Xerox Control Program-Five (CP-V)

Xerox 560 and Sigma 6/7/9 Computers

Time-Sharing
Reference Manual

XEROX

Xerox Control Program-Five (CP-V)

Xerox 560 and Sigma 5/6/7/9 Computers

Time-Sharing

Reference Manual

90 09 07H
90 09 07H-1

September 1978

File No.: 1X23
XG15A, Rev. 0
90 09 07H-1(9/78)

© 1974, 1975, 1976 Xerox Corporation

© 1978, Honeywell Information Systems Inc.

REVISION

This publication documents the F00 version of Control Program-Five (CP-V). The publication consists of the H edition of this manual (90 09 07H, dated November 1976) and the revision package numbered 90 09 07H-1 (9/78). Vertical lines in the margins of pages labeled 90 09 07H-1(9/78) indicate technical changes that reflect the F00 version of CP-V. Vertical lines in the margins of other pages indicate changes that occurred in a previous release of the system.

RELATED PUBLICATIONS

Title	Publication No.
Xerox Sigma 6 Computer/Reference Manual	90 17 13
Xerox Sigma 7 Computer/Reference Manual	90 09 50
Xerox Sigma 9 Computer/Reference Manual	90 17 33
Xerox 560 Computer/Reference Manual	90 30 76
Xerox Control Program-Five (CP-V)/OPS Reference Manual	90 16 75
Xerox Control Program-Five (CP-V)/SM Reference Manual	90 16 74
Xerox Control Program-Five (CP-V)/SP Reference Manual	90 31 13
Xerox Control Program-Five (CP-V)/BP Reference Manual	90 17 64
Xerox Control Program-Five (CP-V)/TS User's Guide	90 16 92
Xerox Control Program-Five (CP-V)/TP Reference Manual	90 31 12
Xerox Control Program-Five (CP-V)/RP Reference Manual	90 30 26
Xerox Control Program-Five (CP-V)/Common Index	90 30 80
Xerox EASY/LN, OPS Reference Manual	90 18 73
Xerox BASIC/LN, OPS Reference Manual	90 15 46
Xerox FLAG/Reference Manual	90 16 54
Xerox Meta-Symbol/LN, OPS Reference Manual	90 09 52
Xerox Assembly Program/Reference Manual	90 30 00
Xerox Extended FORTRAN IV/LN Reference Manual	90 09 56
Xerox Extended FORTRAN IV/OPS Reference Manual	90 11 43
Xerox FORTRAN Debug Package (FDP)/Reference Manual	90 16 77
Xerox ANS COBOL/LN Reference Manual	90 15 00
Xerox ANS COBOL/OPS Reference Manual	90 15 01
Xerox ANS COBOL/On-Line Debugger Reference Manual	90 30 60
Xerox APL/LN, OPS Reference Manual	90 19 31
Xerox Manage/Reference Manual	90 16 10
Xerox Sort-Merge/Reference Manual	90 11 99
Xerox Functional Mathematical Programming System (FMPS)/Reference Manual	90 16 09
Xerox SL-1/Reference Manual	90 16 76
CIRC-AC/Reference Manual and User's Guide	90 16 98
CIRC-DC/Reference Manual and User's Guide	90 16 97
CIRC-TR/Reference Manual and User's Guide	90 17 86
Xerox 1400 Series Simulator/Reference Manual	90 15 02

Manual Content Codes: BP - batch processing, LN - language, OPS - operations, RP - remote processing, RT - real-time, SM - system management, SP - system programming, TP - transaction processing, TS - time-sharing, UT - utilities.

The specifications of the software system described in this publication are subject to change without notice. The availability or performance of some features may depend on a specific configuration of equipment such as additional tape units or larger memory. Customers should consult their sales representative for details.

CONTENTS

APPENDIXES

FIGURES

TABLES

PREFACE

This manual is the principal source of information for the time-sharing features of CP-V. The purpose of the manual is to define the rules for using the Terminal Executive Language and other on-line processors. A closely related manual, the CP-V Time-Sharing User's Guide, 90 16 92, describes how to use the various time-sharing features. It presents an introductory subset of the features in a format that allows the user to learn the material by using the features at a terminal as he reads through the document.

Manuals describing other features of CP-V are outlined below.

- The CP-V Batch Processing Reference Manual, 90 17 64, is the principal source of reference information for the batch processing features of CP-V (i.e., job control commands, system procedures, I/O procedures, program loading and execution, debugging aids, and service processors).

- The CP-V Remote Processing Reference Manual, 90 30 26, is the principal source of information about the remote processing features of CP-V. All information about remote processing for all computer personnel (remote and local users, system managers, remote site operators, and central site operators) is included in the manual.

- The CP-V Operations Reference Manual, 90 16 75, is the principal source of reference information for CP-V computer operators. It defines the rules for operator communication (i.e., key-ins and messages), system start-up and initialization, job and system control, peripheral device handling, recovery and file preservation.

- The CP-V Transaction Processing Reference Manual, 90 31 12, provides information about dynamically modifying and querying a central database in a transaction processing environment. The manual is addressed to system managers, database administrators, application programmers, and computer operators.

- The CP-V System Programming Reference Manual, 90 31 13, describes the CP-V features that are designed to aid the system programmer in the development, maintenance, and modification of the CP-V system.

- The CP-V System Management Reference Manual, 90 16 74, is the principal source of reference information for the system management features of CP-V. It defines the rules for generating a CP-V system (SYSGEN), authorizing users, maintaining user accounting records, maintaining the file system, monitoring system performance, and other related functions.

- The CP-V Common Index (90 30 80) is an index to all of the above CP-V manuals.

Information for the language and application processors that operate under CP-V is also described in separate manuals. These manuals are listed on the Related Publications page of this manual.

COMMAND SYNTAX NOTATION

Notation conventions used in command specifications and examples throughout this manual are listed below.

Notation	Description
lowercase letters	Lowercase letters identify an element that must be replaced with a user-selected value. CRndd could be entered as CRA03.
CAPITAL LETTERS	Capital letters must be entered as shown for input, and will be printed as shown in output. DPndd means "enter DP followed by the values for ndd".
[ ]	An element inside brackets is optional. Several elements placed one under the other inside a pair of brackets means that the user may select any one or none of those elements. [KEYM] means the term "KEYM" may be entered.
{ }	Elements placed one under the other inside a pair of braces identify a required choice. $\left\{\begin{matrix}\text{A}\\ \text{id}\end{matrix}\right\}$ means that either the letter A or the value of id must be entered.
...	The horizontal ellipsis indicates that a previous bracketed element may be repeated, or that elements have been omitted. name[,name]... means that one or more name values may be entered, with a comma inserted between each name value.
⋮	The vertical ellipsis indicates that commands or instructions have been omitted. MASK2 DATA,2 X'1EF' ⋮ BYTE DATA,3 BA(L(59)) means that there are one or more statements omitted between the two DATA directives.
Numbers and special characters	Numbers that appear on the line (i.e., not subscripts), special symbols, and punctuation marks other than dotted lines, brackets, braces, and underlines appear as shown in output messages and must be entered as shown when input. (value) means that the proper value must be entered enclosed in parentheses; e.g., (234).
Subscripts	Subscripts indicate a first, second, etc., representation of a parameter that has a different value for each occurrence. sysid$_1$,sysid$_2$,sysid$_3$ means that three successive values for sysid should be entered, separated by commas.
Superscripts	Superscripts indicate shift keys to be used in combination with terminal keys. c is control shift, and s is case shift. L^{cs} means press the control and case shift (CONTROL and SHIFT) and the L key.
Underscore	All terminal output is underscored; terminal input is not. !RUN means that the exclamation point was sent to the terminal, but RUN was typed by the terminal user.
ESC RET LF.	These symbols indicate that an ESC (ESC), carriage return (RET), or line feed (LF) character has been sent. !EDIT RET means that, after typing EDIT, a carriage return character has been sent.

1. INTRODUCTION

DEFINITION OF CP-V

Control Program-Five (CP-V) is a comprehensive operating system designed for use with Xerox 560 and Sigma 6/7/9 computers and a variety of peripheral devices. CP-V provides for five concurrent modes of operation:

- Time-sharing
- Batch processing
- Remote processing
- Real-time
- Transaction processing

TIME-SHARING

CP-V provides a time-shared computing service that allows over 128 on-line terminals to be connected to the central computer at one time. There are three general categories of time-sharing service provided to on-line users. They are on-line file management, on-line program execution and debugging, and on-line entry of jobs into the batch job stream. The processors that provide these services are listed in Table 1 and are discussed in more detail in the following paragraphs.

Table 1. On-Line User Processors

Processor	Function
TEL	Executive language control of all terminal activities.
EASY	Creation, manipulation, and execution of FORTRAN and BASIC programs and data files.
Edit	Composition and modification of programs and other bodies of text.
FORT4	Compilation of Xerox Extended FORTRAN IV programs.
COBOL	Compilation of ANS COBOL programs.
META	Assembly of Meta-Symbol programs.
AP	Assembly of Assembly Program programs.
BASIC	Compilation and execution of programs or direct statements written in an extended BASIC language.
APL	Interpretation and execution of programs written in the APL language.
FLAG	Compilation of fast "load-and-go" FORTRAN programs.
FDP	Debugging of Xerox Extended FORTRAN IV programs.
Delta	Debugging of programs at the machine language level.
COBOL On-Line Debugger	Debugging of ANS COBOL programs.
PCL	Transfer (and conversion) of data between peripheral devices.
Link	Linkage of programs for execution.
LYNX	Linkage of programs for execution.
LEMUR	Building and manipulation of libraries.
Batch	Submission of file(s) to the batch job stream.
Show	Display of user's system parameters.
Manage[†]	File retrieval, updating, and reporting.
SL-1[†]	Compilation of programs written in a language designed specifically for digital or hybrid simulation.
CIRC[†]	Analysis of electronic circuits.

[†] Program product (see glossary).

TERMINAL EXECUTIVE LANGUAGE

The Terminal Executive Language (TEL), is the principal terminal language for CP-V. Most activities associated with COBOL, FORTRAN, and assembly language programming can be carried out via TEL through requests that take the form of single-line commands and declarations. These activities include such major operations as composing programs and other bodies of text, compiling and assembling programs, linking object programs, initiating execution, and debugging programs. They also include such minor operations as checkpointing on-line sessions, determining program status, and setting simulated tab stops. (Reference: Chapter 3.)

EASY

EASY is a shared processor that enables the user to create, edit, execute, save, and delete program files written in BASIC or FORTRAN. EASY also allows the user to create and manipulate EBCDIC data files. Although intended primarily for Teletype® operations, EASY can be used with any type of on-line terminal supported by the system.

®Registered trademark of the Teletype Corporation.

GLOSSARY

address resolution code a two-bit code that specifies whether an associated address is to be used as a byte address or is to be converted (by truncating low order bits) to a halfword, word, or doubleword address.

batch job a job that is submitted to the batch job stream through the central site card reader, through an on-line terminal (using the Batch processor), or through a remote terminal.

binary input input from the device to which the BI (binary input) operational label is assigned.

conflicting reference a reference to a symbolic name that has more than one definition.

control command any control message other than a key-in. A control command may be input via any device to which the system command input function has been assigned (normally a card reader).

control key-in a control message of the type that must be input from the operator's console.

control message any message received by the monitor that is either a control command or a control key-in.

cooperative a monitor routine that transfers information between a user's program and disk storage.

CRT a term used within this manual to refer to any time-sharing terminal which has a display screen.

data control block (DCB) a table in the user program that contains the information used by the monitor in the performance of an I/O operation.

external reference a reference to a declared symbolic name that is not defined within the object module in which the reference occurs. An external reference can be satisfied only if the referenced name is defined by an external load item in another object module.

file extension a convention that is used when certain system output DCBs are opened. Use of this convention causes the file (on RAD, tape, disk pack, etc.) connected to the DCB to be positioned to a point just following the last record in the file. Thus, when additional output is produced through the DCB, it is added to the previous contents of the file, thereby extending the file.

function parameter table (FPT) a table through which a user's program communicates with a monitor function (such as an I/O function).

ghost job a job that is neither a batch nor an on-line program. It is initiated and logged on by the monitor, the operator, or another job and consists of a single job step. When the ghost program exits, the ghost is logged off.

global symbol a symbolic name that is defined in one program module and referenced in another.

GO file a temporary disk storage file consisting of relocatable object modules formed by a processor.

internal symbol a symbolic name that is defined and referenced in the same program module.

job information table (JIT) a table associated with each active job. The table contains accounting, memory mapping, swapping, terminal DCB (M:UC), and temporary monitor information.

job step a subunit of job processing such as compilation, assembly, loading, or execution. Information from certain commands (JOB, LIMIT, and ASSIGN) and all temporary files created during a job step are carried from one job step to the next but the steps are otherwise independent.

key a data item consisting of 1-31 characters that uniquely identifies a record.

key-in information entered by the operator via a keyboard.

linking loader a program that is capable of linking and loading one or more relocatable object modules and load modules.

load map a listing of loader output showing the location or value of all global symbols entering into the load. Also shown are symbols that are not defined or have multiple definitions.

load module (LM) an executable program formed by the linking loader, using relocatable object modules (ROMs) and/or load modules (LMs) as source information.

logical device a peripheral device that is represented in a program by an operational label (i.e., BI or PO) rather than by specific physical device name.

logical device stream an information stream that may be used when performing input from or output to a symbiont device. At SYSGEN, up to 15 logical device streams are defined. Each logical device stream is given a name (e.g., L1, P1, C1), each is assigned to a default physical device, and each is given default atrributes. The user may perform I/O through a logical device stream with the default physical device and attributes or he may change the physical device and/or attributes to satisfy the requirements of his job.

monitor a program that supervises the processing, loading, and execution of other programs.

object language   the standard binary language in which the output of a processor is expressed.

object module   the series of records containing the load information pertaining to a single program or subprogram (i.e., from the beginning to the end). Object modules serve as input to the Load processor or the Link processor.

on-line job   a job that is submitted through an on-line terminal by a command other than the BATCH command.

operational label   a symbolic name used to identify a logical system device.

option   an elective operand in a control command, procedure call, or on-line command, or an elective parameter in a Function Parameter Table.

parameter presence indicator   a bit in word 1 of a Function Parameter Table that indicates whether a particular parameter word is present in the remainder of the table.

physical device   a peripheral device that is referred to by a name specifying the device type, I/O channel, and device number (also see "logical device").

program product   a compiler or application program that has been or will be released by Xerox. A program product is not required by all users and is therefore made available by Xerox on an optional basis. Program products are provided only to those users who execute a License Agreement for each applicable installation.

prompt character   a character that is sent to the terminal by an on-line language processor to indicate that the next line of input may be entered.

public library   a set of library routines declared at SYSGEN to be public (i.e., to be used in common by all concurrent users).

reentrant   an attribute of a program that allows the program to be shared by several users concurrently. Shared processors in CP-V are map reentrant. That is, each instance of execution of a single copy of the program's instructions has a separate copy of the execution data.

relative allocation   allocation of virtual memory to a user program starting with the first unallocated page available.

relocatable object module (ROM)   a program or subprogram in object language generated by a processor such as Meta-Symbol or FORTRAN.

resident program   a program that has been loaded into a specific area of core memory.

secondary storage   any rapid-access storage medium other than core memory (e.g., RAD storage).

shared processor   a program (e.g., FORTRAN) that is shared by all concurrent users. Shared processors must be established by SYSGEN.

source language   a language used to prepare a source program suitable for processing by an assembler or compiler.

special shared processor   a shared processor that may be in core memory concurrently with the user's program (e.g., Delta or TEL).

specific allocation   allocation of a specific page of unallocated virtual memory to a user program.

SR1, SR2, SR3, and SR4   see "system register", below.

static core module   a program module that is in core memory but is not being executed.

stream-id   the name of a logical device stream.

symbiont   a monitor routine that transfers information between disk storage and a peripheral device independent of and concurrent with job processing.

symbolic input   input from the device to which the SI (symbolic input) operational label is assigned.

symbolic name   an identifier that is associated with some particular source program statement or item so that symbolic references may be made to it even though its value may be subject to redefinition.

SYSGEN   see "system generation" below.

system generation (SYSGEN)   the process of creating an operating system that is tailored to the specific requirements of an installation. The major SYSGEN steps include: gathering the relevant programs, generating specific monitor tables, loading monitor and system processors, and writing a bootable system tape.

system library   a group of standard routines in object-language format, any of which may be incorporated in a program being formed.

system register   a register used by the monitor to communicate information that may be of use to the user program (e.g., error codes). System registers SR1, SR2, SR3, and SR4 are current general registers 8, 9, 10, and 11, respectively.

task control block (TCB)   a table of program control information built by the loader when a load module is formed. The TCB is part of the load module and contains the data required to allow reentry of library routines during program execution or to allow asynchronous entry to the program in cases of traps, breaks, etc. The TCB is program associated and not task associated.

unsatisfied reference   a symbolic name that has been referenced but not defined.

EDIT

The Edit processor is a line-at-a-time context editor designed for on-line creation, modification, and handling of programs and other bodies of information. All Edit data is stored on RAD or disk pack storage in a keyed file structure of sequence-numbered variable length records. This structure permits Edit to directly access each line or record of data.

Edit functions are controlled through single-line commands supplied by the user. The command language provides for insertion, deletion, reordering, and replacement of lines or groups of lines of text. It also provides for selective printing, renumbering records, and context editing operations of matching, moving, and substituting line-by-line within a specified range of text lines. File maintenance commands are also provided to allow the user to build, copy, and delete whole files of text lines. (Reference: Chapter 6.)

XEROX EXTENDED FORTRAN IV

The Xerox Extended FORTRAN IV language processor (FORT4) consists of a comprehensive algebraic programming language, a compiler, and a large library of subroutines. The language is a superset of most available FORTRAN languages, containing many extended language features to facilitate program development and checkout. The compiler is designed to produce very efficient object code, thus reducing execution time and core requirements, and to generate extensive diagnostics to reduce debugging time. The library contains over 180 subprograms and is available in a reentrant version. Both the compiler and runtime library for object programs are reentrant programs that are shared among all concurrent users to improve the utilization of the critical core resources.

The principal features of Xerox Extended FORTRAN IV are as follows:

Extended language features to reduce programming effort and increase range of applications.

Extensive meaningful diagnostics to minimize debugging time.

In-line assembly language code to reduce execution time of critical parts of the program.

Overlay organization for minimal core memory utilization.

Compiler produced reentrant programs.

Full use of CP-V features.

Availability of reentrant version of library.

(Reference: Extended FORTRAN IV/LN Reference Manual, 90 09 56 and Extended FORTRAN IV/OPS Reference Manual, 90 11 43.)

XEROX ANS COBOL

COBOL (COmmon Business Oriented Language) is especially efficient in the processing of business problems. Such problems typically involve relatively little algebraic or logical processing. Instead, they most often manipulate large files of basically similar records in a relatively simple way. This means that COBOL mainly emphasizes the description and handling of data items and input/output records.

COBOL looks and reads much like ordinary business English. The programmer can use English words and conventional arithmetic symbols to direct and control the computer operations. Two typical COBOL sentences follow:

ADD NEW-PURCHASES TO TOTAL-CHARGES

PERFORM FEDERAL-TAX-CALCULATIONS

(Reference: Xerox ANS COBOL/LN Reference Manual, 90 15 00, and Xerox ANS COBOL/OPS Reference Manual, 90 15 01.)

META-SYMBOL

Meta-Symbol is a procedure-oriented macro assembler that provides services available in sophisticated macro assemblers and has special features that permit the user to execute dynamic control over the parametric environment of assembly. Meta-Symbol's highly flexible assembly language gives users full use of the available hardware capabilities.

Under CP-V, Meta-Symbol may be used in batch or on-line mode. In on-line mode, the assembler allows programs to be assembled and executed on-line but does not allow conversational interaction.

One of Meta-Symbol's features is a highly flexible list definition and manipulation capability. Lists and list elements may be conveniently redefined, thus changing the value of a given element.

Another Meta-Symbol feature is the macro capability. Xerox uses the term "procedure" to emphasize the highly sophisticated and flexible nature of this macro capability. Procedures are assembly-time subroutines that provide the user with an extensive function capability. Procedure definitions, references, and recursions may be nested up to 32 levels.

Meta-Symbol also has an extensive set of operators to facilitate the use of logical and arithmetic expressions. These operators facilitate the parametric coding capabilities available with Meta-Symbol (parametric programming allows for dynamic specification of both "if" and "how" a given statement or set of statements is to be assembled).

Users are also provided with an extensive set of directives. These directives, which are commands intrinsic to the assembly, fall into three classes:

1. Directives that involve manipulation of symbols and are unconditionally executed.

2. Directives that allow parametric programming.

3. Directives that do not allow parametric programming.

Intrinsic functions are also included in Meta-Symbol. These give the user the ability to obtain information on both the structure and content of assembly time constructs. For example, the user can acquire information on the length of a certain list. He can inquire about a specific symbol and whether it occurs in a procedure reference. (Reference: Meta-Symbol/LN, OPS Reference Manual, 90 09 52.)

AP

Assembly Program (AP) is a four-phase assembler that reads source language programs and converts them to object language programs. AP outputs the object language program, an assembly listing, and a cross reference (or concordance) listing. AP is available in both the on-line and batch modes.

The following list summarizes AP's more important features for the programmer:

- Self-defining constants that facilitate use of hexadecimal, decimal, octal, floating-point, scaled fixed-point, and text string values.
- The facility for writing large programs in segments or modules. The assembler will provide information necessary for the loader to complete the linkage between modules when they are loaded into memory.
- The label, command, and argument fields may contain both arithmetic and logical expressions, using constant or variable quantities.
- Full use of lists and subscripted elements is provided.
- The DO, DO1, and GOTO directives allow selective generation of areas of code, with parametric constants or expressions evaluated at assembly time.
- Command procedures allow the capability of generating many units of code for a given procedure call line.
- Function procedures return values to the procedure call line. They also provide the capability of generating many units of code for a given procedure call line.
- Individual parameters on a procedure call line can be tested both arithmetically and logically.
- Procedures may call other procedures, and may call procedures recursively.

(Reference: Xerox Assembly Program/Reference Manual, 90 30 00.)

BASIC

BASIC is a compiler and programming language similar to Dartmouth BASIC. It is, by design, easy to teach, learn, and use. It allows individuals with little or no programming experience to create, debug, and execute programs via an on-line terminal. Such programs are usually small to medium size, predominantly arithmetic applications.

BASIC is designed primarily for on-line program development and execution, or on-line development and batch execution. In addition, programs may be developed and executed in batch mode.

BASIC provides two user modes of operation. The editing mode is used for creating and modifying programs. The compilation/execution mode is used for running completed programs. This arrangement simplifies and speeds up the program development cycle.

BASIC statements may be entered via a terminal and immediately executed. During execution, programs may be investigated for loop detection, snapshots of variables may be obtained, values of variables may be changed, flow of execution may be rerouted, and so on. This unique capability also allows an on-line terminal to be used as a "super" desk calculator.

At compile and execute time, the user may specify an array dimension check. In the safe mode, statements are checked to verify that they do not reference an array beyond its dimensions. In the fast mode, this time consuming check is not made. The safe mode is used during checkout; the fast mode is used when the program reaches the production state, to speed up execution.

BASIC provides an image statement that uses a "picture" of the desired output format to perform editing. It also has TAB capability and a precision option to indicate the number of significant digits (6 to 16) to be printed.

BASIC also has an easy-to-use feature allowing the user to read, write, and compare variable alphanumeric data. This is particularly inportant for conversational input processing.

Chaining permits one BASIC program to call upon another for compilation and execution without user intervention. Thus, programs that would exceed user core space may be sequenced and overlay techniques may be employed via the chaining facility. (Reference: BASIC/Reference Manual, 90 15 46.)

APL

APL is an acronym for A Programming Language, the language invented by Kenneth Iverson. It is an interpretive, problem-solving language. As an interpretive language, APL does not wait until a program is completed to compile it into object code and execute it; instead, APL interprets each line of input as it is entered to produce code that is immediately executed. As a problem-solving language, APL requires minimal computer programming knowledge; a problem is entered into the computer and an answer is received, all in the APL language.

Because APL is powerful, concise, easy to learn, and easy to use, it is widely used by universities, engineers, and statisticians. It also has features that make it attractive for business applications where user interaction and rapid feedback are key issues. One of APL's major strengths is its ability to manipulate vectors and multidimensional arrays as easily as it does scalar values. For example, a matrix addition that might require a number of statements and several loops in other languages can be accomplished as A + B in APL. This type of simplification exemplifies APL's concise power.

Xerox APL has been designed to be compatible with competitive APL systems. In addition, it has many salient features not generally found in other APL systems. Some of these features are: both on-line and batch operation, operation from terminals without APL characters, fast formatted output, file input/output, compound statements, unequally spaced tab settings, and so on. (Reference: APL/LN,OPS Reference Manual, 90 19 31.)

FLAG

FLAG (FORTRAN Load and Go) is an in-core FORTRAN compiler that is compatible with the FORTRAN IV-H class of compilers. It can be used in preference to the other FORTRAN compilers when users are in the debugging phase of program development. FLAG is a one-pass compiler and uses the Extended FORTRAN IV library. Included in the basic external functions are the Boolean functions IAND (AND), IEOR (exclusive OR), and IOR (OR), which give the FORTRAN user a bit manipulation capability.

If several FLAG jobs are to be run sequentially, they may be run in a sub-job mode, thus saving processing time normally needed for the Control Command Interpreter (CCI) to interpret the associated control cards. In this mode, FLAG will successively compile and execute any number of separate programs, thereby reducing monitor overhead.

The FLAG debug mode is a user-selected option that generates extra instructions in the compiled program to enable the user, during program execution, to detect errors in program logic that might otherwise go undetected or cause unexplainable program failure. (Reference: FLAG/ Reference Manual, 90 16 54.

FORTRAN DEBUG PACKAGE

The FORTRAN Debug Package (FDP) is made up of special library routines that are called by Xerox Extended FORTRAN IV object programs compiled in the debug mode. These routines interact with the program to detect, diagnose, and in many cases, temporarily repair program errors.

The debugger can be used in batch and on-line mode. An extensive set of debugging commands is available in both cases. In batch operation, the debugging commands are included in the source input and are used by the debugger during execution of the program. In on-line operation, the debugging commands are entered through the terminal keyboard when requested by the debugger. Such requests are made when execution starts or restarts and for all execution stops in which the debugger has control. The debugger normally has control of such stops.

In addition to the debugging commands, the debugger has a few automatic debugging features. One of these features is the automatic comparison of standard calling and receiving sequence arguments for type compatibility. When applicable, the number of arguments in the standard calling sequence is checked for equality with the number of dummies in the receiving sequence. These calling and receiving arguments are also tested for protection conflicts. Another automatic feature is the testing of subprogram dummy storage attempts, to determine if they violate the protection of the calling argument. (Reference: FORTRAN Debugger/ Reference Manual, 90 16 77.)

COBOL ON-LINE DEBUGGER

The COBOL On-Line Debugger is designed to be used with Xerox ANS COBOL. The debugger is a special COBOL run-time library routine that is called by programs compiled in the TEST mode. This routine allows the programmer to monitor and control both the execution of his program and the contents of data-items during on-line execution. The debugger also allows the COBOL source program to be examined and modified.

The debugger can only be used during on-line execution; however, programs that have been compiled for use with the debugger may be run in the batch mode. This is not recommended, though, because of the increased program size when the TEST mode is specified. (Reference: ANS COBOL/On-Line Debugger Reference Manual, 90 30 60.)

DELTA

Although Delta is designed to aid in the debugging of programs at the assembly-language or machine-language levels, it may be used to debug FORTRAN, COBOL, or any other program. It is designed and interfaced with the operating system in such a way that it may be called in to aid debugging at any time (even after a program has been loaded).

Delta operates on object programs and tables of internal and global symbols used by the programs but does not require the tables to be present. With or without the symbol tables, Delta recognizes computer instruction mnemonic codes and can assemble machine-language programs on an instruction-by-instruction basis. The main purpose of Delta, however, is to facilitate the activities of debugging by

1. Examining, inserting, and modifying program elements such as instructions, numeric values, coded information (i.e., data in all its representations and formats).

2. Controlling execution, including the insertion of break points into a program and requests for breaks on changes in elements of data.

3. Tracing execution by displaying information at designated points in a program.

4. Searching programs and data for simple elements or elements within a hierarch.

To assist the first activity, CP-V assemblers and compilers include information identifying the type of data each symbol in the symbol table represents. The type of data includes symbolic instructions, decimal integers, floating-point values, single and double precision values, EBCDIC encoded information, and other types. (Reference: Chapter 7.)

Warning: Debugging a program under Delta which has an associated shared library has certain limitations. See Chapter 7.

PERIPHERAL CONVERSION LANGUAGE

The Peripheral Conversion Language (PCL) is a utility processor designed for operation in a batch or on-line environment. It provides for information movement among card devices, line printers, on-line terminals, magnetic tape devices, disk packs, and RAD storage.

PCL is controlled by single-line commands supplied through on-line terminal input, through a file containing PCL commands, or through command card input in the job stream. The command language provides for single or multiple file transfers with options for selecting, sequencing, formatting, and converting data records. Additional file maintenance and utility commands are provided. (Reference: Chapter 5.)

LYNX

LYNX is a load processor that is available in both the on-line and batch modes. LYNX has the capabilities of the Load loader (a batch mode loader) and also provides the same control over internal and global symbol table construction which is available in the Link loader. LYNX may be viewed as a preprocessor for the Load loader. After it analyzes the user's commands, it constructs a table of loader control information which it then passes to the Load loader. It is the Load loader which actually performs the loading process. (Reference: Chapter 8.)

LINK

The Link processor is provided with CP-V only for compatibility with previous versions of the system. It is recommended that the LYNX loader be used. Link is a linking loader that constructs a single entity called a load module, which is an executable program formed from relocatable object modules. Link is a one-pass linking loader that makes full use of mapping hardware. It is not an overlay loader. If the need for an on-line overlay loader exists, the LYNX loader must be used. (Reference: Chapter 8.)

LEMUR

LEMUR (Library Editor and Maintenance Utility Routine) is a processor that builds and manipulates ROM and load module libraries. The libraries thus built are accessed by LYNX or Load when constructing user load modules which require library routines. LEMUR is available in both on-line and batch modes. (Reference: Chapter 8.)

BATCH

The Batch processor is used to submit a file or a series of files to the batch queue for execution. Through Batch processor commands, the following capabilities are available:

1. Files may be inserted into a file being submitted for execution, thus bringing together more than one file to create a single job.

2. Selected strings and fields existing in files being submitted for execution may be replaced by new strings and fields.

3. The results of string and field replacements can be examined before the job is submitted to the batch stream.

4. Files to be submitted for execution may reside on tape or private disk pack.

5. Jobs may be submitted to run in an account other than the account from which the job is submitted.

The Batch processor may be called in either the on-line, the batch, or the ghost mode. (Reference: Chapter 9.)

SHOW

The Show processor allows the user to display his current maximum system services and resources, the peripheral devices that he has been authorized to use, and several other system user parameters. In the time-sharing mode, Show is called via the TEL processor. (Reference: Chapter 3.)

MANAGE (PROGRAM PRODUCT)

Manage is a generalized file management system. It is designed to allow decision makers to make use of the computer to generate and update files, retrieve useful data, and generate reports without having a knowledge of programming. (Reference: Manage/Reference Manual, 90 16 10.)

SIMULATION LANGUAGE (PROGRAM PRODUCT)

The Simulation Language (SL-1) is a simplified, problem-oriented digital programming language designed specifically for digital or hybrid simulation. SL-1 is a superset of CSSL (Continuous System Simulation Language), the standard language specified by Simulation Councils, Inc., for simulation of continuous systems. It exceeds the capabilities of CSSL and other existing simulation languages by providing hybrid and real-time features, interactive debugging features, and a powerful set of conditional translation features.

SL-1 is primarily useful in solving differential equations, a fundamental procedure in the simulation of parallel, continuous systems. To perform this function, SL-1 includes six integration methods and the control logic for their use. In hybrid operations, SL-1 automatically synchronizes the problem solution to real-time and provides for hybrid input and output.

Because of the versatility of Xerox computing systems and the broad applicability of digital and hybrid simulation techniques, applications for SL-1 exist across the real-time spectrum. The library concept of SL-1 allows the user to expand upon the Xerox-supplied macro set and facilitates the development of macro libraries oriented to any desired application. (Reference: SL-1/Reference Manual, 90 16 76.)

CIRC (PROGRAM PRODUCT)

CIRC is a set of three computer programs for electronic circuit analysis: CIRC-DC for dc circuit analysis, CIRC-AC for ac circuit analysis, and CIRC-TR for transient circuit analysis. The programs are designed for use by a circuit engineer at the installation, and require little or no knowledge of programming for execution.

CIRC can be executed with three modes of operation possible: conversational (on-line) mode, terminal batch entry mode, and batch processing mode. The system manager will determine which of these modes are available to the engineer, based on type of computer installation and other installation decisions.

- The on-line mode offers several advantages since it provides true conversational interaction between the user and computer. Following CIRC start-up procedures, CIRC requests a control message from the user. After the control message is input (e.g., iterate a cycle of calculations with changed parameters), the computer responds (via CIRC) with a detailed request for application data. These requests are sufficiently detailed to virtually eliminate misunderstandings by the engineer. This mode is highly useful in a highly interactive environment that produces a low volume of output and requires limited CPU time.

- The terminal batch entry mode allows efficient handling of high volume output and large CPU time requirements while preserving the advantages of the terminal as an input device. Two files are required: one containing all CIRC input including a circuit description and control messages, and the other directing the execution of CIRC. The job is entered from the terminal into the batch queue and treated like a batch job.

- The batch mode should generally be used for jobs involving large volumes of computations and outputs. It enables the user to concentrate on data preparation with virtually no involvement in programming considerations. The system manager can provide a set of start-up cards that never change, and these will constitute the entire interface between user and executive software. However, the batch mode offers less flexibility in experimenting with a circuit and slower turnaround time in obtaining answers.

(Reference: CIRC-AC/Reference Manual and User's Guide, 90 16 98, CIRC-DC/Reference Manual and User's Guide, 90 16 97, and CIRC-TR Reference Manual and User's Guide, 90 17 86.)

BATCH PROCESSING

Batch processing facilities are described in the CP-V/BP Reference Manual, 90 17 64. Although some facilities and processors are reserved for on-line use and others for batch use, the two classes of service are complementary. Generally speaking, anything that can be done in batch mode can be done on-line, although sometimes in a curtailed manner. In particular, compilers and assemblers are compatible across the two classes of service at source and relocatable levels. For example,

1. Processors for Extended FORTRAN IV, ANS COBOL, and Meta-Symbol are available both in batch and on-line mode.

2. Programs compiled or assembled in batch can be linked with those produced on-line and can be run and debugged on-line.

3. Programs compiled or assembled on-line can be linked and run in batch mode.

(Reference: CP-V/BP Reference Manual, 90 17 64.)

REMOTE PROCESSING

Remote processing facilities are described in the CP-V/RP Reference Manual, 90 30 26. The remote processing system is an extension of the CP-V symbiont system. Its purpose is to provide for very flexible communication between

CP-V and a variety of remote terminals. These terminals can range from a simple card reader and line printer combination to another computer system with a wide variety of peripheral devices. Any CP-V user (batch, on-line, ghost) can communicate with any number of devices at one or several remote sites.

(Reference: CP-V/RP Reference Manual, 90 30 26.)

REAL-TIME PROCESSING

The real-time services provided by CP-V allow user to connect interrupts to mapped programs, control the state of interrupts (e.g., trigger, arm/disarm, enable/disable), clear interrupts either at the time of occurrence or upon completion of processing, and disconnect interrupts no longer required. The user may also request that a mapped program be held in core in order to reduce the time required to respond to an external event (via an interrupt) or to allow various forms of special I/O to occur. Programs may be connected to one of the monitor's clocks such that after a specified period of time, a specified routine is entered. In addition, dedicated foreground memory may be used as inter-program communication buffers or as dedicated memory for unmapped, master mode programs which may be directly connected to external interrupts or real-time clocks.

(Reference: CP-V/SP Reference Manual, 90 31 13.)

TRANSACTION PROCESSING

The transaction processing feature of CP-V is an efficient and economical approach to centralized information processing and is a generalized package that is designed to meet the requirements of a variety of business applications. Transaction processing facilities provide an environment in which several users at remote terminals may enter business transactions, simultaneously utilizing a common data base. The transactions are processed immediately, as they are received, by application programs written especially for the particular installation.

(Reference: CP-V/TP Reference Manual, 90 31 12.)

2. TERMINAL OPERATIONS

INTRODUCTION

The following types of on-line terminals may be used with CP-V:

Xerox Model 7015 Keyboard/Printer.

Teletype Models 33, 35, 37, and 38.

IBM 2741 Terminals.

Tektronix® Models 4010 and 4013.

Datapoint 3300.

Any terminal compatible with any of the above.

The terminal operations described in this chapter apply primarily to Teletype and Xerox Model 7015 terminals (see Figure 1) and to the Terminal Executive Language (TEL). Operations that are unique for 2741 terminals are delineated at the end of the chapter. Terminal communication services to user programs are discussed in Chapter 10.

Seven facets of terminal operations are described in this chapter. They are

1. Initiating and ending on-line sessions.
2. Typing lines.
3. Typing commands.
4. Detecting and reporting errors.
5. Interrupting CP-V.
6. Paper tape input.
7. 2741 and Teletype differences.

INITIATING AND ENDING ON-LINE SESSIONS

An on-line user must establish a connection with CP-V and identify himself properly before he can use TEL or any of the processors. When a connection with CP-V is established, CP-V responds by typing

```
HONEYWELL CP-V AT YOUR SERVICE - site id
time date USER# sysid LINE# line
optional operator message
LOGON PLEASE:
```

If the user is using an ASCII terminal with an APL character set (such as the Diablo Hy-Type based units), he can obtain the correct translation (of the code that his terminal is generating to EBCDIC – and vice versa) by typing ESC A during the log-on sequence. (See the description of ESC A.)

The system then waits for the user identifier to be entered in the following format:

```
account,name[(ext. accounting field)][,password]
```

At this point, character echoing is automatically turned off by the system to facilitate keeping the user's identification secret. If the user wishes to have character echoing on, the ESC E sequence must be entered. In any case, character echoing is automatically turned on as soon as the user has successfully entered the identification information.

The name must be from one to twelve characters in length with an optional extended accounting field of up to 24 characters enclosed in parentheses; account and password must be from one to eight characters. None of the following characters may be used in user account, name or password:

| , ; < > . / = ?

Also, none of the control characters acted upon by the COC may be used.

Underscores count as characters and may print as left-facing arrows (←). Commas are used as separators. After the identifier is entered, the RETURN or LINE FEED key is depressed to return the carriage to the left margin of the next line and to deliver the line to CP-V for examination.

