shared, but a shared data file can be used

only for input (when accessed by other than the pooling user); that is, only the originator of a data file can modify it.

MULTIPLE INPUT/OUTPUT FILES

CALL-OS allows data files to be attached to programs at execution time. These files may also be detached and other files attached, permitting access to all of a user's files during one execution of a program. Up to a maximum of four files per program may be open simultaneously. Each file comprises a number of 3440-byte disk storage units. If the maximum number of units to be allowed for a file is not specified (via a FILE command), a default of four is assumed.

The maximum file allocation as specified in the FILE command cannot be greater than the maximum number of disk storage units allowable for a data file, as established at system initialization time. The maximum value that can be specified by an installation for maximum number of units is 100 units, permitting up to 344,000 bytes per file. If no maximum value is specified at system initialization time, the default maximum number is 100. (Thus, both the maximum that can be specified by an installation and the default maximum are 100.)

CALL-OS BATCH INTERFACE (COBI)

The CALL-OS Batch Interface (COBI) option permits a CALL-OS terminal user to create and maintain OS jobs. Terminal commands are issued to route such jobs to the OS batch-processing environment. The job control language (JCL), programs, and data for a job must be available in the user's library when the user enters his SUBMIT command. It may have been entered from a terminal and saved in the user's library, loaded into the user's 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. The user can specify that JCL or specific user or procedure-defined SYSOUT data sets of the job be retained for scanning. He can inquire about the status of submitted jobs or of data sets associated with a submitted job. After a job has been completed, all or a portion of the specified output can be printed at the terminal. See "CALL-OS Batch Interface (COBI) Option" and Appendix B for a description of the facilities available.

SECURITY

To protect against inadvertent access to or destruction of programs and data files, CALL-OS requires that each terminal user follow a specific sign-on procedure, supplying his user number and password. All resources belonging to a user are protected under this identification.

When a user chooses to make his program available to other CALL-OS users through the system library facilities, he can control the use of that program by issuing a PROTECT command to assign a run-only status to it (see Appendix A). A protected program can be run or merged with other programs, but cannot be listed, stored, or saved by any CALL-OS user other than the contributor of the file. Terminal commands controlling edit operations such as adding strings of characters to ends of lines, finding specified character strings in lines, or moving lines cannot refer to protected programs. A program resulting from a merge involving a protected program is also protected and subject to the restrictions this status implies.

A data file placed in a system library may also be protected by issuance of a PROTECT command. A protected data file cannot be opened

by anyone other than the contributor of the file. Thus, access to the file can be restricted prior to or during a required update process.

CALL-OS program and data files are similarly protected against unauthorized use via CALL-OS utilities. For any program or data file saved or stored through a terminal or created via a CALL-OS utility, a SECURE attribute is set automatically. No one other than the creator of a file can copy, modify, or delete any part or all of the file. To permit his file to be copied by another user, the creator of a file can issue a RELEASE command, causing a RELEASE attribute to be set. After the RELEASE attribute has once been set, it is reset by user issuance of a SECURE command or automatically by a CALL-OS utility that accesses the file.

While the CALL-OS data base is mounted, access to critical user-group data sets can be controlled by specifying DISP=OLD in the data definition (DD) statements for system initialization or the utilities.

PERFORMANCE CONSIDERATIONS

CALL-OS system performance is a function of several variables, including the following:

- Number of terminal lines to be supported
- Size and characteristics of the time-sharing programs
- IBM System/360 or System/370 CPU model
- Core size allotted to CALL-OS
- Amount of background activity
- Frequency and distribution of command execution
- System arrangement of frequently used libraries
- Direct access storage configuration

Great care should be exercised in predicting performance levels in any specific situation.

SYSTEM CONFIGURATION

The hardware configuration for the CALL-OS system is determined by the requirements of OS, CALL-OS, and the objectives of each installation.

MINIMUM CONFIGURATION

The minimum central processing unit (CPU) for the CALL-OS system is one of the following:

- System/360 Model 50HG
- System/370 Model 145H (384K, where K is 1024) with:

  3345 Main Storage Frame
  4901 Main Storage Frame Adapter
  3046 Power Storage
  3910 Extended Precision Floating Point Feature (optional feature, but no charge)

- System/370 Model 155HG

The minimum peripheral equipment required for online operation with each of these CPU's is shown below.

- System/360 Model 50HG:

  One selector channel
  One IBM 2314 Storage Control Model A1
  One IBM 2312 Disk Storage Model A1
  One IBM 2702 or 2703 Transmission Control
  Two terminal consoles (see below)

- System/370 Model 145H (384K, as defined above)

  One IBM 2319 Disk Storage Facility Model A1
  One IBM Integrated File Adapter feature (#4650)
  One IBM 2702 or 2703 Transmission Control
  Two terminal consoles (see below)

- System/370 Model 155HG:

  One block multiplexer channel
  One IBM 2314 Disk Storage Control Model A1
  One IBM 2312 Disk Storage Model A1
  One IBM 2702 or 2703 Transmission Control
  Two terminal consoles (see below)

The 2314 Direct Access Storage Facility must be on the selector channel of a System/360 Model 50 configuration or the block multiplexer channel of a System/370 Model 155 configuration. The CALL-OS system employs the 2314 or 2319 for system residence, for program and data file storage, and for a work area for terminal users. CALL-OS can operate with one IBM 2316 Disk Pack, but it is recommended that more than one disk pack be mounted and available to the system during processing. (For discussion of a configuration that includes 3330 Disk Storage, see "Representative OS Configurations" in this section.)

The 2703 Transmission Control unit (TCU) acts as an interface between the terminal lines and OS. The 2702 TCU, with equivalent features and RPQ's, may be substituted. With the appropriate expansion features, a 2702 TCU can support up to 31 lines. The 2703 can support up to 176 lines, expanding in groups of eight-line units. The IBM 3705 Communications Controller, with emulator program (EP), can serve as a 2703 in a System/360 or System/370 configuration supporting CALL-OS.

The two terminal consoles are used for system communication. One serves as a command console from which the operator issues special operator commands (*commands). The other serves as a communications console for recording system error messages and activity. The OS system console is used to initialize CALL-OS and may serve as the communications console, thus reducing to one the number of terminal consoles required.

The following additional peripherals are required for the system support and maintenance functions performed by CALL-OS:

- One printer output unit, OS-supported, with 120 print positions and graphics equivalent to the PN print arrangement
- One card input unit (as required by an OS minimum system)
- One punched card output unit (as required by an OS minimum system)
- One IBM 2401 Magnetic Tape Unit (any model)

Any peripheral devices, in addition to those given above, will be supported within the limits of OS support. Specifically, CALL-OS can use additional selector channels and any appropriate 2314 Direct Access Storage Facility A or B Series configuration. A System/360 Model 85 or 195 or a System/370 installation can include 3330 Disk Storage. In a System/360 configuration, use of the high-resolution timer RPQ gives the most accurate accounting capability. This RPQ is available on the Models 50 (RPQ E15092) and 65 (RPQ E43528) central processing units.

In addition to the above minimum hardware configuration, the user must have an MVT or MFT system with at least an OS Assembler, a Linkage Editor (F), the OS utilities IEHPROGM, IEHMOVE, and IEBUPDTE, and the hardware required to run OS.

TERMINALS AND CONSOLES

CALL-OS TERMINALS

CALL-OS supports the following terminals:

- IBM 2741 Communications Terminal (Correspondence or EBCD)
- Teletype Units, Type 33 or 35

Any of the above terminals can be used as a user terminal, a command console, or a communications console. Not more than 255 terminals (including the command and communications consoles) can be simultaneously online with CALL-OS.

The 2741 (used for any of the above functions) should have the following features:

- #3255 Dial-Up Feature
- #4708 Interrupt Feature

- #9104 Spacing Feature (10 character/inch)
- #9114 Data Set Attachment Feature: Western Electric Data Set 103A2 or equivalent
- #9435 Line Feed Feature (6 line/inch)
- For the 2741 (Correspondence), RPQ S30006 CALL/360 Printing Element and Keyboard

  For the 2741 (EBCD), RPQ S30021 CALL/360 Printing Element and Keyboard

  Note: The RPQ number is sufficient identification. No additional feature code to specify keyboard type is required.
- 98XX Line Voltage as appropriate

If a 2741 Communications Terminal with all above features except RPQ S30006 or RPQ S30021 is currently installed, a CALL/360 Printing Element with:

- Part No. 1167087 for the 2741 (Correspondence) or
- Part No. 1167643 for the 2741 (EBCD)

should be obtained. Terminal Character Decal GX20-1806 is desirable.

Note: Use of a leased line for the command and communications consoles is recommended. A four-wire IBM Line Adapter or Western Electric 103F may be used. The Dial-Up Feature (#3255) is not required. The #9115 Data Set Attachment (rather than #9114) should be specified for use with Western Electric 103F. Any one of several appropriate IBM Line Adapter feature codes may be specified. Compatible features must be included on the 2702 or 2703.

Teletype units, Types 33 and 35, are standard units. Either terminal may be selected in any of three models: RO (Receive Only), KSR (Keyboard Send-Receive), and ASR (Automatic Send-Receive). RO terminals can be used only for output. KSR and ASR terminals can be used for keyboard input as well as output. In addition, ASR terminals provide capability for input of source programs, data, and terminal commands via punched paper tape. The Reader Control Arrangement feature is required if paper tape input is to be restarted automatically.

COMMAND AND COMMUNICATIONS CONSOLES

Two special types of consoles are used by the system. The first is the command console. Although there are no special hardware features associated with this console, commands not available to the ordinary terminal user (*commands) are initiated from this console for operator communication with the CALL-OS system. A minimum of one command console is required; two terminals may be specified for use as command consoles if desired. Either or both may be operated as user terminals after the initial startup of the system.

The second special type of console used in the CALL-OS system is the communications console. This console is used throughout execution of CALL-OS to record system error messages and activity.

The 2741 or a Teletype unit, Type 33 or 35, can be used for either the command console or the communications console. If a Teletype unit is used, the Type 35 is recommended. The Type 35 RO can be used only as

a communications console. A Type 35 KSR or ASR can be used for either. The OS system console can be used as the communications console.