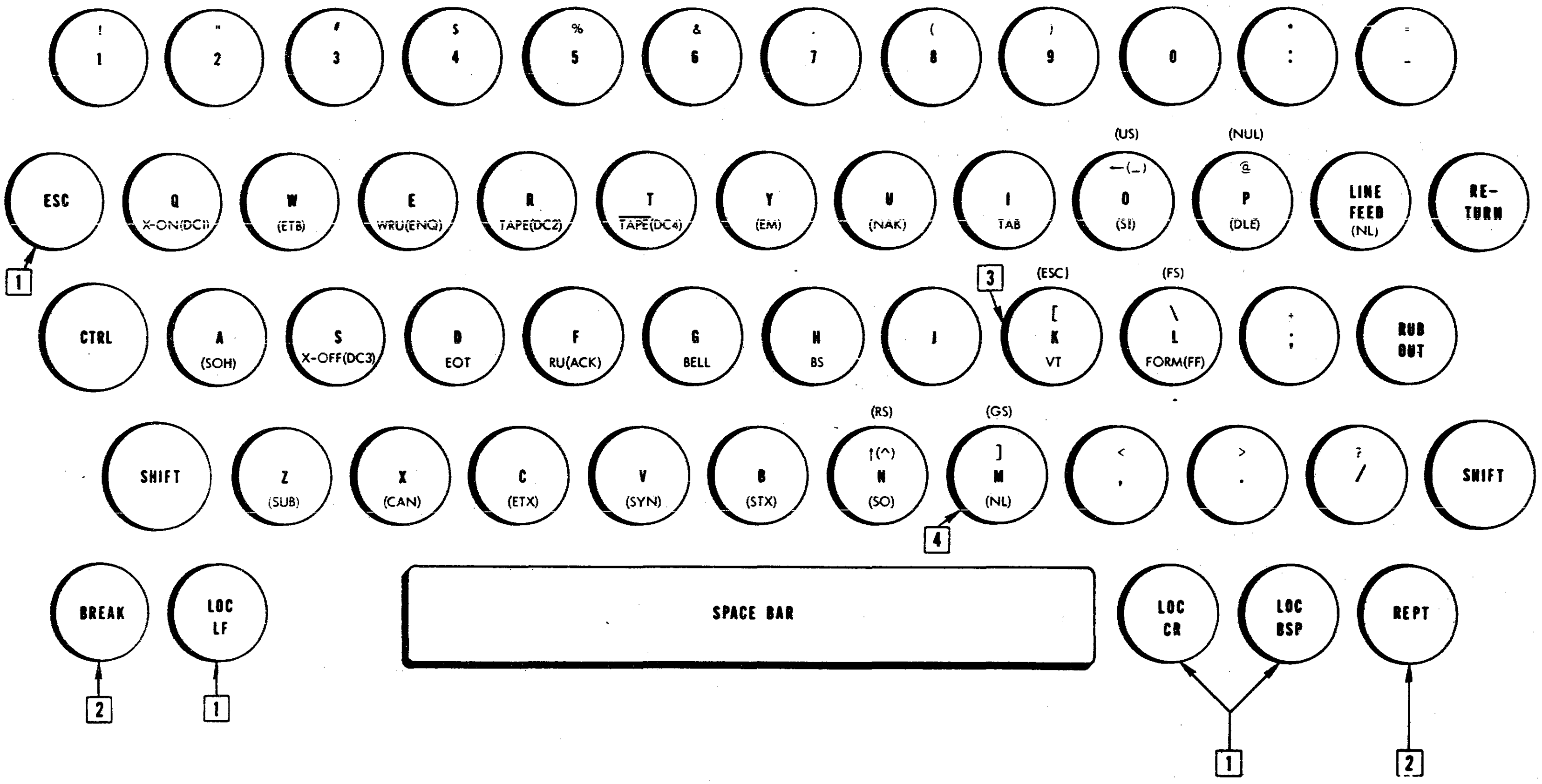
For terminals operated in indirect printing mode, character echoing by the system is normally on but can be turned off (e.g., to suppress printing of security-related information) by striking the ESC E keys. Striking the ESC E keys a second time turns echoing back on. For terminal units operated in direct printing mode, character echoing by the system must be turned off, as above, to suppress duplicate printing of characters.

If the identification is valid and consistent with CP-V records, TEL types its prompt character (!) at the left margin of the top line of the next page and then awaits the first command. If automatic association of a program or processor is specified in the user's log-on record, control passes to that program instead of TEL for identification and command request. The system sends an error message to the terminal and repeats the log-on request if the identification is garbled or otherwise invalid. The error messages are

EH? (Preceded by a repeat of the input for hardware debugging purposes.)

ID?

® Registered Trademark of the Tektronix Corporation.



1 These keys are missing on some models.

2 This key is positioned elsewhere on some models.

3 This is interpreted as | (OR) on Model 7015.

4 This is interpreted as ¬ (NOT) on Model 7015.

Characters obtained by depressing the SHIFT key are shown at the top of the key and characters obtained by depressing the CTRL key are shown at the bottom of the key. Characters obtained by depressing both SHIFT and CTRL keys are shown above the key. On the actual keyboard, all unparenthesized forms appear at the top of the key.

Figure 1. Model 33 Teletype Terminal Keyboard

It may not always be possible to log on. If an error prevents the reading of the log-on file, the message UNRECOVERABLE I/O ON RAD, or ABNORMAL ERROR ON LOGON FILE will be typed. Whenever the user is unable to log on, he may start over by striking the BREAK key and trying again. The system tries five times to log each user on before dismissing him.

Following a successful log-on, if the user has – in a previous session – exhausted his allocated permanent file storage space, he receives the following message:

```
FILE STORAGE LIMIT EXCEEDED
```

This means that no file storing operations can be performed until the user deletes one or more of his files.

If a MAILBOX file (a message file) exists at log-on time, the message CHECK DC/MAILBOX will appear. This MAILBOX file can be examined by copying it to the terminal as follows:

```
!COPY MAILBOX [TO ME]
```

(The underscored exclamation mark is the "prompt character" issued by TEL.)

The password in the log-on file can be set or changed at any time by typing the PASSWORD command.

An on-line session is ended by entering the OFF command. The system sends the following use accounting information to the terminal when a user logs off:

```
CPU=m.mmmm CON=hh:mm:ss INT=nn CHG=xxxx
```

CPU time is expressed in minutes and ten-thousandths of a minute. Terminal time (CON) is expressed in hours and minutes. INT is the number of terminal interactions during the on-line session. CHG is the total number of charge units for the on-line session. (Reference: Chapter 5, CP-V/SM Reference Manual, 90 16 74.)

AUTOMATIC SAVE FOR LINE DISCONNECT

This feature of CP-V preserves a user's program when a line disconnect occurs before the user has entered the OFF command, and provides a method of reconnection to the preserved program when the user calls back. Files remain open and properly positioned so that the program may be continued as if it had never been interrupted.

When a line disconnect occurs, the suspended program image is retained for a fixed length of time. This retention period is established at SYSGEN and may be modified by the operator at any time. If the user's image is being saved, the following will also be typed

```
PROGRAM HELD
```

Suspended program images are retained and named by the user's log-on account/name identifier. Only one suspended image is retained for any given account/name. Thus, if two users are logged on under identical account/names and both hang up, only the image of the first to hang up will be retained. The second to hang up will simply be logged off. Further, the first to call back is given the option of reconnecting to the saved image. Difficulty in this area can be avoided entirely by assigning unique account/names to each user.

The method of retention is to save the user's swap image when the disconnection occurs. File position is maintained for all open files. Since files remain open, update files will be inaccessible to other programs for the duration of the holding period. Files are properly closed at the termination of the holding period.

Terminal I/O in progress, which may amount to several lines, is discarded in the suspension process. If the user affected by the line disconnection is in TEL with no active program, the suspension is disregarded and an ordinary log-off occurs.

The operator may abort a user when the user is in this suspended state. If the affected program has taken exit control via M:XCON, then control is passed to that routine after the operator abort (or, if no abort, after the expiration of the retention period). Thus, if a user reconnects to the suspended program within the retention period, no exit control action occurs.

When the disconnected user logs back onto the system, LOGON recognizes that a swap image exists for his account/name combination and issues the following message:

```
PROGRAM HELD. RECONNECT?
```

The user then responds with either Y or N. If Y, the user is reconnected to the suspended image and continues from the point of the disconnect. (However, I/O going to and from the terminal may have been lost.) If the response is N, the swap image continues to be retained. (The retention time is not changed.)

In the rare instance in which a user's suspended image times out between the time he responds Y to the 'RECONNECT?' message and the time the internal reconnect routine is executed, the message

```
HELD PROGRAM TIMED OUT
```

is printed and the log-on process continues in the normal manner.

TYPING LINES

The rules governing the typing of lines are concerned with operations such as erasing characters or lines, terminating lines, pagination, tabbing, and so on. These rules are common to TEL, all processors, and programs that carry on a line-by-line dialogue with the user. (These rules are controlled by the COC REREAD function, a function which is mentioned in the description of the Edit processor's RR command.)

PROMPT CHARACTERS

When a connection is first established between a terminal and the computer, a message is sent to the terminal requesting the user to log on. As soon as the user has logged on, TEL types a prompt character at the left margin of the next line to indicate that requests may be entered. Thereafter, a prompt character is sent to the terminal following a completed request, an error, or an interruption by the user. If the services of a processor are requested, the processor identifies itself with a different prompt character.

The prompt characters used by TEL and all on-line processors are as follows:

TEL	!
FORT4	>
COBOL	$
META	>
BASIC	>
EDIT	*
FDP	@
DELTA	bell (may be changed by user)
PCL	<
LINK	:
Libraries	?

DIRECT AND INDIRECT PRINTING MODES

There are two modes of terminal operation under CP-V: direct printing mode and indirect printing mode.

In direct printing mode, characters are printed at the same time that they are sent to CP-V; i.e., there is a direct electrical/mechanical connection between keyboard and printer.

In indirect mode, characters input at the terminal are not printed until they are echoed by CP-V; i.e., there is no electrical/mechanical connection between keyboard and printer. Sometimes the characters echoed are different from those entered (e.g., responses to control sequences).

ECHO MODE AND NO-ECHO MODE-ESC E

Note: If half-duplex modems are used, echoing is done by the terminal and the following paragraph doesn't apply.

The system can interact with a terminal in one of two modes: echo mode or no-echo mode. Normally the system is in echo mode and echoes each character sent from the terminal. (Echoing a character means printing it on the terminal.) The terminal user can select the no-echo mode by depressing ESC and E sequentially. In the no-echo mode, CP-V does not echo characters input but does send certain control sequences as necessary (e.g., echoing LF after receipt of CR). Successive use of the ESC E key sequence toggles the echo/no-echo modes.

The diagram below shows the relationship between the terminal direct/indirect printing modes and the echo/no-echo system interaction modes. Each box indicates the number of characters that are printed on the terminal in response to a character that is input. (This illustrates the general rule but control characters are exceptions.)

	Echo	No-echo
Indirect	one character	no character
Direct	two characters	one character

Thus it can be seen that no-echo mode can be selected to suppress printing of passwords or other security-related information when the terminal is in the indirect printing mode. The no-echo mode must be selected for direct printing terminals in order to suppress double printing of characters input.

TOGGLING ASCII APL MODE - ESC A

If the user is using an ASCII terminal with an APL character set (such as the Diablo Hy-Type based units), he may obtain the correct translation (of the code that his terminal is generating to EBCDIC – and vice versa) by typing ESC A.

This may be done during the log-on sequence, thereby making the changing of type-wheels unnecessary. If the user accidentally types ESC A on a terminal with standard ASCII (not APL) translation, the terminal will begin typing incorrectly. To determine whether or not ESC A was inadvertently entered, enter CONROL Y. If ESC A is in effect, TEL's exclamation mark prompt character will print as the character J. Also, a colon will print as the "greater than" character (>).

ERASING CHARACTERS-RUBOUT OR ESC RUBOUT

Depressing the RUBOUT key (or the ESC RUBOUT sequence) erases the last unerased character. The system responds by typing a backslash (\) to indicate that it has effectively backspaced and erased unless backspace-overstrike edit mode is enabled (see ESC O). On terminals that can backspace, backspacing does not erase. Thus, it is possible to overstrike characters as well as to erase them.

There is no specific way to erase an ESC. One can effectively be erased by causing the ESC to do nothing. For example:

ESC S   ESC S

ESC Q

ESC R

ESC RUBOUT and reenter the last character

ERASING THE CURRENT INPUT LINE - ESC X

The current input message (one line or less) is erased by depressing the two keys ESC and X sequentially. If an input operation is pending, the system types a left-facing arrow or underscore, returns the carriage to the position at the beginning of input on the next line, and returns control to the user without further comment. The current message may then be entered.

CANCELLING ALL INPUT AND OUTPUT - CONTROL X

Depressing the CONTROL and X keys simultaneously causes all input (including messages typed ahead) and all output to be deleted. If an input operation was pending, additional action is identical to that for ESC X above.

ENTERING BLANK LINES

Blank lines are usually ignored by processors. However, some processors, in certain modes, treat a blank line as a command to change to a different program control level.

RETYPING THE CURRENT LINE - ESC R

When the ESC R sequence is received, the carriage is returned to the position at the beginning of input on the next line and all characters accumulated will be retyped by the system. The user is then allowed to complete the message.

ENTERING MULTILINE RECORDS – LOC RET, ESC LINE FEED OR ESC RET

On a terminal unit having an inherent line-width limit of less than 140 (e.g., Teletype Models 33, 35, and 37), a single, multiline record may be entered in either of two ways:

1. Using the local carriage return key marked LOC RET, if present, to "break" the input line without releasing it to the system.

2. Using the simulated local carriage return sequence ESC RET or ESC LINE FEED for the same purpose. However, if backspace edit mode is enabled, ESC LINE FEED will position the carriage and the edit point to the beginning of the input message.

TERMINATING LINES

When TEL or a processor that carries on a line-by-line dialogue is in use, an "end-of-message" is signaled by depressing either the RETURN or LINE FEED key. Depressing the CONTROL and L keys simultaneously signals an "end-of-message" and an "end-of-page". FS, RS, US, and GS (see Table A-3) signal "end-of-message" without carriage motion or character printing. ESC and then F signals "end-of-file". Each read operation at the terminal specifies a maximum number of characters to be read (never more than 140). If this number is reached, "end-of-message" is signaled. TEL read operations are restricted to a maximum of 80 characters.

End of message can also occur when a user-controlled timeout on an M:READ occurs. (See the CP-V/BP Reference Manual 90 17 64.)

TYPING AHEAD

COC routines allow 'type-ahead' operations. Key strokes (or paper tape frames) that are input by the user before the system requires them will be retained until an M:READ is issued.

TOGGLING BACKSPACE EDIT MODE - ESC O

The normal mode for processing backspace characters is to pass the characters to the user's buffer and include them in the count of characters received during the read. However, backspaces may be used to position the carriage to the left to replace, delete, and add to the characters previously typed. The backspace edit mode is enabled (via toggling) when the sequence ESC O is received. If a backspace character or ESC LF sequence is received while backspace edit mode is enabled, backspace edit mode will become active. The point in the input buffer where the next input character would have been placed becomes the "edit point". The carriage will normally be positioned directly under the edit point. While backspace edit mode is active, characters are processed in the following manner:

1. Backspace characters move the edit point one column to the left. The first backspace will upspace the carriage one line.

2. An ESC LF sequence will set the edit point to the beginning of the input message, and the carriage will be upspaced one line.

3. DC2 characters (usually CONTROL R, R chosen to signify "right")move the edit point one column to the right, without changing any characters. If the edit point is advanced back to the original edit point, backspace edit mode will no longer be active.

4. A RUBOUT will delete the character at the edit point, and move the edit point one column to the right.

5. End of message characters cause the message to be terminated and the appropriate end of message character to be placed after the rightmost character.

6. A TAB character input when tab simulation is on will move the edit point to the next tab stop, skipping over and not changing any characters in between. If the new edit point is to the right of the original edit point, the character positions from the original edit point up to the new edit point will be filled with spaces and backspace edit mode will no longer be active.

7. Any other character will replace the character at the edit point, and move the edit point one column to the right.

8. ESC R sequences may be entered to display the current line with editing applied.

The backspace edit mode is turned off when the APL processor is called.

Warning: If space insertion is off, backspace editing may yield erroneous results when handling tabs.

TOGGLING CHARACTER INSERT MODE-ESC J

By using character insert mode, the user can insert new characters between characters already typed. Insert mode is entered when an ESC J sequence is typed while backspace edit mode is active. Subsequent characters will be inserted just to the left of the edit point. (A backslash (\) is echoed, which points to the place the characters will go.) Insertion will continue until one of the following happens:

1. The user types a backspace, DC2, TAB, CONTROL X, ESC D, ESC J, ESC X, ESC Z, ESC CR, or ESC LF.

2. An activation condition is met.

RUBOUTS will delete the last character typed. When insert mode is terminated, the edit point will be at the same character as when it was initiated. ESC R sequences may be entered to display the current line with editing applied.

RE-READING AN INPUT MESSAGE-ESC D

ESC D may often be used to retrieve a completed message just entered and passed to the reading program. The message will be transferred back to the monitor's input buffers, and echoed as if the user had just re-typed the message. The user may then edit the message and again release it to the reading program. Re-read handles the transfer in the following manner:

1. When a read is issued to the terminal, the user buffer is inspected. (ESC D will force the read to be re-issued.)

2. Trailing blanks are ignored.

3. Characters are transferred from the program's buffer to the monitor's input buffers until either an activation character is found or a character is found that the monitor would not have placed there.

4. Any typed-ahead input is placed after the re-read characters.

5. The characters in the input buffer are echoed.

Some processors and programs change the buffer they use for input between M:READ requests; ESC D will not function properly when used with them.

(This feature is also usable under program control via the COC,REREAD option on the M:READ procedure. See the CP-V/BP Reference Manual, 90 17 64.)

PAGINATION AND LINEATION - CONTROL L

Depressing the CONTROL and L keys simultaneously requests the system to paginate (i.e., to begin at the top of a new page at the terminal). Non-CRT pagination consists of the following:

1. Blank lines to the page bottom.
2. A heading line containing date, time, user identification, terminal identification, and page number.
3. Five additional blank lines.
4. User heading line, if any.

If the page length established for the terminal is 11 lines or less, CONTROL L has no effect except as described above under "Terminating Lines". (See the TEL PLATEN command for a discussion of setting page length and width.)

SIMULATING TAB STOPS - ESC T

The user can enter tabulation characters into his terminal input, either with the CONTROL and I key combination or ESC I sequence on teletypewriter units, or the TAB key on terminal units that have it. The system simulates tabbing by typing (echoing) successive blank characters. Tab-stop values for this simulation can be set or changed by the TABS command. This tab simulation is under the user's control: it is normally 'on' at the beginning of a terminal session, but the user can turn it off, and back on again, with the ESC T key sequence (i.e., successive use of the ESC T sequence has a toggle-switch effect on tab simulation, each use reversing the previous on or off state). With tab simulation on, any tab characters either sent from the terminal or received for transmission to the terminal are replaced at the terminal by an appropriate string of blanks (in lieu of mechanical tabulation). If no tab-stop values are set, each tab character is replaced by a single blank. The state of tab simulation does not determine whether or not blanks are substituted for a tab character in the input stream received by the processor or program requesting the input.

<u>Note:</u> The tab simulation is turned off when the APL processor is called.

SIMULATING TAB CHARACTERS-ESC I AND CONTROL I

The ESC I and CONTROL I sequence are treated exactly as a tab character. This function is provided for terminals that are not equipped with a TAB key.

INSERTING SPACES - ESC S

One or more spaces are normally inserted into the terminal input stream in place of a tab character (i.e., the tab characters themselves are not normally passed to the processor or program requesting the input). When space-insertion mode is on (initial state), each tab character is replaced in the input stream by an appropriate number of blanks if tab settings are in effect. If there are no tab settings, only a single space is inserted. This space insertion is under the user's control. However, he can turn space-insertion mode off by use of the ESC S key sequence, causing the tab characters to remain in the input stream. (This can result in a significant space saving in large files.) Successive uses of ESC S toggles the space-insertion mode from on to off, off to on, etc.

<u>Note:</u> The space insertion mode is turned off when the APL processor is called.

SETTING THE TAB RELATIVE MODE - ESC C

Normally tabs are considered to be physical carriage positions. If the tab relative mode is active, tabs on input are considered to be offset from the position of the carriage at the beginning of input.

The tab relative mode is toggled on or off by the ESC C character sequence. The tab relative mode is initally set to OFF.

RESTRICTING INPUT TO UPPER CASE - ESC U

When the character pair ESC U is received, a flag that controls alphabetic characters is toggled. When set, all lower case letters received are translated to their upper case counterparts.

INTERPRETING UPPER CASE AS LOWER CASE - ESC)

Receipt of the ESC) sequence causes

1. All subsequent upper case alphabetic characters to be interpreted as the corresponding lower case alphabetic characters.

2. The terminal characters @ [\ ] ^ _ to be interpreted as ' { : } ~ DEL respectively.

This remains in effect until the ESC (sequence is received. The parentheses are echoed, thus bracketing the characters that were interpreted as upper case. This feature is provided to enable terminals that are upper case only to input lower case characters.

EXITING THE LOWER CASE INTERPRET MODE - ESC (

The ESC (character sequence removes the effect of the ESC) sequence (described previously).

CONVERSING WITH A COUPLED TERMINAL - ESC Z

The ESC Z sequence allows coupled terminals to converse freely without feeding any input to running programs or processors. (Terminal coupling is discussed in Chapter 3.)

IGNORE OUTPUT MODE - ESC W

Receipt of the ESC W sequence causes

1. All pending output to be deleted.

2. All future output to be discarded, until a read request is issued to the terminal for which a complete input record has not been typed ahead. ESC W mode is then turned off, and processing continues normally. User type ahead will be echoed when processed.

HALT OUTPUT MODE - ESC H

Receipt of the ESC H character sequence causes output processing to immediately halt. This is useful for CRT terminal users because it gives them a chance to read their output before it is lost from the screen. Receipt of any of the following characters will cause output processing to resume: CARRIAGE RETURN, BACKSPACE, CONTROL Y, ESC, CONTROL X, RUBOUT, or BREAK. When any one of the above characters (except CONTROL Y) is received while in the halt mode, the character itself is ignored except for the resetting of halt mode.

TYPING COMMANDS

Except for a few declaratives, commands take the form of imperative sentences. They consist of an imperative verb followed by a direct object or list of objects. Indirect objects usually follow a preposition but may follow the verb with elision of the implied direct objects. Minor variations of this structure are expressed as encoded parentheticals following either the verb or one of the objects. Individual elements of a list of objects are set off from one another by commas.

Common rules of composition are applicable to commands. Words of the language, numerals, object identifiers, and other textual entities may not be broken by spaces. Otherwise, spaces may be used freely. For purposes of scanning commands, both by machine and by human eye, this rule ha a simple interpretation. Leading spaces are skipped over in a left-to-right scan for the next syntactic element of a command, and trailing spaces are treated as terminators for words, numerals, and other textual entities. In terms of machine scanning, tabs are treated as spaces.

Since it is impossible to determine whether or not trailing characters in a command are in error, a unique code that identifies the end of the command is recognized as a syntactic element. For TEL and processors that carry on a line-by-line dialogue, this is either a LINE FEED or CARRIAGE RETURN code.

DETECTING AND REPORTING ERRORS

The primary object of the error detection procedure is that user information should not be destroyed by an attempt to execute a command that cannot be carried through to completion. To ensure that each command is at least formally valid, TEL and all processors that carry on a line-by-line dialogue always parse an entire command before starting an operation.

Error messages sent to a terminal are as terse as possible since the majority of errors are easily found once the fact that an error exists has been brought to the attention of the user.

The error messages and actions initiated by the errors are contained at the end of each chapter for the processor to which they apply. Many processors use the following format for reporting garbled, malformed, or unintelligible commands:

```
EH ? @ n
```

where n gives the character position at which the confusion was first encountered.

INTERRUPTING CP-V

CP-V can be interrupted whenever it, one of its processors, or a user program has control of the keyboard. Subsequent control depends on which interrupt keys are used and which processor or user program is in control.

CONTROL Y, ESC Y, OR ESC ESC

Regardless of what program is in control of the keyboard, the operation can always be interrupted by simultaneously depressing the CONTROL and Y keys (or by typing the ESC Y sequence or the ESC ESC sequence). The system responds by stopping the current operation as soon as there is a convenient breakpoint and turning control over to TEL (or to the command processor with which the user is associated). All input received prior to this key-in that has not yet been read by the program will be erased. Output that is queued for transmission will also be deleted if CONTROL Y is entered.

BREAK, ESC B

The BREAK key and the ESC B sequence generally perform the same function. The one difference is that output is deleted when a BREAK is entered, whereas it is not deleted when ESC B is entered. Other than that, the discussion about BREAK below applies to ESC B as well.

If CP-V, one of its processors, or a program explicitly requesting break control is in control of the keyboard, the operation can be interrupted by depressing the BREAK key. This action gives control to the program that is currently in communication with the terminal after pending input (and output for BREAK) have been deleted.

A succession of four or more BREAK signals returns control directly to TEL (or to the command processor with which the user is associated)unless the running program resets the BREAK count via the M:STA procedure. There are two reasons for this return. First, some actions can only be stopped at points of convenience and others have so much inertia they cannot be stopped at all. Second, machine or program errors may have disabled the program's response to the BREAK signal. However, it must be emphasized that depressing the BREAK key one time does not constitute a preemptive request for the services of TEL as does depressing the CONTROL and Y keys (or the ESC Y or the ESC ESC sequence).

The precise handling of interruptions by processors is defined by the processors. The handling of interrupts by object programs is defined by the calls these programs can make on system services. If the user does not have break control, interruption of an object program always causes a return to TEL. In general, interruption of the system or any of its processors results in termination of the current operation as soon as possible and return of control to the user after the appropriate prompt character has been typed.

If a BREAK is received before the user has logged on and if additional on-line users are not allowed, the following will be printed:

NEW ONLINE USERS NOT CURRENTLY ALLOWED

If the operator has issued a SEND,ALL or HEADER key-in the contents of the keyed in message will follow.

ESC Q

Teletype users may request acknowledgment from the system at any time by use of the ESC Q sequence. The system will respond by sending a space, an exclamation point (!), the user's scheduling state, and another exclamation point. The scheduling states and their descriptions are listed in Table 2.

Table 2. Scheduler States

State	Description
SC0	Compute – 0
SC1	Compute – 1
SC2	Compute – 2
SC3	Compute – 3
SC4	Compute – 4
SC5	Compute – 5
SC6	Compute – 6
SC7	Compute – 7
SC8	Compute – 8
SC9	Compute – 9
SC10	Compute – 10
SCU	Current User
SIOW	IO Wait
SIOMF	IO Master Function Count Exceeded
SNULL	NULL (no user associated with line)
SQA	Queued for Access
SQFI	Queued for Interrupt
SQR	Queued for Resource
SQRO	Queued for Resource, Out of core
SRT	Real-time
STI	Terminal Input
STIO	Terminal Input, Out of core
STOB	Terminal Output Blocked
STOBO	Terminal Output Blocked, Out of core
SW	Wait

PAPER TAPE INPUT

Paper tape may be punched off-line on Teletype terminals and subsequently read on-line after the user has logged on and a prompt for data on the tape has been issued. The same characters that are keyed in during on-line input may be punched into paper tape. The procedure for reading paper tape on-line is as follows:

1. Insert the paper tape in the paper tape reader.

2. Depress the X-ON (Q^c) key. This will start the reading of the tape by the paper tape reader under control of the computer.

3. Depress the X-OFF (S^c) key to turn off the paper tape mode (read operation).

Rubout characters are ignored during a paper tape read operation. This enables the user to use rubout characters to delete unwanted characters as in normal paper tape operation.

The paper tape read mode is set when a DC1 character (X-ON) is received from the Teletype. It is reset (to normal processing) when a DC3 character (X-OFF) is received. Characters that are input through the keyboard while the Teletype is in the paper tape mode are normally received after the reader reaches the end of the tape or the tape is removed from the reader.

When the system is in the echo mode, X-ON and X-OFF characters are sent to the terminal as necessary to control the input rate and ensure that buffer space is not exhausted resulting in lost characters.

When the system is in the no-echo mode, no X-ON or X-OFF characters are sent to the terminal. When buffer space exhaustion is imminent, however, warning BEL characters are sent to the terminal.

Restrictions:

1. Line feed (LF) characters received after any other activation condition is reached are ignored unless Delta is reading.

2. The full duplex paper tape facility requires the X-ON, X-OFF option on the Teletype.

HALF DUPLEX PAPER TAPE READING MODE

A special mode is available for half duplex terminals that are reading paper tape. (A half duplex terminal does not respond to X-ON and X-OFF characters sent by the system while it is sending paper tape input to the system – even though it is connected to the system with full duplex circuitry.) While in this mode, no attempt is made by the monitor to turn the tape reader off or on. Input is accepted until available buffer space is exhausted. No program output, prompt characters, or echoes are sent to the terminal because the mode renders the terminal incapable of accepting output.

The half duplex paper tape mode is entered upon receipt of an ESC P sequence or upon receipt of an X-ON character while in the non-echoplex mode (controlled by ESC E). The mode is exited by a balancing ESC P sequence or by an X-OFF character if it was initiated by X-ON.

2741 AND TELETYPE DIFFERENCES

In addition to the differing code sets that are translated in a straightforward way, certain unique features of 2741 terminals must be treated in a special way. First, use of the 2741 terminal is proprietary. Both computer and user must, in turn, explicitly release control of the typewriter to the other. Second, multiple code sets are supported and must be properly identified at log-on time. Third, the important functions provided on Teletypes by the ESC and BREAK keys are combined in the 2741 ATTN key.

LINE STATE

Unlike Teletypes, 2741 terminals cannot transmit and receive at the same time. The 2741 operator can type only when the computer has unlocked the keyboard. The computer can type only when the operator has locked the keyboard by ending his message with a carriage return or attention character.

LOG-ON

When a Teletype line is connected to the system, a log-on message is automatically sent to the terminal. Logging on from 2741 lines must be handled differently since the keyboard is initiated for user input when the line is connected and the code set and keyboard arrangement are unknown to the computer at this point. The user at the 2741 terminal must identify the code set and type of keyboard before logging on by sending an asterisk (*) followed by a carriage return character. From this point on, the standard log-on sequence is followed.

BREAK AND ESC

Separate BREAK and ESC keys are not present on 2741 keyboards. On these keyboards, the BREAK and ESC functions are performed by the ATTN key. During input, while the keyboard is unlocked, depressing the ATTN key sends an EOT character to the computer. When an EOT character is input to the computer, an escape sequence is performed. The control function is represented by the character input just prior to the EOT.

If the ATTN key is depressed while the keyboard is locked, the following happens:

1. If no printing was in progress, a break is reported as follows:

   If the user's program contains an M:INT procedure call, control is given to the user at the address specified by the procedure call. If the user's program does not contain an M:INT procedure call, control is given to the monitor.

2. If printing was in process, all unprinted output is deleted. If this is the first ATTN without any other input actions, no break is reported – it is just counted for the "four breaks equals CONTROL Y" rule. If this is a consecutive ATTN, a break is both reported (as described above) and counted.

UPPERCASE/LOWERCASE

Both uppercase and lowercase letters are available on the 2741. There are many cases in which lowercase letters will not be accepted in lieu of uppercase letters (e.g., any letters in id/account for LOGON must be entered in uppercase). If lowercase letters are not required, a u ATTN may be entered, causing all letters tobe accepted as uppercase. (This is not applicable for the APL processor.)

COC ROUTINE

A number of functions are performed in the COC routine to accommodate 2741 terminals. These functions are outlined below.

LOG-ON PROCEDURE

The proper translation table is determined by a special dial-up procedure for 2741 lines. When the asterisk key is depressed, a different code is transmitted for the EBCD and Selectric code sets (both APL and standard versions). This character (*), followed by a carriage return character, is the protocol for 2741 lines to log on with the proper translation table. If the asterisk character is not entered just prior to the carriage return, a space and backspace are transmitted to indicate that the line has been connected but the translation table has not been determined. The procedure can then be repeated.

SPECIAL CHARACTERS

Backspace: The normal mode for processing backspace characters is to put the character in the user's buffer and to include it in the count of characters received. See below for backspace-overstrike editing.

Tab: Tab characters are processed as for a Teletype except that on input, spaces are not echoed to the terminal to position the carrier to the next tab stop since actual physical tabs are available on the 2741 keyboard. Note that the tab settings on the keyboard should correspond to the current system tab settings.

Attention: The ATTN key performs the BREAK and ESC functions of a Teletype terminal (see the section titled "BREAK and ESC"). When a character that represents a control function is detected by the COC handler, a backspace and underscore are transmitted to the terminal to identify the character as part of an escape sequence. An exception is the escape sequence used to retype a line (R ATTN) which results in an R being typed at the terminal before the carrier is returned and the line is retyped.

The ATTN key should also be used any time a character has been entered to which CP-V normally responds immediately (i.e., without a subsequent Carriage Return or New Line). For example, CP-V normally responds to the Delta command characters / and = immediately. On the 2741, the commands should be typed /ATTN and =ATTN. CP-V will respond immediately after the ATTN key is depressed. Other examples of this situation are described later in this manual in the "Break Function" section of the Edit processor description and in the description of the PCL REVIEW command.

Lowercase Carriage Return (Carriage Return): The input message is terminated and a carriage return character (X'0D') is put in the user's buffer.

Uppercase Carriage Return (Line Feed): The input message is terminated and a line feed character (X'15') is put in the user's buffer.

CONTROL CHARACTERS[†]

Certain terminal characters perform control functions if preceded by an ESC on the Teletype or if followed by an ATTN (EOT) on the 2741. For the control characters listed below, the function performed is the same on both types of terminals:

C - Toggle tab relative mode.

F - End of file.

L - Form feed.

R - Retype.

S - Toggle space insertion.

T - Toggle tab simulation.

U - Toggle restrict code to upper case.

Y - Escape to monitor.

(- Upper case shift.

) - Lower case shift.

The functions of the following 2741 characters do not correspond directly to Teletype characters.

BACKSPACE: A BACKSPACE ATTN sequence on the 2741 is the same as the Teletype ESC RUBOUT sequence, except in the backspace edit mode (which is discussed below).

T: The tab simulation mode is switched from on to off or vice versa with the T ATTN sequence. If the mode is on during output, enough blanks are sent to the terminal to move the carrier to the next higher tab position. Since actual physical tabs are available on 2741 keyboards, the tab-simulation mode state is ignored during input.

B: Since a break signal is sent to the computer only if the keyboard is locked, the B ATTN sequence simulates a break when the keyboard is unlocked and is treated like a break signal on a Teletype during input.

SPACE: The input line is terminated and the end-of-message character US (X'1F') is put in the user's buffer when the SPACE ATTN sequence is received.

O: The backspace edit mode is switched from off to on or vice versa when the O ATTN sequence is received. If on, backspace editing is performed inside the COC handler (see below).

X: The X ATTN sequence performs the CONTROL X function of a Teletype terminal. This causes all input and output to be deleted. Note that this is not equivalent to the ESC X sequence on a Teletype terminal which causes only the current input line to be deleted.

N: The N ATTN sequence performs a "local carriage return"; i.e., an N carriage return is output to the terminal and the terminal is again selected for input. The N, the underscore (_), and the carriage return are not sent to the user's program. The N ATTN function is equivalent to the ESC CR function on the Teletype.

BACKSPACE EDITING

The standard mode for processing backspace characters is to pass the character to the user's buffer and to include it in the count of characters received. The backspace edit mode is invoked when the character sequence O ATTN is received. If a backspace is received in this mode, the pointer into the input buffer for the next character is saved and then decremented by one character. Characters are processed in the following manner until this pointer is incremented to its original position:

[†] When the APL processor is used, the control characters are disabled for the 2741.

1. Additional backspace characters decrement the pointer that points to the current character in the input buffer by one.

2. A space increments the pointer that points to the current character in the input buffer by one.

3. The character sequence BACKSPACE ATTN is an explicit blank. The current character in the buffer is replaced with a blank and the pointer that points to the current character in the input buffer is incremented by one. Two SPACE characters are sent to the terminal to correct the carriage position.

4. End of message characters (NL, L ATTN, F ATTN, or SPACE ATTN) cause the message to be terminated and the appropriate end of message characters to be placed at the end of the current line (after the rightmost character).

   Any character other than another backspace, blank, or end of message overlays a character in the buffer; the pointer that points to the current character in the input buffer is incremented by one.

When the pointer that points to the current character in the input buffer becomes equal to its original value, normal processing of input characters resumes.

The backspace edit mode is turned off when the APL processor is called.

Warning: If space insertion is off, backspace editing may yield erroneous results when handling tabs.

SUMMARY OF 2741 AND TELETYPE DIFFERENCES

Table 3 summarizes the differences between 2741 terminals and Teletype terminals. (Refer to Table A-5 for substitutions for characters nonexistent on 2741 terminals.)

Table 3. Summary of Differences Between 2741 and Teletype Services

Function	Teletype	2741
Get log-on message	BREAK (if hardwired line)	* and CRLF if dialing up. ATTN if line is already connected.
Erase line	ESC X	none
Tab relative	ESC C	C ATTN
Suppress lowercase	ESC U	U ATTN
Uppercase shift	ESC (	(ATTN
Lowercase shift	ESC)	) ATTN
Erase last character	RUBOUT	BACKSPACE ATTN
Tab	ESC I, CONTROL I	TAB
End of input	FS, RS, US, GS(L^{cs}, N^{cs}, O^{cs}, M^{cs})	SPACE ATTN
Line continuation	ESC CR, ESC LF, LOC CR	N ATTN
Retype	ESC R	R ATTN
Toggle tab simulation mode	ESC T	T ATTN
Toggle space insertion mode	ESC S	S ATTN
End of file	ESC F	F ATTN
Monitor escape (to TEL)	ESC ESC, CONTROL Y, ESC Y, or 4 BREAKs	Four ATTNS. Also, Y ATTN if input.
Break	BREAK, ESC B	B ATTN on input or ATTN on output.
Toggle backspace edit mode	ESC O	O ATTN
Form feed	ESC L	L ATTN
Half duplex paper tape	ESC P	none
Toggle ECHO mode	ESC E	none
Acknowledge	ESC Q	none
Erase all input and output	CONTROL X	X ATTN
End of transmission	CONTROL D	ATTN
Halt output	ESC H	none
Ignore output	ESC W	none
Ignore input	ESC Z	none
Re-read	ESC D	none
Toggle insert mode	ESC J	none
ASCII APL mode	ESC A	none

3. TERMINAL EXECUTIVE LANGUAGE

INTRODUCTION

The Terminal Executive Language (TEL) is the principal terminal language for the system. Most activities associated with COBOL, FORTRAN and assembly language programming can be carried via TEL commands. Examples are:

Major Operations

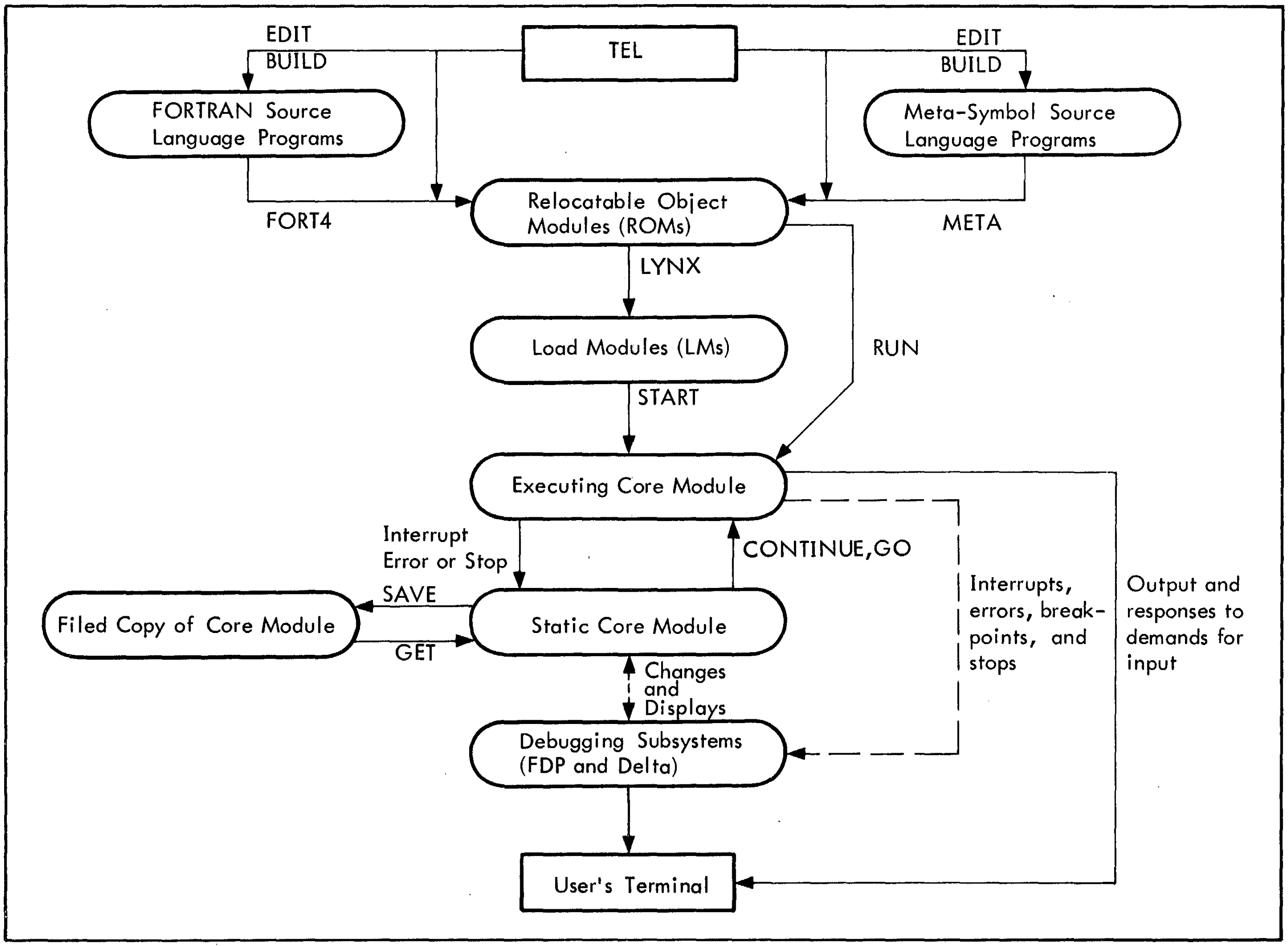
Composing program and data files.
Assembling and compiling programs.
Linking object programs.
Loading programs and initiating execution.
Initiating debugging operations.
Managing and backing up files.
Submitting batch jobs.
Calling processors.
Interrupting, resuming, and terminating execution.

Minor Operations

Logging off.
Changing the log-on password.
Checkpointing on-line sessions.
Assigning I/O devices and DCB parameters.
Modifying logical device stream definitions.
Requesting extended memory mode.
Determining on-line user status.
Listing a file directory.
Listing system load parameters.
Setting simulated tab stops.
Displaying simulated tab stop settings.
Obtaining terminal status.
Changing terminal type.
Changing terminal platen size.
Displaying terminal platen size.
Changing terminal header page number.
Sending messages to operator.
Printing or punching output.

MAJOR OPERATIONS

Figure 2 illustrates the sequence in which major operations normally take place. Capitalized words identify TEL commands and CP-V processors that are used to carry out the various programming activities.



TEL
EDIT
BUILD
EDIT
BUILD
FORTRAN Source Language Programs
Meta-Symbol Source Language Programs
FORT4
Relocatable Object Modules (ROMs)
META
LYNX
Load Modules (LMs)
RUN
START
Executing Core Module
Interrupt Error or Stop
CONTINUE,GO
Interrupts, errors, breakpoints, and stops
Output and responses to demands for input
SAVE
Filed Copy of Core Module
Static Core Module
GET
Changes and Displays
Debugging Subsystems (FDP and Delta)
User's Terminal

Figure 2. FORTRAN and Assembly-Language Programming

A Meta-Symbol or FORTRAN program may be composed on-line in one of two ways. It may be composed and filed away by the Edit processor, which is called by the EDIT or BUILD commands, or entered directly from the terminal one line at a time after Meta-Symbol or FORTRAN has been called with a META or FORT4 command. In both cases, program assembly or compilation is initiated by the META or FORT4 command and a relocatable object module (ROM) and program listing may be produced. Output is directed to the files or devices specified by the user.

Relocatable object modules that have been assembled or compiled separately are put together by the LYNX command to form a load module (LM). On completion of the linking operation, execution is started by the START command. Or, if desired, both linking and execution can be initiated by a single RUN command.

Debugging activities are initiated by starting the execution of a load module under control of one of the debugging processors, Delta or FDP. Delta is most appropriately used for debugging Meta-Symbol programs but may be used for debugging any program. It may always be called into association with an executing user program for aid even after execution has begun. FDP is used for debugging FORTRAN programs.

An executing program becomes a static core module whenever it is interrupted or whenever an error occurs. This static core module can be stored by the SAVE command and retrieved later by the GET command; it can then be restarted with the CONTINUE or GO command.

COMPOSING PROGRAM AND DATA FILES

The Edit processor provides for line-at-a-time composition and editing of files and is called in either of two ways:

E[DIT] [fid]

B[UILD] fid

File identification (fid) has the following format:[†]

$$\text{name}\begin{bmatrix}\text{. account}\\ \text{. account. password}\\ \text{. . password}\end{bmatrix}$$

where

name is the name of the file and may have a maximum of 31 characters. (TEL, Link, and Load allow a maximum of 10 characters.)

account is the account number of the file and may have a maximum of eight characters. A user may not create (BUILD) a file in any account other than the one under which he is running. He may not EDIT a file in another account unless he has write access to the file.

password is the password for the file and may have a maximum of eight characters.

Account and password are optional, defaulting to the log-on account and no password if omitted. The characters not allowed in the three elements of a fid for Edit and all on-line systems are

| ; < > . / = ?

and all control characters acted upon by COC.

When called by EDIT, the Edit processor opens the specified file (fid) for updating (if a file is specified), issues the Edit prompt character (*), and then waits for input of commands. (Reference: Chapter 6.)

When called by BUILD, Edit assumes that a new file is to be entered a line at a time, beginning with line number 1.000 and continuing in increments of one. Edit responds by printing the number of each line at the left margin and waiting for entry of the line. The end of the file is signaled by entering an empty line. Edit is available for corrections and other editing operations after the file has been keyed in.

ASSEMBLING OR COMPILING PROGRAMS

TEL has four commands that permit a program to be assembled or compiled into a single ROM. These commands are

$$\text{META}[\text{sp}]\left[\begin{matrix}\text{ON}\\ \text{OVER}\end{matrix}[\text{rom}]\ [\text{,list}]\right]$$

$$\text{FORT4}\,[\text{sp}]\left[\begin{matrix}\text{ON}\\ \text{OVER}\end{matrix}[\text{rom}]\ [\text{,list}]\right]$$

$$\text{COBOL}\,[\text{sp}]\left[\begin{matrix}\text{ON}\\ \text{OVER}\end{matrix}[\text{rom}]\,[\text{,list}]\right]$$

$$\text{ANSF}\,[\text{sp}]\left[\begin{matrix}\text{ON}\\ \text{OVER}\end{matrix}[\text{rom}]\ [\text{, list}]\right]$$

where

sp specifies a source program and may be either a file identification (fid) or the terminal identification (ME). If no source file is specified, TEL assumes input is from the file/device currently assigned to the M:SI DCB. If the M:SI DCB is not assigned, TEL expects input to come from the terminal (ME). The M:SI DCB is reset at each job step.

ON indicates that ROM output is to be on a new file or to a logical device stream or a device.

OVER indicates that ROM output is to be over an existing file or on a new file.

rom specifies that the relocatable object module produced by compilation is to be directed to a specified file (fid), a logical device stream (stream-id), the card punch (CP), or no file or device (NO). If no rom specification is given, output is directed to a special file that may subsequently be referenced by a dollar sign. If a rom is specified, it is assigned to the M:GO DCB and that assignment remains in effect throughout the job until reassignment by a subsequent SET, META, FORT4, COBOL, ANSF, or lmn command.

list specifies that listing output is to go to a file (fid), a logical device stream (stream-id), the

[†] This definition of file identification is intended whenever fid is used in a command specification; PCL commands allow a larger character set, however.

line printer (LP), the terminal (ME), or no file or device (NO). If list is notspecified, TEL assumes that the listing output is to go to the file/device currently assigned to the M:LO DCB. If the M:LO DCB is not assigned, TEL produces no listing output. Note that the M:LO DCBassignment is made explicitly by the SET command and implicitly by the META, FORT4, COBOL, and lmn commands. Once set, the M:LO DCB assignment remains in effect throughout the job until reassignment by a subsequent SET, META, FORT4, COBOL, or lmn command.

Whenever TEL encounters an input specification (sp) designating the terminal (ME), the program to be assembled or compiled must be entered through the terminal a line at a time. The end of program input is signaled by an end-of-file that is produced at the terminal by the key sequence ESC and F.

Any assignments made at a job step within META, FORT4, ANSF, and COBOL commands apply to all subsequent job steps, except for source input which always reverts to the terminal. These assignments maybe changed bysubsequent assignments either by META, FORT4, ANSF, or COBOL or by the OUTPUT, LIST, and COMMENT commands described below.

CONTROLLING OUTPUTS

Control over output may be exercised before the FORT4, ANSF, COBOL, or META command is entered. This is accomplished by the commands

$$\text{OUTPUT}\left[\begin{matrix}\text{ON}\\ \text{OVER}\end{matrix}\quad\text{fid}\right]$$

$$\text{LIST}\left[\begin{matrix}\text{ON}\\ \text{OVER}\end{matrix}\quad\text{list}\right]$$

$$\text{COMMENT}\left[\begin{matrix}\text{ON}\\ \text{OVER}\end{matrix}\quad\text{list}\right]$$

OUTPUT specifies the destination of ROM output and may designate a file (fid) only. LIST specifies the destination of listing output; COMMENT specifies the destination of error commentary. LIST and COMMENT may designate a disk storage file (fid), a line printer (LP), the terminal (ME), or a logical device stream (stream-id).

Output parameters set up in this way are valid across job steps from the time given until the session is terminated or until reset. They may be reset by other LIST, OUTPUT, COMMENT, and SET commands or by META, FORT4, ANSF, and COBOL commands that specify output.

Whenever output parameters are specified by the LIST, OUTPUT, and COMMENT commands, execution of multiple META, COBOL, FORT4, or ANSF commands without output parameters will continue to place the output from these operations on the same files. This is accomplished by file extension (see "Extension of Output Files"). After the initial log-on, ON is assumed by default.

Examples:

1. Assume that on ANS FORTRAN source program, on file A, is to be compiled. The name of the rom file is C, the name of the list file is D. Both C and D are new files. All files have the user's log-on account and password.

```
!ANSF A ON C,D(RET)
!
```

2. Assume the same conditions as in the previous example except that the output files are to be specified by LIST and OUTPUT commands. Error commentary is to go to the terminal.

```
!OUTPUT ON C(RET)
!LIST ON D(RET)
!COMMENT ON ME(RET)
   :
!ANSF A(RET)
```

Output from an assembly can be interrupted and turned off at any time by one of the following commands:

```
DONT LIST
DONT OUTPUT
DONT COMMENT
```

and turned on again subsequently with LIST, OUTPUT, or COMMENT, respectively. Entering one of these commands while a job step is in effect causes a temporary change that is local to the current job step. Entering a ! RESET command at the end of that job step causes the change to be permanent.

Output need not be directed to the terminal to be controlled by these commands. Error commentary is normally directed to the terminal and accompanies listing output, if specified.

EXTENSION OF OUTPUT FILES

File extension is a convention by which records are added to an output file by successive job steps. Each time the file is opened, the file pointer (RAD, disk pack, labeled magnetic tape, etc.) is positioned to a point immediately following the last record in the file. Thus, when additional output is produced it is added to the previous contents of the file, thereby extending it. File extension simulates output to physical devices, such as line printers or typewriters, when output is actually directed to a file.

File extension takes effect at the time that system output DCBs are opened. The output DCBs that are affected by file extension are those that are normally assigned by default to devices, either in batch or on-line operation, but that are explicitly assigned to a file (e.g., on RAD storage) at the time the DCB is opened. These DCBs include M:AL, BO, CO, EO, LL, LO, PO, SL, and SO. The M:GO DCB is also subject to file extension.

File extension is discontinued when a file is reassigned with a SET or other output-controlling (e.g., OUTPUT) command, or when a file is opened with an OPEN procedure call that specifies an explicit file name. In these cases, a new file is created. Extension of the GO file is terminated following a LYNX or RUN command.

ERROR HANDLING AND END ACTIONS

Whenever an operation is aborted, either because the operation cannot be continued or because a QUIT command is issued, the system restores certain specifications before reporting and returning control to the user. In particular, aborts occurring outside of TEL (within compilers, assemblers, or user programs) result in all previous output specifications and file assignments being restored to the specifications in effect at the beginning of the job step.

When syntax errors are encountered in input messages, the input is erased and an error message is sent to the terminal. An entirely new command must be issued.

Note: TEL ignores all parenthesized fields. This permits the passing of information to a program that would otherwise violate TEL syntax.

ENTERING PROGRAMS FROM TERMINAL

Whenever the input designator ME is encountered, such as in the processing of META, COBOL or FORT4 commands, the carrier is returned to the left margin of the next line and a prompt character is sent to the terminal. A program statement can then be entered. It is followed by a carriage return or line feed character to identify the end of the statement. Error commentary, if any, is sent to the terminal immediately thereafter. The end of source input is signaled either by the ESC and F keys (for META, COBOL, and FORT4) or by the appropriate processor command (such as END).

To aid in formatting, print columns on the terminal's platen are in a one-to-one correspondence with card columns. Trailing blanks are assumed for short lines. The terminal's tab stops should be set by the user to conform to the programming language being used and will be simulated if tab simulation is in effect. For FORTRAN, a single tab stop is set at column 7. For Meta-Symbol, tab stops are set at columns 10, 19, and 37.

The handling and simulation of tab stops is described in Chapters 2 and 11. Briefly, tab simulation works in the following way. Spaces are sent to the terminal to bring the carrier to the position indicated by the next tab position that has been set. Tabbing requested when the carrier is beyond the last set tab position is simulated by a single space.

DEBUGGING INFORMATION

The ROM output of a Meta-Symbol assembly contains sufficient information for subsequent debugging at assembly-language level under Delta. However, for symbolic debugging, a symbol table is needed. The user can get a symbol table by using the SD option during assembly.

To debug FORTRAN programs under FDP, additional information must accompany the compiled code. This information is not normally produced by the compiler since it increases the size of object programs and decreases their execution speed. To produce the information for a specific compilation, the DEBUG option for the FORT4 command must be used (Chapter 4).

LINKING OBJECT PROGRAMS

The on-line linking and loading of programs may be carried out by the LYNX processor. For compatibility with previous versions of the system, it may also be carried out by the Link processor. However, it is recommended that all new programs be linked and loaded with LYNX because LYNX is speed competitive with the Link loader, and in many cases will run faster than Link. The following paragraphs describe a few features of the LYNX processor. Both LYNX and Link are described in detail in Chapter 8.

LYNX constructs a single entity called a load module (LM) which is an executable program formed from relocatable object modules (ROMs). The LYNX processor is called by a LYNX or RUN command given at the TEL level. These two commands are discussed in some detail in this chapter and are described fully in Chapter 8.

ROMs and LMs are both representations of programs and data. ROMs are designed so that they can be efficiently combined with other ROMs, and LMs are designed so that they can be efficiently translated into executable programs and loaded into core. Both may be pictured as bodies of potential machine code to which symbol tables are appended. These symbol tables list the correspondence between the symbolic identifiers used in the original source program and the values of virtual core locations that have been assigned to them. Some of these identifiers are defined and referenced within the same module and are internal symbols. Others are defined (DEF) and referenced (REF) in separate modules and are global symbols.

Functionally, these modules can be compared with black boxes with labeled connectors dangling from them, some pointing out and others in. The labeled connectors are the global symbols associated with the modules; the internal connections have all been sealed and are hidden. In the process of linking modules, internal symbols associated with the constituent parts of the new load module are sealed and hidden, but all global symbols are still visible.

Continuing the black box analogy, if a module is split open, a jumble of internal connections should be visible. If the module has been tested and is ready for production, the internal connections need not be labeled. However, if the module is still in the debugging stage, the labels may be necessary. An option is provided in the LYNX command to indicate that the internal symbols associated with a module are to be kept with the resulting load module.

Note also that the names of ROMs and LMs must consist of from 1 to 10 characters.

LYNX Most commonplace linkages of ROMs can be carried out with a simplified form of the LYNX command

```
          [ON         ]
LYNX [ef[,ef]. . .]   [OVER   lmn ]
```

where

ef may be the file identification (fid) of a ROM, a library load module, a DEFCOM-built load module, a SYSGEN-built load module, or simply a dollar sign ($). If ef is omitted or a dollar sign is specified, the last ROM formed as the result of an assembly or compilation during the current on-line session is used.

lmn (load module name) specifies where the load module is to be placed and may be a file identification (fid) or dollar sign. If lmn is omitted, the resulting load module is placed in a special file and is available for subsequent execution (see Initiating Execution). TEL commands (including their abbreviated forms) cannot be used as load module names.

Example:

Assume that a load module, F, is to be created for execution from files A, B, C, and D.

```
!LYNX  A,B,C,D ON F (RET)
!
```

The complete format of the LYNX command is

```
                   [ON       ]
LYNX [ef[,ef . . .][OVER  lmn][options][;[libname]—┐
  └─ [.[libacct][.password]]]. . .
```

where

ef is as defined above.

lmn is as defined above.

libname specifies the name of a library. :SYS is the default library if no library name, account, or password is specified and if the NL option is not specified. If libaccount is specified, but libname is not specified, then the default libname is :LIB.

libacct specifies the account from which the library is to be obtained. If libname is specified but no libacct is specified, the default account is the log-on account.

password specifies the password for the library if one exists.

options specify options which, for example, determine input to LYNX, affect future access to the load module file, affect the location of the program at execution time, concern the building of symbol tables, and determine how overlay segments will be brought into core at execution time. Options may be specified anywhere in the command except between a preposition and its object. For convenience they are shown immediately following the command verb. All options must be specified within parentheses or preceded by a left parenthesis.

The presence of a semicolon as the last character on an input line indicates that the command is to be continued. LYNX will perform another read from the terminal, prompting the user with a greater than (>) character.

A few of the options for the LYNX command are described in the following sections. The complete set of LYNX command options is given in Chapter 8.

SEARCHING LIBRARIES

Unsatisfied external references are resolved by specifying the order and identification (lid) of libraries to be searched after the input modules have been linked. A list of library identifications, separated by semicolons, is appended to the list of modules in the LYNX command and is separated from the module list by a semicolon.

Additionally, one of two system library search options may be specified:

(L) specifies that the system library is to be searched to satisfy external references that have not been satisfied by the program. (This is a default option).

(NL) specifies that a system library search is not required.

Example:

Assume that a load module, F, is to be created from files A, B, C, and D. Two user libraries, G and H, in the user's account are to be searched to satisfy external references. No search of the system library is required.

```
! LYNX A,B,C,D ON F;G;H (RET)
!
```

LOADING PROGRAMS AND INITIATING EXECUTION

ᵀEL has three commands for loading programs and initiating xecution. The command names are lmn, START, and RUN.

lmn Any stored load module may be loaded into core and started by presenting TEL with the name of the load module (lmn) as a command verb. Additional parameters may be given to specify assignments. The format of the command is the same as for FORT4, COBOL, and META commands with a load module name replacing the processor name.

$$\text{lmn}[\text{sp}]\left[\begin{matrix}\text{ON}\\ \text{OVER}\end{matrix}\;[\text{rom}]\;[,\text{list}]\right]$$

where lmn is the load module name and has the following format:

$$\text{name}\left[.[\text{account}]\;[.\text{password}]\right]$$

TEL commands (including their abbreviated forms) cannot be used as load module names.

When lmn is used as a command verb, the default account is interpreted as follows:

name implies the system account (i.e., the :SYS account).

name. implies the log-on account.

name.account specifies an account and no password.

name.account.password specifies an account and password.

name..password implies the log-on account and specifies a password.

sp is the identification (fid or ME) of the input file to be assigned to the M:SI DCB.

rom is the identification (fid, stream-id, CP, or NO) of the output file to be assigned to the M:GO DCB.

list is the identification (fid, stream-id, LP, ME, or NO) of the output file to be assigned to the M:LO DCB.

TEL scans the parameters in an attempt to create assignments as it does for the FORT4, META, and COBOL commands. Parameters enclosed in parentheses are ignored. If the above syntax is not observed, the scan is not performed.

Examples:

```
! TESTOR (RET)
```

(loads the LM using the system account)

```
! TESTOR. (RET)
```

(loads the LM using the log-on account)

```
! TESTOR.1234 (RET)
```

(loads the LM using account 1234)

```
! TESTOR..SECRET (RET)
```

(loads the LM using the log-on account and the password "SECRET")

```
! TESTOR FILEA ON FILEB,FILEC (RET)
```

(loads the LM using the system account – FILEA is assigned to the M:SI DCB, FILEB to the M:GO DCB, and FILEC to the M:LO DCB)

```
! TESTOR (ABC(DEF(GHI)JK)) (RET)
```

(loads the LM using the system account and passes the line image to the program, starting at JIT word location J:CCBUF)

START The START command loads a load module into core and starts execution at its beginning address. The format of the command is

$$\text{S[TART]}\left[\begin{matrix}\text{lmn}\\ \$\end{matrix}\right]$$

where lmn is the name (fid) of the load module to be executed. If lmn is omitted or a dollar sign is specified, the last load module formed by LINK or LYNX is executed.

Example:

```
!START MYPROG..XWX (RET)
```

(loads the load module MYPROG using the log-on account and the password XWX).

SPECIAL COMMAND FILE HANDLING

The lmn. and START lmn commands may be used to initiate the execution of a command file. (This is the same as issuing an XEQ command.) When a lmn. or START lmn command is issued and the specified lmn is either an unkeyed (consecutive) or Edit-keyed (KEYM = 3) file, an XEQ command is simulated and the command file is executed. A starting record number cannot be specified when the lmn. or START lmn command is used.

Note that this feature cannot be used if the lmn. or START lmn command is issued from within a command file. Also note that this feature may be disallowed by the installation manager.

RUN The RUN command is a combination of the LYNX and START commands. It calls the LYNX loader and instructs it to link, load, and start execution of the designated module. The RUN command has the same format as the LYNX command (except that the command verb is RUN rather than LYNX), and may be continued in the same manner as the LYNX command.

Examples:

1. Assume that file A is to be assembled, loaded, and executed.

```
!META A (RET)
!RUN(RET)
!
```

2. Assume that there are three modules to be loaded: A, B, and C. User library D and the system library are to be searched for external references that have not been satisfied by the program. Public library P2 is to be associated with the program.

```
!RUN (L)(P2)A,B,C;D(RET)
!
```

INITIATING DEBUGGING OPERATIONS

Programs can be executed under the control of one of the two debugging systems, Delta or FDP. The means by which Delta and FDP may be invoked are outlined below.

DELTA

1. Delta may be called by appending the words UNDER DELTA to the START command:

```
S[TART]...[U[NDER DELTA]]
```

Note that UNDER DELTA may be abbreviated to U.

2. Delta may be called by using the U command. This command causes the words UNDER DELTA to be inferred in the command that immediately follows. The command that follows must be of the form:

$$\text{Imn}\,[\text{sp}]\left[\begin{matrix}\text{ON}\\ \text{OVER}\end{matrix}\;[\text{rom}]\,[,\text{list}]\right]$$

Example:

```
!U(RET)
!MYPROG(RET)
```

An important feature of the U command is that it provides the only means by which the words UNDER DELTA may be appended to the Imn command.

For both of the above methods of invoking Delta, control passes to Delta (rather than to the program to be executed). Delta sends an identifying message to the terminal and awaits commands.

3. Delta may also be called when execution has been initiated without it. This is usually done after an interruption by the user or an error comment by the system. In this case, Delta is called by typing the TEL command

```
DELTA
```

Control passes to Delta and execution of the program can be continued in a debugging mode.

FDP

1. FDP may be called by appending the words UNDER FDP to the LINK command:

```
LINK...[UNDER FDP]
```

2. FDP may also be initiated by specifying either of the library-search codes P0 or FDP in a RUN, LYNX, or LINK command. For example,

```
RUN (P0)[rom][,rom]...
```

This command associates public library P1 and FDP with the user program, thus allowing execution of the program under FDP.

3. The FORTRAN debug mode may be set by the TEL command

```
DEBUG
```

When this mode is set, the FORTRAN IV compiler acts as if the DEBUG option had been supplied. LINK acts as if the phrase UNDER FDP had been supplied. Load and LYNX select the correct library according to which REFs are used. Although the mode is set as soon as the command is issued (even when interrupting a FORTRAN or LINK job step), it is not active until the next job step. The debug mode remains set until the user issues the TEL command

```
DONT DEBUG
```

MANAGING AND BACKING UP FILES

File management and information-transfer capabilities are provided by the PCL processor (Chapter 5). PCL can be called implicitly, however, at TEL level, via any of the following commands:

COPY

DELETE

L (which is the TEL form of the PCL LIST command).

COPY and DELETE are discussed in this section. The L command is discussed in a later section.

COPY A very simple form of the COPY command is as follows:

$$\text{C[OPY] sf}\begin{Bmatrix}\text{TO}^{\dagger}\\ \text{OVER}\\ \text{INTO}\end{Bmatrix}\text{df}$$

where

sf specifies an input device or a source file on RAD, labeled tape, or disk pack.

df specifies an output device or a destination file on RAD, labeled tape, or disk pack.

The transfer of information to a printer or to the terminal may be aborted by depressing the BREAK key.

The many additional variations of the COPY command, as described in the section on PCL, are also available through TEL.

DELETE Files can be deleted with the TEL DELETE command. This command has the following format:

$$\text{D[ELETE]}\begin{Bmatrix}\text{[DC/]}\\ \text{DP \#serial no.[-rt]/}\end{Bmatrix}\text{fid[, fid]. . .}$$

where

rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource.

fid is the identification of the file to be deleted. Deletions cannot be interrupted after they have been started.

BACKUP Files created or modified during an on-line session may be saved by using the BACKUP command. The format of this command is

```
BACKUP fid
```

where fid (11-character limit) names the file to be copied to the standard system backup tape. Automatic system restart includes restoration of all entries on the backup tape onto the permanent-file disk.

The backup process is mechanized by an asynchronous process that handles backups for all users. Therefore, there may be a delay between the time the backup command is issued and the time that the file is placed on tape. Also, by rules of simultaneous file access, the file may be unavailable to the user during the time it is being copied by BACKUP. The user-requested backup process delivers error messages as well as successful completion messages to a keyed file (called MAILBOX) in the user's account. The user may print his MAILBOX file using the command:

```
C[OPY] MAILBOX [ON ME]
```

SUBMITTING BATCH JOBS

BATCH The user may compose batch jobs (including all necessary batch job control commands) on-line using Edit. The Batch processor can then be used to submit the jobs to the batch queue for execution. Optionally, the specification field of the JOB control command may be left blank and the Batch processor will supply the missing subfields before submitting the job.

The Batch processor is called by the BATCH command. This command and the four other Batch processor commands are described in detail in Chapter 9. A simplified form of the BATCH command is as follows:

```
BATCH fid[,fid]...
```

where fid is the identification of a job file to be submitted for batch processing. This command may be executed in batch mode as well as on-line mode.

The system responds to this command by assigning each batch job a job identification (jid) and sending one of the following messages to the terminal (on-line) or printer (M:LL) (batch) for each batch job submitted.

```
ID = jid SUBMITTED time-date

WAITING: n TO RUN
```

or

```
ID = jid SUBMITTED time-date

RUNNING
```

JOB The status of one or more jobs submitted on-line to the batch queue may be interrogated at any time by typing

```
JOB jid[,jid]...
```

where jid is the jcb identification reported when the job was submitted using the BATCH command. Response is one of the following:

COMPLETE	if the job has been run.
EH?	if the job is indecipherable.
DOESN'T EXIST	if the job never existed.
RUNNING	if the job is currently in execution.
WAITING: n TO RUN	if the job is waiting to run behind n others.
WAITING TO OUTPUT	if the job has run and symbiont output remains to be printed or punched.

[†] Wherever TO is specified, ON may be substituted.

CANCEL A command is provided to cancel previously submitted batch jobs. The command has the form

```
CANCEL jid[.account][,jid[.account]]. . .
```

where jid is the job identification reported when the job was submitted using the BATCH command and account is the account number under which the job was submitted. A user must have at least C0 privilege to cancel a job under a different account number.

If the file is not an input file or has been processed, the following message is issued on the user's console:

```
COMPLETED OR NOT INPUT
```

If the file is input and waiting to be scheduled, it is deleted. If the job is already running, it is aborted. In both cases, a message is sent to the operator's console.

If the specified job was submitted under some other account, the following message is output on the user's console:

```
NOT YOUR FILE
```

CALLING PROCESSORS

Most processors are called by typing the processor identification. The processors respond by identifying themselves and then typing their prompt character at the left margin of the next line before returning control to the user. The processor identification and prompt character for each user interactive processor that may be called at the TEL level are listed below.

```
ANSF

APL (no prompt)

FORT4 >

COBOL $

META >

BASIC >

FLAG >

EDIT *

DELTA bell

PCL <
```

Example:

Assume that the PCL processor is to be called.

```
!PCL RET

PCL D00 HERE

<
```

INTERRUPTING, RESUMING, AND TERMINATING EXECUTION

There are several courses of action that may be taken whenever a major operation, a processor operation, or an executing user program has been stopped or interrupted and control has been returned to TEL.

1. Any of the following commands may be given:

```
BACKUP               PASSWORD
CANCEL               PLATEN
DELTA                PRINT
DISPLAY              PROMPT
[DONT] COMMENT       SAVE
[DONT] LIST          STATUS
[DONT] OUTPUT        TABS
JOB                  TERMINAL
MESSAGE              TERMINAL STATUS
PAGE
```

   The interrupted operation may then be resumed by one of the following commands:

```
CONTINUE
GO
PROCEED
```

2. The interrupted operation may be given up completely by entering one of the following:

```
Q[UIT]
END
STOP
```

   In this case, TEL restores certain specifications before returning control to the user (see "Error Handling and End Actions").

3. The interrupted operation may be given up completely by entering a request for a load module followed by a NEW LINE character (rather than a RETURN character). The NEW LINE character is effectively an implicit QUIT command. The requested load module is then executed.

4. The interrupted operation may be given up completely by entering any of the following commands:

```
APL      BYE      EDIT     OFF
BASIC    COBOL    FORT4    PCL
BATCH    COPY     LINK     RUN
BUILD    DELETE   META     START
```

   In this case, the effect is the same as if a QUIT, END, or STOP command had been given. In addition, operation of the specified command begins.

5. One of the following commands may be given:

   COMMENT $\begin{Bmatrix}\text{ON}\\\text{OVER}\end{Bmatrix}$ list

   GET

   LIST $\begin{Bmatrix}\text{ON}\\\text{OVER}\end{Bmatrix}$ list

   SET dcb [options]

```
           {ON  }
OUTPUT     {OVER}  fid
RESET
```

TEL will respond with the message

```
COMMAND LEGAL ONLY AT JOB STEP-QUIT
OR GO
```

To resume the interrupted operation, the user responds with a GO, CONTINUE, or PROCEED command. To terminate the current job step, the user responds with a Q[UIT], END, or STOP command. When the job step is terminated, the rejected command may be re-entered.

6. The user may request the execution of a new load module by entering:

   ```
   lmn. [account] [.password]
   ```

   TEL will respond with the message

   ```
   QUIT?
   ```

   If the user responds to the QUIT? message with a RETURN or NEW LINE character by itself, the interrupted operation will be aborted and the command just issued will take effect. (The user may respond with a QUIT, END, or STEP command and the interrupted operation will be aborted, but the user will have to reenter the desired command.) If the user responds, with a GO, CONTINUE, or PROCEED command, the interrupted operation will be resumed.

MINOR OPERATIONS

Minor operations consist of the operations that support on-line programming. They include checkpointing, assigning I/O devices and DCB parameters, determining current user status, and so on.

LOGGING OFF

{OFF / BYE} The user terminates an on-line session by typing either of the following commands:

```
OFF
BYE
```

The user is logged off and a summary of accounting information for the session is printed.

CHANGING THE LOG-ON PASSWORD

PASSWORD A password is an optional part of the information by which a user identifies himself when logging onto the system. The purpose of a log-on password is to allow a user to protect his resources and files by preventing unauthorized use of his name and account. The PASSWORD command allows the user to assign, modify or delete his password. The format of the command is

```
PASSWORD, old-password, new-password
```

where old-password is the current password associated with the user (or null if none currently exists) and new-password is the password to be associated with the user's name and account number. Passwords must be one to eight characters in length and may not contain any of the following characters:

```
I , ; < > . / = ?
```

or any of the COC control characters (see Chapter 2).

To assign a password when none already exists, the command format is

```
PASSWORD,,new-password
```

To cancel the current password, the command format is

```
PASSWORD,old-password
```

An associated password must thereafter be used when logging on until it is changed or cancelled.

It is important for the user to remember his password because only the system manager can establish a new password when the old one is forgotten.

CHECKPOINTING ON-LINE SESSIONS

SAVE During interruptions of execution, core images of programs may be saved (checkpointed) on disk storage for subsequent recall and continuation. The SAVE command is used for this purpose. The format of this command is

```
    {ON  }
SAVE{OVER}  fid
```

where fid is the identification of the file in which the image should be saved.

{GET / RESTORE} A checkpointed core image may be recalled for continuation by either the GET or RESTORE commands. The formats of these commands are:

```
GET fid
RESTORE fid
```

where fid is the identification of the file to be recalled. This file must be in the user's log-on account.

CONTINUE A program that has been SAVEd and RESTOREd (with RESTORE or GET) may be restarted with a CONTINUE command which has the format

```
CONTINUE
```

SAVE is implemented in such a way that execution of a program is unaffected by a SAVE-CONTINUE sequence of operations except for the time delay. Especially important is the fact that open files are not closed or repositioned.

A GET or RESTORE operation, however, requires that any current execution be terminated and all files using default close options be closed. This means that current position information is lost for IN and INOUT files (they are effectively rewound) and OUT and OUTIN files are released. Thus, whenever a GET or RESTORE command is issued, the user must take responsibility for repositioning of IN files and re-creation of OUT files that were open at the time of the save in whatever way is appropriate to continuation of his program.

The collection of I/O assignments made during the job (up to the point of SAVE) and collected in the user's assign-merge table is not preserved, but the active DCBs are. The effect of the current assign-merge activity is therefore carried over to the GET or RESTORE operation through DCBs. The assign-merge table current at GET or RESTORE time has no effect on the retrieved DCBs.

Symbiont output that has been produced, say for printer or punch, is packaged for delivery to the appropriate device whenever a SAVE command is given.

SAVE remembers the identifying numbers of any shared processors associated with the program that are to be saved. The processors having these same numbers are reassociated by the GET or RESTORE command. If the shared processor has changed in the elapsed time between the SAVE and the GET (or RESTORE), proper continuation may not be achieved.

Programs that have been SAVEd can only be fetched and run under the same version of CP-V. SAVE records the release version of the monitor at the time of the SAVE. GET (or RESTORE) checks the release version of the fetched program against that of the current running monitor and aborts the GET operation if the system versions are different.

ASSIGNING I/O DEVICES AND DCB PARAMETERS

{SET
RESET} DCB assignments to files or devices and many other DCB parameters may be set from an on-line terminal. This includes most of the parameters that are set by a batch ASSIGN command and many of the parameters that are set by OPEN and DEVICE procedure calls in a batch program. The on-line command that sets these assignments and parameters is the SET command.

The system retains all information supplied by SET commands in a permanent table associated with each user. This table is called the assign/merge table and is stored on disk. At each job step (i.e., each time a new user program or processor is loaded), the information in the assign/merge table is merged into the DCBs associated with the program. An entry for a DCB that is currently in the assign/merge table may be deleted by the command

```
SET dcb [0]
```

This allows the default assignment (if any) for the specified DCB to take effect. The command

```
R[ESET]
```

resets all DCBs back to their system defaults. Any temporary changes to the listing control flags (i.e., LIST, DONT LIST, etc .) that were made during the previous job step become permanent changes.

DCB assignments are either to a device or to a file. If a DCB that has already been assigned to a device is assigned to a file, the new information replaces the old information in the assign/merge table. The same procedure applies to device assignments for DCBs currently assigned to files. Each DCB assignment requires an entry in the assign/merge table. The total number of DCBs that may be assigned is limited to 12.

Changes to device parameters are added to DCBs assigned to devices. Changes to device parameters for DCBs assigned to files yield an error message.

SET commands may be issued only between job steps, i.e., not during interruptions thereof. Once issued, the information specified by the command for all but the M:SI DCB remains in effect until revoked, regardless of whether one or many job steps are included in the session.

The several formats of the SET command are:

```
SET dcb [0]

        ┌oplabel             ┐
SET dcb │device              │[;dopt]...
        │stream-id           │
        └tapecode[tapeid]    ┘

        ┌┌tapecode[tapeid][-rt]┐     ┐
SET dcb │└filecode[-rt]        ┘ /fid┘[;fopt]...

SET dcb JR/fid
```

where

dcb identifies a DCB and is in the form M:x or F:x where x is 1 to 31 characters. (Assignments of M:UC, M:OC, and M:XX are not allowed.)

oplabel specifies an operational label (BI, C, CI, etc.). (See Tables 4 and 5.)

device specifies a device code (CP, PL, LP, etc.). (See Table 4.)

stream-id specifies the name of a logical device stream (C1, L1, P1, etc.). (See Table 4.)

tapecode specifies a magnetic tape code (LT, AT, or FT). (See Table 4.)

filecode specifies a secondary storage code (DP). (See Table 4.)

tapeid if followed by /fid, specifies a serial number for a labeled tape and has the form #serial number. The tape is accessed with the serial number applying as both an INSN and an OUTSN. (Serial numbers may contain alphanumeric characters. Xerox labeled tape serial numbers are 1-4 characters in length. ANS labeled tape serial numbers must be six characters in length.) If not followed by /fid, it specifies an external reel number for free-form tape.

JR specifies a common journal. (Refer to the CP-V/TP Reference Manual, 90 31 12.).

rt specifies the 2-character identifier of a mountable device that was defined at SYSGEN to be a resource (e.g., 7T, 9T, SP, etc.).

/fid specifies the name of a file on tape or secondary storage. A maximum of 11 characters is allowed. The form is

$$\text{name}\begin{bmatrix}\text{.account}\\ \text{.account.password}\\ \text{..password}\end{bmatrix}$$

If not preceded by a tapecode or filecode, /fid implies public disk storage by default.

dopt specifies a device option. (See Table 6.)

fopt specifies a file option. (See Table 7.)

Spaces may be arbitrarily used in a SET command between numbers, words, and identifiers but may not be embedded within them.

Example:

1. Assume that the monitor DCB for listing output is to be assigned to disk storage file N under account A with password P.

```
!SET M:LO/N.A.P RET
!
```

Table 4. DCB Assignment Codes – SET Command

Type	Codes	Description
Operational Label	BI, BO, C, CI, CO, DO, EI, EO, LL, LO, OC, PO, SI, SL, SO, UC (see Table 4) (and others defined at SYSGEN) NO	When the DCB is assigned to one of the system operational labels, the actual device connected to the DCB is that implied by the operational label, if any, for on-line mode. No assignment, i.e., no default is to be applied.
Device	CP LP PL (and others defined at SYSGEN)	Card punch. Line printer. Plotter.
Logical Device Stream	L1 C1 P1 (and others defined at SYSGEN)	Line printer. Card reader. Card punch.
Magnetic Tape	LT AT FT	Xerox labeled tape. ANS labeled tape. Free form tape.
Secondary	DP	Disk pack storage. This requests the default disk device type defined at SYSGEN if the rt field is not specified.

Table 5. Operational Label Conventions

Label	Reference	Comments	Typical On-Line Device Assignment[†]
BI	Binary input	Binary coded input will be received from the device to which this label is assigned.	NO
BO	Binary output	Binary coded output will be transmitted to the device to which this label is assigned.	NO
C	Control input	Input from the device to which this label is assigned will be monitored, so that all control commands will be recognized by the monitor.	ME
CI	Compressed input	Compressed symbolic input will be received from the device to which this label is assigned.	NO

Table 5. Operational Label Conventions (cont.)