INTERVAL TIMER CONSIDERATIONS

CALL-OS is designed to operate with either the high-resolution (13.02 microsecond resolution) or standard (16.667 millisecond resolution) interval timer available on the System/360 Model 50 or 65, or the standard interval timer available on the System/370 Models 145, 155, and 165 (3.3 millisecond resolution).

Time-slice values, controlling time allotments for compilation and execution of user programs, are established by a user installation at system initialization time and then interpreted and applied during system operation by CALL-OS and OS timer routines. Since the execution of a job may be fragmented into many small pieces, some of which may be less than 16 milliseconds duration, use of the standard interval timer on the System/360 Model 50 or 65 could lead to failure to record such fragments during some executions. Therefore, use of the high-resolution interval timer is recommended for repeatable program execution times and most accurate accounting on the System/360 Models 50 and 65.

RPQ SUMMARY

Engineering Change Levels (EC's) and several RPQ's are important to CALL-OS hardware:

- Any IBM 2703 used in the CALL-OS environment must be at Engineering Change Level 307695 or higher.
- Similarly, any IBM 2702 used in the CALL-OS environment must be at Engineering Change Level 307575 or higher.
- For an IBM 2703 with attached Model 33/35 Teletype units, the Telegraph Terminal Control Type II (#7912) requires RPQ E62376. This RPQ causes the terminal control to recognize carrier returns and control X characters instead of X-ON and X-OFF, respectively.
- Similarly, for an IBM 2702 with attached Model 33/35 Teletype units, the Telegraph Terminal Control Type II (#7912) requires RPQ E62920. This RPQ modifies the terminal control to recognize two pluggable characters (carrier return and control X).
- High-resolution timers recommended are RPQ E15092 for the Model 50 CPU and RPQ E43528 for the Model 65.
- To utilize the ASR paper-tape feature of any attached TTY in TAPE mode, RPQ E54838 for the 2702 or RPQ Z16087 for the 2703 is required to modify the Telegraph Terminal Control Type II on the transmission control unit. For the 2702, RPQ EA3120 is required to inhibit interrupt on X-OFF if the TAPE ALL command is to be used.

REPRESENTATIVE OS CONFIGURATIONS

A representative System/360 configuration supporting CALL-OS includes:

- One primary command console and one alternate command console, usually consisting of one 2741 Communications Terminal (Correspondence or EBCD) and one Teletype unit, Type 35 ASR; one communications console, either a 2741 Communications Terminal (Correspondence or EBCD) or a Teletype unit, Type 35 RO

- One 2314 Storage Control, Model A1
- One or more 2313 Disk Storage Modules, Model A1
- System/360 Model 50 with Model I core storage
- High-resolution timer.

A representative System/370 configuration includes:

- One primary command console, one alternate command console, and one communications console, as described above
- One 3830 Storage Control, Model 001
- One or more 3330 Disk Storage Modules, Model 001
- System/370 Model 155IH

CUSTOMER RESPONSIBILITIES

During the installation of the CALL-OS system, it is necessary to contact the local telephone company and arrange for communication service for the data center and for the various user installations. The local telephone company should be apprised of the number of lines that will be required for the data center, the size of the line groups, and the system characteristics. The customer is responsible for ordering and installing the communications equipment required to support CALL-OS.

The customer is also responsible for the following functions:

- An OS system generation to include support for CALL-OS devices, the storage protect function, the interval timer function, and one "user" SVC (provided with the CALL-OS system)
- Ordering and installing at least one of the three CALL-OS language processors
- Allocating direct-access storage space for the CALL-OS Executive and libraries
- Building a CALL-OS Executive and data base
- Ensuring that proper steps are taken to maintain the security of his confidential program and data files

If the CALL-OS Batch Interface (COBI) option is utilized, the customer is also responsible for:

- Building COBI data sets (via a COBI utility program)
- Converting catalog procedures for use by COBI

TERMINAL PROCESSING

SIGN-ON

To use the CALL-OS system, a user dials the system telephone number. Subsequent action is determined by terminal type as follows:

1. If the user is at a Teletype unit, the system prints a sign-on message as shown below and requests entry of user number and password.

2. If the user is at a 2741 Communications Terminal (Correspondence or EBCD), he may type LOGON or LOG and press the carrier return. The system determines the terminal type and prints a sign-on message in the appropriate 2741 code.

   If the user does not type LOGON or LOG, followed by a carrier return or pressing of the Attention (ATTN) key, the system assumes that the 2741 is as defined in the system startup deck and prints a message in the appropriate code.

   If the default assumption is not valid, the sign-on message is garbled. The user can perform either of two actions to cause the terminal to be identified correctly:

   a. Type LOGON or LOG.

   b. Type his user number and password.

A user number consists of three alphabetic and three numeric characters (such as ABC123). The password, which the user defines and controls himself, has up to eight characters in any combination (such as WXYZ90+). This user number and password are checked and decoded to confirm that he is a valid user.

After confirmation, the system responds with READY. The entire sign-on procedure might look like this:

```
ON AT 9:30  09/13/71  MONDAY  PAO LINE 60
USER NUMBER,PASSWORD--ABC123,WXYZ90+
READY
```

Note: ABC123,WXYZ90+ is typed by the user.

TERMINAL COMMAND LANGUAGE

When the user has successfully signed on, he may enter terminal commands or source-language statements. A command begins with an alphabetic character. A language statement begins with a line number.

The terminal command language is designed to facilitate communication between the terminal user and the computer. This language provides commands that permit the terminal user to:

- Log on and log off the system with password security
- Select the CALL-OS language appropriate to the solution of his problem
- Develop problem solutions (user programs) at his own pace

- Create, modify, and save source programs
- Perform various edit operations on one or a number of source-program files
- Store programs in object format
- Develop and use multiple data files within his own programs
- Share both program and data files with other CALL-OS users by pooling them in system libraries
- Protect program and data files pooled by him into system libraries
- Purge program and data files from his library
- Obtain listings of program files
- Cause a program to be compiled and/or executed
- Obtain information relative to terminal connect time, CPU time, and program and data files
- Use routines residing in CALL-OS system libraries
- Submit jobs for OS batch processing via the COBI option
- Scan (print), inquire about, and scratch or keep all or a portion of the output from jobs submitted for OS batch processing

CALL-OS terminal commands, together with brief descriptions of their functions, are listed in Appendices A and B. The commands are explained in detail in the CALL-OS Terminal Operations Manual and summarized on the CALL-OS Terminal Command Language Reference Card.

To disconnect from the system, the user types OFF. The terminal and CPU processing time used are then printed automatically.

CALL-OS PROGRAMMING LANGUAGES

Upon initial sign-on to CALL-OS, the system assumes that CALL-OS BASIC with short-form arithmetic is the programming language to be used. If a user desires to use another language, he types the terminal command ENTER followed by the language name (FORTRAN, PL/I, or BASICL, for BASIC with long-form arithmetic).

In all of the CALL-OS languages, each line entered through the terminal must begin with a line number which:

- Identifies the line to the system as a language statement rather than a command
- Specifies to the system the order in which lines are to be considered and acted upon

Two types of data files are created by user programs within CALL-OS: external (EBCDIC) and internal (binary). The former can be created by programs written in CALL-OS FORTRAN or CALL-OS PL/I, and can be used by programs written in the language that created them. The latter can be created and used interchangeably by programs written in CALL-OS FORTRAN, CALL-OS PL/I, and CALL-OS BASIC. The created data should, of course, be in a form supported within the structure of the language used.

CALL-OS BASIC LANGUAGE

CALL-OS BASIC is designed for ease of use. The experienced programmer or the terminal user having little or no experience with computers or programming languages will find CALL-OS BASIC simple to learn and use.

CALL-OS BASIC has the following features:

- Short- and long-form arithmetic
- Simple and precise control of printed output
- Acceptance of input from an internal data list, a terminal, or a disk file
- Built-in mathematical functions and constants
- Statements for matrix manipulation

Sample CALL-OS BASIC Program

Shown below is a CALL-OS BASIC program to compute compound interest using the following formula:

$$A = P\left(1 + \frac{i}{100}\right)^{N}$$

where P is the principal (amount originally invested), i is the annual interest rate, and N is the length of time of the investment in years.

```
ON AT 9:30  05/15/72  MONDAY  PAO LINE 60
USER NUMBER,PASSWORD--ABC123,WXYZ90+
READY

10 PRINT "ENTER LENGTH OF TIME OF INVESTMENT"
20 READ P, I
30 DATA 1000.00, 5
40 INPUT N
50 LET A = P* (1 + I/100) **N
60 PRINT USING 70, N,A
70 :IN ## YEARS, THE AMOUNT WILL BE $####.##
80 END
RUN
            10:35   05/15/72   MONDAY PAO LINE 60


ENTER LENGTH OF TIME OF INVESTMENT
?25
IN  5 YEARS, THE AMOUNT WILL BE $1276.28
TIME  0 SECS.
```

Figure 1. Sample CALL-OS BASIC program

CALL-OS PL/I LANGUAGE

CALL-OS PL/I is an efficient problem-solving tool for both commercial and scientific users. It combines many of the features of the full PL/I-F language with the advantages of a remote terminal system and provides the following capabilities:

- A large number of built-in functions and subroutines
- Extended array facilities
- Capability of handling a variety of data types including character strings and complex numeric data
- Flexible stream-oriented input/output facilities
- Free-field format for entering programs

Sample CALL-OS PL/I Program

The problem of solving the roots of the quadratic equation $AX^2+BX+C = 0$ illustrates fundamental CALL-OS PL/I elements and concepts. If $A \neq 0$, this equation always has a solution of the form

$$X = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

A sample procedure for solving this problem follows.

```
ON AT 9:30  05/15/72  MONDAY  PAO LINE 60
USER NUMBER,PASSWORD--ABC123,WXYZ90+
READY

ENTER PL/I
READY

100 EXAMPLE: PROCEDURE;
110 GET LIST(A,B,C);
120 D = B**2-4*A*C;
130 IF D < 0 THEN GO TO ERR;
140 ROOT1=(-B+SQRT(D))/(2*A);  ROOT2=(-B-SQRT(D))/(2*A);
150 PUT LIST
160 (A,B,C,ROOT1,ROOT2);
170 GO TO STOP;
180 ERR: PUT LIST('THE ROOTS OF THIS EQUATION ARE COMPLEX');
190 STOP: END;
RUN
            10:40   05/15/72   MONDAY PAO LINE 60

?7,17,7
 7.00000E+00   1.70000E+01     7.00000E+00     -5.25453E-01
 -1.90311E+00
TIME 1 SECS.

RUN
            10:40   05/15/72   MONDAY PAO LINE 60

?7,-17,7
 7.00000E+00  -1.70000E+01    7.00000E+00    1.90311E+00
 5.25453E-01
TIME 1 SECS.

RUN
            10:41   05/15/72   MONDAY PAO LINE 60

?7,17,80
THE ROOTS OF THIS EQUATION ARE COMPLEX
TIME 1 SECS.
```

Figure 2. Sample CALL-OS PL/I program

Each CALL-OS PL/I statement must end with a semicolon, indicating to the system where the statement ends. Using the semicolon in this manner permits the terminal user to write more than one statement on the same line (see line 140), or to use more than one line to write a single statement (see lines 150 and 160 above).