Label	Reference	Comments	Typical On-Line Device Assignment[t]
CO	Compressed output	Compressed symbolic output will be transmitted to the device to which this label is assigned.	NO
DO	Diagnostic output	Diagnostic program dumps will be output on the device to which this label is assigned.	ME
EI	Element input	Element file input will be received from the device to which this label is assigned.	ME
EO	Element output	Element file output will be transmitted to the device to which this label is assigned.	NO
LL	Listing log	All control commands and system messages, including accounting information for the job, will be output on the device to which this label is assigned.	ME
LO	Listing output	Source and object listings for assemblies and compilations will be output on the device to which this label is assigned.	ME
OC	Operator's console	All JOB, MESSAGE, and FIN control commands, and all job termination messages will be output on the device to which this label is assigned. OC may not be assigned to another operational label, but may be assigned to another physical device.	ME
PO	Punch output	BCD or binary coded output will be transmitted to the device to which this label is assigned (normally a card punch).	NO
SI	Source input	Symbolic (source language) input will be received from the device to which this label is assigned.	ME
SL	Source listing	A listing of symbolic (source language) input will be transmitted to the device to which this label is assigned.	ME
SO	Source output	Symbolic (source language) output will be transmitted to the device to which this label is assigned.	NO
UC	User's console	This is for on-line use. The batch mode defaults to OC (operator's console).	ME

[t]These device assignments are standard in CP-V but may be changed at SYSGEN.

Table 6. Device Options – SET Command

Format	Description
ASC[II] EBC[DIC]	ASC[II] specifies code conversion (between ASCII on tape and EBCDIC in core). EBC[DIC] specifies no code conversion. EBCDIC is assumed by default and ASCII is legal only for tapes having this feature.
BCD, BIN	Controls the binary-BCD mode for device read and write operations. BIN used in conjunction with DRC will invoke the transparent mode. (See Transparent Mode section of Chapter 11.)
COUNT = value	Turns on page counting and specifies the column number at which the page number is to be printed.

Table 6. Device Options – SET Command (cont.)

Format	Description
DATA = value	Controls the beginning column for printing or punching and is a decimal value. The maximum value is 144.
DEN=$\begin{Bmatrix}800\\1600\end{Bmatrix}$	Specifies the density that will be used on a dual density tape device.
DRC, NODRC	Turns the special formatting of records on and off. DRC specifies that the monitor is not to do special formatting of records on read or write operations. NODRC specifies the monitor is to do special formatting. If neither DRC nor NODRC is specified, NODRC is assumed by default. DRC used in conjunction with BIN will invoke the transparent mode. (See Transparent Mode section of Chapter 11.)
FBCD, NOFBCD	Controls the automatic conversion between external Hollerith code and internal EBCDIC code (FORTRAN BCD conversion). NOFBCD is assumed by default.
IN	Specifies the input mode.
OUT	Specifies the output mode.
INOUT	Specifies the input and output mode (i.e., the update mode).
OUTIN	Specifies the output and input mode (i.e., the scratch mode).
L, NOL	Identifies the device type. L specifies that the device must be listing type. NOL specifies that it need not be listing type. NOL is assumed by default.
LINES = value	Specifies the number of printable lines per page and is a single decimal value. The maximum value is 255.
PACK, UNPACK	Controls the packed or unpacked mode of writing 7-track tape. PACK is assumed by default.
RECL = value	Specifies the default record length, in bytes. The greatest value that may be specified is 32,767. IF RECL is not specified, a standard value (appropriate to the type of device used) will apply. The value specified in a SET command will override that assembled into the DCB but will not override the RECL specification of an M:OPEN call or the SIZE specification of an M:READ or M:WRITE procedure call.
SEQ[= value]	Specifies that sequence numbers are to be punched in columns 77-80 of punched output. Four characters of nonblank sequence identification may be given for columns 73-76. Fewer than 4 characters are left-justified and blank filled.
SN[= value [,value] [,value]]	Specifies the serial numbers of volumes that are to be used for input or output. The serial number may be from 1 to 4 characters except for ANS labeled tape serial numbers which must be 6 characters. A maximum of 3 serial numbers may be specified. If a serial number is specified with the tapeid, it is included in the 3 allowed. An existing list of serial numbers may be removed by specifying the SN option with no arguments.
SPACE = value	Specifies the number of lines of space after printing and is a single decimal value. Values of 0 or 1 result in single spacing. The maximum value is 255.
TAB = tab[, tab]...	Specifies simulated tab stops and is followed by a list of up to 16 decimal numbers, separated by commas, giving the column position of the stops. If all 16 stops are not specified, the stops given are assigned to the first stops and the remainder are reset.
TRIES = value	Specifies the maximum number of recovery tries to be performed for any I/O operation. The greatest value that may be specified is 255. The default value is 10.
VFC, NOVFC	Controls the formatting of printing by using the first character of each record. VFC specifies that the first character of each record is a format-control character. NOVFC specifies that records do not contain a format-control character. NOVFC is assumed by default.

Table 7. File Options – SET Command

Type	Format	Disk	Xerox Tape	ANS Tape	Description
Organization	CONSEC KEYED RANDOM	X X X	X X		Consecutive record organization. Keyed record organization. Contiguous relative-sector addressed organization.
Access	SEQUEN DIRECT	X X	X X		Records will be accessed sequentially. Records will be accessed by key.
Function	$\mathrm{IN}\left[,\begin{Bmatrix}\mathrm{SHARE}\\ \mathrm{EXCL}\end{Bmatrix}\right]$	X	X	X	File is read only. SHARE specifies the share mode for the DCB which allows more than one IN user of the file. EXCL specifies the exclusive mode for the DCB which prohibits more than one IN user of the file. EXCL is assumed by default.
	OUT	X	X	X	File is write only.
	$\mathrm{INOUT}\left[,\begin{Bmatrix}\mathrm{SHARE}\\ \mathrm{EXCL}\end{Bmatrix}\right]$	X	X	X	File is to be updated. SHARE specifies the share mode for the DCB which allows more than one INOUT user of the file. EXCL specifies the exclusive mode for the DCB which prohibits more than one INOUT user of the file. EXCL is assumed by default.
	OUTIN	X	X	X	File is scratch.
Record Length	RECL = value	X	X		Specifies the default record length, in bytes. The greatest value that may be specified is 32,767. If RECL is not specified, a standard value (appropriate to the type of device used) will apply. The value specified in a SET command will override that assembled into the DCB but will not override the RECL specification of an M:OPEN call or the SIZE specification of an M:READ or M:WRITE procedure call.
	$\begin{Bmatrix}\mathrm{LRECL}\\ \mathrm{REC}\end{Bmatrix}$ = value			X	Specifies the logical record size in bytes. The value may be in the range 1 through 32,767.
Block Size	BLK[L] = value			X	Specifies block size in bytes. The value may be in the range 1 through 32,767. If a value less than 18 bytes is specified, 18 bytes are written.
Recovery Tries	TRIES = value	X	X	X	Specifies in decimal the maximum number of recovery tries to be performed for any I/O operation. The greatest value that may be specified is 255. The default value is 10.
Disposition	REL	X			OUT or OUTIN file is to be released on closing.
	SAVE	X			OUT or OUTIN file is to be saved on closing.
	JOB	X			Temporary file saved across job steps but released when the job ends.

Table 7. File Options – SET Command (cont.)

Type	Format	Disk	Xerox Tape	ANS Tape	Description
Size	RSTORE = value	X			Specifies the number of granules allocated to the RANDOM file. The value must be in the range 1 through 16,777,215 $(2^{24} - 1)$.
Storage Control	CYLINDER	X			Specifies that the data blocks of a public file are to be allocated from public disk packs having cylinder allocation.
Key Length	KEYM = value	X	X		Specifies the maximum length, in bytes, of the keys associated with records within the file. A key may consist of up to 31 characters. The default value is 11.
Key Storage	NOSEP	X			Specifies that index blocks of a public file are to be allocated in the same manner as data blocks. (Disk pack if possible; otherwise RAD.)
Additional Key Space	SPARE = value	X			Specifies in bytes the amount of spare space to be left unused at the end of each index block while a keyed file is being created or updated with sequential access. Value may not exceed 255 and the default is 102 bytes.
Expiration	EXP[IRE]= $\left\{\begin{array}{l}\text{mm, dd, yy}\\ \text{ddd}\\ \text{NEVER}\end{array}\right\}$	X		X	Specifies either an explicit expiration date, the number of days to retain the file, or that the file is never to expire.
Index Structure	NEWX = slides[, consecutive slides]	X			The "slides" argument specifies the number of blocks that can be added to the file's index since the current higher-level index structure was built; if the specified value is exceeded, the higher-level index structure will be rebuilt when the file is closed. If a value of 255 is specified, the higher-level index structure will not be built (or rebuilt). If NEWX is not specified, the value 254 is used in default. The "consecutive slides" argument specifies the number of contiguous blocks that can be added to the file's index since the current higher-level index structure was created; if the specified number is exceeded, the higher-level index structure will be rebuilt when the file is closed. If the number is not specified, 2 is used in default.
Execute Accounts	EX[ECUTE] $\left[=\left\{\begin{array}{l}\text{acct[,acct]...}\\ \text{ALL}\\ \text{NONE}\end{array}\right\}\right]$	X			Specifies the account numbers of the accounts that may execute the load module. A maximum of 8 accounts may be specified. The value ALL may be used to specify that any account may execute the file. The value NONE may be used to specify that no other account may execute the file. In all of the above cases, READ, NONE is implied in the absence of any READ specification. This

Table 7. File Options – SET Command (cont.)

Type	Format	Disk	Xerox Tape	ANS Tape	Description
Execute Accounts (cont.)					option with no arguments resets all previous execute account entries in the DCB.
Read Accounts	R[EA]D [= {acct[,acct]... ALL NONE}]	X	X		Specifies the account numbers of those accounts that may read but not write the file. This option is applicable to OUT and OUTIN files. A maximum of 8 read accounts may be specified. The value ALL may be used to specify that any account may read the file. The value NONE may be used to specify that no other account may read the file. This option with no arguments resets all previous read account entries.
Write Accounts	WR[ITE] [= {acct[,acct]... ALL NONE}]	X	X		Specifies the account numbers of those accounts that may have both read and write access to the file. This option is applicable to OUT and OUTIN files. A maximum of 8 write accounts may be specified. The value ALL may be used to specify that any account may have write access to the file. The value NONE may be used to specify that no other account may have write access to the file. This option with no arguments resets all previous write account entries.
Volume Serial Number	SN[= value[,value] [,value]]	X	X	X	Specifies the seria! number of volumes that are to be used for input or output. The serial number may be from 1 to 4 characters, except for ANS labeled tape serial numbers which must be 6 characters. A maximum of 3 serial numbers may be specified. If serial number is specified with tapeid, it is included in the 3 allowed. An existing list of serial numbers may be removed by specifying the SN option with no arguments.
Code Conversion	ASC[II] EBC[DIC]		X	X	ASCII specifies code conversion between ASCII on tape and EBCDIC in core. EBCDIC specifies no code conversion. EBCDIC is the default. ASCII is legal only for tapes having the code conversion feature.
Recording Density	DEN = {800 1600}		X	X	Specifies the density that will be used on the dual density tape device.
Initial Volume	VOL = value		X	X	Specifies which tape reel in the SN list is to be used initially. A value of 1 designates the first, a value of 2 the second, etc. If VOL is omitted, a value of 1 is assumed.

Table 7. File Options – SET Command (cont.)

Type	Format	Disk	Xerox Tape	ANS Tape	Description
Concatenate Tape Files	[CON] CAT = value			X	Specifies the number of identically named files that are to be read as one logical file (concatenated). The value may be in the range of 2 through 255.
Tape Format	{FORMAT FMT} = character			X	Specifies the record format. The character may be: F = fixed length; D = variable specified in decimal; V = variable specified in binary; or U = undefined.
Block Count Errors	ABCERR			X	Specifies that block count errors for ANS labeled tapes are not to result in an unconditional abort.
Execution Vehicle	UN[DER][= name]	X			Specifies the name of the only processor that may access this file if the user does not own the file. The name may be from one to ten characters. The processor may be any shared processor or any load module in the :SYS account. If EXECUTE accounts are specified and UNDER is not specified, the file is presumed to be a load module and UNDER=FETCH is implied by default. FETCH is the name of the monitor routine that places a program into execution.

2. Assume that F:IN1 (a user constructed DCB) is to be assigned to file M on a Xerox labeled tape with the serial number 4003.

```
!SET F:IN1 LT#4003/M(RET)
!
```

3. Assume that the monitor DCB for compressed input (M:CI) is to be assigned to file JJ on an ANS labeled tape with the serial number B12345. Also, the tape was recorded at 1600 bpi on a device known to the system as BT and the BT device was defined at SYSGEN to be a resource.

```
!SET M:CI AT#B12345-BT/JJ(RET)
!
```

4. Assume that tab positions 27, 38, 47, and 75 are to be added to the listing output DCB. In addition, the first character of each record of the listing is to control vertical format and the listing is to be double spaced.

```
!SET M:LO;TAB=27,38,47,75;VFC;SPACE=2(RET)
!
```

If the M:LO DCB is not assigned when the above changes are made, an error message will be sent to the terminal.

5. Assume that DCB F:1 is to be assigned to an output file XXXX which spans private disk volumes A2, A3, and A4. An expiration date of NEVER is to be assigned.

```
!SET F:1 DP#A2/XXXX;OUT;SN=A3,A4;
     RD=F14,F22X;EXPIRE=NEVER(RET)
```

or the equivalent

```
!SET F:1 DP#A2/XXXX;OUT;EXPIRE=NEVER(RET)
!SET F:1;SN=A3,A4;RD=F14,F22X(RET)
```

6. Assume the same case as Example 5 above, but assume that the second read account number (F22X) should have been F22Y. The mistake can be corrected after the initial assignment of F:1 by the following sequence of SET commands:

```
!SET F:1;RD(RET)          (Reset all read accounts.)

!SET F:1;RD=F14,F22Y(RET)    (Correction accom-
     plished by new assignment.)
```

DCB ASSIGNMENT CODES

A device assignment is made whenever a SET command contains an expression with an operational label or device code, or a tapecode/tapeid not followed by a file identification. For each assignment, an assign/merge table entry is made or an existing entry is modified. DCB assignments are specified by the two-letter codes in Table 4.

DEVICE OPTIONS

SET commands specifying device options may be issued only between job steps. The device options take effect on subsequent input or output through the DCB. The options are then in effect from job step to job step until reset.

The device options allowed for the SET commands are listed in Table 6. Options corresponding to the M:DEVICE options PAGE, FORM, SIZE, and HEADER are not provided.

FILE OPTIONS

When a DCB is assigned to a disk file, Xerox labeled tape, or ANS labeled tape, any options that are valid for the batch ASSIGN command are also valid for the on-line SET command. Table 7 contains the list of file options. The options are the same as those allowed in batch with the exception of READ, WRITE and EXECUTE account options.

Each of these options may be used to specify up to 8 account numbers for that individual option. The lists of READ, WRITE, and EXECUTE accounts are cumulative as long as they do not exceed the maximum of 8 entries per option.

Variable length parameters for READ, WRITE, EXECUTE, UNDER, and SN options may be set, extended, reset and respecified as needed (see Example 5).

Alternatively, PCL compatible keywords (as shown in Table 7) may also be used. As an example, FORMAT is a valid ANS labeled tape option in batch DCB assignments and TEL's SET command will honor either the keyword FORMAT or the PCL form of the keyword FMT.

However, the keyword PASSWORD is not recognized by the SET command because the password is obtained from the filename.account.password field.

MODIFYING LOGICAL DEVICE STREAM DEFINITIONS

LDEV A logical device stream is an information stream that may be attached to any symbiont device that the user specifies. (Symbiont devices include devices such as the line printer, card reader, card punch, plotter, and all devices at remote sites that are accessed via remote processing.) At SYSGEN, up to 15 logical device streams may be defined. Each is given a name (e.g., C1, L1, P1), each is assigned to a physical device, and attributes are defined for the physical device. The user may perform I/O through a logical device stream with the default physical device and attributes or he may change the physical device assignment and/or attributes to satisfy the requirements of his job. He makes any necessary changes through use of the LDEV command. The information about the logical device stream is stored in a cooperative context block, providing for centralized information about the physical device even though I/O to that device may arise through more than one DCB within a job.

The format of the LDEV command is

```
LDEV stream-id [,(option)]. . .
```

where

stream-id specifies the name of one of the logical device streams defined during SYSGEN (e.g., C1, L1, P1).

options are as defined in Table 8. The options may appear in any order.

The LDEV command may be continued by ending the line with a semicolon. LDEV will prompt for succeeding lines with a dollar sign ($) character.

Table 8. LDEV Command Options

Option	Description
AINIT	Specifies that the attributes for the stream are to be initialized with the attributes specified on this LDEV command and that system defaults are to be supplied wherever an attribute is not specified. Any attributes specified for the stream on a previous LDEV command are to be ignored. AINIT is the default for the AINIT, ASAVE, and AREL options.
AREL	Specifies that the system table containing the attributes of this stream (which may have been set as the result of previous LDEV commands) is to be released and that the attributes are not to be reinitialized. Any other options specified (except DELETE) in this command will be ignored.
ASAVE	Specifies that the attributes for the stream are to be set only by options explicitly specified on this LDEV command. Other LDEV-specifiable attributes (which may have set as the result of previous LDEV commands) are not to be changed. ASAVE cannot be used for the LABEL option. Restrictions for using ASAVE with the DEV and WSN options are discussed in the Remote Processing Reference Manual, 90 30 26.

Table 8. LDEV Command Options (cont.)

Option	Description
CONCURR	Places the symbiont output stream in concurrent output mode, a mode in which output is broken into groups ("chunks") and released to the symbiont stream for output. Once this stream has been selected by the symbiont for printing or punching, the particular device is held until all output produced by the job has been processed, except as otherwise directed by an operator key-in. If CONCURR is not the only option specified, then already prepared output will be packaged for printing in its entirety and a newly bannered stream will be created for subsequent output. The COPIES option may not be specified when CONCURR is specified.
COPIES, value	Specifies the number of times the file is to be processed to produce multiple copies. The range of values that may be specified is 1-255. The default value is 1.
COUNT, tab	Specifies that page counting is to be done and specifies the column in which the most significant digit of the page count is to be listed. The value of "tab" must be appropriate for the particular physical device. (Note that ifCOUNTis specified for the LOdevice and a TITLE control command is also specified, the page count will be superimposed on the title line.) The default is no page counting.
DELETE	Specifies that if output currently exists for this stream but has not yet been dispatched for processing, it is to be deleted. (If such a stream exists and DELETE is not specified, the output for the stream is dispatched for processing.) If an input stream with the same name currently exists, any part of the stream that has not been read will automatically be deleted whether or not DELETE is specified.
DEV, type	Specifies the device type, where type is the two-character mnemonic of the device to be associated with the stream. Valid mnemonics are either type mnemonics of the central site or of a remote workstation. Central site mnemonics are those defined for symbiont devices during SYSGEN (e.g., CR, LP). Remote mnemonics are those specified when defining a workstation with Super (e.g., OC, LP).
DRC	Requests that monitor logical record formatting implied by the DEV option not be performed. Any record formatting necessary will be supplied by the user. If DRC is not specified, the monitor will perform logical record formatting.
FFORM, name	Specifies the future form name (as below, with FORM) of the form to be used when the form change procedure (M:DEVICE (FORM/FNAME)) is specified in the program for the stream. When M:DEVICE (FORM/FNAME) is encountered, the stream will be dispatched for processing and restarted with name as the stream form. The default is none.
FORM, name	Specifies the one- to four-character name of an installation-determined paper form or card stock and is used in output scheduling for the device. The default is to have no special scheduling (i.e., the operator will determine which form to use). If used on input, name specifies the one- to four-character name of a noncontrol input file. (FORM and NAME may be used interchangeably.)
LABEL, text	Specifies text that is to be added to the user-identification banner for the stream. A line printer stream has a one-page banner; a card punch stream has a one-card banner. Each line or card contains the user's log-on identification name and account and the date and time; the text specified by the user will be appended to each line immediately following the time. The text may not include a semicolon, period, right parenthesis, or a COC control character. Up to 255 characters of text may be specified; however, the maximum length of a line on the specified stream device will limit the number of characters of text actually used.
LINES, value	Specifies the number of printable lines per logical page. A maximum of 255 lines per page may be specified. The default is determined at SYSGEN.

Table 8. LDEV Command Options (cont.)

Option	Description
NAME,name	Specifies the one- to four-character name of an installation-determined paper form or card stock and is used in output scheduling for the device. The default is to have no special scheduling (i.e., the operator will determine which form to use). If used on input, name specifies the one- to four-character name of a noncontrol input file. (FORM and NAME may be used interchangeably.)
NOBANNER	Specifies that no user-identification banner is to be associated with output for this stream. A FORM name must also be specified for NOBANNER to be operative.
NOVFC	See VFC, below.
SEQ[,id]	Specifies that punched output is to have decimal sequencing in columns 77-80. If a user-defined id is specified, it will be punched in columns 73-76 of each card. Sequencing begins with 0000.
SPACE, value[,top]	Specifies the spacing between lines (value) and between the top of each page and the first line printed (top). A value of 1 indicates that lines are to be single spaced. The greatest value that may be specified (for 'value' and 'top') is 15.
SRCB	Specifies that the user will supply a device-dependent control byte as the first byte of each record if this is an output stream, or that the user wishes to receive it as the first byte of records if the stream is input. This is only applicable to remote processing.
VFC, NOVFC	Specifies whether or not vertical format control characters are to be used. (These two options are only legal for line printers.) VFC requests that a default vertical format control character be added to all records. NOVFC requests that the format character be stripped from the record if present. The default is VFC.
WSN, $\left\{\begin{array}{l}\text{name}\\ \$\end{array}\right\}$	Specifies the workstation name of the remote device that is to receive the stream, where name can be from one to eight alphanumeric characters. The default is local output. If a dollar sign ($) is specified, the name of the workstation on the JOB command (if one is specified) effectively replaces the dollar sign. If no workstation name was specified on the JOB command or if no JOB command was used, the name of the workstation from which the job was submitted effectively replaces the dollar sign. (The dollar sign option allows a job to be run from more than one workstation without necessitating respecification of the workstation name on the LDEV command.)

Examples:

1. The following command requests association of L1 with the local line printer and specifies that the number of printable lines per page is to be 60. All other attributes are to be supplied by default.

```
!LDEV L1,(DEV,LP),(LINES,60) (RET)
```

2. The following command requests association of L5 with the line printer at remote workstation LAX. All other attributes are to be supplied by default.

```
!LDEV L5,(WSN,LAX),(DEV,LP) (RET)
```

NONCONTROL INPUT FILES

There are two types of symbiont input: that which is a job control stream and that which is not. Card readers are usually defined to be control-type devices and are used to input job control streams. However, noncontrol input streams may be entered from the card reader if the first card of the input deck is

```
!INCTL    [name]
```

where name specifies the one- to four-character name of the noncontrol input file.

In this case, the input deck is read until a !FIN is encountered. If any job control cards exist in the deck, they are treated as noncontrol information. That is, the entire deck is simply read into the input symbiont. This feature provides, among other things, a means of inputting a job that is to be run at a later time and a means of allowing on-line users to make use of the card reader.

A file created in this manner must be accessed via the LDEV command or M:LDEV procedure using any logical device stream except C1. If the user gives the file a name or requests the operator to do so the user can access the file using the NAME,name or FORM,name option. (The operator gives the file a name using the key-in Syydd, F'name' where the name must be identical to the name on the FORM or NAME option.) If the file is not given a name by the operator, the next noncontrol file in the queue that has no name will be returned to the user.

REQUESTING EXTENDED MEMORY MODE

EXTEND The space available for an on-line user program is by default virtual address X'A000' through X'1BFFF'. The user may request the extended memory mode by entering the following command

```
EXTEND
```

The EXTEND command frees the virtual core area normally reserved for special shared processors (i.e., TEL, LINK, Delta, and core libraries), making it available to the on-line user program. The additional area is at virtual address X'1C000' through X'1FFFF'.

The EXTEND command is valid at a job step only and remains in effect for the following operation only. Once the extended memory mode is in effect, the current operation may not be interrupted and then resumed by the CONTINUE, GO, or PROCEED command.

The EXTEND command is invalid if the load module to be extended has been built by Link, requires the association of a core library, or is to be executed under Delta.

DETERMINING ON-LINE USER STATUS

STATUS The current accounting records applying to an on-line session can be displayed by entering the following command into a terminal:

```
ST[ATUS]
```

Output is similar to that produced at log-off time and includes:

1. CPU time in minutes and ten-thousandths of a minute.
2. Console time in hours, minutes, and seconds.
3. Number of interactions.
4. Total charge units.
5. Number of CALs executed on behalf of the user.

The format of output is

```
CPU=M.MMMM CON=hh:mm:ss INT=nn CHG=xxxxCALS=xxxx
```

DISPLAYING USER AND DCB INFORMATION

SHOW It is possible, through the SHOW command, to display certain information about the currently logged-on user. Included are the user system service and system resource limits as well as the DCB assignments.

The format of the SHOW command is:

```
SHOW[option[,option]...]
```

The legal options are:

USER displays the log-on account, name, auto-call processor, user id, and COC line, and user accumulated space on both RAD and disk.

PRIV displays the user accumulated space on both RAD and disk, the default and maximum file retention periods, the extended accounting field, and the user privilege, service limits, resource limits, and device and feature authorization for both batch and on-line operation.

PROC displays the current setting of the on-line processor options OUTPUT, COMMENT, LIST, SEND, and DEBUG.

DCBS displays all the user DCB assignments in SET command format.

M:xx or F:xx displays the individual DCB requested in SET command format.

ALL displays all of the information requested by the USER, PRIV, PROC, and DCBS options and is assumed if no options are specified.

If SHOW is not able to access the system defaults, only the user specific values are displayed. In addition, the following message is output:

```
'CAN'T GIVE YOU SYSTEM DEFAULT VALUES
```

SHOW output is through the M:UC DCB.

LISTING A FILE DIRECTORY

L The file directory for RAD, tape, or disk pack may be listed using the PCL LIST command at the TEL level. The command has the following format:

```
  ┌ LT#reel-id[-rt][(s)]                                  ┐
  │ [DC][.acct][(s)]                                      │
  │ LT#serial no.[-rt][(s)]/fid[(s)][,fid[(s)]]. . .      │
L │ fid[(s)][,fid[(s)]]. . .                              │
  │ DP#reel-id[-rt][(s)]                                  │
  │ DP#serial no.[-rt]/fid[(s)][,fid[(s)]]. . .           │
  └ FT#serial no.[-rt][(s)]                               ┘
```

where

s may be A or EA.

rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource.

Note that at the TEL level, the mnemonic must be L; whereas at the PCL level, the mnemonic may be either L or LIST. The various specifications listed above are described in detail in the PCL chapter.

LISTING SYSTEM LOAD PARAMETERS

DISPLAY System load parameters supply information about current system operation, such as the number of users currently active and the current values of interactive and compute response times. The format of the command used to display this information is

```
DI[SPLAY]
```

Output is

```
USERS = xxxx

ETMF = xxxx

RESPONSE 90% < xxxx MSECS

RADS = xxxx GRANULES
```

where

USERS is the number of currently active on-line users.

ETMF is the execution multiplier currently relating program CPU time to job throughput time. ETMF is calculated and updated each minute. It is a moving average covering the preceding minute calculated by summing the time spent computing plus the time spent waiting in high priority ready-to-run queues by all users, and dividing by the sum of time spent computing. Note that since the value is averaged over all users, it is only an approximate measure of how much slower a given process will run due to the time-sharing environment.

RESPONSE gives the number of milliseconds that just exceeds the response time of 90 percent of the responses to terminal requests.

RADS gives the number of unused RAD granules that were available in the user's account at the time he logged on.

TERMINAL COUPLING

This facility provides for coupling (linking) of indirect printing mode terminals (e.g., Teletypes but not IBM 2741s) in such a way that the input and output of one terminal is displayed on both. All typing at both terminal keyboards appears on the paper of both. If the two terminal users are typing concurrently, then mixed (but identical) lines of characters appear on the two terminals. However, a running program of a particular terminal "sees" only the input of that terminal. Conversations may be carried on between linked terminals by terminating lines with cancel (X^c) so as not to affect a reading program. Terminal page heading output is not coupled. The link is broken if either line is disconnected.

This facility includes mechanisms for accepting, rejecting, creating, and terminating couplings. Both terminal commands and program procedures are provided so that the user may control coupling either using TEL commands typed at the terminal or through the operation of the program. The TEL commands are described in this section. The program procedures are described in Chapter 10.

COUPLE This command puts the user's terminal in a mode such that attempts to couple to it will be accepted. When the user logs on, the default is to disallow other user's attempts to couple. The format of the command is

```
COUPLE
```

The message

```
COUPLE FROM line#
```

is printed on the coupled terminals when coupling is successful.

If the object of the couple is refusing couples (see DONT COUPLE below), if the couple is to the wrong type of terminal, or if the line being coupled to is not connected, one of the following messages is typed on the terminal of the user requesting the couple:

```
COUPLE REFUSED

COUPLE REFUSED. LINE NOT ON
```

Whenever a user reaches TEL, the most recently rejected couple attempt (if any) is reported to the user with the message

```
COUPLE REJECTED FROM line#
```

This message indicates that a couple was attempted from another terminal but was rejected. The line number is supplied so that the rejecting terminal may couple back when desired.

DECOUPLE This command may be used by either coupled terminal to release the connection. The format of the command is

```
DECOUPLE
```

DONT COUPLE This command puts the terminal in a mode such that attempts to couple to it will be rejected. Any current coupling is released. The format of the command is

```
DONT COUPLE
```

COUPLE line This command establishes a link between the user's terminal and another terminal. The format of the command is

```
COUPLE line
```

where line specifies the line number of the terminal to which the user's terminal is to be linked. A user may determine his own line number by reference to the sysid-line# pair printed just preceding the page number on each terminal page heading. Line numbers are also printed on the operator's console. The command described below allows the user to determine the line number of another user.

WHERE This command allows the user to determine the line number of a user given the account and name. The format of the command is

```
WHERE  account,name
```

The system responds by typing either the user's line number or the message NOT ON.

INFORM This command permits the typing of the message 'COUPLE REJECTED FROM line #' on a user's terminal when a couple is attempted from another terminal but is rejected. The format of the command is:

```
INFORM
```

DONT INFORM This command prevents the typing of the message 'COUPLE REJECTED FROM line#' on a user's terminal when a couple is attempted from another terminal but is rejected. This is the default. The format of the command is:

```
DONT INFORM
```

SETTING THE DEFAULT PROMPT CHARACTER:

PROMPT A prompt character may be set for use by programs that don't issue an M:PC procedure call. The format of the command is

```
PROMPT  [character]
```

where character is the character to be used for prompting when otherwise no prompt character would be issued. If no character is specified in the command, the default prompt character mechanism is reset.

SETTING SIMULATED TAB STOPS

TABS Simulated tab stops for a terminal are set by the TABS command. The format of this command is

```
TABS  s[,s]...
```

where s is a column position where a tab stop is to be placed.

Up to 16 tabs, in ascending sequence, may be set. Whenever a tab character is sent to or received from a terminal, spaces are sent to the terminal to position the carrier to the next stop that is higher than the current position (if tab simulation is in effect). The setting applies until superseded by another TABS command or by an M:DEVICE procedure call in a program. (The tabs are set in the M:UC DCB.)

Any tab position may be respecified by reentering the entire list of tabs up to and including the new tab value. All of the previously set tab values which are higher than the new tab value will not be altered.

Example:

```
!TABS 5,9,15,20 (RET)    (the tabs are set initially)

!TABS 5,10 (RET)         (the value 9 is changed to 10
                         in the list of tabs)
```

The result of the above example is that the tab values 5, 10, 15, and 20 are in effect after the second TABS command.

Tab settings may be temporarily disabled by using the command

```
TABS  0
```

At a later time, the tab settings can be put into effect again simply by typing the command and specifying the first tab setting only.

Example:

```
!TABS  10,20,30,40 (RET)   (the tabs are set initially)
:
!TABS  0 (RET)             (the tab settings are disabled)
:
!TABS  10 (RET)            (the tab settings are in effect
                           again)
```

After the third TABS command in the above example, the tab settings 10, 20, 30, and 40 are in effect again.

DISPLAYING SIMULATED TAB STOP SETTINGS

TABS Current tab settings may be displayed by entering the TABS command in the following format:

```
TABS
```

Example:

```
!TABS 12,108 (RET)
!TABS (RET)
 12,108
```

OBTAINING TERMINAL STATUS

TERMINAL STATUS The parameters that reflect the operational status of the terminal may be listed by the following command:

```
T[ERMINAL] STAT[US]
```

The information that is output is

line

line speed

timing algorithm

$\left\{\begin{matrix}\text{ACCEPT}\\\text{REJECT}\end{matrix}\right\}$ COUPLES

TRANSLATION TYPE type

ECHOPLEX state

TAB SIMULATION state

UPPER CASE RESTRICT state

PAPER TAPE state

SPACE INSERTION state

LOWER CASE SHIFT state

PARITY CHECK state

RELATIVE TABBING state

BACKSPACE EDIT state

where

type is the type of terminal and may be one of the following:
33
35
37
7015
EAPL
ESTD
SAPL
SSTD
C360
APL
VP72

or the name given to an installation-supplied translation table

state is either ON or OFF.

CHANGING TERMINAL TYPE

TERMINAL Whenever the type of terminal used with the system is changed from the type specified at SYSGEN time, the system must be informed. The system uses this information to adjust character tables and in responses to line-delete and character-delete options. The format of the command used to identify the terminal type is

T[ERMINAL] type [,algorithm]

where

type specifies the terminal type and may be any type specified in Table 9 or an installation-supplied type name.

algorithm specifies the timing algorithm number for the terminal. The value may be in the range 0-6. The seven timing algorithms that are supplied with CP-V are listed in Table 10. The timing algorithm determines how many (if any) special timing characters ("idles") are to be sent to the terminal and when they are to be sent. These characters provide a delay to give the terminal's carriage time to get to the correct position after tab and new-line characters are sent. 2741-type terminals are sent IDLE (X'16') characters and the other types of terminals are sent RUBOUT (X'FF') characters. If algorithm is not specified in the command, the timing algorithm is set to the default listed in Table 9 or is not changed.

Table 9. Types of Terminals

Type	Default Timing Algorithm	Translation
33	5	Teletype Model 33
35	5	Teletype Model 35
37	5	Teletype Model 37
7015	5	Xerox 7015 Keyboard Printer
APL	no change	ASCII APL
C360	no change	IBM2741 CALL/360
DATA POINT]	0	Teletype Model 33
EAPL	1	IBM 2741 EBCD APL
ESTD	1	IBM 2741 EBCD Standard
EXEC[UPORT]	2	Teletype Model 33
MEMO[REX]	3	Teletype Model 37
SAPL	1	IBM 2741 Selectric APL
SSTD	1	IBM 2741 Selectric Standard
TI	5	Teletype Model 37 (TI is Texas Instrument's Model 725)
VP72	no change	Honeywell VIP 7200/7205

Table 10. Timing Algorithms

<table>
<tr><th>Timing Algorithm Number</th><th>Usage</th><th>Idles[†]</th></tr>
<tr><td>0</td><td>Teletype Models 33, 35, and 37 and alphanumeric displays.</td><td>None</td></tr>
<tr><td>1</td><td>IBM 2741 and 2741-compatible equipment.</td><td>Before carriage return none
After carriage return (curpos[††]+15)/10
After tc b character (new position-old position+15)/10</td></tr>
<tr><td>2</td><td>Execuport, Dataport, and TI Model 33 terminals.</td><td>
<table>
<tr><th></th><th>0-10 cps[†††]</th><th>11-15 cps</th><th>16-30 cps</th><th>31-60 cps</th><th>61- cps</th></tr>
<tr><td>Before carriage return</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>After carriage return</td><td>1</td><td>4</td><td>8</td><td>12</td><td>16</td></tr>
<tr><td>After tab character</td><td>1</td><td>1</td><td>2</td><td>4</td><td>8</td></tr>
</table>
</td></tr>
<tr><td>3</td><td>Memorex terminals</td><td>
<table>
<tr><th></th><th>0-10 cps[†††]</th><th>11-15 cps</th><th>16-30 cps</th><th>31-60 cps</th><th>61- cps</th></tr>
<tr><td>Before carriage return</td><td>7-curpos[††]</td><td>10-curpos</td><td>21-curpos</td><td>40-curpos</td><td>40-curpos</td></tr>
<tr><td>After carriage return</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>After tab character</td><td>1</td><td>1</td><td>2</td><td>4</td><td>8</td></tr>
</table>
</td></tr>
<tr><td>4</td><td>This algorithm is a combination of the others and may be used to ensure than in inexperienced user can utilize the system without any character loss. It also supports an experienced user until a change in terminal type can be entered. It is suggested that installations with mixed types of high-speed terminals use this algorithm as the default for high-speed lines.</td><td>
<table>
<tr><th></th><th>0-10 cps[†††]</th><th>11-15 cps</th><th>16-30 cps</th><th>31-60 cps</th><th>61- cps</th></tr>
<tr><td>Before carriage return</td><td>7-curpos[††]</td><td>10-curpos</td><td>21-curpos</td><td>40-curpos</td><td>40-curpos</td></tr>
<tr><td>After carriage return</td><td>1</td><td>4</td><td>8</td><td>12</td><td>16</td></tr>
<tr><td>After tab character</td><td>1</td><td>1</td><td>2</td><td>4</td><td>8</td></tr>
</table>
</td></tr>
<tr><td>5</td><td>This algorithm is used for terminals that require a number of idles roughly proportional to the carriage movement distance. It may be used for Teletypes and other equipment of similar mechanical design, and is sometimes a better algorithm than number 0 for such equipment.</td><td>
<table>
<tr><th></th><th>0-10 cps[†††]</th><th>11-15 cps</th><th>16-30 cps</th><th>31-60 cps</th><th>61- cps</th></tr>
<tr><td>X =</td><td>60</td><td>50</td><td>17</td><td>15</td><td>2</td></tr>
</table>
Before carriage return = none
After carriage return (curpos+15)/X
After tab character (new position-old position+15)/X</td></tr>
<tr><td>6</td><td>Teletype Model 40 hardcopy printer.</td><td>
<table>
<tr><th></th><th>0-10 cps[†††]</th><th>11-15 cps</th><th>16-30 cps</th><th>31-60 cps</th><th>61-120 cps</th><th>121-240 cps</th></tr>
<tr><td>Before carriage return</td><td>3-curpos[††]</td><td>5-curpos</td><td>9-curpos</td><td>17-curpos</td><td>34-curpos</td><td>67-curpos</td></tr>
<tr><td>After carriage return</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>After tab character</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>
</td></tr>
</table>

[†] Many high-speed terminals require a delay before sending a carriage return, after sending a carriage return, or after sending a tab character. In such a case, the COC handler must send "idle" characters, the number of which depends upon line speed, carriage position, and characteristics of the particular terminal.

[††] Current carriage position.

[†††] Characters per second.

CHANGING TERMINAL PLATEN SIZE

PLATEN The PLATEN command changes the maximum number of characters to be written per line on the terminal and the number of lines to be printed between each automatic page heading. The format of the command is

```
PLATEN [w] [,[l] [, [lb] [, la]]]
```

where

w is the maximum number of characters to be written per line on the terminal. If more than w characters are written, a line feed and carriage return character sequence is inserted to break up the output into segments no longer than specified by w. If w is 11 or less, no line feed and carriage return sequence is supplied. If the w field is omitted, the current width setting is unchanged. The maximum value for w is 140.

l is the number of lines per page of terminal output (excluding heading) and must be within the range 0-255. If no l value is given, then the number of lines per page remains unchanged. If l is set to 11 or less, no heading is produced and the page length is unlimited. If l is between 1 and 10, no heading is produced but l many lines will be upspaced for a VFC character of "1".

lb is the number of lines upspaced before the page heading, and must be in the range 0-255. Default value for lb is 5 lines.

la is the number of lines upspaced after the page heading, and must be in the range 0-255. Default value for la is 5 lines.

The default case when a user logs on is equivalent to PLATEN 132,0 if 2741 terminal or to PLATEN 72,0 if a Teletype-compatible terminal. This means that no line feed and carriage return sequence is supplied, that no heading is produced, and that the pagè length is unlimited.

Page width and length may be set so as to provide 8-1/2 by 11-inch pages with 1-inch margins at the top and bottom of each page. For a 10 pitch terminal with six lines per inch, the command would be

```
PLATEN 65,54
```

For a 12 pitch terminal with six lines per inch, the command would be

```
PLATEN 78,54
```

Pagination may also be applied to CRT (display) terminals. The l value should be set to the number of lines that may be viewed minus one. M:STA may be used to set the pagination type to the proper value for the particular type of terminal. (See "CAL Control of Terminal Modes" in Chapter 10.)

Examples:

!PLATEN 72,54 RET	sets line width to 72; lines per page to 54.
!PLATEN ,20 RET	sets printable lines per page to 20; width remains unchanged.
!PLATEN 27 RET	sets width to 27; lines remain unchanged.
!PLATEN ,10 RET	turns off page heading; width remains unchanged.
!PLATEN 2 RET	prints full line width.

DISPLAYING TERMINAL PLATEN SIZE

PLATEN The current platen size may be displayed by entering the PLATEN command in the following format:

```
PLATEN
```

Example:

```
!PLATEN RET
  WIDTH= 012
  LINES= 099
```

CHANGING TERMINAL HEADER PAGE NUMBER

PAGE The PAGE command resets the current page number in the terminal header to the number specified. The header of the next page is then given the number n + 1 and the number is incremented by one for subsequent pages. The format of the command is

```
PAGE n
```

where n specifies the new current page number.

SENDING MESSAGES TO THE OPERATOR

MESSAGE The MESSAGE command causes a message to be sent to the machine operator. The format of the command is

```
M[ESSAGE] text
```

The text may be from 1 to 72 characters, the first of which may not be a period (.). If the text exceeds 72 characters, the first 72 characters are transmitted rather than reinsertion of the message being required. The terminal user is not informed that 72 characters have been exceeded. (If the message text exceeds 44 characters, the message is broken into two lines on the operator's console. The first line contains the first 44 characters of the message and the second contains the remainder.)

CONTROLLING MESSAGES FROM THE OPERATOR

SEND The SEND command allows messages to be sent from the operator to the user terminal at any time. SEND is the default setting. The format of the command is

```
SEND
```

DONT SEND The DONT SEND command disallows messages to be sent from the operator to the user terminal. Global broadcasts will be deferred until TEL is in control (i.e., will not interrupt a user program). DONT SEND also disallows use of the MESSAGE command since a response is impossible. The format of the command is

```
DONT SEND
```

PRINTING OR PUNCHING OUTPUT

PRINT Normally the output destined for symbiont devices such as the line printer and the card punch from all on-line compilations, assemblies, PCL operations, Delta, dumps, etc., is accumulated on RAD or disk pack until the user logs off. When the user logs off, this output is put in the print and punch queues and is printed or punched when it becomes first in the queue. The PRINT command causes output accumulated for symbiont devices to be placed in the queue at once. The format of the command is

```
PRINT
```

ERASING PRINT OUTPUT

ERASE The ERASE command deletes any unwanted printer output for an on-line user. The format of the command is

```
ERASE
```

SETTING PSEUDO SENSE SWITCHES

SWITCH The SWITCH command allows the user to set or reset any of six pseudo switches. (It is the equivalent of the batch !SWITCH control card.) The format of the command is

```
SWITCH [S [ET] =n [,n] ... [;R [ESET] =n [,n] ...]]
```

with n ranging from 1 to 6. With no arguments, the current setting of the user's pseudo-switches are displayed.

MAKING A TERMINAL AVAILABLE FOR TRANSACTION PROCESSING

TP The TP command logs off a time-sharing terminal and makes it available as a slave Transaction Processing terminal. The system must be sysgened for TP and TP must be active for this command to take effect. The format of the command is

```
TP
```

If the line may be given to TP, the message 'LOGGING OFF TO FREE YOUR LINE FOR TP' is output prior to the accounting summary.

COMMAND FILE PROCESSING

XEQ The XEQ command initiates the execution of TEL commands read from a keyed or consecutive file called a command file. It provides a convenient method of executing a frequently used sequence of commands. The format of the command is

```
XEQ     fid[,record number]
```

where

fid specifies the command file identification. The name of the file cannot exceed 11 characters.

record number specifies the record number at which command execution is to start. The records are (logically) numbered consecutively beginning with 1 for purposes of this command. (Even if the record has Edit keys, for example, the keys do not apply when specifying the beginning record number.) If record number is not specified, command execution begins with the first record of the command.

Command file execution may also be initiated by executing the command file as if it were a load module, unless this feature is disabled by the system manager. (See SPECIAL COMMAND FILE HANDLING, page 25.)

As each command in the file is executed, it is printed on the terminal preceded by its record number if command printing isn't suppressed. (See the ECHO and DONT ECHO commands.)

Example of terminal printout:

```
0001 – !SET F:IN/XFILE

0002 – !SET F:OUT LP

0003 – !STATUS

    .
    .
    .
```

Command files executed by XEQ must consist of valid TEL command preceded by a ! character in the first position of the record.

Command file processing will be terminated when any of the following conditions occur:

1. End-of-file on the command file.
2. !QUIT command (or any of its synonymns) either in the command filè or after a CONTROL Y or BREAK.
3. Read error while reading the command file.
4. Syntax error in a command in the command file.
5. Another XEQ command (within the command file). The second XEQ command terminates execution of the current command file and starts processing the new command file.
6. BREAK key if TEL is in control. (If TEL is not in control, a BREAK key returns control to TEL and interrupts execution of XEQ as described below.)

Upon termination of command file processing, the following message is output at the terminal

```
*** XEQ TERMINATED ***
```

When command file processing is interrupted by CONTROL Y or ESC ESC (or BREAK when TEL is not in command), TEL outputs the message

```
*** XEQ FILE INTERRUPT ***
```

The user may issue commands as described in the section "Interrupting, Resuming, and Terminating Execution". Execution of the interrupted process may be resumed by typing GO, CONTINUE, or PROCEED. However, the remainder of the command file will not be executed.

ECHO This command causes printing of TEL commands being executed from a command file via the XEQ command. The format of the command is

```
ECHO
```

DONT ECHO This command suppresses printing of the TEL commands being executed from a command file. This is the default. The format of the command is

```
DONT ECHO
```

ERROR This command has two effects:

1. If SEND is in effect, MOUNT messages, device error messages and operator key-ins for tapes and packs that are in use by the user are output on the user's terminal. Messages output by the system are preceded by S: and key-ins are preceded by O:. Device messages include the serial number for tapes and private packs and PUBLIC for public packs in parentheses.
2. The user may type CONTROL Y to exit from conditions requiring operator intervention when dealing with tapes. The CONTROL Y causes control to return to TEL but the condition is treated as an error. (The current I/O operation is treated as if the operator had performed an error key-in for the device.) If the user attempts to resume the program (via GO, etc.), the error condition is in effect. The format of the command is

```
ERROR
```

DONT ERROR This command prevents the user from using CONTROL Y to exit from conditions requiring operator intervention when dealing with tapes. This is the default. The format of the command is

```
DONT ERROR
```

ERROR MESSAGES

During each TEL session, checks are made for a variety of error conditions. Some of these error conditions are detected by TEL and some by the monitor. The messages that are output by TEL for error conditions are listed in Table 11. Monitor error messages are listed in Appendix B.

Most error messages are variable and may be changed by the management of an installation through a terminal that is logged on with a special identification and account. The procedure for changing error messages is defined in the CP-V/SM Reference Manual, 90 16 74.

TEL COMMAND SUMMARY

Table 12 is a summary of TEL commands. The left-hand column gives the command format, the right-hand column gives the command function and option codes.

Table 11. TEL Error Messages

Message	Description
BACKUP RECORD FULL, CAN'T ADD MORE	The BACKUP record, containing the names of files to be backed up, is full. Wait and retry command.
BACKUP OF FILE ALREADY EXISTS	The file was not backed up as requested because it has not been modified since the last backup (by the FILL system management processor).
BAD GET FILE – INVALID JIT	The specified GET file cannot be used because of illegal format.
BAD GET FILE – INVALID LIMITS	The specified GET file cannot be used because of illegal format.
CAN'T CHANGE MORE THAN 6 PSEUDO SWITCHES	The SWITCH command specifies a non-existent pseudo switch (not 1 to 6).

Table 11. TEL Error Messages (cont.)

Message	Description
CAN'T CREATE OR MODIFY DCB – A/M RECORD FULL	The total size of all DCB assignments is too large.
CAN'T SAVE EXECUTE-ONLY PRGM, QUIT OR GO	The user tried to save a load module via the SAVE command which is in an execute only load module.
COMMAND LEGAL ONLY AT JOB STEP – QUIT OR GO	The command can be issued in between job steps only.
COMMAND FROM FILE MUST BEGIN WITH A BANG (!)	The format of a record in the command file is incorrect.
CONTINUE WHAT?	The CONTINUE, GO, or PROCEED command can be issued only when a major operation, a processor, or executing user program has been stopped or interrupted. An attempt to use it at other times such as between job steps will result in an error message.
DCB NOT ASSIGNED – CAN'T UPDATE	The SET command cannot be used to update a DCB that has not been assigned.
ERROR, CODE = 03xxxx	Error having no message text, where xxxx is the code and subcode of the error.
ERROR READING :USERS FILE, COMMAND ABORTED	As a result of a SHOW or PASSWORD command, a read was attempted on the :USERS file. An error occurred during the attempt and the command was aborted.
ERROR WRITING :USERS FILE, COMMAND ABORTED	As a result of a PASSWORD command, a write was attempted to the :USERS file. An error occurred during the attempt and the command was aborted.
EXECUTE ACCOUNTS EXCEED 8 ENTRIES	A maximum of 8 execute account numbers may be specified for each DCB assignment.
EXPANDED INPUT EXCEEDS 80 CHARACTERS	An abbreviated command (D, E, C, B, L) exceeded 80 characters when the common mnemonic was expanded.
FILE DOES NOT EXIST	The file specified by the BACKUP command does not exist.
FILE INELIGIBLE FOR BACKUP	The file will not be backed up because a current backup already exists or the file was designated exempt from backup requests when the file was created.
FILENAME: ME IS ILLEGAL	The terminal may not be used for the requested purpose.
GET CAL FAILED – PROBABLY BAD DCBs	The specified GET file cannot be used because of illegal format.
GET FILE DOES NOT MATCH THIS SYSTEM VERSION	On a GET command, the current system is not the same version as that at the time of the SAVE.
GET WHAT?	TEL cannot find the GET file.
ILLEGAL OR INCONSISTENT RESOURCE NAME	On a SET command, either the device is not an assignable resource type or the device is inconsistent with the tapecode/filecode field.
IMPROPER FORMAT FOR SET COMMAND	A format error has been made in the SET command.
INPUT ERROR – RETRY	TEL received a parity error in the input from the terminal.

Table 11. TEL Error Messages (cont.)

Message	Description
xxxx IS NOT A VALID OPTION	xxxx was specified in the option field of the SHOW command and is not a legal option. SHOW will continue execution if any valid options were specified.
LOGGING OFF TO FREE YOUR LINE FOR TP	An informational message to indicate that the TP command was successful.
M:xx or F:xx IS NOT ASSIGNED	User requested from SHOW the assignment of an unassigned DCB. SHOW will continue execution if any valid options were specified.
MAXIMUM NUMBER OF SN'S EXCEEDED	The maximum number of volume serial numbers that may be specified on a SET command is 3.
MESSAGE DISALLOWED BY DONT SEND	A MESSAGE command was given but DONT SEND status is in effect.
NO SAVE FILE NAMED	TEL cannot find the SAVE file.
NOTHING TO SAVE	No program in execution–user is at job step.
ON FILE fid ILLEGAL	The file following the preposition ON already exists.
opt OPTION ILLEGAL FOR {ANS LABELED TAPE DEVICES FILES JOURNAL TAPE UNKNOWN FILE TYPE XEROX LABELED TAPE}	The option specified is not relevant for the type of assignment made.
dopt OPTION LEGAL ONLY FOR DEVICES	Specified device option is not relevant for files, Xerox labeled tape, or ANS labeled tape.
PASSWORD CHANGE REQUIRES ',OLD,NEW' PASSWORDS	Invalid format on the PASSWORD command.
PASSWORD CHANGE SUCCESSFUL	The change specified by the PASSWORD command has been made.
PASSWORD SUPPLIED DOESN'T MATCH CURRENT PASSWORD	The old password specified does not match the user's current password; or no old password was specified when one exists; or one was specified when none exists.
QUIT?	The last command was issued during a Y^c interrupt and would abort the previous command if executed. For example, assume a LINK command is interrupted. !LINK A, B ON F (RET) Y^c !PROGLM. QUIT? ! !FORT4 AA ON BB (RET)
READ ACCOUNTS EXCEED 8 ENTRIES	A maximum of 8 read account numbers may be specified for each DCB assignment.
READ ERROR ON COMMAND FILE	TEL was unable to read the specified command file.

Table 11. TEL Error Messages (cont.)

Message	Description
START WHAT?	Either the START command did not specify a load module or it specified a dollar sign and there was no previous link operation.
TEL ISSUED SINGLE USER ABORT ON YOU	TEL detected a problem, requested monitor to abort user.
TERMINAL TYPE NOT VALID	The TERMINAL command specified a terminal type other than 33, 35, 37, 7015, APL, C360, DATA POINT, EAPL, ESTD, EXECUPORT, MEMOREX, SAPL, SSTD, TI, or an installation-supplied translate table name.
TP IS NOT CURRENTLY ACTIVE	The TP command was given but transaction processing is not currently up and running.
TP IS NOT IN THIS SYSTEM	The TP command was given but transaction processing is not available in this system.
UNABLE TO GET COMMON PAGE FOR SHOW	TEL was unable to get common page to pass SHOW information. Execution of SHOW command is terminated.
UNABLE TO GET PAGE FOR BACKUP RECORD	TEL was unable to obtain a page for the BACKUP record. Wait and retry command.
UNABLE TO OPEN :USERS FILE, COMMAND ABORTED	TEL was unable to open the :USERS file for a PASSWORD command. Execution of the command is terminated.
:USERS FILE BUSY, TRY AGAIN LATER	If this message is returned for a PASSWORD command, it indicates that TEL cannot read the user's file because it is open. If the message is returned for a DELETE command, it indicates that no password was specified for a file that is passworded.
WRITE ACCOUNTS EXCEED 8 ENTRIES	A maximum of 8 write account numbers may be specified for each DCB assignment.

Table 12, TEL Command Summary

Command	Description
ANSF [sp] $\left[\begin{matrix}\text{ON}\\ \text{OVER}\end{matrix}\ [\text{rom}][,\text{list}]\right]$	Compiles an ANS FORTRAN source program. Options: sp may be fid or ME. rom may be fid, stream-id, CP, or NO. list may be fid, stream-id, LP, ME, or NO.
BACKUP fid	Saves the specified file on a system tape. In case of a crash in which files are lost, files on the tape will be restored.
BATCH fid[,fid]... (Simplified format)	Enters the specified file(s) in the batch job stream. (See Batch processor chapter for complete description).
B[UILD] fid	Allows a new file to be created from the terminal using the Edit processor.
BYE	Disconnects the terminal from the system and provides an accounting summary. This command is equivalent to the OFF command.

Table 12. TEL Command Summary (cont.)

Command	Description
CANCEL jid[,account][,jid[,account]]...	Cancels previously submitted batch jobs.
COBOL [sp] [ON OVER [rom] [,list]]	Compiles an ANS COBOL source program. Options: sp may be fid or ME. rom may be fid, stream-id, CP, or NO. list may be fid, stream-id, LP, ME, or NO. Output may be interrupted and continued by the following commands: LIST COMMENT DONT OUTPUT CONTINUE OUTPUT DONT LIST DONT COMMENT
COMMENT [ON OVER list]	Directs error commentary to the specified device or counteracts the preceding DONT COMMENT command. Option: list may be fid, LP, ME, or stream-id.
CONTINUE	Continues processing from the point of interruption. This command is equivalent to the GO and PROCEED commands.
C[OPY] sf{TO[†] OVER INTO} df (Simplified format)	Copies a file or device input to the specified file or device. Options: sf may be fid or device code. df may be fid or device code. (See PCL section for complete description.)
COUPLE	Allows other terminals to couple to this terminal.
COUPLE line	Establishes a link between the user's terminal and the terminal specified by line.
DECOUPLE	Releases the coupling between two terminals.
D[ELETE] {[DC/] DP# serial number[-rt]/} fid[,fid]...	Deletes the specified file(s). Option: rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource.
DELTA	Calls the Delta processor.
DI[SPLAY]	Lists the current values of various system parameters.
DONT COMMENT	Stops error commentary output.
DONT COUPLE	Causes attempts to couple to the terminal to be rejected.
DONT ECHO	Suppresses printing of TEL commands during command file processing.
DONT ERROR	Prevents the user from using CONTROL Y to exit from conditions requiring operator intervention when dealing with tapes.

[†] Whenever TO is specified, ON may be substituted.

Table 12. TEL Command Summary (cont.)

Command	Description
DONT LIST	Stops listing output.
DONT OUTPUT	Stops object output.
DONT SEND	Disallows messages from the machine operator to the user's terminal. Global broadcasts are deferred until TEL is in control. Also disallows the MESSAGE command.
ECHO	Allows printing of TEL commands during command file processing.
E[DIT][fid]	Calls Edit to modify a file.
END	Terminates the current job step. This command is equivalent to the STOP and QUIT commands.
ERASE	Deletes the accumulated output for the line printer.
ERROR	Allows the user to enter CONTROL Y to exit from conditions requiring operator intervention when dealing with tapes.
EXTEND	Sets the extended memory mode; i.e., appends the special processor area to the available user area.
FORT4[sp] [ON OVER [rom][,list]]	Compiles a Xerox Extended FORTRAN IV source program. Options: sp may be fid or ME. rom may be fid, stream-id, CP, or NO. list may be fid, stream-id, LP, ME, or NO. Output may be interrupted and continued by the following commands: LIST COMMENT DONT OUTPUT CONTINUE OUTPUT DONT LIST DONT COMMENT
GET fid	Restores the previously saved core image. This command is equivalent to the RESTORE command.
GO	Continues processing from the point of interruption. This command is equivalent to the CONTINUE and PROCEED commands.
JOB jid[,jid]...	Requests the status of jobs that were submitted to the batch queue via the Batch processor.
L [LT#reel-id[-rt][(s)] [DC][.acct][(s)] LT#serial no.[-rt][(s)]/fid[(s)][,fid[(s)]]. . . fid[(s)][,fid[(s)]]. . . DP#reel-id[-rt][(s)] DP#serial no.[-rt]/fid[(s)][,fid[(s)]]. . . FT#serial no.[-rt][(s)]]	Lists file names and, optionally, attributes from the account directory, tape, or disk pack. Options: s may be A or EA. rt specifies the 2-character identifier of a device that was defined at SYSGEN to be a resource.

Table 12. TEL Command Summary (cont.)

Command	Description
LDEV stream-id[,(option)]...	Modifies a logical device definition. Options: see Table 8.
LIST [ON OVER list]	Directs the listing output to the specified device, or counteracts the preceding DONT LIST command. Option: list may be fid, LP, ME, or stream-id.
lmn[sp] ON OVER [rom] [,list]	Initiates execution of a load module. Options: lmn has the form: name [.[account] [.password]] absence of period and account specifies system account. presence of period and absence of account specifies log-on account. sp is assigned to M:SI DCB. rom is assigned to M:GO DCB. list is assigned to M:LO DCB.
LYNX [ef[,ef]. . .] [ON OVER lmn] [options][; libname][.[libacct][.password]]...	Constructs a load module from the elements (ef) specified. See Chapter 8.
M[ESSAGE] text	Sends the specified message to the operator. The first character of the message may not be a period (.).
META[sp][ON OVER [rom] [,list]]	Assembles the specified Meta-Symbol source program. Options: sp may be fid or ME. rom may be fid, stream-id, CP, or NO. list may be fid, stream-id, LP, ME, or NO. Output may be interrupted and continued by the following commands: LIST DONT LIST CONTINUE OUTPUT DONT OUTPUT COMMENT DONT COMMENT

Table 12. TEL Command Summary (cont.)

Command	Description
OFF	Disconnects the terminal from the system and provides an accounting summary. This command is equivalent to the BYE command.
OUTPUT [ON OVER fid]	Directs object output to the specified file, or counteracts the previous DONT OUTPUT command.
PAGE n	Resets the terminal header page number to the value specified by n.
PASSWORD, old-password, new-password PASSWORD, , new-password PASSWORD, old-password,	Assigns, changes, or deletes a log-on password for the user. The password is 1-8 characters excluding I , ; < > . / = ? and all COC control characters. A null field is used to specify nonexistence of either an old or new password.
PLATEN[w][,l]	Sets the value of the terminal platen width and/or page length if w and/or l are specified; or displays the terminal platen width and page length values if neither w nor l is specified. (Page length does not include header.)
PRINT	Sends accumulated symbiont output, such as output for the line printer or the card punch, to the output device.
PROCEED	Continues processing from the point of interruption. This command is equivalent to the GO and CONTINUE commands.
PROMPT [character]	Sets the default prompt character, or resets the default prompt character mechanism if a character is not specified.
Processor Calls	These calls are entered while TEL is in control of the terminal. They turn over control of the terminal to the processor. Examples are: ANSF FLAG LYNX APL BASIC PCL
Q[UIT]	Terminates the current job step. This command is equivalent to the STOP and END commands.
R[ESET]	Resets all DCBs back to their system default values.
RESTORE fid	Restores the previously saved core image. This command is equivalent to the GET command.
RUN [options][ef[,ef]. . .] [ON OVER] lmn [;[libname][.[libacct][.password]]]. .	Calls the LYNX loader and instructs it to link, load, and start execution of the designated module. See Chapter 8.

Table 12. TEL Command Summary (cont.)

Command	Description
SAVE $\begin{Bmatrix}\text{ON}\\\text{OVER}\end{Bmatrix}$ fid	Saves the current core image on the designated file.
SEND	Allows messages from the machine operator to be printed on the user's terminal.

SET dcb[0]
SET dcb $\begin{bmatrix}\text{oplabel}\\\text{device}\\\text{stream-id}\\\text{tapecode[tapeid]}\end{bmatrix}$ [;dopt]. . .
SET dcb $\begin{bmatrix}\text{tapecode[tapeid][-rt]}\\\text{filecode[-rt]}\end{bmatrix}$/fid[;fopt]. . .
SET dcb JR/fid
	Assigns file or device to a DCB or sets DCB parameter.
	Options:
	rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource.
	other options, see Tables 3, 4, 5, and 6.

SHOW [option[,option]. . .]	Displays information about currently logged-on user. Options: USER, PRIV, DCBS, M:xx or F:xx, or ALL.
S[TART] $\begin{bmatrix}\text{lmn}\\\text{\$}\end{bmatrix}$ [U[NDER DELTA]]	Loads a load module into core and starts execution of the program, either with or without an associated debugger.
ST[ATUS]	Displays the current accounting values.
STOP	Terminates the current job step. This command is equivalent to the END and QUIT commands.
SWITCH S[ET]= n[,n]. . .[R[ESET]=n[,n]. . .]]	Controls setting and resetting of the user's pseudo sense switches, where n ranges from 1 to 6. With no arguments the command displays the pseudo sense switch settings. This command is the equivalent of the batch !SWITCH command.
TABS[s[,s]. . .]	Sets simulated tab stops for the terminal if s values are specified; or displays the simulated tab stop settings if no s value is specified.
T[ERMINAL] type[,algorithm]	Sets the terminal type for proper I/O translations. Type may be 33, 35, 37, 7015, APL, C360, DATA[POINT], EAPL, ESTD, EXEC[UPORT], MEMO[REX], SAPL, SSTD, TI, or an installation-supplied translation table name.
T[ERMINAL] STAT[US]	Lists the terminal type and the current values of parameters associated with its operation.
TP	This command logs off a time-sharing terminal and makes it available as a slave Transaction Processing terminal.
U	Causes the words UNDER DELTA to be inferred in the next command.
WHERE account, name	Returns the line number of the specified user (if the user is logged on).
XEQ fid[,record number]	Initiates processing of TEL commands from a command file.

4. LANGUAGE PROCESSORS

INTRODUCTION

The Meta-Symbol, ANS FORTRAN, Extended FORTRAN IV, ANS COBOL, APL, and BASIC processors may be used in either the on-line or batch mode. The command processor EASY operates in the on-line mode only. An introduction to the on-line operating features of these processors is given in this chapter. Complete descriptions of the processors are given in the following manuals:

Xerox Meta-Symbol/LN, OPS Reference Manual, 90 09 52

CP-V ANS FORTRAN/LN Reference Manual, 90 32 00

CP-V ANS FORTRAN/OPS Reference Manual 90 32 01

Xerox Extended FORTRAN IV/LN Reference Manual, 90 09 56.

Xerox Extended FORTRAN IV/OPS Reference Manual, 90 11 43.

Xerox ANS COBOL/LN Reference Manual, 90 15 00.

Xerox ANS COBOL/OPS Reference Manual, 90 15 01.

Xerox APL/LN, OPS Reference Manual, 90 19 31.

Xerox BASIC/LN, OPS Reference Manual, 90 15 46.

Xerox EASY/LN, OPS Reference Manual, 90 18 73.

META-SYMBOL

The Meta-Symbol assembler is called from an on-line terminal by the following command:

```
META [sp] [ON   [rom] [,list]]
          [OVER               ]
```

where

sp specifies a source program and may be either a file identification (fid) or the terminal identification (ME). If no source file is specified, TEL assumes input is from the file/device currently assigned to the M:SI DCB. If the M:SI DCB is not assigned, TEL expects input to come from the terminal (ME). The M:SI DCB is reset at each job step.

ON indicates that ROM output is to be on a new file or to a logical device stream or a device.

OVER indicates that ROM output is to be over an existing file or on a new file.

rom specifies that the relocatable object module produced by compilation is to be directed to a specified file (fid), a logical device stream (stream-id), the card punch (CP), or no file or device (NO). If no rom specification is given, output is directed to a special file that may subsequently be referenced by a dollar sign. If a rom is specified, it is assigned to the M:GO DCB and that assignment remains in effect throughout the job until reassignment by a subsequent SET, META, FORT4, COBOL, or lmn command.

list specifies that listing output is to go to a file (fid), a logical device stream (stream-id), the line printer (LP), the terminal (ME), or no file or device (NO). If list is not specified, TEL assumes that the listing output is to go to the file/device currently assigned to the M:LO DCB. If the M:LO DCB is not assigned, TEL produces no listing output. Note that the M:LO DCB assignment is made explicitly by the SET command and implicitly by the META, FORT4, COBOL, and lmn commands. Once set, the M:LO DCB assignment remains in effect throughout the job until reassignment by a subsequent SET, META, FORT4, COBOL, or lmn command.

This command replaces the control cards that are needed to perform the equivalent operations through batch processing. The replaced cards are

```
!JOB
!ASSIGN M:SI. . .
!ASSIGN M:LO. . .
!ASSIGN M:GO. . .
!METASYM SI, LO, GO
```

When the assembler is entered, it sends a request for options (WITH >) to the terminal. If there are no options, a carriage return character may be entered following the request. This initiates the assembly, providing additional inputs are not required by the assembler.

Three META options are assumed by default: SI, GO, and LO. These options cause reading from source, production of ROM output, and listings on the device or file given in the META command.

If assembly options are desired, the codes (Table 13) for the desired options are entered following the request for options. These codes are separated by commas and terminated by a carriage return or line feed character which initiates assembly. If a concordance option (CN in Table 13 has been specified, additional input is required. Meta-Symbol sends a prompt character to the terminal to request each concordance command. Assembly is initiated only after the last concordance control command (.END) has been entered.

Examples:

1. Assume that file A is to be assembled with ROM output going to B and list output going to the terminal. No special assembly options are desired, and no additional input is required by the assembler.

```
!META A  ON  B(RET)
WITH>(RET)
*      ERROR SEVERITY: 0
* NO ERROR LINES
!
```

Table 13. Meta-Symbol Assembly Options[†]

Option	Description
AC (ac$_1$,ac$_2$,...,ac$_n$)	Specifies alternate accounts that are to be searched when the assembler must access system files that are not logged either under the system (:SYS) account or under the user's log-on job account. The ac items are alternate accounts that are searched first; then by default, the :SYS account and finally the log-on account are searched as necessary.
DC	Specifies that a "standard" concordance is to be produced on the LO device. The "standard" listing does not include operation code names, but otherwise includes all symbol references, including function and command procedure names and intrinsic functions.
CN	Requests that a symbolic cross-reference listing be included with the assembly listing. When this option is given, the assembler sends a prompt character to the terminal to indicate that concordance control records identifying special concordance options should be entered. One control record, preceded by a period, is entered following each prompt character. The last control record must be an END record. The concordance control commands are as follows: IO Include all or a selected set of operation codes. SS Suppress all or a selected set of symbols. OS Include only a selected set of symbols. DS Produce a modified LS listing, displaying only lines that reference a selected set of names. END Terminate concordance control commands.
CO	Causes the assembler to produce a compressed version of the input program on the file specified in the M:CO DCB. This DCB must have previously been assigned by a SET command.
LU	Requests that the assembler include a listing of the Meta-Symbol update records with the program listing.
NS	Requests that no assembly summaries be included with the listing.
SD	Causes the assembler to produce symbolic debugging object code for use with the Delta debugging processor. The object code is included with the standard binary output ROM.
SO	Causes the assembler to create a source output file corresponding to the input program. The input program may be Edit-source, compressed, or compressed with updates. The M:SO DCB must have been previously assigned.
CI	Causes the assembler to access M:CI for compressed input. Typically it would be specified if the user wishes to update the compressed file with the contents of the source file assigned to M:SI (via, e.g., the META command). The source input on the M:SI file must be terminated with a +END statement. The M:CI DCB must have been previously assigned by a SET command. Consult the Meta-Symbol/LN,OPS Reference Manual, 90 09 52 for a full discussion of the assembler's operation when both SI and CI inputs are specified.

[†] For additional details concerning assembly options, refer to the Meta-Symbol/LN,OPS Reference Manual, 90 09 52.

2. Assume that a disk storage file, called SOURCE, is to be assembled. ROM output is to go to BIN and list output is to go to the line printer. A cross-reference is to be included with list output. The cross-reference is to exclude symbols X1 and X2 and to include operation code CAL3.

```
!META SOURCE ON BIN, LP (RET)
WITH> CN (RET)
>.SS X1, X2 (RET)
>.IO CAL3 (RET)
>.END (RET)
```

3. Assume that a source program, called SOURCE, is to be assembled with ROM output going to BIN and listing output going to the line printer. The following assembly options are desired:

   a. A source output file (SOURCEOUT) corresponding to SOURCE.

   b. A compressed version of SOURCE.

   c. A symbolic cross-reference.

   d. A symbolic debugging object code for Delta.

```
!SET M:SO DC/SOURCEOUT (RET)
!SET M:CO CP (RET)
!META SOURCE ON BIN, LP (RET)
WITH> SO, CO, CN, SD (RET)
>.END (RET)
```

When input is from a keyed Edit file, a decimal representation of the sequence number for each record is placed in the assembly listing. This representation is placed in the position normally occupied by columns 73-80 of an input card.

It is possible to make use of Meta-Symbol's internal editor in conjunction with compressed source files while running on-line. The internal editor and source compression facility are oriented toward card image batch processing but can be useful to on-line operation when backup files must be kept on cards or when work must be done in strictly a BPM-compatible fashion. These Meta-Symbol features are described in Chapter 12 of the Meta-Symbol/LN,OPS Reference Manual, 90 09 52.

If a program in compressed format exists on RAD or disk pack storage, either as the output of the assembler or as a result of a file management operation, it can be assembled with on-line Meta-Symbol simply by specifying it as input (sp) in a META command. Meta-Symbol distinguishes between the keyed source format of the Edit files and the sequential binary format of compressed files.

Example:

Assume that CI-FILE is a program file in compressed format. This file is to be assembled with ROM output going to BO-FILE and list output going to the line printer.

```
!META CI-FILE ON BO-FILE, LP (RET)
WITH> (RET)
```

It is also possible to maintain an update file through the use of Edit and to use the file to modify a compressed file. In this case the former would be assigned to M:SI and the latter to M:CI, via a SET command and the CI assembly option.

Example:

Assume that an update file (UPDATE-FIL) is being maintained under Edit and is to be used to update a CI-FILE on labeled tape. ROM output is to go to BIN and list output is to go to the line printer.

```
!BUILD UPDATE-FIL (RET)
    1.000 + 4,6 (RET)
    2.000           BANZ EXIT (RET)
    3.000 + 10,10 (RET)
    4.000 + END (RET)
    5.000 (RET)
*END (RET)
!SET M:CI LT#1234I/CI-FILE (RET)
!META UPDATE-FILE ON BIN,LP (RET)
WITH>CI (RET)
```

ANS FORTRAN

The ANS FORTRAN Compiler is compatible with most features of the forthcoming (new) ANS Standard FORTRAN language which includes many extensions to the 1966 ANS FORTRAN Standard Language. It is operable under CP-V as a shared processor, offering services to both the batch user and the on-line user. The user may request, as an option, that the compiler produce either ROM output or program execution (LOAD and GO).

Advantageous features of the ANS FORTRAN compiler are

- Compiler speed on the order of 2K-3K lines per minute.
- Compressed input/output capability.
- Addition of INCLUDE (system) capability.
- Conversational characteristics for time-sharing.
- New ANS FORTRAN compatibility.
    - CHARACTER variables.
    - Expanded READ/WRITE capabilities.
    - OPEN and CLOSE statements.

The ANS FORTRAN compiler is called from the on-line terminal by the following TEL command:

$$\text{ANSF}\ [\text{sp}]\left[\begin{matrix}\text{ON}\\ \text{OVER}\end{matrix}\ [\text{rom}][,\text{list}]\right]$$

ANS FORTRAN is described in the ANS FORTRAN/LN Reference Manual, 90 32 00, and the ANS FORTRAN/OPS Reference Manual, 90 32 01.

FORTRAN IV

The Xerox Extended FORTRAN IV compiler is called from an on-line terminal by the following TEL command:

$$\text{FORT4}\ [\text{sp}]\left[\begin{matrix}\text{ON}\\ \text{OVER}\end{matrix}\ [\text{rom}]\ [,\text{list}]\right]$$

where

sp specifies a source program and may be either a file identification (fid) or the terminal identification (ME). If no source file is specified, TEL assumes input is from the file/device currently assigned to the M:SI DCB. If the M:SI DCB is not assigned, TEL expects input to come from the terminal (ME). The M:SI DCB is reset at each job step.

ON indicates that ROM output is to be on a new file or to a logical device stream or a device.

OVER indicates that ROM output is to be over an existing file or on a new file.

rom specifies that the relocatable object module produced by compilation is to be directed to a specified file (fid), a logical device stream (stream-id), the card punch (CP), or no file or device (NO). If no rom specification is given, output is directed to a special file that may subsequently be referenced by a dollar sign. If a rom is specified, it is assigned to the M:GO DCB and that assignment remains in effect throughout the job until reassignment by a subsequent SET, META, FORT4, COBOL, or lmn command.

list specifies that listing output is to go to a file (fid), a logical device stream (stream-id), the line printer (LP), the terminal (ME), or no file or device (NO). If list is not specified, TEL assumes that the listing output is to go to the file/device currently assigned to the M:LO DCB. If the M:LO DCB is not assigned, TEL produces no listing output. Note that the M:LO DCB assignment is made explicitly by the SET command and implicitly by the META, FORT4, COBOL, and lmn commands. Once set, the M:LO DCB assignment remains in effect throughout the job until reassignment by a subsequent SET, META, FORT4, COBOL, or lmn command.

The naming of files sp, rom, and list can be thought of as simple assignments for the DCBs used by the compiler. The DCBs M:SI, M:GO, and M:LO are used by FORTRAN for its input and output operations and CP-V directs the data to and from the respective files. The specifications sp, rom, and list are used for these assignment purposes and have no effect on the operation of the compiler. The control of th compilation rests with the compiler options described below.

In the absence of a specification for rom or list, CP-V will direct the data to or from the last file or device to which GO or LO was assigned. In this way a user may make these assignments at the beginning of a job and they will remain in effect until changed. If an identifier is not specified for ROM, the object program produced by the compilation will be written on a scratch file which may be referenced later by the name $.

When the FORTRAN IV compiler is entered in the on-line mode, it sends a request for options to the terminal by typing

OPTIONS>

The user may then enter the option codes (Table 14) to be used for this compilation. The codes are separated by commas and terminated by a carriage return. If no option codes are entered before the terminating carriage return, the compilation will be done as though the single option PS has been typed. The PS option ensures that the user is aware of the size of his program and the first and last card while producing a minimum of output. A source input file is always expected and an object program is always produced when operating from an on-line terminal.

After the option request has been completed, the compiler reads the source program from the sp file. Input continues until an END statement or end-of-file (ESC F keys) is encountered. The program summary and object program are then output as requested and control is returned to TEL. If the source file contains more than one program, subsequent compilations can be obtained from it by using the BC option.

When used from an on-line terminal, the compiler accepts horizontal tab characters in source program records. It replaces each tab character with the correct number of spaces to locate the next input character at the position specified by the next tab stop. At the on-line terminal, this positioning is done by the monitor so that the typist is aware of the tabbing action. Internally, the compiler performs a similar action by inserting the correct number of spaces into the source record image. Any characters in the record following the tab character are shifted to the right. The listing output and source output from the compiler do not contain the character. They contain the spaces which were inserted into the image.

Table 14. FORTRAN IV Compilation Options[t]

Option	Description
ADP	Causes all real operations to be done in double precision and all complex operations to be done in double complex. (See Extended FORTRAN IV/LN Reference Manual, 90 09 56.)
BC [(n)]	Permits a number of programs to be compiled from the source file. When this option is used, the compiler reads source programs until the conditions of the option are met. Thus, a number of different programs may be compiled using only one FORT4 command. The suboption, n, allows the BC option to specify compilation of the first n programs from the source file.
BO	Causes a binary object deck to be produced (via M:BO). If the BO option is used, the correct assignment for M:BO must be ensured. There is no default assignment for this DCB.
COMS	Causes COMMON symbols to be generated for use with Delta.
DEBUG	Causes the compiler to generate linkages, such as internal symbol tables, to the FORTRAN Debug Package.
GO	This option is redundant. In on-line operation, a binary object deck is produced for all programs via the M:GO DCB.
LO	Lists the object program on the LO device.
LOCS	Causes local symbols to be generated for use with Delta.
LS	Lists each source program and compilation summary on the LO device.
NMP	Causes the generated code of the object program to be a control section with protection type 00 instead of 01.

[t] For more details concerning compilation options, refer to the Extended FORTRAN IV/OPS Reference Manual, 90 11 43.

Table 14. FORTRAN IV Compilation Options[†] (cont.)

Option	Description
NS	Eliminates the compilation summary map and the printing of the first and last card of the source program. To eliminate the entire listing of a compilation, NS or PS should be specified and LS or LO should not be specified.
PS	Causes the first and last cards and a partial summary map of the program to be printed. The partial summary map includes NUMBER OF ERROR MESSAGES:n NUMBER OF STATEMENTS DELETED:m } These are printed only if there were errors in the program. HIGHEST ERROR SEVERITY: { 0 (NO ERRORS) 4 (NO MAJOR ERRORS) 7 (MAJOR ERRORS) 10 (MAJOR ERRORS) } DEC WORDS HEX WORDS GENERATED CODE: ddddd xxxxx CONSTANTS: ddddd xxxxx LOCAL VARIABLES: ddddd xxxxx TEMPS: ddddd xxxxx ----- ----- TOTAL PROGRAM: ddddd xxxxx
RT	Causes reentrant object code to be generated for the real-time user.
S	Specifies that in-line assembly code is to be accepted on cards that have an S in column 1. For information concerning the rules for in-line symbolic code, see the Extended FORTRAN IV/LN Reference Manual, 90 09 56.
SB(xxxx)	Permits control of the sequencing of binary output. The xxxx field specifies the id to be used in the sequencing.
SBIT	Preserves the integrity of the maximum negative number expressible on a Sigma computer.
SI	Specifies source input. This is unnecessary but is acceptable for compatibility.
SO	Reproduces the source program on the source output file (via M:SO).
X	Compiles records with X in column 1.