CALL-OS FORTRAN LANGUAGE

CALL-OS FORTRAN implements most features of the FORTRAN language and provides the following terminal-oriented extensions:

- Statement format, as entered from the terminal, is free-form.
- The characters percent (%), quote (") and up-arrow (↑) have been added to the character set to denote continuation, comments, and exponentiation, respectively. The hyphen (-) character may also be used to denote continuation and may be used interchangeably with the percent character.

- The characters <, ≤, >, ≥, and ≠ have been added to the character set to denote relational operations. The use of the character = has been extended to denote relational operation as well as assignment.

- A syntactic variation of the READ and WRITE statements has been defined to allow free-formatted input/output for terminal and data files.

A line entered using CALL-OS FORTRAN consists of a line number, an optional statement number, and a FORTRAN language statement. A blank must follow the line number. Line numbers serve the same purpose in CALL-OS FORTRAN as in CALL-OS BASIC and CALL-OS PL/I. They order the source statements of the program.

The statement number, if entered, is used for reference within the CALL-OS FORTRAN program. Its actual numerical value does not affect the ordering of program statements during the compilation process. Comments may be entered between statements in the program for purposes of explanation without affecting program execution.

Sample CALL-OS FORTRAN Program

The following program illustrates some of the features of the CALL-OS FORTRAN language.

```
ON AT 10:05 05/15/72  MONDAY  PAO LINE 60
USER NUMBER,PASSWORD--ABC123,WXYZ90+
READY

ENTER FORTRAN
READY

10 "THIS PROGRAM COMPUTES THE REAL ROOTS
20 "OF A QUADRATIC EQUATION
30 10 READ(5,*) A,B,C
40 D=B**2-4*A*C
50 IF(D<0) GO TO 15
60 ROOT1 = (-B+SQRT(D))/%
70 (2*A)
75 ROOT2 = (-B-SQRT(D))/(2*A)
80 WRITE(6,*) A,B,C,ROOT1,ROOT2
85 GO TO 10
90 "IF THE BRANCH IS TAKEN AT THIS POINT
100 "THE ROOTS OF THE EQUATION ARE COMPLEX
110 15 WRITE(6,20)
120 20 FORMAT(' THE ROOTS OF THIS EQUATION ARE %
130 COMPLEX')
140 GO TO 10
900 END

RUN
            10:15   05/15/72   MONDAY  PAO LINE 60

?7,17, 7

7.  17.  7.  -.525454  -1.90312
?7,-17,7

7.  -17.  7.  1.90312  .525454
?7,17,80

THE ROOTS OF THIS EQUATION ARE COMPLEX
?             (Hit Attention Key)
STOP.
RAN 1 SECS.
```

Figure 3. Sample CALL-OS FORTRAN program

CALL-OS BATCH INTERFACE (COBI) OPTION

The program files of CALL-OS can be used for the creation and maintenance of jobs to be submitted to the OS batch-processing environment. When the user has completed preparation of an OS job, he can submit the job to be run in the OS batch-processing environment as though it were entered via the card reader. When the job has been completed, he can scan (print) all or a portion of its output at the terminal.

COBI input simulates data punched in 80-column cards. A CALL-OS program file created to be sent to OS batch processing is shown in Figure 4. One space or tab character must follow each line number. The next position is treated as though it were column 1 of an input card.

```
100 //AAA123XY JOB (.....,...),etc.
110 //ABC  EXEC  ASMFCLG
120 //SYSIN DD *
130 CNVT  TITLE 'THIS ROUTINE IS A HEX CONVERSION ROUTINE'
140 HEXCNVRT  CSECT
150   ...
160   ...
...
...
...
300  END
310 /*
```

Figure 4. Sample CALL-OS JOB to be submitted for OS batch processing

When input is initially entered into CALL-OS via a 2741 Communications Terminal or Teletype, the tabulation function available with this terminal can be used 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 correspond to the label, operation, operand, comments, and continuation fields of a source-program statement line. The entry 73-80 in the $$TABS control statement indicates that each CALL-OS program-file line number is to be converted to an eight-digit number with leading zeros and placed in columns 73-80 of the output record corresponding to that line. Thus, the output is formatted to be compatible with IEBUPDTE and other OS utilities.

For this $$TABS control statement, 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.

If the user presses the TAB key on the 2741 when the type element is positioned between the left margin and eleventh position on the line, the next character will be placed in the eleventh position of the line, and so on. When the user enters the tab character (control I) from the Teletype keyboard, neither printing nor spacing occurs at the terminal. A $$TABS control statement must precede the initial use of tab characters it is to control in the input stream.

Use of the tabulation function permits efficient skipping of columns which are to be blank (during initial keying of input). Tab characters are automatically placed in the internal record for later deblocking control, the input is compressed to an optimum reduced format, and disk storage requirements are lessened accordingly. In addition, the readability of subsequent printer output is enhanced through use of the tabulation function.

The functions of CALL-OS terminal commands designed specifically for COBI users are explained in Appendix B. For additional information, see the CALL-OS Terminal Operations Manual.

RELATIONSHIP TO OS

All OS concepts and facilities as explained in IBM Systems Reference Library publications are available to the installation using CALL-OS. Program design and job, data, and task management features of OS are fully operational.

The CALL-OS system is considered a normal job by the OS job management facilities. It is initiated by job control language (JCL) statements. Once loaded, the CALL-OS resident modules remain in core during the CALL-OS period of operation. Transient routines are called for as needed by CALL-OS overlay management facilities. After the CALL-OS job is terminated, control of its resources is returned to OS.

CALL-OS has the facility of segmented operation in high- and/or low-speed core under OS Hierarchy Memory Support. It will benefit from instruction retry but will not benefit from the remaining facilities offered by the OS Recovery Management System.

CALL-OS operates in the task environment of OS. The CALL-OS task is divided into two segments. One segment, called the Executive area, is dedicated to CALL-OS system modules and subroutines; the other, called the user program area, is dedicated to compilation and execution of user programs. Figure 5 is a simplified diagram of the structure of core when the CALL-OS task is operating.

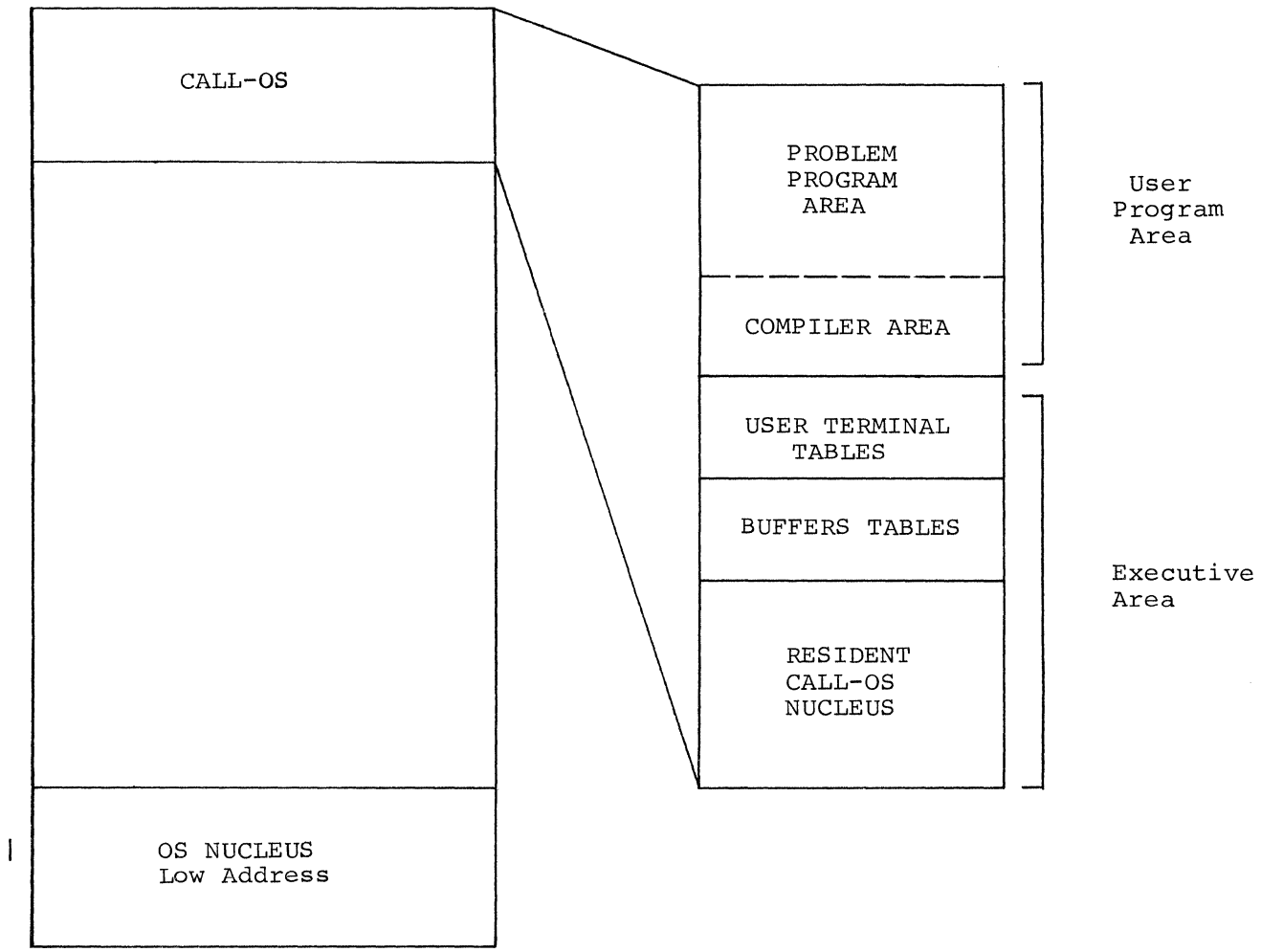

Figure 5. Core relationship of OS and CALL-OS

DATA BASE CONCEPTS

The CALL-OS data base consists of three logical sections:

- The CALL-OS index, which is a one-track data set that identifies all data sets used by the CALL-OS system. This data set is created and modified by CALL-OS utility programs, which are run under OS when the CALL-OS time-sharing system is not operating. The data set is also used by CALL-OS at initialization time to verify that JCL in the system startup deck is complete and accurate.
- System data-base data sets, which are used for compiler storage, work/swap areas, and overlay modules
- User data-base data sets, which contain user catalogs, source programs, object programs, data files, and shared libraries

In addition, the CALL-OS Batch Interface (COBI) option utilizes four OS data sets. The first of these provides an index containing job status information. The second provides storage for requested JCL. The remaining two are used alternately by CALL-OS and OS. They provide storage of jobs submitted for OS batch processing and are read by OS using a standard OS reader procedure.

The organization of user data-base data sets is based on three logical groupings: a set of 99 users (subscription group, or sub group), a set of sub groups (user group), and a set of user groups (cluster).

A sub group consists of 99 users whose six-character user numbers are identical for the first four characters. Each user number that is validated with the numeric digits 00 as its last two characters creates a dummy user catalog (called a directory) for use by the entire sub group. This directory is then available for the pooling and use of shared program and data files by all members of the same sub group. Thus, for example, when user number ABC600 is validated, a directory is created and available for pooling of program and data files by all users having user numbers in the range ABC601 through ABC699. The sub group is the smallest set of user numbers for which pooling is allowed. The directory is referred to as the *Library (single-star library) for that sub group.

Note: Two other libraries are designed for pooling and use of shared program and data files in CALL-OS. These are the **Library and ***Library. Any program and data files pooled in these libraries are available for system-wide usage. To ensure data set integrity, a program entered into any library can be marked as run only (by means of a PROTECT command). The program can be executed by other users, and it can be merged with other programs, but it cannot be listed, stored, or saved. A program resulting from a merge involving a protected program is also protected and subject to the restrictions this status implies. A pooled data file can be protected to prevent users other than the creator of the file from opening it for input prior to or during a required update process. When a file need no longer be protected, an ALLOW command can be issued to permit other CALL-OS users to access the file.

A user group consists of a range of user numbers (such as AAA000 through CCC999). This range can include as few as 1000 user numbers (AAA000 through AAA999) or more than 17 million of them (AAA000 through ZZZ999). Each user group is defined by the user installation to include a set of related users. Only user numbers ending with 000 or 999 can serve as bounds. Each CALL-OS user group is assigned a collection of single-volume OS data sets (not all of which need reside on the same volume). These data sets are used for the storage of catalogs, programs, and data for all users in the group. A given user is assigned disk storage space only from the data sets assigned to his group, thus ensuring data set integrity.

To permit a user installation to group its users in alternate ways, and/or to allow a user group to be allocated alternate data sets for backup purposes, CALL-OS additionally defines each organization of user groups as a cluster. Two clusters may be defined, either of which may be activated for a particular session of CALL-OS. The same users and the same user group organization may be included in each cluster, or one cluster may differ from the other.

The CALL-OS data base may be built and organized to fit the specific needs of any user installation. It may reside on 2314, 2319, or 3330 disk storage units, or on any combination thereof. To meet its immediate needs, a new installation may choose one of three default data

bases to use until defining its own data base according to its objectives. The default data base may use one pack, two packs, or three packs. The installation specifies pack volume ID's and the number of lines to be supported.

All default data bases include CALL-OS FORTRAN, CALL-OS PL/I, and CALL-OS BASIC. The number of packs chosen determines the number of user groups to be defined. For example, if the two-pack default data base is used, two user groups, AAA000-MZZ999 and NAA000-ZZZ999, are defined.

The default data base allows an installation to initialize CALL-OS using a proven, easy method. The system can be placed in operation within a minimum amount of time.

SYSTEM PROCESSING

CALL-OS provides its own internal task switching. The OS Input/Output Supervisor (IOS) is utilized at the execute channel program (EXCP) level. CALL-OS also supplies its own data management facilities.

EXECUTIVE FUNCTIONS

The CALL-OS Executive operates in the Supervisor state and utilizes protect-key zero. Compilers and user programs operate with the protect key assigned by OS to the CALL-OS task area. This approach ensures the integrity of the CALL-OS system and of OS.

The Executive performs the following functions (invoked by the compilers and user programs through the use of a Type 1 SVC).

- Handles the CALL-OS overlay facilities, invoking nonresident modules as needed
- Performs multiprogramming within the CALL-OS task area
- Queues and dequeues direct-access I/O requests
- Allocates direct-access storage space within CALL-OS data sets
- Schedules and allocates for compilation and execution of programs
- Allocates and manipulates three types of files on 3330/2319/2314 disk storage:

  Source-program files consisting of lines beginning with line numbers

  Object-program files created by the STORE command

  Data files defined by the FILE command
- Analyzes and executes terminal commands entered by the terminal user
- Analyzes and executes operator commands (*commands) entered from a command console
- Allocates and manipulates four COBI-related files, or data sets, on 3330/2319/2314 disk storage:

  COBI index data set used for recording submitted OS jobs

  JCL data set used for storing JCL of completed OS jobs

Two SYSIN data sets used for OS input job stream

These tasks are performed on a priority basis such that work requiring the fastest response time is assigned the highest priority.

The Executive maintains control of terminal lines by means of information in the user terminal table (UTT). There is one UTT for each terminal line available at system initialization. It contains all information pertinent to a specific line.

TIME SLICING

Time slicing is a technique whereby each terminal job is allowed to execute for only a fixed length of time (a time slice).

Within CALL-OS, CPU time is allocated according to two priorities. Higher priority is given to jobs that may complete execution within an initial time slice (new jobs). Lower priority is given to jobs whose execution has exceeded an initial time slice (old jobs). The percentage of CPU time given to each priority level within CALL-OS is determined by the installation at system initialization.

Time slices for job execution do not include compilation time. A separate time slice, the length of which is dependent on the compiler being used, is given to the compiler for compilation of the job. The length of the compiler time slice is variable and can be specified by the installation at system initialization.

An additional option available to the user installation permits a percentage of the times allotted (1) for old-job execution, or (2) for old-job execution, new-job compilation, and new-job execution to be reassigned to OS background jobs. The compilation and execution times of CALL-OS jobs and of OS background jobs are affected accordingly. Thus, time slicing within CALL-OS can be defined to meet the needs of an installation.

JOB QUEUE MANAGEMENT

Two queues are used within CALL-OS to control the compilation and execution of jobs--the new job queue and the old job queue. Both are serviced primarily on a first-in, first-out basis.

An entry is made for a job at the end of the new job queue for one of the following reasons:

- A RUN or STORE command is entered.
- Compilation of the job is not completed within the compiler time slice.
- Data is entered in response to an input request.
- A user output buffer for the job is filled.

When a job enters execution after completion of compilation, it is given a full new-job time slice. It continues to execute until it requests input, or the output buffer is full, or the new-job time slice expires.

If execution of the job is completed within the initial new-job time slice, the job is ended. If execution is not completed, an entry for the job is placed at the end of the old job queue. Each time the entry reaches the top of the old job queue, the job is allowed to execute for

an old-job time slice. If the job is not finished, and is not entered in the new job queue because an input request is made or an output buffer is filled, another entry for the job is made at the end of the old job queue.

A job in either queue is read into the user program area of core when it is ready to be compiled and/or executed. At all other times, the job resides in the user's work/swap area on disk storage.

Figure 6 illustrates a job flow through the system in response to either a RUN command or a request for input data.