[†] For more details concerning compilation options, refer to the Extended FORTRAN IV/OPS Reference Manual, 90 11 43.

As many tab characters as are required may be entered but care should be taken to ensure that tab stops are provided. If the compiler cannot match a tab character with a corresponding tab stop position during its internal expansion operation, the tab character will remain in the record and will cause a syntax error. Tab stops may be set from the on-line terminal by the TEL commands TAB and SET.

When accepting source input from an on-line terminal, the compiler normally checks each record as it is typed and immeditaely prints out any diagnostic on the following line. However, if a statement is to be continued, this error checking is not done until the continuing statements have all been input. When a statement is to be continued, the last character preceding the carriage return must be a colon (:) to indicate that this record is continued. The next record, the continuation record, must follow standard FORTRAN rules and have blanks in columns 1 to 5 and a continuation character in column 6. Statements containing errors and continued over several records have their error diagnostics printed following the last record. The colon used to indicate that a record is continued is removed from the record and replaced with a blank character.

Since the colon is contained in the Extended FORTRAN IV standard character set, it is possible to use it in a FORTRAN statement. Some difficulty might be expected in statements which end with a colon, such as, A = 4HABC:. This problem can be overcome by typing an extra blank following the colon and before the carriage return.

Examples:

Assume a program is to be compiled with the source input read from file SOURCE, the relocatable object module written onto file DECK, and the listing written onto the user's terminal.

```
!FORT4 SOURCE ON DECK,ME(RET)
OPTIONS>LS(RET)
```

Since the compiler always expects an SI file and always generates a GO file, the only option (LS) is used to cause a listing of the source program at the user's terminal. If any errors occur they are printed at the terminal.

ANS COBOL

The COBOL compiler is called from an on-line terminal by the following command:

$$\text{COBOL } [\text{sp}]\left[\begin{matrix}\text{ON}\\ \text{OVER}\end{matrix}[\text{rom}][,\text{list}]\right]$$

where

sp specifies a source program and may be either a file identification (fid) or the terminal identification (ME). If no source file is specified, TEL assumes input is from the file/device currently assigned to the M:SI DCB. If the M:SI DCB is not assigned, TEL expects input to come from the terminal (ME). The M:SI DCB is reset at each job step.

ON indicates that ROM output is to be on a new file or to a logical device stream or a device.

OVER indicates that ROM output is to be over an existing file or on a new file.

rom specifies that the relocatable object module produced by compilation is to be directed to a specified file (fid), a logical device stream (stream-id), the card punch (CP), or no file or device (NO). If no rom specification is given, output is directed to a special file that may subsequently be referenced by a dollar sign. If a rom is specified, it is assigned to the M:GO DCB and that assignment remains in effect throughout the job until reassignment by a subsequent SET, META, FORT4, COBOL, or lmn command.

list specifies that listing output is to go to a file (fid), a logical device stream (stream-id), the line printer (LP), the terminal (ME), or no file or device (NO). If list is not specified, TEL assumes that the listing output is to go to the file/device currently assigned to the M:LO DCB. If the M:LO DCB is not assigned, TEL produces no listing output. Note that the M:LO DCB assignment is made explicitly by the SET command and implicitly by the META, FORT4, COBOL, and lmn commands. Once set, the M:LO DCB assignment remains in effect throughout the job until reassignment by a subsequent SET, META, FORT4, COBOL, or lmn command.

This command replaces the control cards that are needed to perform the equivalent operations through batch processing. The replaced cards are

```
!JOB. . .
!ASSIGN M:SI. . .
!ASSIGN M:LO. . .
!ASSIGN M:GO. . .
!COBOL LS,GO
```

When the compiler is entered, it sends a request for options (OPTIONS) to the terminal. If compilation options are desired, the word COBOL must be entered followed by the codes (Table 15) for the desired options. These codes are separated by commas and terminated by a carriage return or line feed character which initiates compilation.

Example:

Assume that file PAYROLL is to be compiled with ROM output going to PAYPROG and the listing going to the line printer.

```
!COBOL PAYROLL ON PAYPROG,LP(RET)
OPTIONS?
$ COBOL LS,GO(RET)
```

Table 15. ANS COBOL Compilation Options

Option	Description
BO	Specifies that a permanent copy of the object program is to be written via M:BO.
CS(name)	Specifies the name of the COMMON-STORAGE SECTION.
DEBUG	Specifies that program debugging statements (e.g., TRACE, EXHIBIT) are to be compiled if they exist.
DIAG	Specifies that trivial diagnostic messages are to be listed.
DMAP	Causes a data division map to be produced.
DQ	Specifies that the quote character (instead of a single apostrophe) is to be used as the quote character.
GO	Specifies that a load-and-go copy of the object program is to be written via M:GO.
LIB(accounts)	Specifies library accounts for the COPY verb.
LO	Requests an object program listing via M:LO.
LS	Requests a source program listing via M:LO.
MAIN	Specifies that this is a main program.
MAPS	Specifies that both a data division map and a procedure division map are to be produced.
PMAP	Causes a procedure division map to be produced.
SEG	Causes priority segments to be honored.
SEQCHK	Requests that the compiler sequence check the source program.
SO	Requests source output via M:SO.
SORT	Requests code to be generated to interface with co-resident sort.
SUB	Specifies that this is a subprogram.
SYN	Requests compilation for syntax checking only. (No code will be generated.)
XREF	Requests a cross-reference listing via M:LO.

APL

The APL processor was designed to be used at a terminal that has a special APL typeball. It is possible to use APL at a standard terminal, but the user must be aware of a set of substitute characters and mnemonics that replace APL characters that are either illegal or missing on his terminal. (These characters are documented in Appendix B of the APL/LN, OPS Reference Manual, 90 19 31.)

At an APL terminal, TEL prompts with a small circle (∘) rather than with the normal exclamation point (!). The APL processor is called by entering the following command:

```
APL
```

APL acknowledges control by typing the message 'APL-date' and a message indicating whether a clear workspace is available or the CONTINUE workspace has been loaded.

Example:

```
∘APL (RET)
APL-03/16/73
CLEAR WS
```

The user can now enter APL assignments, statements, function definitions, system commands, etc.

The user exits from APL via any of the following APL system commands:

```
)OFF
)CONTINUE
)OFF HOLD
)CONTINUE HOLD
```

CP-V BASIC

CP-V BASIC may be operated in on-line or batch mode. The on-line mode is the normal mode. Batch operations are limited to those requiring no user intervention and differ from on-line operations primarily in the assignment of input/output devices.

The BASIC system is called from a terminal in the following way:

```
!BASIC (RET)

>
```

When the system is ready to accept input, it prompts with a "greater than" character (>). At this point, BASIC is in editorial mode with no program text.

In the on-line mode, BASIC returns to TEL only if terminal input failure occurs, the BREAK key is activated twice without any intervening terminal input or the SYS[TEM] command is typed. In batch mode, exit to the monitor also occurs after a compilation that contains errors, or after a run-time error.

While using BASIC in on-line mode, the user fully controls the flow of activity via the terminal. The normal mode for doing so is to respond to prompt characters that indicate the system is prepared for input. Two prompt characters are used: a question mark and a "greater than" symbol. A question mark indicates that execution is in progress and input data is required. A "greater than" symbol indicates that the system is ready for editorial input or commands.

In some instances, such as during the output of an extended listing or when a program is suspected of being in a loop, it is desirable to acquire terminal control without waiting. A facility is provided via the BREAK key activation to interrupt current activity.

For additional detail about BASICoperations, refer to the BASIC/Reference Manual, 90 15 46.

CP-V EASY

CP-V EASY is an on-line command processor that enables the terminal user to create, edit, execute, save, and delete program files written in BASIC or FORTRAN. It also allows the terminal user to create and manipulate EBCDIC data files.

CP-V EASY is called from a terminal in the following way:

```
!EASY RET

NEW OR OLD-- RET
```

The user types NEW if he wishes to create a new file, or OLD if he wishes to access an existing file.

EASY then asks for the file name, and the user types it. The user may begin creating or updating a BASIC or FORTRAN program, or he may wish to utilize one of the EASY commands. These commands are described in the Xerox EASY/LN, OPS Reference Manual, 90 18 73.

When the terminal user first calls EASY, it is assumed that he wishes to create or modify a BASIC or FORTRAN program. If he wishes to deal with a data file, he enters the DSM command after naming the file. In addition, programs are executed in BASIC unless FORTRAN is requested (via the SYSTEM FORTRAN command).

Since CP-V EASY is a control processor, pressing the BREAK key or CONTROL Y returns the terminal user to EASY, not to TEL. The only way to return to TEL is to type the TEL command. The system responds with the TEL prompt (!) to indicate the successful transfer of control to TEL.

5. PERIPHERAL CONVERSION LANGUAGE

INTRODUCTION

The Peripheral Conversion Language (PCL) is a utility processor designed for operation in a batch or on-line environment. It provides for information movement among card devices, line printers, on-line terminals, magnetic tape devices, and RAD or disk pack storage.

PCL is controlled by commands supplied through on-line terminal input, through a file containing PCL commands, or through command card input in the batch job stream. The command language provides for single or multiple file transfers with options for selection, sequencing, formatting, and conversion of data records. Additional file maintenance and utility commands are provided. The actual input/output operations are carried out using standard system CALs.

For batch operation, PCL is activated by a !PCL control command card in the job stream. Once active, PCL reads subsequent command cards directly through the M:SI DCB until terminated by an END command card or some other control command card. Input and output is done through the M:EI and M:EO DCBs respectively. Error messages are transmitted to the device currently assigned to the M:DO DCB.

For on-line operation, PCL is called by typing "PCL" while TEL is in command of the terminal. PCL responds by typing "PCL version HERE" followed by a prompt character (<) at the left margin of the next line. This indicates that PCL is ready to accept a command.

Example:

```
!PCL (RET)
PCL F00 HERE
<
```

When accepting or processing a command on-line, PCL is in the command state. Entry to this state is always indicated by the display of the PCL prompt character. Once a valid command begins execution, PCL enters the active state. In this state, PCL prompts for input, if required, with a period (.). This state remains in effect until execution of the command terminates, at which time PCL reenters the command state, issues a < prompt character, and waits for the next command. As in batch operation, user input and output is processed through the M:EI and M:EO DCBs; error messages go to the M:UC DCB and commands are received through the M:SI DCB.

The user has the option of building a file of PCL commands and having the commands executed by preceding the call to PCL by an ASSIGN or SET command that assigns M:SI to the file of commands. In this case, PCL will not prompt the on-line user for input, but will print each command, preceded by a prompt character (<), as it begins execution of the command.

Example:

```
!SET M:SI/CMDFILE (RET)
TPCL (RET)
PCL D00 HERE
< first command from CMDFILE
   .
   :
```

The on-line user may avoid using the SET command by using the following command to call PCL:

```
!PCL CMDFILE
```

In this case, Tel will perform the M:SI assignment.

The following description of PCL is oriented toward the on-line user. For the batch user, communication is established with input through the job stream and output through the M:LO DCB with no user interaction. Thus, all user prompting and terminal-specific operations given here may be ignored by the batch user.

SYNTAX CONVENTIONS

PCL is a free form language with a few restrictions imposed for simplicity in implementation and use. These restrictions are outlined below:

1. Blanks preceding or following an argument field are permitted; embedded blanks are not permitted except within quotes, which delimit a character string.
2. At least one blank must follow each command verb, except REW and REM when followed by a number (#) character, and must precede and follow each command preposition (TO, ON, INTO, or OVER).
3. A command may be continued from one line to the next by ending the continued line with a semicolon (i.e., a semicolon must be the last nonblank character). There is no limit on the number of continued lines; however, a command that contains more than 1024 characters (exclusive of the semicolon continuation characters) will be rejected.

   Example:

   ```
   <COPYALL LT#1#2#3#4 TO LT#; (RET)
   <A#B#C#D (RET)
   ```

4. "End-of-command" is indicated by the end of the input record (column 72) for card input or by a carriage return or line feed character for either card or on-line terminal input.
5. Only one input device and only one output device may be open at any given time.
6. Any command that begins with an asterisk is treated as a comment line and is output through the M:LL DCB.

SOURCE AND DESTINATION SPECIFICATION

Most PCL commands require the specification either of a source alone or of a source and a destination. These specifications can take one of two forms: simple or complex.

A simple specification consists of a device type, a logical device stream-id, or an operational label.

Device types that are known and checked by PCL are listed in Table 16. Other device type codes may be specified and are checked for validity by the monitor. These device type codes correspond to unformatted unit record equipment and their use results in action that is very close to direct device access.

Table 16. PCL Device Types

Device Type	Description
CR	Card reader (not available for on-line operations). For batch operations, files are separated by two successive EOD control cards.
CP	Card punch.
LP	Line printer.
ME	For time-sharing mode, on-line terminal. (Input is terminated by an ESC F — end-of-file — code.) For batch processing mode, card reader for input and line printer for output.

Any logical device stream that was defined at SYSGEN may be specified and is identified by its stream-id (e.g., L1, C1, P1).

Any operational label that was defined at SYSGEN may be specified. The standard system operational labels and their default device assignments are listed in Table 5.

A complex specification is required for devices with mountable volumes (magnetic tapes and private disk packs) or for devices which may hold logically connected groups of records called files. In most cases, each file has a name by which it is known to the system. Files with names may be contained on RAD, public disk pack, private disk pack, Xerox labeled tape, or ANS labeled tape. Files without names may exist on magnetic tape. Mountable volumes carry internally a serial number and the creator's account number.

Complex specifications allow the user to uniquely identify device and file combinations by organization type, volume identification, resource type, and file identification. When a complex specification is required, the user needs only to provide enough information to uniquely identify the source or destination of the data.

The general form of a complex specification is:

ot#vol-id[-rt][/fid]   for sources

ot[#vol-id][-rt][/fid]   for destinations

where each of the fields are described briefly below and in detail in the sections that follow:

ot   is the organization type.

vol-id   is the volume identification.

rt   specifies a resource type and is the 2-character identifier of a device that was defined at SYSGEN to be a resource.

fid   is a file identification.

There are some exceptions to this general format. Some PCL commands allow options to be specified which are specific to the particular command. In most cases, the options follow the complete source or destination specification. However, in some cases, the option may be embedded in the complex specification. In addition, some commands allow multiple volume-ids and lists of files to be specified.

ANS tape specifications are a special case. They have the format

AT[#serial no.][-rt][/filename]

The serial number is optional if the file name is present. Normally, both the file name and serial number are specified. The serial number may be omitted only when the file name specified is that of the first file on the tape. In this case, the serial number of the tape should be communicated to the operator (e.g., on the job sheet or via a MESSAGE command).

ORGANIZATION TYPE

The organization type specifies the type of disk or magnetic tape that the data resides on. Valid organization types for PCL are listed in Table 17.

Table 17. PCL Organization Types

Organization Type	Description
DC	RAD storage. (See Default Disk Pack, page 68.)
DP	Disk pack storage. (See Default Disk Pack, page 68.)
LT	Xerox labeled tape.
AT	ANS labeled tape.
FT	Free form tape. (Files are separated by an EOF mark.) Note: When keyed files are copied to free form tape, the keys are lost

FILE AND VOLUME IDENTIFICATION

A file identifier (fid) has three parts: name, account and password[†]. A file name consists for PCL of 1 to 31 character[††], which in general may be any characters except the following PCL delimiters:

blank . ( ) ; / ,

[†] An ANS tape file identifier consists of a name only.

[††] Note that most on-line processors allow a maximum of 10 characters for a file name. ANS tape file names are limited to 17 characters.

However, any character including these delimiters may be used in a file name if the name is delimited by single quotes, e.g., '(A)'. Single quotes within such a file name must each be represented by paired quotes.

A hexadecimal format may be used to represent a file name that contains one or more unprintable characters, e.g., X'00E7'.

PCL translates any two-character names that start with an * into three-character names. The first two characters (of the three-character name), which replace the *, are the user's job id. For example, *G is the GO file.

When PCL outputs a file name, account, or password, it prints the string in hexadecimal format if any of the characters do not belong to the EBCDIC 57-character set, unless such characters have been read as input to PCL.

Account and password are one to eight characters from the same set and may also be written as hexadecimal or character strings. The various combinations are written as follows:

name	file in log-on account.
name.account	file in specified account.
name..password	file in log-on account with password.
name.account.password	file in specified account, with password.

In general, a job may create, delete, read, or modify files in the account in which it is running. However, files in different accounts can only be read – not created, deleted, or modified. A file identifier is the same whether the file is on RAD, disk pack, or labeled tape. However, in order to access a file on labeled tape, the physical volume identifier must in general also be given.

To access a file on a private disk pack, the volume identifier of the primary volume must be given. When creating files on a disk pack, all volume identifiers for the volume set must be specified. The following description of a volume identifier applies to disk pack as well as to labeled tape.

A volume identifier (vol-id) consists of two parts: a serial number and an account number.

The account has the same format as described above, while a serial number for devices other than ANS tape is one to four alphanumeric characters of the same character set as file identifier, except that the number sign (#) may not be used unless the serial number is enclosed in quotes. An ANS tape serial number may be up to six alphanumeric characters. The two permissible forms for a volume identifier are as follows:

#serial no. [#serial no.]...[#serial no.]

Volume(s) created, or to be created, in log-on account.

#serial no. [#serial no.]...[#serial no.] .account

Volume(s) created in specific account.

The # is a syntactic identifier used to introduce the serial number, e.g.,

```
#MEFA
#MEF1#MEF2.C7308300
```

The optional serial numbers are used to indicate a multi-volume file or set of files. A maximum of 50 serial numbers is allowed.

In general, a job cannot create files on a labeled tape or disk pack in a different account than that in which it is executing. However, it may read tapes or disk packs that were created in different accounts.

Therefore, in subsequent command descriptions, the following convention is adopted. If a volume identifiei is used in an input sense, where either of the above representations is valid, then it will be symbolized as "#reel-id". However, if it is used in an outpjt sense, where only a serial number is valid, then "#serial no." will be used explicitly. In either case, up to 50 serial numbers may be specified if a multi-volume file is involved. Free form tape (FT) only needs to be identified by a serial number.

SCRATCH TAPES

Although it is not shown in the syntax descriptions of the PCL commands, a volume identifier is never actually required for any command. The absence of a volume identifier on a labeled tape or free form tape specification implies that a scratch tape is to be used. After the first occurrence of a scratch tape specification in an output sense, the output volume identifier of the tape is communicated to the on-line user ir order that this tape may be referenced by subsequent commands. If a scratch tape is used for the first time in an input sense, an I/O error is reported. If a scratch tape has been written, a command in the same PCL session that specifies a tape without a volume identifier, in either an input or output sense, is interpreted by PCL as referring to the same scratch tape. PCL must be reentered if a second scratch tape is needed.

DISK PACK DEFAULT

Although it is not shown in the syntax descriptions of the PCL commands, a volume identifier is not required if the organization type code is DP. If the file is random, the absence of a volume identifier on the disk pack specification indicates that the public disk pack is to be used. For other types of files, the absence of a volume identifier causes the DP organization type code to be treated the same as DC.

RESOURCE TYPE

The resource type must be a valid 2-character mnemonic for a device which was defined at SYSGEN to be a resource (e.g., 7T, 9T, BT). Resource type is a qualifier to the organization type and is necessary in order to uniquely identify the device. It is needed only for devices with mountable volumes when more than one type of device of the same organization type are present on the system. Thus, when a system has only 800 bpi 9-track tape drives, the organization type LT, FT, or AT uniquely identifies the device type and the resource type specification is unnecessary. However, if the system has, for example, both 9-track and 7-track tapes, then the resource type (7T or 9T) must be specified.

SPECIFICATION EXAMPLES

The following examples illustrate simple and complex specifications. The user should remember that a source or destination specification requires only the minimum information necessary to uniquely identify the source or destination.

1. A file called MYFILE in the user's account with the password SECRET.

   ```
   DC/MYFILE..SECRET
   ```

   In most cases, the DC/ is not needed to identify the file. However, it is required when specifying a file name which could be confused with a PCL or CP-V reserved name. Thus, DC/LP is a file; LP is a line printer.

2. A file called MYFILE contained on a two-volume Xerox labeled tape set for which the serial numbers are 123, and 456. The tapes are 1600 bpi and the tape device identification was SYSGENed as BT.

   ```
   LT#123#456-BT/MYFILE
   ```

3. The same file as 2 above on ANS tape.

   ```
   AT#'123      '#'456      '-BT/MYFILE
   ```

4. A file called MYFILE on a private disk pack with the serial number PAK1 whose device identification was SYSGENed as DA.

   ```
   DP#PAK1-DA/MYFILE
   ```

5. A user wishes to list the disk pack in Example 4. The pack was created in account F65426QL which is not the user's account.

   ```
   DP#PAK1.F65426QL-DA
   ```

CAPABILITIES

The following is a list of available functions in PCL defined in terms of the actual command verbs:

COPY device(s) and/or file(s) TO[†] device or new file.

COPY device(s) and/or file(s) OVER or INTO device or existing file.

COPYALL files in specified account on RAD or disk pack TO labeled tape(s) or to a device.

COPYALL files in specified account on RAD or disk pack TO log-on account on RAD.

COPYALL files on labeled tape(s) TO RAD or disk pack.

COPYALL files on labeled tape(s) TO files on labeled tape(s) or to a device.

COPYSTD performs a copy of a control file and all files indicated within the control file.

DELETE specified files on RAD or disk pack.

DELETEALL deletes all or a portion of the user's files on RAD or disk pack.

ERRORS SAVE/REL controls the disposition of output files when errors occur during copying commands.

LIST a file directory for RAD, tape, or disk pack.

MOUNT causes designated tape or disk pack to be mounted.

PRINT sends any waiting output for symbiant devices to them.

REVIEW user's file directory on RAD or disk pack.

SPF space file ±n files on free form (unformatted) magnetic tape.

SPR skip records ±n records on free form (unformatted) magnetic tape.

WEOF write end-of-file on current output device.

REW rewind designated tape.

SPE space to end of last file on labeled tape.

REM remove designated tape or disk pack.

TABS define tab settings for tab expansion.

MODE OPTION COMPATIBILITY

In the current version of PCL, 7T and 9T are resource types and are no longer used as mode options. However, for compatibility with previous versions of PCL, 7T and 9T may still be specified as mode options in all of the commands for which they were previously applicable. If 7T or 9T is specified as a mode option, it will be treated exactly as though it had been specified as a resource type.

BREAK FUNCTION

The function of the BREAK key under PCL (as under TEL) is to interrupt current activities. If the BREAK key is pressed while PCL is in the active state, PCL usually terminates what

[†] Wherever TO is specified, ON may be substituted.

it is doing, such as printing or copying, passes control to the terminal, and reverts to the command state. If the BREAK key is pressed while PCL is in the command state, PCL ignores the current command as if X^c had been pressed. The effect of the interruption or the termination varies with the command being executed and is discussed in detail with each command, where necessary. If no mention is made of the effect, the BREAK key is assumed to have no effect on execution of the command.

FILE COPY COMMAND

The file COPY command permits single or multiple file transfers to take place between peripheral devices or between file storage and peripheral devices. Options are included for selecting, formatting, and converting data records. When more than one keyed file is copied to a single file, PCL can either merge or concatenate the files (see "Record Sequencing", below).

COPY COMMAND FORMAT (GENERALIZED)

The COPY command is of the form

```
                            ┌TO  ┐
C[OPY] source[,source...]   │OVER│ destination
                            └INTO┘
```

where

source may be an input device such as card reader (CR), a RAD file (e.g., ALPHA), a file on private disk pack, or a file on Xerox or ANS labeled tape or free form tape. File concatenation or merging may be performed by specifying more than one source device or file.

destination may be an output device such as card punch (CP), a public disk file, a file on private disk pack, or a file on Xerox or ANS labeled tape or free form tape. Absence of a destination specification is allowed and will normally cause file extension to occur.

If the purpose of the COPY is to replace a RAD or disk file currently existing in the user's account directory, PCL will require that the preposition OVER be used in the command. That is, COPY TO, OVER, or INTO will create a file, but for the user's protection only COPY OVER can replace an existing file. After this check, PCL opens the source devices and files one at a time in the order given, and copies them to the destination device or file. Source files are closed after they have been copied. The destination device or file is closed at the same time.

If the BREAK key is pressed during execution of the COPY command, PCL responds by typing the message 'ENTER X TO ABORT'. Any character typed, except X, causes continuation of the command. Typing an X aborts the command and causes the partially created output file to be released unless the ERRORS SAVE command has been specified.

Note that the command preposition and the destination are optional. If the COPY command contains only a source specification, PCL uses the destination device or file defined on the most recently issued COPY command containing a destination specification. (This is illustrated in the sixth COPY example.) It should be noted that file extension will occur in this case. Any PCL command except COPYALL may be used between the COPY defining the destination specification and the COPY with this specification omitted, since the output specification will not be changed by these commands.

If the destination is an unmanaged device (CP or FT), two end-of-files (EOD, TM) are written when the device is closed.

File extension may also be accomplished by using the INTO preposition in the command.

If a COPY command is used without a destination specification and a destination has not been defined by a previous command, the default destination is to the terminal.

The message '..COPYING' prints at the terminal when the copy operation begins if neither the input nor the output device is the terminal.

COPY COMMAND FORMAT (SPECIFIC)

The specific format of the COPY command is

```
        ←——— Source 1 ———→

  C[OPY] sd[(s)][/fid[(s)][,fid[(s)]]...]

  ←——— Source 2 ———→

  [sd[(s)][/fid[(s)][,fid[(s)]]...]]...

   ←———Destination———→

  ┌TO                       ┐
  └OVER dd[(s)][/fid[(s)]]  ┘
```

where

sd represents the device portion of a <u>source</u> specification and may be a device type (Table 16), a logical device stream-id, an operational label, or one of the following:

```
DP
DC
DP#serial no.[-rt]
LT#serial no.[-rt]
AT[#serial no.][-rt]
FT#serial no.[-rt]
```

where rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource.

/ separates a PCL identification code from the associated file specifications. The slash is only required if both device (sd or dd) and file (fid) specifications are given.

fid represents file identification and has the form

```
     [[.[account].password]]
name [                     ]
     [. account            ]
```

The DC identification code is optional on a COPY command referencing a RAD or public disk file. For example, RAD file A may be specified in one of two formats: DC/A or A. However, this flexibility makes the codes in Table 16 reserved words. For example, file CR must be referred to as DC/CR or 'CR', never simply as CR. The fid is not optional for ANS tapes.

, separates files on the same device.

; separates devices. (Interpreted as a continuation character if last nonblank character of a line.)

(s) represents specifications for data encoding: data codes (Table 18), formats (Table 19), modes (Table 20), record sequencing (Table 21), accounts (Table 22), ANS tape options (Table 23), expiration option, and record selection. It has the form

```
(option[,option]...)
```

Specifications given at the device level apply to all files on that device. Those given at the file level apply to that file only and have precedence if a conflict occurs between levels.

Data encoding is discussed in detail below.

dd represents the device portion of the destination specification and may be a device type (Table 16), a logical device stream-id, an operational label, or one of the following:

```
DP
DC
DP#serial no.[-rt]
LT[#serial no.][-rt]
AT[#serial no.][-rt]
FT[#serial no.][-rt]
```

where rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource.

Examples:

1. Assume that three consecutive files, each terminated by a double !EOD mark, are to be copied from a card reader to an existing RAD storage file called ALPHA. (This would only be allowed in batch.) The PCL command would be:

```
COPY CR;CR;CR OVER ALPHA
```

or

```
COPY CR OVER ALPHA

COPY CR

COPY CR
```

2. Assume that a Meta-Symbol source program file, called SOURCE, is to be copied from RAD storage to the terminal. The command could be coded as

```
≤COPY SOURCE TO ME (RET)

START LW, R1 ALPHA

        AI, R1 5

        CW, R1 BETA
```

This command could also be typed as

```
≤C SOURCE TO ME (RET)
```

3. Assume that successive cards are to be copied from the card reader to a new RAD storage file with the following file identification: KD.2024.PLEASE. (This would only be allowed in batch processing.) Two !EODs are used to signal the end of the card file. The COPY command would be:

```
C CR TO KD.2024.PLEASE
```

4. Assume that files B and C from 1600 bpi labeled tape No. 57 are to be copied, in that order, to a new RAD storage file called B..PASS.

```
≤C LT#57-BT/B,C to B..PASS(RET)
..COPYING
```

5. Assume file A from labeled tape No. 5, file D from RAD storage, and all files on free form tape No. 8 up to the next double end-of-file are to be copied to file A on labeled tape Nos. 6 and 7. Tape No. 7 is to be used only if No. 6 overflows.

```
≤C LT#5/A;D;FT#8 TO LT#6#7/A(RET)
..COPYING
```

6. Assume three successive sets of files, each separated by a double end-of-file, are to be punched in cards from free form tape No. 7236. Two !EODs are written when the output device is closed.

```
≤C FT#7236 TO CP(RET)
..COPYING
≤C FT#7236(RET)
..COPYING
≤C FT #7236(RET)
..COPYING
```

or

```
≤C FT#7236;FT#7236;FT#7236 TO CP (RET)
..COPYING
```

DATA ENCODING

The COPY command may contain various codes and specifications which either describe certain characteristics of input and output files or devices, or which request various types of data conversion or format changes in the output to be produced. Partial files may be copied by use of record selection and output records may have sequence identification inserted or deleted.

A description of the available codes and specifications follows:

DATA CODES

Data codes (Table 18) describe the source or destination data types to be expected or produced for devices only.

Table 18. Data Codes

Code	Meaning
E	EBCDIC (default data code)
H	Hollerith (FORTRAN BCD conversion)

DATA FORMATS

Data formats (Table 19) describe the source or destination record formatting to be expected or produced.

Table 19. Data Formats

Code	Meaning
X	Hexadecimal dump
C	Meta-Symbol compressed
CRPT (seed)	Encryption seed

The X option produces a single-spaced dump on the line printer or terminal. The presence of an asterisk following the word count in the dump indicates that omitted lines are identical to the preceding line.

A C option on an input specification indicates that input is in compressed format and is to be decompressed on output. A C option on an output specification indicates that input is in symbolic form and is to be compressed on output.

The CRPT option is followed by from 1 to 8 hexadecimal characters which specify the seed for data encryption for keyed and consecutive files. Since separate algorithms are used for keyed and consecutive files, a keyed file that is encrypted cannot be decrypted if its keys are stripped. Data encryption is described in the CP-V/BP Reference Manual, 90 17 64.

MODES

Mode codes dictate the control modes for the specified files or devices. They are shown in Table 20.

Table 20. Mode Codes – Copy Command

Mode	Description
BCD, BIN	Binary-coded decimal or binary mode. These codes are valid for cards, paper tape, and magnetic tape.
PK, UPK	7-track binary tape packed or unpacked.
SSP, DSP, VFC	Single, double, or variable format controlled spacing on line printer or terminal.
NC	No carriage return. Removes carriage-control character (X'15' or X'0D'), if present, from each record on output. This mode is the default mode if input is from the terminal.
NB	No trailing blanks. Removes trailing blanks (X'40'), if present, from each record on output. This operation is performed after NC, if specified.
VOL(n)	Volume number. The value n specifies the volume to use for a multi-volume tape set.
CR	Retains carriage return. Must be specified if carriage returns are to be retained when copying 'ME' to a file or device.
TX	Tab expansion. Values specified on a PCL TABS command are used. If a PCL TABS command was not issued, the tab values in the M:UC DCB are used. If no tab values are specified, single spaces replace tabs on output.
FA, NFA	File attributes. These codes specify whether or not the attributes (i.e., variable-length parameter list except name, account, and password) of the source file are to be carried over to the destination file. If the file name remains the same from source to destination and neither FA nor NFA is specified, the attributes are copied. If the names of the source and destination files are different, the attributes are not normally copied; information specified in ASSIGN or SET commands takes effect.
DEOD	Double end-of-file. Multiple source files are copied into a single output file. Thus, while COPY FT copies files including single end-of-file marks up to a double end-of-file, COPY FT (DEOD) copies files to a double end-of-file without copying the single end-of-file marks.
K	Print keys. If the file has a 3-byte key, the listing is not to be in hexadecimal form and the destination is a printer or terminal; the file is assumed to be an Edit format file. The use of the K option on output causes the key to be decoded as an Edit line number in the form xxxx.xxx and to be printed on the same line with the record contents (Edit listing format). A record sequence number pre-

Table 20. Mode Codes – COPY Command (cont.)

Mode	Description
K (cont)	cedes the key. For other types of keyed files, the key is not decoded and prints on the line preceding the record contents. If the file is not keyed, only the record sequence number precedes the record contents.
DEN(800)	Dual density drive is to be written at 800 bpi.
DEN(1600)	Dual density drive is to be written at 1600 bpi.
ASCI	Conversion of code between EBCDIC in core and ASCII on tape is to be done.
EBCD	No code conversion is to take place. EBCDIC code is used on tape.
JOB	JOB file. Specifies that the file is to disappear at job termination. If the 2-3 characters of file name are ::, they will be replaced by the user's sysid.
LC, UC	Translate alphabetic characters to the indicated case, lower case or upper case.
NF	No formatting. PCL does not produce any output that is not in the input data.

Examples:

1. Assume that file A is to be copied on to a 1600 bpi labeled tape No. 4 with exactly the same attributes it had on RAD storage.

```
≤ C A TO LT#4/A (DEN(1600))(RET)
..COPYING
```

2. Assume that RAD storage file A is in compressed form and is to be converted to symbolic and listed on the printer with double spacing.

```
≤ C A (C) TO LP(DSP)(RET)
..COPYING
```

3. Assume that line images are to be read from RAD storage file A, converted from EBCDIC to Hollerith, and written on a 7-track scratch tape in BIN mode.

```
≤ C DC/A TO FT-7T(BIN,H) (RET)
..COPYING
```

4. Assume that a source file, SOURCE, containing tab characters was created on-line and is to be punched with tab characters expanded and carriage return characters removed.

```
≤ C SOURCE TO CP(TX,NC) (RET)
..COPYING
```

RECORD SEQUENCING

Insertion or deletion of sequence identification for output data records is accomplished by using record sequencing specifications (Table 21). These specifications are available only as output options. All of these options are mutually exclusive.

PCL can either merge or concatenate keyed files. If the LN option is specified for the output file, concatenation will occur with the new keys as specified in the LN option. If the NLN option is specified for the output file, concatenation will occur with the output file being a consecutive (not keyed) file. If no record sequencing option (i.e., neither LN nor NLN) is specified for the output file, a merge will occur. In this case, if records with duplicate keys exist, the record from the first specified input file will be replaced (in the output file) with the record from the next specified input file. Thus the sequence in which the input files are specified will determine which of the identically keyed records appears in the output file. When concatenating a keyed file and a consecutive (unkeyed) file, the LN or NLN option should be used.

Examples:

1. Assume that a file called SORC on labeled tape #25 is to be sequenced and punched into cards. The card identification is SRCE, the initial value is 1, and the increment is 1. Thus, logical records are to be given sequential identification as follows: SRCE0001, SRCE0002, SRCE0003, etc.

```
≤ C LT#25/SORC TO CP (CS(SRCE, 1, 1)) (RET)
..COPYING
```

2. Assume that PCL is to read successive records from free form tape #73, to assign line numbers starting at 5, in increments of 5, and to write the records on RAD storage file A.

```
≤ C FT#73 TO DC/A(LN(5,5)) (RET)
..COPYING
```

3. Assume that two keyed files A and B, are to be concatenated into file C and assigned new keys. Default keys are to be assigned.

```
≤ C A,B TO C(LN)(RET)
..COPYING
```

4. Assume that files A and B are to be merged into a new keyed file C with the output records alternately coming from A and B.

```
≤C        A  TO C(LN(1,2))(RET)
..COPYING
≤C        B  INTO C(LN(2,2))(RET)
```

Table 21. Record Sequencing Options—COPY Command

Code	Description
CS[(id[,n,k])]	Card sequencing in columns 73-80. id is identification (0-4 characters) n is initial value k is increment The identification (id) is left-justified in the field (73-80) and is followed by the sequence number, which is right-justified in the same field. The identification may be written as a character string containing one to four characters; e.g., '..XY'. Precedence is given to the sequence number if overlapping occurs. The default values for id, n, and k are null, 0, and 1, respectively.
NCS	No card sequencing. This specification strips columns 73-80 from each output data record.
LN[(n,k)]	Line numbering. Sets organization to keyed. The file starts at n and continues in sequential steps of k. Line number and increment formats are as in the Edit processor. Line numbers must be between 1 and 9999. Increments may range from .001 through 100.000. The default values for both n and k are 1.
NLN	No line numbering. Sets organization to consecutive.

ASSIGNMENT OF ACCOUNTS

The combined list of read accounts, write accounts, execute accounts, and the name of a processor under which the file is to be run (see Table 22) must not exceed 16 entries.

Table 22. Account Options — COPY Command

Code	Description
RD(ac_1[,ac_2,...])	Adds read account(s) on output. ALL or NONE may be specified in place of an account.
WR(ac_1[,ac_2,...])	Adds write account(s) on output. ALL or NONE may be specified in place of an account.
EX(ac_1[,ac_2]...)	specifies the account numbers of those accounts that may execute the file. The value ALL may be used to specify that any account may execute the file. The value NONE may be used to specify that no other account may execute the file. In all of the above cases, RD(NONE) is implied in the absence of any RD specification.
UN(name[,name]...)	specifies the name(s) of the processor(s) that may access this file if the user does not own the file. The name may be from one to ten characters in length. The processor may be any shared processor or any load module in the :SYS account. If EXecute accounts are specified and UNder is not specified, the file is presumed to be a load module and may be executed by any user running in an EXecute account but not under Delta.

Examples:

1. Assume that file A is to be copied to labeled tape No. 4 with the same attributes it had on RAD storage plus the addition of read accounts ONE and TWO.

```
<C A TO LT#4/A(RD(ONE,TWO)) (RET)
..COPYING
```

2. Assume that read account ALPHA, write accounts X and Y, execute accounts ONE, TWO, and THREE, and the name of a load module BETA under which the file SRCE is to be run are to be added as attributes of file SRCE.

```
≤C SRCE OVER SRCE(RD(ALPHA),WR(X,Y); (RET)
,EX(ONE,TWO,THREE),UN(BETA)) (RET)
..COPYING
```

ANS TAPE OPTIONS

Special options for ANS tapes are described in Table 23. These options pertain to record blocking, concatenation of files, and changing the record formats. Unblocking is always performed when copying from an ANS input tape.

FMT, BLK and REC may be specified for any input or output device to perform ANS-type blocking/deblocking. REC alone causes all records to be truncated or padded (with blanks up to 140 characters) to the specified length.

Table 23. ANS Tape Options—COPY Command

Code	Description
FMT(f)	Output format. The value of f must be . F – fixed-length records, blocked. D – variable length records, decimal size word, blocked. V – variable length, binary size halfword, blocked. U – unblocked. The default is U unless input is from ANS tape, in which case the input's format is used.
BLK(n)	Block size. The value n specifies the maximum block size to be built for FMT(F), FMT(D), and FMT(V), where 1≤ n≤32,767 bytes. The default is 2048 and if n is less than 18, 18 will be used. The default for ANS input is the value from the input file.
REC(n)	Record size. The value n specifies the size of records for FMT(F) only, where 1≤n≤32,767 bytes. Records will be truncated or padded to conform, but padding with blanks will extend only for 140 bytes. The default is 128 except for ANS input with F format, for which the value from the input file is used. The block size must be a multiple of the record size.
CAT(n)	Input option that causes n files of the specified name on ANS tape to be concatenated to produce a single output file or to be output to the named device. (All of the input files must have the same format.) The value for n may range from 2 to 128.