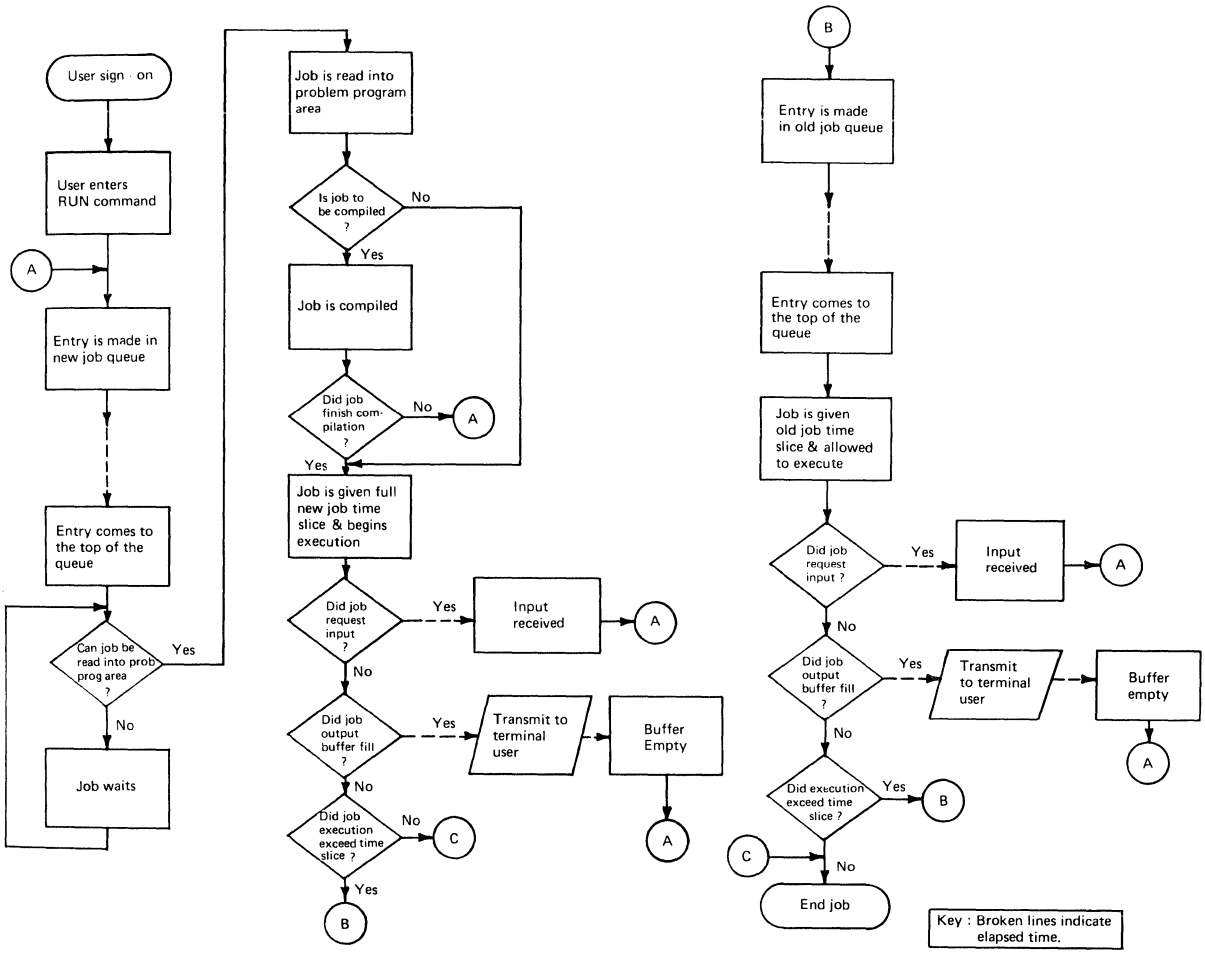

Figure 6. Job flow diagram

SYSTEM BUILD AND INITIALIZATION

An IBM Operating System generated to support the CALL-OS system must include the following optional control-program features:

- Storage protection
- Interval timing
- CALL-OS Type 1 SVC

If CALL-OS is used in an MFT environment, the OS subtasking option is required.

If the 2702 Transmission Control is used, the appropriate SAD command (SAD0, SAD1, or SAD2) must be specified.

A recommended feature that may be included is trace.

In addition, CALL-OS permits the user installation to:

- Specify the system resources (for example, main storage, direct access storage devices) to be allocated to CALL-OS
- Format the system data base
- Specify the number of terminal lines to be supported
- Specify which of the supported language processors are to be part of the active CALL-OS system
- Format data sets required for the COBI option

The CALL-OS system is started via the normal OS job and task management facilities. At the time CALL-OS is loaded into core, the user installation can:

- Select resident and nonresident modules from a set of previously established combinations
- Select which of the language processors included at system-build time are to be available during this period of CALL-OS usage
- Select time-slice lengths for new job execution, old job execution, background job execution, and compiler time slices
- Eliminate CALL-OS trace entries
- Select which portions of the CALL-OS system are to be resident in high- or low-speed core

If the COBI option is included at system-build time, the installation can:

- Specify that COBI is to be inactive during this period of CALL-OS usage
- Change various COBI default options

After the system has been initialized, a message to indicate that the system is ready is displayed on the OS system console.

SYSTEM TERMINATION

The CALL-OS means for normal termination is the *OFF operator command. This command is used as the final step in system shutdown. All CALL-OS closeout procedures are invoked, the OS environment is restored, and a final return to OS is made.

In the event of a system failure, there are no built-in restart procedures. The operator at the central computer installation must reinitialize the system.

ACCOUNTING

CALL-OS accounting facilities enable each CALL-OS user to obtain the following information (per user number):

- Accumulated terminal connect time
- Accumulated CPU time
- Maximum disk storage space used
- Tape journal of accounting information
- Printed journal of monthly billing information (output to the system print unit)

The user can request that accounting information be reset to zero. Normally, this is done after monthly billing information is provided.

STATISTICS

A statistical report can be obtained by entering a *REPORT operator command at a command console. The report is directed to the SYSPRINT data set and contains information regarding current operational characteristics of the CALL-OS system. Statistics such as the following are provided:

- Total number of disk tracks assigned to each user group
- Number of disk tracks currently available to each user group
- Number of calls received on each terminal line
- Number of lines in use

DATA BASE BUILD AND MAINTENANCE

CALL-OS utilities perform the following CALL-OS data base manipulation and maintenance functions:

- Format the CALL-OS index to X'FF's (hexadecimal) prior to indexing the data-base data sets
- Initialize compiler, work/swap, overlay, system-group, and user-group data sets and make corresponding entries in the CALL-OS index

- Maintain the CALL-OS index when reorganization of the system data base is desired
- Reorganize the system group or one or more user groups of one cluster of the CALL-OS data base into the system group or a user group of the other cluster from-to: 2316/2316, 2316/3330, 3330/2316, 3330/3330
- Re-create part or all of a CALL-OS data base from a backup tape
- Convert CALL-OS program files to a format compatible with OS batch compilers and convert CALL-OS data files to OS data sets; converted program files are partitioned data sets in card-image format, and converted data files are sequential data sets in fixed, fixed blocked, variable, or variable blocked format
- Punch program and data files in card format, converted to a format compatible with OS batch compilers
- List program files, data files, and control information from the CALL-OS data base
- Delete any CALL-OS data-base records (program files, data files, directory entries, or catalog entries) on a selective or comprehensive basis, for a single user or a range of users
- Insert or replace a program or data file in the CALL-OS data base by means of card, tape, or disk input
- Write all or a portion of a CALL-OS data base to a backup tape
- Validate a single user or a range of users
- Cancel a single user or a range of users
- Provide user disk-storage space-usage accounting statistics
- Update a user catalog with respect to current COBI jobs

These functions permit a great deal of flexibility in the organization, manipulation, and maintenance of the CALL-OS data base. They are invoked by means of user-provided control cards during normal OS batch processing. Security is maintained by means of an installation-controlled password.

PAPER TAPE FACILITIES

Once entered into CALL-OS by means of the 2741 Communications Terminal (Correspondence or EBCD) or a Teletype unit, Type 33 or 35, infrequently used programs need not necessarily be carried in online storage. A complete program can be stored on paper tape and reentered into CALL-OS whenever it is required.

A complete program, CALL-OS terminal commands, and run-requested data can be punched in paper tape for subsequent entry into CALL-OS by either offline or online punching operations. For the latter, the user enters a PUNCH ON command causing a required line-ending character sequence to be appended to each line of listed or runtime output. Later, when the paper tape is to be read into CALL-OS, the user issues a TAPE ALL command. It is possible to read, compile, and execute the program without operator intervention. See the CALL-OS Terminal Operations Manual for specific examples of CALL-OS paper tape facilities.

Notes: 1. The 2741 has a larger character set than either Teletype unit. Therefore, some characters of a 2741 program may not be meaningful when displayed as Type 33 or 35 printed output. However, this has no effect on the punched output produced as a result of punching operations.

2. Readjusting line width may be necessary if a program created at a 2741 is listed at the Teletype (see WIDTH command).

APPENDIX A: CALL-OS TERMINAL COMMAND LANGUAGE SUMMARY

The terminal user tells the computer, via CALL-OS terminal commands, what he wants the computer system to do. A summary of these commands follows. Most of the commands can be invoked by specifying only their first three characters (and any required parameters). All exceptions are noted in the descriptions below. A complete description of the CALL-OS terminal command language is given in the CALL-OS Terminal Operations Manual.

- ADD

  Adds a specified string of characters to the end of a single line, the end of each of a group of consecutive lines, or the ends of all lines in the program in the user work area.

```
ADD 10,30,55,'THIS'                    (terminal entry)
READY                                  (system response)
```

  The word THIS is added at the ends of lines 10, 30, and 55.

```
ADD 10 THRU 30,55,'THIS'               (terminal entry)
READY                                  (system response)
```

  The word THIS is added to lines 10 through 30 inclusive and to line 55.

```
ADD ALL,'THIS'                         (terminal entry)
READY                                  (system response)
```

  The word THIS is added to all lines of the program in the user work area.

- ALLOW program or data filename

  Removes protection from a program or data file whose filename is in a CALL-OS system library. (See PROTECT command.)

```
ALLOW PROG1                            (terminal entry)
READY                                  (system response)
```

- CATALOG

  Prints a list of program and data files stored in the user's library.

```
CATALOG                                (terminal entry)
16:18   05/18/72   THURSDAY            (system response)
PROG1    PROG6
```

- CATALOG *
  CATALOG **
  CATALOG ***

  Prints a list of the program and data filenames in the designated system library.

```
CATALOG *                                              (terminal entry)
16:18   05/18/72   THURSDAY                            (system response)
PROG4    MYPROG     STATPROG     NEWDATA    /OBJP
```

- CATALOG ALL

  Instructs the system to print a list of the program and data files in the user's library. The command may be given in full or abbreviated CAT ALL. The example below illustrates the descriptive information that is provided.

```
CATALOG ALL                                            (terminal entry)
16:18   05/18/72   THURSDAY                            (system response)
FILENAME  TYPE  LANGUAGE  PROGRAM SIZE  FILE UNITS  DAYS SINCE USED

PROG1     PROG  BASIC            599                        6
PROG2     PROG  BASICL           975                       27
PROG3     PROG  PL/I             670                       55
FORECAST  DATA  PL/I                          10             55
/PROG4    OBJ   FORTRAN                       12             17
```

- CLEAR

  Clears the user work area.

```
    CLEAR                                              (terminal entry)
    READY                                              (system response)
```

- DELETE

  Deletes one or more lines from the program currently in the user work area, through references to line numbers. Single lines or selected groups of lines may be deleted.

```
    DELETE 10,30,65,80                                 (terminal entry)
    READY                                              (system response)
```

  The single lines with line numbers 10, 30, 65, and 80 are deleted from the program in the user work area.