Examples:

1. Assume that file ABC is to be copied to file X on ANS tape number 123456. The tape is to be on a 1600 bpi drive. Only the first 72 characters of each record are to be copied, and the block size is to be 720.

```
≤C ABC TO AT# 123456-BT/X(FMT(F),;(RET)
≤BLK(720),REC(72))(RET)
```

2. Assume that four files named A are to be copied from ANS tape number '1   ', '2   ', '3   ', '4   ', and '5   ' into a single RAD file B. (Unblocking is performed if the input is blocked.)

```
≤C AT#'1       '#'2      '#'3       '#'4       ';(RET)
≤#'5       '/A(CAT(4)) TO B (RET)
```

EXPIRATION OPTION

The expiration option specifies an expiration time for the output file of the COPY command. It has the format

$$\text{EXP}\left(\begin{Bmatrix}\text{mm, dd, yy}\\ \text{ddd}\\ \text{NEVER}\end{Bmatrix}\right)$$

where

mm, dd, yy   specifies a particular date: mm is month and may be one or two digits with a value from 1 to 12; dd is day and may be one or two digits with a value from 1 to 31; yy is year and may be one or two digits with a value from 0 to 99. (The format mm, dd, yy may also be written mm/dd/yy.)

ddd   specifies the number of days to retain the file. It may be from one to three digits in length with a value from 1 to 999.

NEVER   specifies that the file is never to expire (i.e., it is to have the maximum expiration period as specified at SYSGEN).

RECORD SELECTION

This specification permits selection of the logical records to be copied by giving the sequential position of the records within the file. The specification has the form

x[-y]

All records within the file that have a position, n, satisfying the condition $x \leq n \leq y$ are selected. Multiple selections may be specified if separated by commas (e.g., 1-5, 10, 20-21). Selections do not have to be in sequential order (but nonsequential selection is very slow for tape operations). The maximum number of selections is ten for each input file.

Example:

Assume that sections of two files, N1 and N2, are to be combined to form a third file, N3. Records 20-30 and 40-100 of N1 followed by records 50-75 of N2 are to be copied, in that order, to N3. The job account is assumed for files N1 and N3; N2 is from account 34 under password PA.

```
≤C N1(20-30, 40-100), N2.34.PA(50-75);(RET)

≤TO DC/N3 (RET)

..COPYING
```

VALID OPTION COMBINATIONS

Not all combinations of source and destination devices, data types, formats, modes, or sequencing codes are valid. Table 24 shows the valid combinations, the invalid combinations, and the default provisions for the various possible combinations that are checked by PCL. Other combinations may be allowed, particularly for resource types, and are checked for validity by the monitor. If an invalid combination is found, an error message is produced. Execution of the command may or may not continue, depending on the severity of the error encountered (see Error Messages).

ACCOUNT COPY COMMAND

This command allows all files, or a specified subset of files, in the log-on or some other account to be copied from a file-type device (RAD, labeled tape, or disk pack) to any valid output device. It has the general form

```
COPYALL files TO device
```

where

files may be one of the following:

```
[DP][.acct][(s)][/r]
[DC][.acct][(s)][/r]
DP#reel-id[-rt][(s)][/r]
LT#reel-id[-rt][(s)][/r]
```

If 'files' is not specified, DC is assumed.

Device may be one of the following:

```
DP[(a)]
DC[(a)]
DP#serial no.[-rt][(a)]
LT[#serial no.[-rt]][(a)]
FT[#serial no.[-rt]][(a)]
LP
ME
CP
```

L1, P1, or any other logical device stream-id defined at SYSGEN

If 'device' is not specified, DC is assumed. Device must be specified if options are specified.

In the above specification,

s may be KEY to copy keyed files only; or SEQ to copy sequential files only; or RAN to copy random files only; and/or PHY to copy in physical order from tape. All input options valid for COPYing from DC, LT, or DP are also permitted here.

r may be b, e; or b; or , e.

where

b is a fid (see COPY command) representing the beginning of a range of files to be copied.

e is 1 to 31 characters representing the end of a range of files to be copied.

Both b and e are used as sort keys only and generally do not have to name an existing file. They may be written in character string or hexadecimal notation (e.g., A, 'A', or X'C1' all represent A). The e field must be equal to or greater than the b field. Files on tape are assumed to be in alphanumeric order unless the PHY option is used.

If PHY is specified, the b and e fields define a physical range of files on tape instead of an alphanumeric range and therefore must be file names. If the b field is null, copying begins wherever the tape is positioned. If the e field is null, copying continues to end of tape. If the file in the b field does not exist, the command is aborted. If the file in the e field does not exist, copying continues to end of tape.

Each of the PCL identification codes listed in Table 16 and 17 is a reserved word and may not be used as a range specification unless it is enclosed in single quotes or unless a DC or DP identification is specified in the command. For example, key DC may be legally specified as 'DC', DC/DC, DP/DC, DC/ABC,DC, etc., but never simply as DC.

Note: The introductory slash (/) is optional if no codes or options precede it.

a may be any output option valid for the COPY command.

PCL copies all files from the input device to the output device. Files protected by passwords cannot be copied with this command unless correct password is specified in the range specification. The BREAK key terminates execution of this command and causes PCL to type the identification of the last file copied.

A synonym file is copied to RAD or disk pack only if the parent file was copied or previously existed on the destination device. A synonym file is always copied to tape regardless of whether the parent file is present on the tape. If a range is specified on the command, the synonym files within the range are copied if the above conditions are met.

Table 24. Valid Option Combinations

Option	Codes	Source Device								Destination Device								
		CR	PR	DC	LT	DP	FT	AT	ME	DC	LT	DP	FT	AT	ME	LP	CP	PP
Data codes	E	d	x	d	d	d	d	d	d	d	d	d	d	d	d	d	d	x
	H	x	-	-	-	-	x	-	-	-	-	-	x	-	-	-	x	-
Data formats	X	-	-	-	-	-	-	-	-	-	-	-	-	-	x	x	-	-
	C	x	x	x	x	x	x	x	-	x	x	x	x	x	-	-	x	x
	CRPT	-	-	x	-	x	-	-	-	x	-	x	-	-	-	-	-	-
Modes	None	-	d	d	-	d	-	-	d	d	-	d	-	-	-	-	-	d
	BCD	d	-	-	-	-	x	x	-	-	-	-	x	x	-	-	x	-
	BIN	x	-	-	d	-	d	d	-	-	d	-	d	d	-	-	x	-
	7T†	-	-	-	x	-	x	x	-	-	x	-	x	x	-	-	-	-
	9T†	-	-	-	d	-	d	d	-	-	d	-	d	d	-	-	-	-
	PK	-	-	-	d	-	d	d	-	-	d	-	d	d	-	-	-	-
	UPK	-	-	-	-	-	x	x	-	-	-	-	x	x	-	-	-	-
	SSP	-	-	-	-	-	-	-	-	-	-	-	-	-	d	d	-	-
	LC	-	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x	x
	DSP	-	-	-	-	-	-	-	-	-	-	-	-	-	x	x	-	-
	VFC	-	-	-	-	-	-	-	-	-	-	-	-	-	x	x	-	-
	UC	-	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x	x
	NC	-	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x	x
	NB	-	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x	x
	NF	-	-	-	-	-	-	-	-	-	-	-	-	-	x	x	-	-
	CR	-	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x	x
	VOL	-	-	-	x	-	x	x	-	-	x	-	x	x	-	-	-	-
	K	-	-	-	-	-	-	-	-	-	-	-	-	-	x	x	-	-
	FA	-	-	-	-	-	-	-	-	x	x	x	-	-	-	-	-	-
	NFA	-	-	-	-	-	-	-	-	x	x	x	-	-	-	-	-	-
	TX	-	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x	x
	DEOD	-	-	-	-	-	x	-	-	-	-	-	-	-	-	-	-	-
	ASCI	-	-	-	x	-	x	x	-	-	x	-	x	x	-	-	-	-
	EBCD	-	-	-	d	-	d	d	-	-	d	-	d	d	-	-	-	-
	DEN	-	-	-	-	-	-	-	-	-	x	-	x	x	-	-	-	-
	JOB	-	-	x	-	x	-	-	-	x	-	x	-	-	-	-	-	-
Sequencing	None	-	-	-	-	-	-	-	-	d	d	d	d	d	d	d	d	d
	CS	-	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x	x
	NCS	-	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x	x
	LN	-	-	-	-	-	-	-	-	x	x	x	-	-	-	-	-	-
	NLN	-	-	-	-	-	-	-	-	x	x	x	-	-	-	-	-	-
Accounts	EX	-	-	-	-	-	-	-	-	x	x	x	-	-	-	-	-	-
	RD	-	-	-	-	-	-	-	-	x	x	x	-	-	-	-	-	-
	UN	-	-	-	-	-	-	-	-	x	x	x	-	-	-	-	-	-
	WR	-	-	-	-	-	-	-	-	x	x	x	-	-	-	-	-	-
Expiration	EXP	-	-	-	-	-	-	-	-	x	x	x	-	x	-	-	-	-
Selection	x-y	x	x	x	x	x	x	x	x	-	-	-	-	-	-	-	-	-

Legend: d = default x = optional - = error, not available, unreasonable

†For compatibility with previous versions of PCL.

A parent file of a synonym file within the range is not copied unless it is also within the range. If files are copied by organization (KEY, SEQ, or RAN option), synonym files are not copied.

If files are being copied to the terminal or line printer, each file copy is preceded by the name of the file. If files are being copied to any device other than the terminal, the files are listed through M:LO as they are copied.

If there are no files present in the specified account, the following message prints:

```
NO FILES IN DIRECTORY
```

If a file cannot be copied, the file is listed followed by the error or abnormal code and subcode.

PCL indicates completion of the command by printing a message of the form

```
..nnnnnn FILES COPIED
..ssssss FILES SKIPPED
```

where nnnnnn is the number of files copied and ssssss is the number of files skipped during execution of the command.

Examples:

1. Assume that all files listed in the user's account directory are to be copied to labeled tape Nos. 3 and 4. Tape No. 4 is to be used only if No. 3 overflows.

   ```
   <COPYALL TO LT#3#4 (RET)
   ```

   Note that RAD (or disk) storage space previously occupied by this account can be released for other use after the files have been copied.

2. Assume that files are to be restored on RAD storage under the job account from labeled tape Nos. 3 and 4, created under account :SYSGEN.

   ```
   <COPYALL LT#3#4. :SYSGEN (RET)
   ```

3. Assume that an exact copy of labeled tape No. 3 is to be written on tape No. 4. The record size must fit the allowable installation-set allocation of core to a single job.

   ```
   <COPYALL LT#3 TO LT#4(RET)
   ```

4. Assume that all keyed files on disk pack #5 are to be written to a scratch tape.

   ```
   <COPYALL DP#5 (KEY) TO LT (RET)

   OUTPUT SERIAL NUMBER   xxxx
   ```

5. Assume that all files on RAD between the sort keys C and L are to be copied to the line printer. Each file name will print before the file copy. It is assumed that records are in BCD format.

   ```
   <COPYALL C, L TO LP
   ```

6. Assume that all files on RAD are to have read accounts 123 and X'00C6' and write account XY added as attributes.

   ```
   < COPYALL TO DC(RD( 123, X'00C6'),;
   <WR(XY)) (RET)
   ```

CONTROL FILE COPY COMMAND

The control file copy command allows the copying of files whose identifiers appear in a control file. The command is called "copy standard" and has the form

```
COPYSTD input [TO output]
```

where

input specifies the control file and may be one of the following:

```
[DC/]fid
DP#serial no.[-rt]/fid
LT#serial no.[-rt]/fid
```

output may be one of the following:

```
DC (default)
DP#serial no.[-rt]
LT[#serial no.][-rt]
FT[#serial no.] [-rt]
LP
ME
CP
```

L1, P1, or any other logical device stream name defined at SYSGEN

rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource.

PCL opens the control file named in the input specification and unless this file is a RAD or disk file in the user's account and the output device is 'DC', the file will be copied to the specified output device. Subsequently the files named in the control file are copied to the output device using the running account and the same file names as appear in the standard file for output.

The format of a control file record is an initial character followed by name, account, and password separated by periods. For example:

```
*NAME.ACCT.PASS
*NAME.ACCT
*NAME
```

The initial character is unused in the copy operation. If no account is specified, then the source account for the file is assumed to be the same as the account of the control file itself. Commentary may appear on each record.

Files named with the control file may be from labeled tape, disk pack, or RAD; in fact all variations allowed for the input specification field of a COPY command are valid for these devices. Device codes and accounts present in the record override the one present on the COPYSTD command.

When files are copied from tape, their names should be listed in the control file in the same order as the files are stored on the tape. Otherwise, rewinds will occur between files.

If files are being copied to the terminal or line printer, each file copy is preceded by the name of the file. If files are being copied to any device other than the terminal, the files are listed through M:LO as they are copied.

If a file does not exist or cannot be copied, the file name is listed followed by the error code:

$$\text{filename} \quad \text{errorcode} \begin{Bmatrix}\text{IN}\\ \text{OUT}\end{Bmatrix}$$

The file is then bypassed and execution of the command continues.

The BREAK key terminates execution of the COPYSTD command and causes PCL to type the identification of the last file copied.

PCL indicates completion of the COPYSTD command by printing a message of the form

```
..nnnnnn FILES COPIED
..ssssss FILES SKIPPED
```

where nnnnnn is the number of the files copied and ssssss is the number of files skipped during execution of the command including the standard file itself.

Examples:

1. Assume that all files listed in file STDF on labeled tape No. 5 are to be copied to RAD storage. The format of file STDF is

   ```
   *A     COMMENTARY
   *B
   *C
   ```

   The command to be used is

   ```
   ≤COPYSTD LT#5/STDF(RET)
   ```

   On completion of the command, the files STDF, A, B, and C, will have been copied from tape No. 5 to the user's RAD account.

2. Assume that all files listed in file ST in the user's RAD account are to be copied to his account. The format of file ST is

   ```
   .ALPHA.ACCT.PASS,BETA.:SYSGEN
   :LT#5/B,C
   ```

   The command to be used is

   ```
   ≤COPYSTD ST(RET)
   ```

   On completion of the command, four files will have been copied: ALPHA, BETA, B, and C.

3. Assume that all files listed in file :STD in account :SYSGEN are to be copied to the line printer. The files listed are all in account :SYSGEN. The format of file :STD is

   ```
   =ALPHA,BETA,GAMMA
   ```

   The command to be used is

   ```
   ≤COPYSTD :STD. :SYSGEN TO LP(RET)
   ```

On completion of the command, files :STD, ALPHA, BETA, and GAMMA will have been copied from account :SYSGEN to the printer.

OTHER COMMANDS

This group of commands provides file deletion, file positioning, and other manipulation and maintenance functions.

DELETE The DELETE command deletes complete files and has the form

```
D[ELETE]{ [DC/]                    }fid[,fid]...
        { DP#serial no.[-rt]/      }
```

where

rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource.

fid specifies the identification of the file to be deleted. Each of the PCL identification codes listed in Table 15 is a reserved word for this command and may not be used as a fid unless it is enclosed in single quotes or unless a DC or DP identification is specified in the command. For example, file DC may be legally specified as 'DC', DC/DC, DP/DC, DC/ABC,XYZ,DC, etc., but never simply as DC.

Example:

Assume that RAD storage file SOURCE is to be deleted. This file is assumed to have been set up under the log-on account with password PLEASE.

```
≤ D SOURCE..PLEASE (RET)

..     1 FILES DELETED, 2 TOTAL GRANULES
```

Depressing the BREAK key terminates execution of the command. The summary message tells how many files were deleted.

DELETEALL Another delete command deletes all files, or a specified range of files, in the log-on account. The form of the command is

```
           { [DP/]                }
DELETEA[LL]{ [DC/]                } [range]
           { DP# serial no. [-rt]/}
```

where

rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource.

range specifies a range of files to be deleted and is described in detail for the COPYALL command.

The commands

DELETEALL DC and DELETEALL DP

delete all the user's files from public storage. The command DELETEALL DP with a serial number specified deletes all the user's files on the specified private disk pack.

A synonym file within the range is deleted only if its parent file is within the range.

A confirmation, YES$, is required in the on-line mode. (This is shown in the examples below.)

If there are no files in the log-on account, PCL responds to the command with the following message:

```
NO FILES IN DIRECTORY
```

If a file cannot be opened due to a password requirement, the following message prints:

```
CAN NOT ACCESS FILE xxx
```

The file is then bypassed and execution of the command continues.

After the delete function is performed, the following message prints:

```
..nnnnnn FILES DELETED
..ssssss FILES SKIPPED
..tttttt TOTAL GRANULES
```

The count (nnnnnn) does not include synonym files which were deleted.

Examples:

1. Assume that all files in the log-on account are to be deleted.

   ```
   ≤ DELETEALL (RET)
   DELETEALL.account? YES$ (RET)
   ..  8 FILES DELETED,25 TOTAL GRANULES
   ```

2. Assume that all files in the inclusive range B through H are to be deleted.

   ```
   ≤ DELETEALL B,H (RET)
   DELETEALL FROM B.account TO H?  YES$ (RET)
   ..  4 FILES DELETED,13 TOTAL GRANULES
   ```

Depressing the BREAK key terminates execution of the command and causes PCL to type the identification of the last file deleted.

LIST The LIST command is of the form

```
       [{LT}                                           ]
       [{AT} #reel-id [-rt][(s)][range]                ]
       [{DP}                                           ]
       [[DC][.acct][(s)] [range]                       ]
L[IST] [{LT}                                           ]
       [{AT} #serial no.[-rt][(s)]/fid [(s)][,fid[(s)]]...]
       [{DP}                                           ]
       [ fid(s)],fid[(s)]]...                          ]
       [ FT#serial no.[-rt]                            ]
```

All listed output goes through the M:LO DCB.

The first two formats of this command allow range specification which designates a range of files to be listed. The format of the range specification is the same as for the COPYALL command. If a range is specified, R must be included in the s specification.

The action for the various specifications is as follows:

1. $\left\{\begin{array}{l}\text{LT}\\ \text{AT}\\ \text{DP}\end{array}\right\}$ reel-id[-rt][(s)] (list file directory)

   Resource type (rt) is the 2-character identifier of a device that was defined at SYSGEN to be a resource.

   Device option (s) may be A, EA, Cn, or R (separated by commas).

   PCL scans the tape or private pack and lists the names of all files contained on it. If option A has been requested, the attributes of each file are also listed. These attributes include for LT and DP:

   Size in granules.
   Record count.
   Organization (keyed or consecutive).
   Read accounts, if other than 'ALL'.
   Write accounts, if other than 'NONE'.

   Modification time and date.
   Parent name of synonyms.
   Maximum key length (for keyed files only).
   Execute accounts (if other than 'ALL').

   Execute vehicles (if present).

   If option EA (extended attributes) has been requested, the following attributes are listed in addition to those described above:

   Expiration date.
   Creation date.
   Backup date.
   Last access date.

   For ANS tapes (AT), A and AE are equivalent and list the following attributes:

   Format (F, D, V, or U).

   Block length.

   Record length (F format).

   Block count.

   The Cn option controls the list format for a name-only list. C0 indicates that each name is to be listed on a separate line. The value of $1 \le n \le 9$ specifies the number of four-character columns to be occupied by each file name. The default is C3. Names longer than the allotted space will occupy more than one space.

   If a file requires a password or account and none is given, this will be noted.

2. [DC][.acct][(s)] (list file directory)

   Device option(s) may be A, EA, Cn, or R (separated by commas).

   PCL scans the user's RAD or public disk pack file directory and lists the names of all files. If device options have been specified, the files are listed as in 1.

3. $\left\{\begin{array}{l}\text{DP}\\ \text{LT}\\ \text{AT}\end{array}\right\}$ #serial no. [-rt] (s)]/fid[(s)][, fid[(s)]]. . .

   (list file attributes)

   This is a request for the attributes ot the indicated files. Resource type (rt) is the 2-character identifier of a device that was defined at SYSGEN to be a resource. File options (s) may be A or EA. If an account is required, it must be included in the file identifier. PCL prints an attribute summary for each file, as in 1.

4. fid[(s)j[,fid[(s)]]. . . (list file attributes)

   This is a request for the attributes of the one or more RAD or public disk pack files named. Options may be A or EA. PCL prints an attribute summary for each file, as in 1.

5. FT#serial no.[-rt][(s)]

   Resource type (rt) is the 2-character identifier of a device that was defined at SYSGEN to be a resource. Serial no. can be a fake. If the tape conforms to Xerox labeling conventions, PCL prints the serial number, account, and contents (file names) of the tape. The tape remains positioned after the last file, thus enabling the user to add files. Device option (s) may be A, EA, Cn or R (separated by commas).

If only the command LIST is given, and no specification follows, then the command executes as though it were LIST DC. LIST (A) and LIST .acct are also valid commands. All output, except for completion messages, is written through the M:LO DCB.

The BREAK key terminates execution of this command.

PCL indicates completion of the command by printing a message of the form

```
..nnnnnn FILES LISTED
```

where nnnnnn is the number of files listed during execution of the command.

If attributes of all files in a directory are listed, one of the following messages also prints:

```
..xxxxxx TOTAL GRANULES      (DC or DP)

..xxxxxx TOTAL RECORDS       (LT)

..xxxxxx TOTAL BLOCKS        (AT)
```

Examples:

1. Assume that all files on RAD under the log-on account are to be listed.

```
<L(RET)

ALPHA     BETA       GAMMA
... 3 FILES LISTED
```

2. Assume that files on 7-track labeled tape Nos. 3 and 4 are to be listed. These tapes were created under the account :SYSGEN.

```
≤L LT#3-7T.:SYSGEN(CO)(RET)
SOURCE
ALPHA
XYZ
..  3 FILES LISTED
```

3. Assume that the attributes of files ALPHA and BETA on RAD are to be listed. The attributes listed have the following meaning:

ORG	C = consecutive, Knn =keyed file (nn specifies the maximum key length), R = random file.
GRAN	Number of granules of RAD space (1 granule = 512 words).
REC	Number of records in file.
LAST MODIFIED	Modification time and date.
Name	File name.

Read and write accounts print on a separate line if necessary and will print only if they have other than default values.

```
ORG   GRAN   REC   LAST MODIFIED      NAME
 C      2     71   16:35 22 JUL 71    ALPHA
K3     14    590   01:10  1 AUG 71    BETA
.. 2 FILES LISTED
```

4. Assume that the extended attributes of file ABC on disk pack No. 2 are to be listed. This file has had write account 123 assigned previously.

```
     <L DP#2/ABC(EA)(RET)
ORG  GRAN  REC  LAST MODIFIED  NAME
C     28   385  04:24 16 AUG 71 ABC  WRITE=123

          WILL EXPIRE       31 DEC 71

          CREATED ON         2 AUG 71

          BACKED UP ON      10 AUG 71

          LAST ACCESS ON    18 AUG 71

          ..  1 FILES LISTED
```

5. Assume that a tape requires identification. The fake serial no. X is used in the command.

```
< L FT#X(CO)(RET)

INSN = 8522

ACCT = :SYSGEN

ONE

TWO

THREE

FOUR

FIVE

SIX

..  6 FILES LISTED
```

6. Assume that the files A through B of account X are to be listed with attributes.

```
≤L(R,A)A.X,B

12:41   18 AUG '76   ACCOUNT=X

ORG   GRAN   REC   LAST MODIFIED     NAME

C     26     25    10:37  28 MAY 76  AB:PS

K3    2      3     10:20  10 FEB 76  AXS

... 2 FILES LISTED

..  28 TOTAL GRANULES
```

REVIEW This command lists files in the specified account (or the user's account if none is specified) and waits for a user response after listing each file name to allow the option of deleting the file. The format of the command is

```
          ⎧[DC/]               ⎫
REV[IEW]  ⎨DP# serial no.[-rt]/⎬ [(s)] [range]
          ⎩.acct               ⎭
```

vhere

rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource.

range specifies a range of files to be reviewed and is described in detail for the COPYALL command.

s may be A or EA (as in LIST command)

This command may be used in the batch mode and will function identically to 'LIST' with a range specification.

The BREAK key or an 'E' response terminates execution of this command.

Example:

```
 REV N,X

--ENTER D TO DELETE FILE.

NAY

P.

W99 D *DELETED*

..3 FILES LISTED, 1 FILE DELETED, 2 TOTAL GRANULES
```

Each file name within the inclusive range N through X is listed and a wait occurs. Only one character must be typed as a response. If a D is typed, the confirmation message *DELETED* prints, and the next file name is listed. If a D is typed before the typing of the name has completed, the D is ignored and the name is listed again (on the same line). If any character other than D or E is typed, including carriage return or line feed (LF), the file is not deleted. If an E or BREAK is typed, the review is terminated.

Note that except for 2741 terminals, PCL responds immediately to the character that is typed (the period (.) and the D in the example above) and that a carriage return should not be used if another character is typed. On the 2741 terminal, the ATTN key should be depressed after the character is typed. The carriage return that occurred at the end of the line

```
P.
```

was provided by PCL.

If a file has a password or is open by another user, this is noted by an appropriate message when an attempt is made to delete the file.

PRINT This command causes output accumulated for symbiont devices to be placed in the output queue to be output immediately. (Normally, the output destined for symbiont devices is not output until the user logs off or issues a TEL PRINT command.) The format of the command is

```
PRINT
```

ERRORS The ERRORS command controls the disposition of output files when a fatal error occurs during a copy operation. It has the form

```
         ⎧SAV[E]     ⎫
ERR[ORS] ⎨REL[EASE]  ⎬
         ⎩hhhhhh     ⎭
```

where

SAVE causes all subsequent output files to be saved even if a fatal error occurs during their creation

RELEASE causes all subsequent copy operations which abort to release the output file. RELEASE is in effect when PCL is first entered.

hhhhhh is a hexadecimal error code whose meaning is to be typed.

SPF
SPR Those commands position free form tape forward or backward a designated number of files (SPF) or records (SPR). The form of the command is:

```
(SPF)
(SPR) [FT]#serial no. [-rt][,±n]
```

where

rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource.

+ specifies forward direction.

- specifies backward direction.

n is the number of files or records to be skipped.

If the direction is not given, forward direction is assumed. If an error condition is encountered prior to completion, an error message is sent to the terminal. If n is not specified, the value 1 is assumed.

Example:

Assume that free form tape No. 2076 is to be positioned forward two files.

```
≤SPF FT#2076,2(RET)
```

SPE This command skips to the position following the last file on (Xerox) labeled, free form, or ANS labeled tape. The form of the command is

```
    ⎧LT⎫
SPE ⎨FT⎬#serial no.[-rt]
    ⎩AT⎭
```

where rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource.

Prior to issuing this command, the user must make sure that the tape is not write protected, i.e., the operator must be informed to insert a ring in the tape if it is a saved tape.

If an error occurs on the command and the command is from a command file, PCL exits to TEL after issuing the message.

<u>PCL ABORT</u>

Example:

Assume that labeled tape No. 5 is to be positioned past the last file on the tape so that additional files may be added.

```
≤SPE LT#5(RET)
```

WEOF WEOF writes an end-of-file. This is an end-of-file mark for free form tape units, !EOD for card or paper tape punches, or top-of-form for line printers. If no device is specified the current output device will be used.

```
        ⎡FT #serial no.[-rt]⎤
WEO[F]  ⎢LP                 ⎥
        ⎢CP                 ⎥
        ⎣PP                 ⎦
```

(Note that only one output file will be open at a time.)

REW This command rewinds the specified magnetic tape reel. It has the form

```
   ⎧⎡LT⎤#serial no.[-rt]           ⎫
REW⎨⎣FT⎦                           ⎬
   ⎩AT[#serial no.][-rt][/filename]⎭
```

where rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource. LT or FT must be specified if rt is specified as other than 7T.

Example:

Assume that magnetic tape reel No. 205 is to be rewound.

```
≤REW#205(RET)
```

MOUNT This command mounts a magnetic tape or disk pack. The form of the command is

```
         ⎧⎡LT⎤ #serial no.[-rt]               ⎫
MOU[NT]  ⎨⎣FT⎦                                ⎬[(RING)]
         ⎪AT #serial no.[-rt]/[file name]     ⎪
         ⎩DP #serial no.[-rt] [.account]      ⎭
```

where rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource and RING specifies that the device is to be mounted with write access.

Example:

Assume that tape reel #2075 is to be mounted with a write ring.

```
≤MOU #2075 (RING)(RET)
```

REMOVE This command removes a magnetic tape or disk pack no longer needed, thus releasing the drive or spindle for other purposes. The form of the command is

```
          ⎧⎡LT⎤#serial no.[-rt]            ⎫
REM[OVE]  ⎨⎣FT⎦                            ⎬
          ⎪AT[#serial no.][-rt][/filename] ⎪
          ⎩DP#serial no.[-rt][.account]    ⎭
```

where rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource.

If a tape is removed, the tape is rewound and a dismount message is sent to the computer operator. If a disk pack is removed, the user's interest in that spindle is released; however, no message is sent to the operator.

Example:

Assume that magnetic tape reel No. 2075 is to be rewound and removed.

```
≤REM#2075(RET)
```

TABS This command sets tab values to be used in conjunction with the TX (tab expansion) option. As many as 16 values may be specified. The form of the command is

```
TAB[S] s[,s]... .
```

where s is a column position to be used in expanding a line.

Example:

Assume that tabs are to be set for expansion in the standard Meta-Symbol list format.

```
≤TABS   10,19,37(RET)
```

TERMINATION OF PCL

PCL operations are terminated by the END command. This command returns control to TEL and has the format:

```
E[ND] or

X
```

Example:

```
_END (RET)

!
```

ERROR MESSAGES

PCL reports two types of error conditions. One type consists of the I/O error and abnormal conditions as listed in Appendix B. The other type consists of errors arising out of the use of PCL commands. These conditions are defined in Table 25.

A severity level of 1, 2, 3, or 4, is attached to each error and has the following effect on the execution of the command in question:

1. Warning

   PCL continues execution. The message will be printed only if a higher error severity level occurs during execution of a command.

2. Invalid Syntax or I/O Error

   This level terminates execution of the command but continues the syntax edit of the command for both on-line and batch operations.

3. Format Error

   This level terminates the command.

   In the case where a command is terminated (severity level 2 or 3), PCL reverts to the command state if the error occurs during on-line operations; it reads the next command card if the error occurs during batch operations.

4. Fatal Error

   This level causes PCL to abort the job.

Example:

Assume that a file is to be copied from RAD storage file A to the card punch. In entering the command, the device code for the RAD is entered as CC instead of DC.

```
<COPY CC/A TO CP (RET)
```

Error message printout:

ILLEGAL DEVICE CODE

PCL COMMAND SUMMARY

Table 26 is a summary of PCL commands. The left-hand column gives the command formats. The right-hand column gives the command function and options.

Table 25. PCL Error Codes

Hexadecimal Code	Message	Severity Level
10100	ARGUMENT GREATER THAN 31 CHARACTERS	2
10200	ILLEGAL IDENTIFICATION CODE	2
10300	INVALID REEL NUMBER SPECIFICATION	2
10400	ILLEGAL FILE NAME SPECIFICATION	2
10500	ILLEGAL ACCOUNT NUMBER SPECIFICATION	2
10600	ILLEGAL PASSWORD SPECIFICATION	2
10700	TOO MANY FIELDS IN A FILE IDENTIFICATION SPECIFICATION	2
10800	INVALID FILE RANGE SPECIFICATION	3
10900	MORE THAN TEN RS FIELDS[t]	2
10A00	VOLUME NUMBER BEYOND END OF SNS	2
10B00	ILLEGAL DECIMAL NUMBER	2

[t] RS signifies record selection.

Table 25. PCL Error Codes (cont.)

Hexadecimal Code	Message	Severity Level
10C00	CS ID-FIELD GREATER THAN FOUR CHARACTERS	2
10D00	ERROR ON N OR K VALUE OF CS OPTION	2
10E00	IMPROPER TERMINATION WITHIN RS, LN, OR CS OPTION	3
10F00	)) MUST TERMINATE RS, LN, OR CS OPTION	3
11000	SPECIAL ARGUMENTS MUST HAVE) AS TERMINATION CHARACTER	3
11100	EH?	3
11200	UNDEFINED COMMAND	2
11300	ILLEGAL INPUT DEVICE	3
11400	NO DEFINED OUTPUT DEVICE	3
11500	ILLEGAL OUTPUT DEVICE	2
11600	REEL NUMBER SPECIFICATION NOT VALID	2
11700	FILE SPECIFICATION NOT VALID	2
11800	DATA CODE SPECIFICATION NOT VALID	2
11900	MODE SPECIFICATION NOT VALID	2
11A00	SEQUENCE SPECIFICATION NOT VALID	2
11B00	RECORD SELECTION SPECIFICATION NOT VALID	2
11C00	PK BIN/7T COMBINATION NOT VALID	2
11D00	NULL ARGUMENTS (TWO DELIMITERS IN A ROW)	2
11E00	IMPROPER TERMINATION OF THE COMMAND	1
11F00	ONE REEL NUMBER MUST BE SPECIFIED ON THIS COMMAND	2
12000	'TO' 'INTO' OR 'OVER' NOT SPECIFIED	3
12100	RECORD SIZE EXCEEDS AVAILABLE MEMORY	3
12200	INVALID DEVICE TYPE FOR THIS COMMAND	3
12300	TOO MANY REEL NUMBERS SPECIFIED	3
12400	'TO' FILE EXISTS	3
12500	INVALID DIRECTION INDICATOR ON 'SPF' COMMAND	3
12600	INPUT RECORD SIZE LARGER THAN 32767 BYTES	3
12700	INVALID OPTION FOR THIS COMMAND	2
12800	TOO MANY SN, RD, WR, EX, UN SPECIFICATIONS	3
12900	RS SPECIFICATION BEYOND END OF FILE	2
12A00	ERROR IN COMPRESSED INPUT	3
12B00	PCL NEEDS AT LEAST TWO DATA PAGES TO RUN	4
12C00	TOO MANY ERRORS – PROCESS ABORTED	4
12D00	INVALID TAB SPECIFICATION	3
12E00	OVERFLOW ON EDIT LINE NUMBER	3
12F00	ZERO INCREMENTS ON CS OR LN OPTION	2
13000	TX OPTION USED WITHOUT TABS COMMAND	2

Table 25. PCL Error Codes (cont.)

Hexadecimal Code	Message	Severity Level
13200	CONFLICTING OR DUPLICATE OPTION	2
13300	MORE THAN 16 TAB VALUES	2
13500	TOO MANY CHARACTERS IN THE COMMAND	3
13600	INVALID VALUE FOR ANS OPTION	2
13900	TAPE DENSITY SPECIFICATION IS IN ERROR	2

Table 26. PCL Command Summary

Command	Description
C[OPY] sd[(s)][/fid[(s)][,fid[(s)]]...][;sd[(s)] [/fid[(s)][,fid[(s)]]...]]... TO OVER INTO dd[(s)][/fid[(s)]]	Copies file(s) between devices or between public storage and devices. Options: sd may be DC, CR, ME, operational label, stream-id, or: DP#serial no.[-rt] AT #serial no. [-rt] LT#serial no.[-rt] FT#serial no.[-rt] where rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource. s may be a data code (E, H); a data format (X, C); a mode (BCD, BIN, PK, UPK, SSP, DSP, VFC, NC, CR, FA, NFA, TX, DEOD, K, ASCI, EBCD, DEN); a sequence (CS, NCS, LN, NLN); an account (RD, WR, EX, UN); an ANS tape option (BLK, REC, FMT, CAT); an expiration time (EXP); or selection (x-y). dd may be DC, CP, LP, ME, operational label, stream-id, or: DP#serial no.[-rt] AT[#serial no.][-rt] LT[#serial no.][-rt] FT[#serial no.][-rt] where rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource.
COPYALL {[DC][.acct][(s)][/r] DP#reel-id[-rt][(s)][/r] LT#reel-id[-rt][(s)][/r]} [TO {DC[(a)] DP#serial no.[-rt][(a)] LT[#serial no.][-rt][(a)] FT[#serial no.][-rt][(a)] LP ME CP stream-id}]	Copies files from RAD, labeled tape, or disk pack to any output device. Options: s may be KEY, SEQ, RAN, PHY, and COPY input options. r is a range specification. rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource. a may be COPY output options.

Table 26. PCL Command Summary (cont.)

Command	Description
COPYSTD [DC/]fid LT#serial no.[-rt]/fid DP#serial no.[-rt]/fid [TO DC DP#serial no.[-rt] LT[#serial no.][-rt] FT[#serial no.][-rt] LP ME CP stream-id]	Copies a control file and all files named within the file. Option: rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource.
D[ELETE] [DC/] DP#serial no.[-rt]/ fid[,fid]...	Deletes the specified files. Option: rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource.
DELETEAL[L] [DC/] DP#serial no.[-rt]/ [range]	Deletes all files or a specified range of files. Option: rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource.
E[ND]	Returns control to the monitor.
ERR[ORS] SAV[E] REL[EASE] hhhhhh	Controls the disposition of aborted copy output. The default is RELEASE.
L[IST] [LT#reel-id[-rt][(s)][range] [DC][.acct][(s)] [range] DP#reel-id[-rt][(s)][range] LT#serial no.[-rt][(s)]/fid[(s)][,fid[(s)]]... fid[(s)][,fid[(s)]]... DP#serial no.[-rt]/fid[(s)][,fid[(s)]]... FT#serial no.[-rt][(s)]]	Lists file names and, optionally, attributes from the account directory, tape, or disk pack. Options: rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource. s may be A, EA, R, and Cn.
MOU[NT] [LT FT] #serial no.[-rt] AT#serial no.[-rt]/[file name] DP#serial no.[-rt]/[.account] [(RING)]	Mounts a magnetic tape or disk pack. Options: rt is the 2-character identifier of a device that was defined at SYSGEN as a resource. RING specifies that the device is to be mounted with write access.
PRINT	Sends accumulated symbiont output to the output device.
REM[OVE] [LT FT] #serial no.[-rt] AT[#serial no.] [-rt][/filename] DP#serial no.[-rt]	Removes a magnetic tape or disk pack. Option: rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource.

Table 26. PCL Command Summary (Cont.)

Command	Description
REV[IEW] {[DC/] DP#serial no.[-rt]/ .acct} [(s)] [range]	Reviews all or a specified range of files. Options: rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource. s may be A or EA (as in LIST command)
REW {[LT FT] #serial no.[-rt] AT[#serial no.] [-rt][/filename]}	Rewinds tape reel. Option: rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource.
SPE {LT FT AT} #serial no.[-rt]	Spaces to the end of the last file on (CP-V) labeled, free form, or ANS labeled tape. Option: rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource.
{SPF SPR} [FT]#serial no.[-rt][,[±]n]	Positions free form tape forward or backward a designated number of files (SPF) or records (SPR). Option: rt is the 2-character identifier of a device that was defined at SYSGEN to be a resource.
TAB[S] s[,s]...	Sets tab values for tab expansion.
WEO[F] [FT#serial no.[-rt] LP CP DP]	Writes an end-of-file on the current output device.

6. EDIT

INTRODUCTION

Edit is a line-at-a-time context editor for on-line creation, modification, and manipulation of files of EBCDIC text. All Edit data is stored on disk in a keyed file structure of sequence-numbered variable-length records, which permits Edit to directly access each line or record of data. Edit functions are controlled via single-line commands from the user. The command language provides for the following:

1. Creating a sequenced EBCDIC coded text file.
2. Inserting, reordering, and replacing lines or groups of lines of text.
3. Selective printing and renumbering.
4. Reordering groups of records within a file.
5. Merging part of one file into another.
6. Context editing operations that allow matching, moving and substituting character strings within a specified range of text lines.
7. Maintaining files (allowing the user to build, copy, and delete whole files of text lines).