  Note: A single line can also be deleted by entering its line number, followed by a carrier return.

```
    DELETE 10 THRU 100,120 THRU 130                    (terminal entry)
    READY                                              (system response)
```

  The system deletes the lines with line numbers 10 through 100 and 120 through 130 inclusive.

- ECHO

  Checks transmission between the terminal and the computer system. The text entered by the user should be printed out at the terminal.

```
    ECHO    TEST    TRANS                              (terminal entry)
    ECHO    TEST    TRANS                              (system response)
```

- ECHOX

  Checks transmission between the terminal and the computer system. The terminal should print exactly what was entered, repetitively, until the terminal user presses the ATTN or BREAK key (on the 2741 or Teletype, respectively). This command cannot be abbreviated.

```
ECHOX   TEST    TRANS                   (terminal entry)
ECHOX   TEST    TRANS                   (system response)

ECHOX   TEST    TRANS

ECHOX   TEST    TRANS
  .       .       .
  .       .       .
  .       .       .
```

- ENTER language-name

  Selects the programming language to be used to compile a program. Valid language names are PL/I, FORTRAN, BASIC, BASICL, and DATA. 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. The information can be retained in a user's library by means of a SAVE command; the language-name DATA is retained with the file.

```
ENTER PL/I                              (terminal entry)
READY                                   (system response)
```

  Upon initial entry (sign-on) to CALL-OS, BASIC is assumed.

- EXTRACT

  Extracts lines from a program. Single lines or selected groups of lines may be extracted, using line numbers as references. All other lines are cleared from the user work area.

```
EXTRACT 10,15,25,40                     (terminal entry)
READY                                   (system response)
```

  The system retains only the lines with line numbers 10, 15, 25, and 40 of the program in the user work area.

```
EXTRACT 10 THRU 120,180 THRU 220        (terminal entry)
READY                                   (system response)
```

  The system retains the lines with line numbers 10 through 120 and 180 through 220, inclusive.

- FILE filename[,n]

  Names a data file to be used with a program at runtime, increases the maximum number of disk storage units that can be allocated to an existing data file, or determines the maximum number of disk storage units that can be allocated to an existing data file.

  If the command is being used to name a new data file, n specifies the maximum number of disk storage units to be allowed for the file. If n is omitted, a value of four is assumed. Thus, DATA1

is established as a new file for which a maximum of seven disk storage units will be allocated as shown below.

```
FILE DATA1,7                                    (terminal entry)
READY                                           (system response)
```

If the command is being used to increase the permissible maximum amount of storage that can be allocated for a file, n specifies the maximum number of disk storage units to be allowed. The entry n must be specified and greater than the current n. Assuming that DATA2 is an existing file for which less than nine storage units have been specified, the following command can be used to increase the maximum number of disk storage units that can be allocated for the file to nine.

```
FILE DATA2,9                                    (terminal entry)
READY                                           (system response)
```

If the command is being used to determine the maximum amount of storage that can be allocated for a file, no entry should be given for n. Assume, for example, that the user wishes to know how many disk storage units can be allocated for the existing file DATA1. The appropriate terminal entry and the system response generated for this entry are shown below, where xx is the total number of storage units that can be allocated for the file.

```
FILE DATA1                                      (terminal entry)
DATA1 LIMIT = xx UNITS                          (system response)
```

- FIND

Locates the use of a string of characters throughout a program or in a specified line or lines of the program. A character string enclosed in single quotes in the FIND command is located wherever it appears in the specified search area. A string enclosed in double quotes (a qualified character string) is located only when bounded at either end by one of a set of delimiting characters. If the optional parameter NOTEXT (or its abbreviated form, N) is specified in the FIND command, only the line numbers of lines in which the string is found are listed at the terminal. If the optional parameter is omitted, the entire text of each line in which the string appears is printed at the terminal.

```
FIND 10 THRU 30,50 THRU 75,'ABCD',N     (terminal entry)
10      25      65                      (system response)
READY
```

Lines 10 through 30 and 50 through 75, inclusive, are searched for the character-string ABCD. This response indicates that the string of characters is contained in lines 10, 25, and 65.

```
FIND ALL,"X1"                           (terminal entry)
30  X1 = B + C;                         (system response)
70  ROOT = TOTAL/X1;
    . . .
    . . .
    . . .
```

All lines of the program are searched for the qualified character string X1. The text of all lines containing X1, appropriately delimited, is printed at the terminal.

- HELP

  Instructs the user to take installation-determined action.

```
HELP                                          (terminal entry)
PLEASE CONTACT YOUR INSTALLATION MANAGEMENT FOR ASSISTANCE
```

- INSERT

  Inserts a specified line in the program currently in the user work area. The new line is assigned the line number specified in the INSERT command. If multiple line numbers are specified, the new line is inserted at the location indicated by each specified line number. Any current line having a specified line number is deleted.

```
INSERT 10,15,'PRINT X'                        (terminal entry)
READY                                         (system response)
```

  The statement PRINT X is assigned line number 10 and inserted in the current program. A duplicate statement is assigned line number 15 and inserted at its appropriate point in the program.

- KEY

  Restores all functions which were inhibited by the TAPE or TAPE ALL command in order to permit paper tape input from the Teletype (see TAPE); should be entered only from Teletype.

```
KEY                                           (terminal entry)
READY                                         (system response)
```

- LIST

  Prints all or part of a program currently in the user work area.

```
LIST                                          (terminal entry)
LIST 120                                      (terminal entry)
```

  In the first example, the system responds by printing the complete program. In the second example, the program is listed beginning with line number 120. Either listing begins with a heading line that contains the name of the program and the current date.

- LIST-NO-HEADER

  Identical to LIST except that the heading line is not printed. The command may be given in full or abbreviated LIST-N.

```
LIST-NO-HEADER                                (terminal entry)
LIST-N 120                                    (terminal entry)
```

- LIST-TEXT

  Identical to LIST except that line numbers do not print; only text is provided as output. The command may be given in full or abbreviated LIST-T.

```
LIST-TEXT                                   (terminal entry)
LIST-T 120                                  (terminal entry)
```

Note: The user can combine use of the LIST-N and LIST-T commands by specifying either LIST-TN or LIST-NT. Either causes the program in the user work area, beginning with a specified line number (if any), to be printed without headers and without line numbers.

- LOAD program-name

  Loads the named source program from the user's library into the work area so that it can be examined or modified. The work area is automatically cleared before loading.

```
LOAD MYPROG                                 (terminal entry)
READY                                       (system response)
```

- LOAD *program-name
  LOAD **program-name
  LOAD ***program-name

  Loads the named source program from the system library designated by asterisks into the work area, clearing the work area automatically before loading.

```
LOAD *STARP                                 (terminal entry)
READY                                       (system response)
```

- LOCK program or data filename

  Prevents a program or data file in the user's library from being accidentally destroyed by any CALL-OS user. A named program cannot be deleted from the user's 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.

```
LOCK PROG1                                  (terminal entry)
READY                                       (system response)
```

- LOGON

  This command is used to:

  1. sign-on at a 2741 Communications Terminal after successfully dialing in the system, or

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

  In the former case, the system determines the type of the 2741 Communications Terminal (Correspondence or EBCD) by analysis of the transmission codes generated by the LOGON or LOG command. It prints out a sign-on message in correspondence or EBCD code 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 at the terminal accordingly.

In the latter case, the user types LOGON or LOG without disconnecting the line to the system. The system responds by printing the off message, time, and sign-on message at the terminal. The terminal is then used by a second user to sign-on the system from the same terminal without performing the disconnect sequence.

In either case, the user then enters a user number and password. After the LOGON command has been completed successfully, 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 72.

```
LOGON                                          (terminal entry)
OFF AT 14:38                                   (system response)
PROC. TIME...   2 SEC.
TERM. TIME...  17 MIN.

ON AT 14:39    05/15/72  MONDAY  PAO LINE 60
USER NUMBER,PASSWORD--ABC213,WXYZ+90
READY
```

- MERGE

Allows the user to merge from two to nine source programs into a single combined program. Any source programs available to the user, including those in system libraries, may be merged. If a run only (protected) program is selected for a merge operation, the entire resulting program is assigned a run only status and is subject to all restrictions that this status implies.

Each subprogram is renumbered to fit in its indicated place in the main program. If possible, an integral value which allows the entire subprogram to fit between successive bordering main-program statements is selected for use as an increment. If this is not possible, the subprogram is renumbered using an increment of 1. If the renumbering causes existing main program line numbers to be overlapped (that is, if new subprogram line numbers are greater than or equal to existing main program line numbers), successive main program lines are renumbered until overlapping is eliminated. As far as possible, existing main program line numbers are retained in the resulting program.

```
MERGE PROG1,PROG2,10,PROG3,200          (terminal entry)
READY                                   (system response)
```

PROG2 is inserted into PROG1, starting after the line with line number 10 in PROG1, and PROG3 is inserted into PROG1, starting after the line with line number 200 in PROG1.

- MOVE

Moves a single line or block of consecutive lines to a new location in the working program. The new location cannot be within the range of the lines to be moved. If an increment is specified as a third parameter of the MOVE command, the moved lines (and succeeding lines as necessary to prevent line-number overlap) are renumbered using this increment. If no increment is specified, a default of 1 is assumed.

```
MOVE 50,20                                     (terminal entry)
READY                                          (system response)
```

Line 20 is moved to follow line 50 and renumbered 51.

```
MOVE 50,10 THRU 20,10                    (terminal entry)
READY                                    (system response)
```

Lines 10 through 20, inclusive, are moved to follow line 50 and renumbered using an increment of 10 (60,70,...).

- NAME program-name

  Assigns a name to the program in the user work area. A program name is required for a program that is to be saved or stored.

```
NAME PROG1                               (terminal entry)
READY                                    (system response)
```

- OFF

  Causes the user's terminal to be disconnected from the computer system. CPU processing and terminal connect times for the session are printed. (A session is the total period of time from user sign-on to user sign-off.)

```
OFF                                      (terminal entry)
OFF AT 17:11                             (system response)
PROC. TIME...    11 SEC.
TERM. TIME...    24 MIN.
```

- PASSWORD password

  Permits a terminal user to change his password.

```
PASSWORD NEWPASS                         (terminal entry)
READY                                    (system response)
```

- POOL *filename
  POOL **filename

  Permits a user's saved or stored program or data file to be made available to other CALL-OS terminal users, through the shared library facility. Users other than the pooling user can use a shared data file only for input.

```
POOL *PROG1                              (terminal entry)
READY                                    (system response)
```

- PROTECT program or data filename

  Specifies that a program or data file pooled into a CALL-OS system library is protected. A protected program is assigned a run only status. It may not be listed, saved, or stored by any CALL-OS terminal user other than its originator. A protected data file may not be opened by anyone other than its originator. Thus, a CALL-OS user can prevent use of his file prior to or during a required update process.

```
PROTECT PROG1                            (terminal entry)
READY                                    (system response)
```

- PULL *filename
  PULL **filename

  Removes a named program or data file from a CALL-OS system library, so that it is no longer available to other terminal users.

```
PULL *PROG6                                (terminal entry)
READY                                      (system response)
```

- PUNCH OFF

  Resets the paper-tape punch mode-switch so that a conventional two-character line-ending sequence (C/R, L/F) is appended to each line of listed or runtime output (see PUNCH ON). The command may be abbreviated PUN OF. It is meaningful only when entered from the Teletype.

```
PUNCH OFF                                  (terminal entry)
READY                                      (system response)
```

- PUNCH ON

  Causes a six-character line-ending sequence to be appended to each line of listed or runtime output. These characters (C/R, L/F, X-OFF, and three RUBOUT characters) are required to produce paper tape (program statements, terminal commands, and program-requested data) reenterable in TAPE ALL mode. The command may be abbreviated PUN ON. It is meaningful only when entered from the Teletype.

```
PUNCH ON                                   (terminal entry)
READY                                      (system response)
```

- PURGE filename

  Deletes the named program or data file from the user's library unless the program or data file is locked. It also deletes any references to the program in system libraries.

```
PURGE MYPROG                               (terminal entry)
READY                                      (system response)
```

- RELEASE filename

  Allows users other than the creator of a file to copy any part or all of the file by means of a CALL-OS utility. (If this command is not issued, the SECURE attribute for the file is set and remains set automatically; the file cannot be copied by other users via a CALL-OS utility.)

```
RELEASE PROG1                              (terminal entry)
READY                                      (system response)
```

- RENUMBER

  Renumbers the lines of the program in the user work area. A new line number, old line number, and increment can be specified. One, two, or all parameter entries may be omitted; defaults of 100, 0, and 10 for the new line number, old line number, and

increment, respectively, are assumed. The comma that otherwise follows a parameter should be retained if specified parameters follow.

```
RENUMBER 200,10,20                          (terminal entry)
READY                                       (system response)
```

The system assigns line number 200 to the line identified by line number 10 in the user work area. Subsequent lines are renumbered 220, 240, and so on.

- REPLACE

  Replaces a first-named character string by a second-named character string wherever it occurs in a specified line, consecutive group of lines, or complete program. If only one string is specified, the string is deleted wherever it occurs in the specified lines or in the complete program.

```
REPLACE 10,25 THRU 50,'THIS','NEW'      (terminal entry)
READY                                   (system response)
```

  The characters THIS are replaced by NEW wherever THIS occurs in line 10 and lines 25 through 50, inclusive.

```
REPLACE ALL,'THIS'                      (terminal entry)
READY                                   (system response)
```

  The characters THIS are deleted wherever THIS occurs throughout the program in the user work area.

- RUN

  Causes compilation and execution of the program currently in the user work area. When serious errors are detected in program statements, error messages are printed at the terminal; then the compilation/execution process is terminated. Statements in error may be corrected, and the RUN command may be entered again.

```
RUN                                                    (terminal entry)
PROG1        16:18    05/18/72    THURSDAY (system response)
```

- RUN program-name

  Causes the named program to be located in the user's library. A source program is loaded, compiled, and executed. A program in object form is executed directly from the library. If a source program contains serious errors, subsequent actions are as described above. Either source or object programs can be rerun immediately without reentering the program name.

```
RUN PROG1                                              (terminal entry)
PROG1        16:18    05/18/72    THURSDAY (system response)
```

- RUN *program-name
  RUN **program-name
  RUN ***program-name

  Causes the named program to be located in the system library designated by asterisks. A source program is loaded, compiled, and executed. A program in object form is executed directly from

the library. Either source or object programs can be rerun immediately without reentering the program name.

```
RUN *PROG1                                    (terminal entry)
PROG1       16:18   05/18/72   THURSDAY (system response)
```

- SAVE [program-name]

  Writes a source program from the work area into the user's library. The program-name parameter need not be specified if the program in the work area has already been assigned the name to be associated with it.

```
SAVE                                          (terminal entry)
READY                                         (system response)
```

- SECURE filename

  Sets the SECURE attribute for a file so that it cannot be copied by any CALL-OS utility user other than the creator of the file.

```
SECURE PROG1                                  (terminal entry)
READY                                         (system response)
```

- STATUS

  Prints information about the program currently in the user work area. The amount of terminal (connect) time and processor (CPU) time used during the user's current session are also printed.

```
STATUS                                        (terminal entry)
TERMINAL NO..32                               (system response)
USER NO..ABC123
PGM NAME..EXAMPLE
PGM LENGTH..4823 CHARS.,191 LINES
LANGUAGE..FORTRAN
TERMINAL TIME..33  MIN
PROCESSOR TIME..3  SEC
```

- STORE [program-name]
  STORE ?

  Compiles the source program in the work area and stores it in object format in the user's library and prints the number of disk storage units (each 3440 bytes in length) required. When the second form is used, the number of disk storage units that would be required to store the object program is printed, but the program is not stored. If the program-name parameter is not specified in the first form of the command, and the current source program name is not more than seven characters in length, the object program is stored under a name consisting of a slash followed by these characters.

```
STORE                                         (terminal entry)
/PROG1      10:15   05/26/72   FRIDAY   (system response)
OBJ PROG STORED - 12 STORAGE UNITS REQUIRED
TIME 8 SECS.
```

```
STORE ?                                          (terminal entry)
/PROG2      10:40   05/26/72   FRIDAY   (system response)
10 STORAGE UNITS REQUIRED TO STORE /PROG2
TIME 8 SECS.
```

- TAPE

  Allows input of source-program statements from paper tape using the Teletype. After this command has been issued, output line feeds, carrier returns, and messages are inhibited. Only the TAPE ALL, OFF, and KEY commands are acted upon. All other commands and unnumbered statements are ignored.

```
TAPE                                             (terminal entry)
READY                                            (system response)
```

- TAPE ALL

  Allows input of source-program statements, terminal commands, and data from paper tape via the Teletype on a line-by-line basis. The command may be given in full or abbreviated TAP A. If the Type 33 or 35 Reader Control Arrangement feature is installed, the paper tape is restarted automatically whenever it is halted after one of the following:

  1. A source-program statement
  2. A successfully completed LIST, LIST-NO-HEADER, LIST-TEXT, or RUN operation
  3. A terminal command that terminates with a READY response to the user
  4. A request for input from an executing program (the system prints a question mark at the terminal)

  In any other situation, the system awaits an operator decision to manually restart the tape or to enter data from the keyboard.

```
TAPE ALL                                         (terminal entry)
READY                                            (system response)
```

- TIME

  Causes the following month-to-date information for the user to be printed at the terminal: CPU time, terminal time, and maximum number of disk storage units in use at any time during the month. (Current session time may be obtained by using the STATUS command.)

```
TIME                                             (terminal entry)
PROCESSOR TIME MONTH-TO-DATE..74  SEC.   (system response)
TERMINAL TIME MONTH-TO-DATE..570  MIN.
DISK STORAGE MONTH-TO-DATE..0  UNIT(S)
```

- UNLOCK program or data filename

  Removes protection from a previously locked program or data file in the user's library.

```
UNLOCK PROG1                                     (terminal entry)
READY                                            (system response)
```

- WEAVE

  Weaves up to nine source programs together according to the existing line numbers of the program statements. Each program must be saved in a library available to the user. None of the programs may be run only (protected); if a protected program is named, the WEAVE command is rejected. The combined program resides in working storage. All lines retain their original line numbers. If lines have duplicate line numbers, the line from the rightmost of the programs specified in the WEAVE command and having this line number is retained. Other duplicate-numbered lines are deleted.

```
WEAVE PROG1,PROG2                    (terminal entry)
READY                                (system response)
```

  Assume that PROG1 contains lines 10, 30, and 80 and that PROG2 contains lines 20, 30, 40, and 45. This command yields a program with line numbers as follows: 10, 20, 30, 40, 45, 80. Line 30 of PROG1 is eliminated.

- WIDTH

  Sets the width of a print line to be used for printing information at the terminal. This command is used only if the line width desired is other than 72 positions. The line width specified by the WIDTH command remains in effect during the current session unless another WIDTH command is given. Although the program accepts a maximum of 255 positions, the actual, effective maximum number of positions is the maximum line width for the particular terminal. Thus, effective maximum limits are 130 positions for the 2741 with ten positions per inch and 72 positions for the Teletype.

```
WIDTH 130                            (terminal entry)
READY                                (system response)
```

APPENDIX B: CALL-OS TERMINAL COMMAND LANGUAGE FOR COBI

The CALL-OS Batch Interface (COBI) option permits CALL-OS terminal users to create OS jobs and submit those jobs to the OS batch-processing environment. At this time also, the user may specify that job control language (JCL), procedure-defined SYSOUT data sets, and user SYSOUT data sets of the job be saved for subsequent scanning (printing) at the terminal.

Each job is assigned an internal job number of the form #nnnnn by the system when it is submitted. A message indicating this job number is printed on the user's terminal. The job number assigned to the job must be specified by the user in subsequent CALL-OS terminal commands pertaining to the job. Specific capabilities are invoked by means of CALL-OS terminal commands as summarized below.

- CANCEL

  Provides a means by which the terminal user can cancel a job submitted via CALL-OS for OS batch processing. If a SCRATCH parameter is specified in the CANCEL command, the COBI index record and any data sets created for the job are scratched.

```
CANCEL #78                                  (terminal entry)

CANCEL #78,SCRATCH                          (terminal entry)
```

  The system responds with the first message shown below if the job can be cancelled before being sent to OS batch processing. The second message is printed if the job has been sent to OS batch processing and OS has been requested to cancel it. The third message is printed if the job has already been processed by OS. In either of the last two cases, unless the SCRATCH parameter was specified in the CANCEL command, the user is responsible for scratching any created data sets.

```
JOB CANCELLED                               (system response)

JOB CANCELLATION REQUEST INITIATED          (system response)

JOB COMPLETED                               (system response)
```

- DSSTATUS

  Prints the names, serial numbers, and status of the SYSOUT data sets pertaining to all completed jobs for a terminal user, to a specified completed job, or to a number of specified completed jobs. The first command shown below causes the status of the data sets of job #120 to be printed on the terminal (provided the job has been completed). The second causes the status of the data sets of jobs #3276, #441, and #1085 to be indicated. The third causes the status of all jobs submitted by the user to OS/360 batch processing, and completed, to be described.

```
DSSTATUS #120                               (terminal entry)

DSSTATUS #3276,#441,#1085                   (terminal entry)

DSSTATUS                                    (terminal entry)
```

A representative response to the second command above is printed below. In the SYSOUT column, entries preceded by U refer to user SYSOUT data sets; entries of the form nPmm refer to SYSOUT data sets defined in cataloged procedures. n is the procedure number, and mm is the number of the data set.

```
JOB ID:     SYSOUT:     VOL ID:     STATUS:

#3276         U003       EXEC99      OFF-LINE
#441          1P05       111111      ON-LINE
#1085         U007       222222      ON-LINE
              3P01       RTOSLK      OFF-LINE
```

- JOBSTATUS

Prints the status of all jobs that the user has submitted for OS batch processing, of a specified job, or of specified jobs. The first command shown below causes the status of job #100 to be indicated. The second causes the status of jobs #123, #48, and #16 to be indicated. The third causes the status of all jobs the user has submitted to OS batch processing to be described.

```
JOBSTATUS #100          (terminal entry)

JOBSTATUS #123,#48,#16                    (terminal entry)

JOBSTATUS                                 (terminal entry)
```

A representative response to the third command above is printed below, showing the four possible status conditions that may be noted. CODE refers to system completion code. User ABEND codes are preceded by U. In the rightmost column, entries preceded by U refer to user SYSOUT data sets. Entries of the form nPmm refer to SYSOUT data sets defined in cataloged procedures.

```
JOB ID:  STATUS:     CODE:     DATA SET:

#13     CANCEL RQ                1P06,1P13,...
#100    COMPLETED 0C5,U0130 JCL,3P01,U003,...
#123    JCL ERROR           JCL,U027,3P04,...
#17     NOT DONE                 1P01,1P02,...
  .        .          .             .
  .        .          .             .
  .        .          .             .
```

- NOTIFY

Sends a message to the CALL-OS communications console or to one or more OS operator consoles. Messages are directed to the CALL-OS communications console by a command of the form shown below.

```
NOTIFY CALL,'message'                     (terminal entry)
READY                                     (system response)
```

To send a message to a single (or default) OS operator console, a command of the following form is issued.

```
NOTIFY OS,'message'                       (terminal entry)
READY                                     (system response)
```

Messages to one or more OS operator consoles are directed to specific consoles by means of OS routing codes. This capability

is available only when the Multiple Console Support (MCS) feature is included at system generation. Multiple consoles may be specified in one NOTIFY command as shown below.

```
NOTIFY 2,4,'message'                    (terminal entry)
READY                                   (system response)
```

If no message destination is specified in a NOTIFY command, CALL is assumed.

Note: A NOTIFY command containing CALL, or in which CALL applies by default, can be issued by any CALL-OS terminal user. This form of the NOTIFY command is available 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.

- SCAN

Causes all or a portion of an OS data set to be scanned and listed at the terminal. The data set may be an output of an OS job submitted via CALL-OS or a data set created by OS batch-processing facilities not initiated via CALL-OS. The JCL of a submitted job can also be listed at the terminal. Entries preceded by U refer to user SYSOUT data sets. 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. n may be omitted; if so, 1 is assumed.

The user may stop a scan operation at any time by pressing the ATTN or BREAK key (on the 2741 Communications Terminal or Teletype, respectively). If he does so, or when the scan operation is completed, the system prints a question mark at the terminal. The primary purpose of the question mark is to remind the user that he has control of operations and is responsible for disposition of the data set. The user must, in response to the question mark, enter another scan request pertaining to the same data set or exercise his option to KEEP or SCRATCH the data set. A data set scanned by specifying a DSN parameter (see below) cannot be scratched.

Note: The system will respond to one other user action 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 accomplished more rapidly.

```
SCAN #120,3P2,3500 THRU 3600            (terminal entry)
list of specified data                  (system response)
    .
    .
    .
?
```

The content of print lines (logical records) 3500 through 3600 from SYSOUT data set 2 of the third procedure of job #120 are printed at the terminal.

```
SCAN #130,JCL                                (terminal entry)
list of specified data                       (system response)
    .
    .
    .
?
```

The JCL of job #130 is printed at the terminal.

```
SCAN DSN=OSFU.FORTRAN.IBM020,VOL=CARS09,MEM=PROG1   (entry)
list of specified data                        (system response)
    .
    .
    .
?
```

The content of member PROG1 of data set OSFU.FORTRAN.IBM020 from volume CARS09 is printed at the terminal, provided that one of the qualifiers of the data set name (in this case, of course, IBM020) is the user number of the user issuing the SCAN command. A request of this type generally pertains to an OS data set not created using CALL-OS. When a data set name is specified but the volume identification is omitted, the data set is assumed to be cataloged and searched for via the OS catalog.

The DSN parameter of a SCAN command can also be used to request scanning of a data set whose name contains, as its first qualifier, one of two installation-specified high-level index names. The user should contact the central computer installation to learn the index names that have been specified.

- SCRATCH

Enables the user to scratch one or all of the SYSOUT data sets attached to a specified job via a SUBMIT command. These may be user, procedure-defined, or JCL (message class) data sets.

```
SCRATCH #200          (terminal entry)
READY                                        (system response)
```

All data sets attached to job #200 are scratched.

```
SCRATCH #210,U2                              (terminal entry)
READY                                        (system response)
```

User SYSOUT data set 2 of job #210 is scratched.

```
SCRATCH #200,2P1                             (terminal entry)
READY                                        (system response)
```

The first data set of the second procedure for job #200 is scratched.

- SUBMIT

Copies a requested job from CALL-OS program files to a sequential data set for input to OS batch processing. The job may consist of from one to twelve job segments (programs, JCL, and/or data) previously placed in the CALL-OS data base. The user may specify

options in the SUBMIT command to save data sets for scanning (printing) at his terminal as follows:

- an entry of the characters JCL refers to the JCL for the job
- an entry of the form Unnn refers to a user SYSOUT data set
- 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. n may be omitted; if so, 1 is assumed.

A maximum of eight SYSOUT data sets (not counting JCL) may be specified. The system assigns a job number of the form #nnnnn to the input. A message is printed at the terminal, informing the user of this job number, which must be used in all terminal commands referring to the job. The message also indicates the name by which OS recognizes the job. In the examples below, JOBNAME is:

- the user's job name from his JOB card if ANYJNAME is specified in the system startup deck
- aaannnxx where aaannn is a user number and xx is a unique system-generated identifier if ANYJNAME is not specified in the system startup deck

```
SUBMIT JOB1,(P2,U1)                    (terminal entry)
#300 SUBMITTED AS JOBNAME              (system response)
```

JOB1 is copied to a sequential data set for input to OS. User SYSOUT data set 1 and SYSOUT data set 2 from the first procedure of the job will be saved for subsequent scanning. Job number #300 is assigned to the job and must be used to refer to it.

```
SUBMIT JOB2,(JCL,P1,3P2)               (terminal entry)
#310 SUBMITTED AS JOBNAME              (system response)
```

JOB2 is copied to a sequential data set for input to OS. JCL, SYSOUT data set 1 from the first procedure, and SYSOUT data set 2 from the third procedure of the job will be saved. Job number #310 is assigned to the job and must be used to refer to it.

```
SUBMIT ↑JOB3                           (terminal entry)
#320 SUBMITTED AS JOBNAME              (system response)
```

The up-arrow identifies an indirect submit. This form of the SUBMIT command makes it easier for the terminal user to submit a job again, once it has been set up with its options for OS batch processing. The example above indicates that a $$SUBMIT control statement will be the first statement in the program file JOB3 and will control submission of the job. For example, the $$SUBMIT control statement may appear as follows:

```
10 $$SUBMIT PROG1,PROG2,PROG3,(JCL,P1,P2,U1,U2,U3,U4,4P1)
```

If options are specified in the SUBMIT command, they override options specified in the $$SUBMIT control statement. Since none are specified in the SUBMIT command above, the options specified in the $$SUBMIT statement control the saving of data sets. In the example, job number #320 has been assigned and must be used to refer to the job.

INDEX

CALL-OS Version 2.1 SDM Printed in U.S.A. GH20-0673-5

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 Version 2.1

GH20-0673-5

System Description 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.

GH20-0673-5

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 Version 2.1 SDM Printed in U.S.A. GH20-0673-5

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]