A user may edit files under his own account (i.e., the one under which he logged on) or under accounts to which he has been granted write access by the file creator. Attempting to edit a file in another account for which the user has read access but not write access or one that is not keyed will result in the file being opened for input only. The user will be notified that he is in this mode by a message

```
--FILE OPEN FOR INPUT ONLY; CANNOT
  UPDATE
--FILE NOT KEYED; CANNOT UPDATE
```

This allows those commands that do not alter a file to be executed normally (e.g., TY, FT, etc.). Any command that normally alters a file will be aborted and the file will not actually be changed.

A user may copy his own files or those to which he has read access. Under the rules of CP-V file access, a file may not be created (i.e., built or copied to) under an account number different than that used for log-on.

In using Edit, it must be stressed that the editing takes place as the commands are given; the file is edited in place. Therefore, a backup file should be kept to protect against user or machine errors.

An on-line user may call the Edit processor either directly,

```
!E[DIT]
```

or indirectly through one of two executive-level commands:

```
!E[DIT] fid        (edit an existing file)
!E[UILD] fid       (build new file)
```

CALLING EDIT

The first executive-level command allows the user to call Edit for updating an existing file. Edit first opens the specified file and then prompts for command input by typing its identifying mark, the asterisk (*). The second executive-level command allows the user to call Edit for on-line creation of a text file. Edit opens the specified file and prompts for command input by typing the first line number at the left margin of a fresh line. The user is expected to enter the text lines of the new file.

If an Edit command is given at the executive level without a file identifier, Edit types EDIT HERE and prompts for further commands by typing an asterisk (*).

Edit may also be called via the M:LINK system procedure. The following assembly language segment calls Edit and instructs Edit to process file ABC as Edit commands.

```
                M:LINK 'EDIT', ':SYS', (CMD, EDITCMD)
                  .
                  .
                  .
 EDITCMD TEXTC    'EDIT XEQ ABC'
```

When Edit encounters an END command in file ABC or the end of file ABC, it returns to the calling program.

RECORD FORMATS

The editing process is based on a sequence number associated with each line. Unsequenced files of text lines may be sequenced via the Edit COPY command. Sequence numbers for inserting new lines may be generated automatically by Edit or may be supplied by the user.

Sequence numbers consist basically of an integer and three fractional digits. However, the user may write a sequence number with one or more fractional digits omitted and Edit will automatically assume sufficient trailing zeros to complete the sequence number. For example:

Sequence Number	Implies
50	50.000
50.01	50.010
50.5	50.500
50.008	50.008

Edit writes variable-length records, with a maximum record size of 256 characters including the(RET). Trailing blank characters in a record are not written on the files unless the RP ON command has been given.

Edit files are stored on disk as keyed records, with the keys being binary representations of the sequence numbers. The sequence number DDDD.DDD is taken as a seven-digit decimal integer and converted to binary, giving a key with a length of three bytes. For example, the following record created in a BUILD operation would have a key value of 8000_{10} and a record length of 20 bytes (assuming that (RET) is in column 20):

```
8.000     B2     LI,5     0(RET)
```

If the (RET) is preceded by a number of blanks, they will not be carried in the output unless the RP ON command has been given. The record terminator can be either (RET) or (LF) and is carried in the record as X'15' if the CR ON command has been given. (Both RP and CR have the default OFF.)

MULTILINE RECORDS

On a terminal unit having an inherent line-width limit of less than 140 (e.g., Teletype models 33, 35, and 37), a single, multiline record may be entered into a file (using the BUILD or IN commands, for example) in either of two ways:

1. Using the local carriage return key marked LOC CR, if present, to "break" the input line without releasing it to the system.

2. Using the simulated local carriage return sequence (ESC) (RET) for the same purpose.

Either method permits entering a record of up to 139 characters plus (RET) on virtually any terminal unit.

An example of a multiline record is presented in Figure 3.

Note that editing a file with records greater than 140 characters does not alter the maximum number of characters that may be input or output at the terminal unit. The COC routines truncate terminal records at 140 characters. However string substitutions etc. may be made up to column 255. This allows editing of files created by other processors.

BREAK FUNCTION

The BREAK key always causes an immediate interruption in Edit activity, with any partially completed input and output being discarded. ESC B has the same effect except that waiting output will be printed. Edit stops any command in progress and reverts to accepting command input from the user.

If commands are being accepted from an Edit XEQ file, the current command will be displayed in XEQ ECHO format. See the Edit XEQ command.

For commands that product no display while operating on a range of records, the point of interrupt is reported by a message which denotes the sequence number(s) of the record(s) being processed at the time of the interrupt. Edit then types this message

```
--X TO ABORT
```

and prompts for input. The user may enter an X followed by a (RET) to abort the operation, or any other character (or none) and a (RET) to continue with the operation. If commands are being accepted from an Edit XEQ file and the command is aborted, command input will revert to the terminal.

If a command is being executed and the BREAK key causes an interrupt during an I/O operation (e.g., READ, WRITE, OPEN, DELETE record), the I/O operation is completed. After the I/O is completed, the user may continue execution of the command to normal conclusion or may immediately terminate the command. With record or intrarecord commands (see Command Structure), the current Edit file remains open. All file commands terminate by closing all files.

LINE FEED FUNCTION

If only a line feed or GS (usually control-shift-M) is entered and an SE range has previously been specified, the SE range will be reset to the first record after the current SE range and the record will be typed in the format specified by the last RR, TC, TS, or TY command entered. If no such commands have been entered, the record wili be typed in the TY command format.

Line number 4.000 is input as a multiline record in the following manner:

```
 4.000 THIS IS AN EXAMPLE OF A MULTILINE(ESC)(RET)
RECORD. A RECORD CAN CONTAIN UP TO 140 (ESC)(RET)
CHARACTERS INCLUDING THE CARRIAGE RETURN.(RET)
```

If this record were displayed by Edit, it would appear as

```
  4.000 THIS IS AN EXAMPLE OF A MULTILINERECORD. A RECORD CAN CONTAIN U
P TO 140 CHARACTERS INCLUDING THE CARRIAGE RETURN.
```

Note that the user did not type a space after the word 'multiline' and that Edit did not assume a space. Also, the system "folds" the record indiscriminately when the physical line width limit is reached.

Figure 3. A Multiline Record

If a line feed only is entered when in step mode (SS or ST command), the effect is the same as entering the NO command.

MULTILINE COMMANDS

A command line can be continued by typing a semicolon as the last character on the line. The last command on the line must be complete and must normally terminate with a semicolon. The continuation line is prompted with >*.

The following examples are identical in function:

```
*/A/S/B/;C/S/D/;TS
*/A/S/B/;
>*C/S/D/;TS
```

The following examples are illegal:

```
*SE1-99,/A/OR;
*/A/S/;
```

UP-ARROW FUNCTION

If only an up-arrow (↑ or ∧) followed by a carriage return or an EOT (control D) is entered and an SE range has previously been specified, the SE range will be reset to the first record before the current SE range and the record will be typed in the format specified by the last RR, TC, TS, or TY command entered. If no such commands have been entered, the record will be typed in the TY command , format.

EDIT COMMANDS

COMMAND STRUCTURE

Edit commands fall into the following three categories:

1. File commands: Commands that apply to an entire file. These commands may be given at any time.
2. Record commands: Commands that act upon one record or a group of records within a file. These commands may be given only after a file has been selected for editing.
3. Intrarecord commands: Commqnds that make changes within an individual record. These commands generally manipulate character strings and may be given only after a specific set of records has been selected by a command of type 2, above (either the SE, SS, or ST command).

FILE COMMANDS

The file commands will be discussed in the following order:

EDIT	Select file for editing.
BUILD	Create a new file.
COPY	Copy file 1 to file 2.
DELETE	Delete file.
MERGE	Merge files.
XEQ	Execute commands from a file.
ECHO	Set XEQ echo mode.
CRPT	Set data encryption seed.
END	Exit to executive.
X	Exit to executive.
CR	Set carriage return mode.
TA	Set tab positions.
TABX	Set tab expansion mode.
RP	Set record size preservation mode.
BP	Set blank preservation mode.
DL	Limit Multi-record Deletes.
L	List.

EDIT Edit File

EDIT opens a file to be edited and has the format:

E[DIT] fid

The EDIT command must be used to enter the record editing mode and to identify the file that is to be edited.

Use of any of the following commands terminates the record editing mode: BUILD, DELETE, MERGE, and COPY. If an EDIT command is given while in the record editing mode, the previously open file is closed and the specified file is opened. In both situations, the following message is printed by Edit:

```
..EDIT STOPPED
```

Edit then processes the new command.

BUILD Build New File

The BUILD command enables the user to create a new file. The command may be given at the TEL level with the form

B[UILD] fid

or it may be given at the Edit processor level with the form

B[UILD] fid[, [n][, i]]

where

fid	is the identifier of the file to be created.
n	is the sequence number at which the new file is to start. The default value is 1.
i	is the value by which sequence numbers for the new file are to be incremented. The default value is 1.

The system prompts by typing a sequence number, and the user then types in the corresponding line. A null line (indicated by RET alone) terminates the buildoperation and closes the file. If the BUILD command is used at the executive level, control returns to the executive level after typing a null record (RET alone). If the BUILD command is invoked while in the Edit processor, control returns to the Edit processor after typing a null record.

Example:

```
*BUILD SOFILE RET
1.000              SYSTEM    SIG5 RET
```

```
2.000        DEF    B (RET)
3.000        REF    A (RET)
4.000        B      A (RET)
5.000        END (RET)
6.000 (RET)         The null record, consisting of
*                   only a carriage return, termi-
                    nates the command and does not
                    appear in the output file.
```

COPY Copy File

COPY causes Edit to copy a specified file and has the format:

$C[OPY]\ fid_1 \begin{Bmatrix}ON\\OVER\end{Bmatrix}\ fid_2[,[n][,i]]$

where

fid$_1$ identifies the file that is to be copied.

fid$_2$ identifies the file to which fid$_1$ is to be copied.

n is the starting sequence number for the new file. If omitted, the old sequence numbers of fid$_1$ are retained in the copy.

i is the sequence number increment for the new file. The default value is 1.

If ON is specified, a new file is created (and must not already exist). If OVER is specified, fid$_2$ may exist; and if it does, it will be deleted and replaced by the copy of fid$_1$.

Example 1:

```
*COPY  PROG1  ON  PROG2(RET)
..COPYING
..COPY DONE
```

(This example copies file PROG1 onto a new file PROG2. The two files will have the same record keys.)

Example 2:

```
*COPY FILEA OVER FILEA, 1, 1(RET)
..COPYING
..COPY DONE
```

(This example copies FILEA over itself, giving FILEA record sequence numbers beginning with 1 and incremented by 1.)

A short form of the copy command is available for copying a file over itself, and has the format:

C[OPY] fid[,n[,i]]

where

fid identifies the file to be copied over itself.

n is the starting sequence number for the new file. If omitted, a value of 1 will be used for n and i.

i is the sequence number increment for the new file. The default value is 1.

Example 3:

```
*C FILEA(RET)
..COPYING
..COPY DONE
```

(This example provides the same function as Example 2.)

When a file is copied over itself, the READ and WRITE account numbers are preserved.

DELETE Delete File

DELETE causes Edit to delete a specified file from the log-on account. The DELETE command has the format shown below.

D[ELETE] fid

Example:

```
*DELETE   PROG1(RET)
..DELETED       The file has been deleted.
```

MERGE Merge Files

MERGE causes Edit to transfer records between specified files. The MERGE command has the form shown below.

M[ERGE] fid$_1$[,n$_1$[-n$_2$]] INTO fid$_2$,n$_3$[-n$_4$][,i]

Records n$_1$ through n$_2$ from file fid$_1$ are merged into file fid$_2$ where they replace records n$_3$ through n$_4$. In the target file the new records are numbered from n$_3$ in steps of i. The source file, fid$_1$, must be keyed format or else Edit aborts the command. If no range specification is attached to fid$_1$, all of its records are subject to the move. If a range specification exists, Edit checks that at least one record is contained in it.

Example:

*MERGE	fid$_1$	INTO	Merges all of fid$_1$.
*MERGE	fid$_1$, 10	INTO	Merges record 10.000 of fid$_1$.
*MERGE	fid$_1$, 10-12.5	INTO	Merges records 10.000 through 12.500.

After validity checks are made on fid$_1$, Edit checks for the existence of fid$_2$. If fid$_2$ does not exist, Edit creates a file identified by fid$_2$ and then moves the appropriate record set from fid$_1$ into fid$_2$, resequencing from n$_3$ and incrementing by i. (If no value for i is specified, the value is 1 by default.) This operation is similar to a COPY operation, except for the selection of records from fid_1. If fid_2 exists, Edit deletes from it all records in the range n_3-n_4 and then replaces them with the appropriate records from fid_1, starting at sequence n_3 and incrementing by i.

Example:

```
*MERGE ALPHA. ACCT1,100-120 INTO BETA, 400-440(RET)
..MERGE STARTED
--DONE AT 420      420 is the last sequence number as-
                   signed in BETA.
```

If (when fid_2 exists) the number of records to be transferred at the specified increment causes Edit to equal or exceed the next higher existing sequence number above the destination range n_3-n_4, the merge is stopped with the message

```
--CUTOFF AT n5(n6)
```

where

n_5 is the last sequence number assigned in fid_2.

n_6 is the sequence number of the last record moved from fid_1.

The user may then give subsequent commands to investigate how to move the remaining records.

XEQ Execute Commands From A File

XEQ causes Edit to obtain commands from a file instead of from the terminal. The commands will be displayed as they are processed, if desired. (See the Edit ECHO command.) Commands will be processed from the file until the end of the file is reached or the user aborts the command via the BREAK function; command input then reverts to the terminal.

The XEQ command has the format shown below.

```
XEQ fid
```

where fid is the identifier of the file containing the Edit commands.

If ECHO is on, each command will be displayed as it is processed in the format shown in the example below.

```
*XEQ ABC (RET)
XEQ:     1.000 TY20.01

  20.010 ASDFJKL
*
```

ECHO Set XEQ Echo Mode

ECHO sets the XEQ echo mode on or off and has the format:

$$\text{ECHO}\begin{Bmatrix}\text{ON}\\ \text{OFF}\end{Bmatrix}$$

If turned on, commands accepted from an Edit XEQ file will be displayed in XEQ format. (See the Edit XEQ command description.)

CRPT Set Data Encryption Seed

CRPT causes Edit to set or reset the data encryption seed. When data encryption is set, the seed is used by the CP-V monitor to encrypt and decrypt all data read from and written to the files being used by Edit, except the XEQ file. The same seed that is used when the file is created must be used for subsequent file accesses. The seed must be remembered by the user; there is no way to later determine what seed was used in creating or updating a file. The correct seed must also be used when copying a file or the data may be destroyed. The CRPT command has the format shown below.

```
CRPT [seed]
```

where seed is from 1 to 8 hexadecimal digits long. If seed is omitted, data encryption is turned off.

END Exit Edit

END causes Edit to close all active files and return control to the terminal executive language (TEL). The END command has the format:

```
END
```

Example:

```
*END(RET)
!      Any TEL command may now be given.
```

X Exit Edit

The X command performs exactly the same function as an END command and has the format

```
X
```

CR Set Carriage Return Mode

The CR command controls the inclusion of the CR (X'15') character at the endof each record in the user's output file. The CR command has the form shown below.

$$\text{CR}\begin{Bmatrix}\text{ON}\\ \text{OFF}\end{Bmatrix}$$

where

ON includes the X'15' terminator in the user's output file.

OFF excludes the X'15' terminator from the user's output file and is the default setting.

The carriage return is normally not included since this is provided by the COC routines. However, if the user wishes to reproduce the file on cards or tape (for later use by other than CP-V software), he may want the carriage return. Inclusion of the carriage return character will have no effect on the typing of records on the terminal, however.

The CR command may be given at any time except during "set and step" operations. (Set and step operations are controlled by the Edit SS and ST commands described later.)

TA Set Tab Positions

TA causes Edit to set or reset the terminal tab stops and has the format:

$$\text{TA}\begin{Bmatrix}\text{F}\\ \text{M}\\ \text{S}\\ \text{C}\end{Bmatrix}$$

where

F implies FORTRAN and tabs set at columns 7, 16, and 34.

M implies Meta-Symbol and tabs set at columns 10, 19, and 37.

S implies Meta-Symbol, short-form, and tabs set at columns 8, 16, and 30.

C implies COBOL and tabs set at columns 8, 12, and 36.

These tab settings correspond to record column numbers and are offset to provide for the line number produced at the left margin of the user terminal. The TA command may be given at any time except during "set and step" operations. (Set and step operations are controlled by the Edit SS and ST commands described later).

When the programmer uses the terminal to build a file, he can columarize the instructions as if he were typing them on a coding sheet. However, unlike the TAB key on most typewriters, the TAB key on many terminals does not move the carriage across the page. Therefore, a CP-V service is provided to simulate tabbing action when the TAB key is struck. To achieve simulation the user must do these things:

1. Tell the system where the tab stops are by using the executive command TABS or the Edit command TA.
2. Be sure tab simulation is on to cause the appropriate number of spaces to be sent on output and echoed on input whenever a tab character is detected. (Tab simulation is discussed in Chapter 10.)
3. Set space insertion mode. If space insertion mode is on, an appropriate number of spaces will be inserted into the input record. If space insertion mode is off, the tab character (X'05') will be inserted into the input record. (Space insertion mode is discussed in Chapter 2.)

If space insertion is off, Edit puts the actual tab character (X'05') into the file being constructed whenever the TAB key is struck, regardless of whether simulation is carried out.

If tab expansion is turned off, tab characters are not given any special treatment (see Edit TABX command). Otherwise when using intraline commands to edit text that contains tab characters, the user must give a TA or TABS command so that Edit will know how to interpret the tab characters it finds. Edit then uses this information to expand the records by inserting an appropriate number of blanks for each tab character it finds. (See the discussion of the blank preservation command, BP, later under "Intrarecord Editing Commands".) If tab stops have not been set by a TA or TABS command and Edit finds a tab character, the user is notified with the message

-TAB CHARACTER FOUND. NO TAB STOPS SET.

When editing a file containing tab characters, care should be taken to set the tabs to the same columns as were used when the file was created.

TABC Set Tab Compression Mode

The TABC command sets the tab compression mode on or off and has the format:

$$\text{TABC}\begin{Bmatrix}\text{ON}\\ \text{OFF}\end{Bmatrix}$$

The default is off.

When TABC mode is on, blank fields in all new or modified records written to the file are replaced with tab characters, based on the tab stops in effect. TABX mode is effectively forced on. When TABC mode is off and TABX mode is on, tab compression is performed on modified records.

TABX Set Tab Expansion Mode

The TABX command sets the tab expansion mode on or off and has the format:

$$\text{TABX}\begin{Bmatrix}\text{ON}\\ \text{OFF}\end{Bmatrix}$$

The default is on.

When tab expansion mode is off, tab characters read from the file will not be replaced with blanks, and blank fields will not be replaced with tab characters before being written to the file.

The TABX OFF command will reset space insertion mode, and TABX ON will set space insertion mode. When the user exits EDIT, space insertion mode will be reset to the mode in effect when EDIT was first called. (Space insertion mode is discussed in Chapter 2.)

RP Set Record Preservation Mode

The RP command sets the record preservation mode on or off and has the format:

$$\text{RP}\begin{Bmatrix}\text{ON}\\ \text{OFF}\end{Bmatrix}$$

The default is OFF.

When record preservation mode is off, records are shortened or lengthened as necessary when they are edited and trailing blanks are deleted.

When record preservation mode is on, all records are maintained at their current record size. Trailing blanks in records are not truncated. Strings added or moved beyond the end of record are truncated. Strings deleted or shifted left cause blank fill to the end of record if necessary. Records which are replaced entirely (via move, merge, insert, etc.) take the length of the replacement records.

RP is a file command and should be specified if desired before any editing is begun.

BP Set Blank Preservation Mode

BP sets the blank preservation mode on or off and has the format shown below.

$$\text{BP}\begin{Bmatrix}\text{ON}\\ \text{OFF}\end{Bmatrix}$$

When "on", all strings of blanks are preserved during intrarecord operations. When "off", blank strings are compressed to a single blank or expanded as required to retain column alignment of nonblank fields. The default mode is "off".

When a string is inserted or replaced in a manner that changes the number of characters in a record, the record format is adjusted as follows.

When the blank preservation mode is off, the blanks between two successive strings are not preserved. When a string operation causes the first of two strings to be expanded or contracted, the number of blanks between the two strings are decreased or increased so that the second string stays in the same columnar position. (If the first string expands, the number of blanks between the two strings decreases; if the first string contracts, the number of blanks increases.) At least one blank must be left between strings.

When the blank preservation mode is on, the blanks between the two strings are preserved. That is, when the first string expands or contracts, the second string is moved to the left or right so that the same number of blanks remains between the two strings.

For example, the following string substitution command

```
*/8/S/LINK/(RET)
```

substitutes the string "LINK" for the string "8" in the instruction

```
$ 10   BAL, 8   SUB
```

adjusting blanks as indicated below:

old new (BP-OFF) new (BP-ON)	$10 BAL,8 SUB $10 BAL,LINK SUB $10 BAL,LINK SUB

The BP command may be given at any time except during "set and step" operations. (Set and step operations are controlled by the Edit SS and ST commands described later.)

DL Limit Multi-Record Deletes

The DL command limits the number of records that EDIT will delete during a DE, MK, or MD command and has the format shown below.

DL n

where n is the number of records to be deleted.

L List

The L command invokes PCL's LIST command, and has the same format. (See the PCL LIST command.)

RECORD EDITING COMMANDS

The record editing commands may only be given after a file has been opened for editing via the EDIT command. If the user does not open a file for editing before giving a record editing command, Edit prints the message.

-NO FILE NAMED

The record editing commands will be discussed in the following order:

IN IP IS	Insert records.
DE	Delete records.
AD	Add to end of record.
RR	Reread record.
TY TC TS	Type individual records.
MD MK	Reorder records within a file.
FD	Delete records containing a specified character string.
FT	List sequence numbers and contents of records containing a specified character string.
FS	List sequence numbers of records containing a specified character string.
RN	Renumber record.
CM	Insert commentary.
CT	Type and insert commentary.
SE	Select a group of records for character operations.
SS ST	Select records for step mode operation.

IN Insert New Records

IN causes Edit to insert new records into a file. The IN command has the format shown below.

IN [n] [,i]

New records are inserted starting at the record with sequence number n, with each successive record being sequenced from n with increment i. (If i is omitted, the increment size specified in the most recent record editing command is used. If no such commands have been given, the value 1 is assumed by default. If n is omitted, the starting record is calculated by incrementing by i the sequence number of the last record of the most recently established intrarecord selection range.) If a record with sequence number n exists in the file, it is replaced by the newly inserted record n.

Edit prompts the user console with the first sequence to be inserted, and repeats the prompt for each subsequent insertion, increasing the sequence number by the increment i.

The insertion can be terminated in one of three ways. If a null record ((RET) only) is supplied, the insertion terminates. An equivalent action takes place if an incremented sequence equals or exceeds a sequence existing in the file. In the latter case, the console bell is rung. The insertion is also terminated (and an error message is printed) if an attempt is made to insert a record having a record number greater than 9999.999. In any case, EDIT enters the intrarecord mode as though an SE command for the last inserted record had been given.

Example:

```
*EDIT SOURCEFILE (RET)
*IN 100, .1 (RET)
100.000     10 A = 2.5(RET)  Replaces the existing record.
100.100        B = 0.(RET)
*                            Record insertion terminates
                             because sequence number
                             100.200 existed previously;
                             the console bell is rung.
```

IP Insert New Records, Protected Mode

The IP command is identical to the IN command except that if a record with sequence number n exists in the file the command is immediately aborted with the message:

```
-P1:REC EXISTS
```

The format of the IP command is:

```
IP[n][,i]
```

IS Insert New Records

The IS command is identical to the IN command in function and format except that Edit prompts with a period (.) instead of with sequence numbers. The format of the IS command is:

```
IS[n][,i]
```

Example:

```
*EDIT SOMEFILE
*IS 100, .1
.10 A = 2.5
.    B = 0
*
```

DE Delete Records

DE causes Edit to delete all records whose sequence numbers lie in a specified range. The DE command has the form shown below.

```
DE n[-m]
```

where

n specifies the number of the first record to be deleted.

m specifies the number of the last record to be deleted. If m is omitted, only record n is deleted.

Example:

```
*DE 50 (RET)          Deletes record 50.000 only.

*DE 50-60.5 (RET)     Deletes all records in the range
                      50.000 through 60.500, inclu-
                      sive.
```

AD Add To Record, Sequence Number Included

AD causes Edit to type the contents of each record in the range of n to m. The carriage will remain positioned at the end of each record. The user may then enter text to be placed after the end of the original record.

The AD command has the format shown below.

```
AD n[-m]
```

If m is omitted, only record n is processed.

RR Re-read Record, Sequence Number Included

RR causes Edit to type the contents of each record in the range n to m, and allow the user to modify each record using the editing functions described in the section "Typing Lines" of Chapter 2. A carriage return should be entered when the editing is complete. If the user enters a null line (e.g., types CONTROL X followed by a carriage return) the record will not be changed. (This command makes use of the COC routine's REREAD function.) The RR command has the format shown below.

```
RR [n[-m]]
```

If m is omitted, only record n is processed. If n and m are omitted, RR is processed as an intrarecord command.

TY Type Records, Sequence Numbers Included

TY causes Edit to type the sequence numbers and the contents of specified columns of one or more records. In addition, it causes Edit to enter the intrarecord mode as though an SE command had been given. The TY command has the format shown below.

```
TY [n[-m][,c[,d]]]
```

Edit type records in the range n to m, and types only the portions between columns c and d. If m is omitted, only record n is typed. If n and m are omitted, TY is processed as an intrarecord command. If the values for c and d are not given, c has a value of 1 and d has a value of 140 by default.

Example:

```
*EDIT SOURCEFILE (RET)
*TY 1-2,4,8 (RET)
1.000    EQU
1.200    SYST
1.400    REF
1.600    DEF
1.800    PAGE
2.000    ITIAL
*
```

TC Type Compressed

TC causes Edit to type the sequence numbers and the contents of specified columns of one or more records. Any nonblank strings within the columns are shifted to the left to compress each blank string to a single blank. This compression affects only the typed output; the records themselves are not affected. TC is the same as TY with all blank strings compressed to a length of one. Like TY, TC causes Edit to enter the intrarecord mode as though an SE command had been given. The TC command has the format:

TC [n[-m][,c[,d]]]

Edit types records in the range n to m, and types only the portions between columns c and d. If m is omitted, only record n is typed. If n and m are omitted, TC is processed as an intrarecord command. If the values for c and d are not given, c has a value of 1 and d has a value of 140 by default.

Example:

```
*EDIT SOURCEFILE (RET)
*TC 1-2,1,7 (RET)
1.000 A EQU
1.200  SYS
1.400 B REF
1.600 C DEF
1.800 PAGE
2.000 *INITIA
*
```

TS Type Records, Sequence Numbers Not Included

TS causes Edit to type the contents of specified columns of one or more records, without accompanying sequence numbers. In addition, it causes Edit to enter the intrarecord mode as though an SE command had been given. The TS command has the format shown below:

TS[n[-m][, c[, d]]]

Edit types records in the range n to m, and types only the portions between columns c and d. If m is omitted, only record n is typed. If n and m are omitted, TS is processed as an intrarecord command. If the value of c and d are not given, c has a value of 1 and d has a value of 140 by default.

Example:

```
*EDIT SOURCEFILE (RET)
*TS 1-2,1,8 (RET)
A    EQU
     SYS
B    REF
C    DEF
     PAC
*INITIA
*
```

MD Move and Delete Records

MD causes Edit to move records from one specified range to another. The original records are deleted as they are moved. Records in the destination range are also deleted. The MD command has the form shown below.

MD n[-m] , k[-p][, i]

where

n specifies the sequence number of the first record that is to be moved and deleted.

m specifies the sequence number of the last record that is to be moved. If omitted, only n is moved and deleted.

k specifies the lower limit (i.e., sequence number) of the range of destination records that will be deleted.

p specifies the upper limit of the range of records to be deleted. If omitted, only k is deleted. However, records from the range n-m are still moved to record k and following until a record is encountered, that originally followed record k in the file. (When such a record is encountered, no more records are moved.)

i specifies the increment value to be used for renumbering records. If omitted, the most recent increment value specified in a record edit command is used. If no such commands have been given, the default value is 1.

The first record (n) is renumbered as k. Successive records from the range n-m are renumbered consecutively higher, incremented by i.

It is important to note that the ranges n-m and k-p may not overlap.

As each record from the range n-m is moved, it is deleted from the original range (n-m). At the end of this operation, a message is printed specifying the new sequence number of the last record moved from the range n-m and the total number of records moved.

Example:

```
*EDIT BETA
*MD 5-21,  100-101,  .02
--DONE AT 100.32
     17 RECORDS MOVED
```

If the increment is too large to permit all records in the range n-m to be moved into the space between k and the next record after p, a message is printed specifying the total number of records deleted and the sequence numbers, from both ranges, of the last record moved. In this case the original contents of range k-p will be lost, but only those records in the range n-m that have actually been moved will have been deleted. Thus, the user can perform another move (with a smaller increment) to move the remaining records in the range n-m.

Example:

```
*EDIT BETA
*MD  10-30, 100-110,  1
      10 RECORDS DELETED
--CUTOFF AT 110.  (20.)      20 is the number of the
                              last record that was moved.
```

MK Move and Keep Records

MK is identical to MD except that the records in the range n-m are not deleted as they are moved; thus a copy of records in the range n-m is made. The MK command has the form shown below.

```
MK n[-m],k[-p][,i]
```

FD Find and Delete Records

FD causes Edit to search for a specified string or specified strings between specified columns. If the search is satisfied, the record containing the string(s) is deleted from the file. The FD command has the form shown below.

```
FD n[-m],{/string/}[,c[,d]]
         { sse     }
```

where

n specifies the sequence number of the first record to be searched.

m specifies the sequence number of the last record to be searched. If omitted, only record n is searched.

/string/ specifies the character string identifying the record to be deleted.

sse specifies the string selection expression and is described in the SE command description.

c specifies the lower limit (i.e., column number) of the field to be searched. The default value is 1.

d specifies the upper limit of the field to be searched. The default value is 140.

The specified string must be entirely contained within columns c through d to cause deletion. At the end of this operation, a message is printed telling how many records were deleted.

```
*EDIT FILEA
*FD  5-20.4,/DATA/, 10, 18
--006 RECS DLTED
```

If there are no records in the specified range containing the indicated string, Edit prints the following message:

```
--NONE
```

FT Find and Type Record and Sequence Number

FT causes Edit to search for a specified string or specified strings between specified columns. If the search is satisfied, Edit types out the sequence number and the contents of the record. (The string must be entirely contained within the specified columns.) The FT command has the format:

```
FT n[-m],{/string/}[,c[,d]]
         { sse     }
```

The parameter specifications are the same as those for the FD command.

Example:

```
*EDIT SOMEFILE
*FT  1-100,/LW/,10
5.000           LW,3          DATA
9.000           LW,2          TABLE,7
21.480          LW,10         LOC+5,8
73.000          LW,9          FLAG
*
```

If there are no records in the specified range containing the indicated string, Edit prints the message

```
--NONE
```

Find and Type Sequence Number

FS causes Edit to search a given range of records for a specified character string or specified strings between designated columns. Edit will type the sequence number of each record satisfying the search criteria. The FS command has the format:

$$\text{FS}\ \ n[-m],\begin{Bmatrix}/\text{string}/\\ \text{sse}\end{Bmatrix}[,c[,d]]$$

The parameter specifications are the same as those for the FD command.

Example:

```
*EDIT SOMEFILE (RET)
*FS  10-20,/BE/, 10, 11 (RET)
15.000
18.000
*
```

If there are no records in the specified range containing the indicated string, Edit prints the following message:

```
--NONE
```

RN Renumber Record

RN causes Edit to renumber a specified record. The RN command has the form shown below.

```
RN n, k
```

This has the same effect as deleting record n and then entering a new record with sequence number k with the same contents as n. Sequence number k must not already exist.

CM Insert Commentary

CM causes Edit to insert commentary into specified columns of each successive record beginning at a specified sequence number. The CM command has the format shown below.

```
CM n, c
```

where

n is the record number.

c is the column number.

The sequence number of each record is typed and then the user types in the data he wants inserted starting at column c. The data he types in is blank filled to the right through column 140, as required. A null record terminates the command. It is not necessary to delimit commentary with slashes.

CT Type and Insert Commentary

CT causes Edit to insert commentary into specified columns of each successive record in the same manner as the CM command except that the contents of the record up to the specified column is typed, along with the sequence number, prior to accepting the user's data. The CT command has the format:

```
CT n, c
```

The parameters and functioning of the command are the same as those for the CM command.

Example:

```
*EDIT SOURCEFILE (RET)
*CM 37.6, 40 (RET)
37.600  *  COMMENT 1 (RET)
37.800  *  COMMENT 2 (RET)
40.500  * (RET)
*CT 50, 40 (RET)
50.000 ALPHA LW, 1 BETA   *COMMENT 3 (RET)
50.100          AI, 1  X'15'  *  (RET)
*
```

SE Set Intrarecord Mode

SE causes Edit to accept successive lines of intrarecord commands. The SE command has the format shown below.

```
SE n[-m][,sse][,c[,d]]
```

Each input line of intrarecord commands is applied, in order, to columns c through d of every selected record in the range n through m. If m is missing, only record n is processed. The default values for c and d are 1 and 140, respectively.

The SE command may also be used with no parameters, which is equivalent to the command:

```
SE 0-9999.999.
```

sse is a string selection expression of the format:

$$[\text{NOT}]/\text{string}/\ .\ .\ .\ \left[\begin{Bmatrix}\text{AND}\\ \text{OR}\\ \text{EOR}\end{Bmatrix}\begin{bmatrix},\text{NOT}\\ \not{b}\text{NOT}\end{bmatrix}/\text{string}/\right]\ \ldots$$

A record is selected only if the string selection expression yields a logical "true" result, defined by the following:

1. /string/ is a string selection operator, and will yield a true result if the specified string is present in the record. String selection operators may also include hexadecimal representations of EBCDIC character values, as described under "INTRARECORD EDITING COMMANDS".

2. NOT/string/ is a string selection operator, and will yield a true result if the specified string is not present in the record.

3. There is an accumulator that is initially set to the logical state of the first string selection operator. As each binding operator (AND, OR, or EOR) is encountered, it is evaluated with its string selection operator and the accumulator as inputs, and the accumulator is updated. If AND is specified, both the accumulator and the string selection operator must be true for the accumulator to result in the true state.

4. If OR is specified, either the accumulator or the string selection operator being true will result in setting the accumulator to the true state.

5. If EOR is specified, the accumulator will be set true if the accumulator or the string selection operator is true, but not if both are true.

If a record is not selected, it is skipped and processing resumes with the next record in the range of n through m.

If several commands are entered on one line, all commands on the line are executed on one record before the next record is processed. The first occurrence of a file or record-editing command terminates the effect of the SE command. All commands executed in the intrarecord mode apply only to the strings lying entirely within columns c through d.

SE may be used on the same input line with other intrarecord commands, but when so used, it must be the first command on the line.

The following example shows an SE command with a string selection expression:

```
SE1-100,/A/OR/B/OR/C/OR/D/AND/E/AND,
NOT/F/;73O/ABC/;TC
```

This command would overwrite with ABC columns 73 through 75 and type in compressed format each record that:

1. contained an A or a B or a C or a D

2. and contained an E

3. and did not contain an F.

SS Set and Step

SS causes Edit to start at a specified record and proceed to each record in succession, accepting one line of intrarecord commands to update the current record. The SS command has the format shown below.

$$\text{SS n } [\text{-m}], \left\{\begin{matrix}\text{/string/}\\ \text{sse}\end{matrix}\right\} [\text{,c } [\text{,d}]\,]$$

The first record to be updated has the sequence number n. If m is not specified, records n through the last record in the file are selected. If m is specified, records n through m are selected. Intrarecord commands will only be effective on strings that lie wholly within columns c through d. The default values for c and d are 1 and 140, respectively.

Edit prompts for commands for each successive record with the sequence number, followed by a double asterisk. The SS command is terminated by typing a null record in place of an intrarecord command.

ST Set, Step, and Type Record

This command is similar to SS except that the content of each record is typed in the format of the last TS, TY, TC or RR command used, prior to accepting a command. The ST command has the format shown below.

$$\text{ST n } [\text{-m}], \left\{\begin{matrix}\text{/string/}\\ \text{sse}\end{matrix}\right\} [\text{,c } [\text{,d }]\,]$$

The parameters of the command and the error messages which Edit types are the same as those for the SS command.

INTRARECORD EDITING COMMANDS

The intrarecord commands make changes within an individual record. They generally manipulate character strings. These commands may only be given after the user selects an intrarecord mode with the SE, SS, or ST commands.

The intrarecord commands will be discussed in the following order:

IF	Conditional command execution.
EI	End IF.
EL	Else if.
CL	Set column limits .
RL	Repeat command line.
S	Substitute string.
D	Delete string.
P	Insert string preceding.
F	Insert string following.
O	Overwrite string.
E	Overwrite string; blank fill.
R and L	Shift string.
A	Align strings.
DE	Delete record.
AD	Add to record
CP	Copy Current Record, Protected.
CI	Copy Current Record, Interlaced.
RR	REREAD record.
TC TS TY	Type individual records.
TX	Type changes records.
JU	Jump to new sequence.
NO	No change.
QR	Quit processing current record.
RF	Reverse blank preservation flag.

Commands in the intrarecord group may be linked together on a single line through use of the semicolon (;). The following command sequence would select a line, type the original, edit, and type the new version:

```
*SE 100; TY; /TEMP/S/B/;/JK/F/+BETA/;TY
```

The following conventions are used with intrarecord commands:

1. j/string/x

   means that command x is to operate on the jth occurrence of the indicated string found between columns c through d as specificied by a CL, SE, SS, or ST command. If j = 0, this means that the command is to operate on all occurrences of the string between columns c and d. If j is missing, the default is 1.

Strings are composed of any combination of the following constructs.

/string/ specifies the exact character string between the slashes. A single / may be included in the string by typing two successive slashes.

"string" specifies the exact character string between the double quotes. A double quote may be included in the string by typing two successive double quotes.

#hh specifies a character with an EBCDIC hexadecimal value of hh.

? specifies any character string. ? may be used for any string selection, matching, or replacement. The ? is satisfied by any string of any length (including the zero-length, or null string). The ? string always exists in all records.

?n specifies a character string of any n characters. ?n may be used for any string selection, matching, or replacement. The ?n string is satisfied by any string of exactly n characters in the record.

Each of the following examples describes the string: ABCD

```
/ABCD/
"ABCD"
#C1#C2#C3#C4
/A/"BC"#C4
/A/#C2"CD"
```

Each of the following examples describes the string: A/B+C/D

```
/A//B+C//D/
"A/B+C/D"
```

The string /AB/?/CD/ exists in any record that contains the string AB followed by the string CD. Any number of characters (including none) may exist between AB and CD.

The string /AB/?1/CD/ exists in any record that contains the string AB immediately followed by any single character immediately followed by the string CD.

2. kx

   means that command x is to operate on the character contained at column k, where k must lie between columns c and d of the SE, SS, or ST command,

Whenever an S, D, P, F, O, E, R, or L command is given, Edit responds in one of the following ways:

1. A prompt alone indicates that one change occurred as a result of the command, or that the changes have been made to all specified records for a kx format command (i.e., changes are not optional).

2. The message

   x STRINGS CHANGED

   followed by a carriage return and a prompt indicates that x changes occurred as a result of the command.

The following general errors are possible:

-MISSING SE No SE command was given. Either an SE, SS, or ST command must be given in response to this message.

--Cn:COL > LIMIT The value specified for k is greater than d for the nth command.

--Cn:COL < LIMIT The value specified for k is less than c for the nth command.

Before reading the intrarecord command descriptions, it is important to note the following information:

Note: In any intrarecord command that seeks a matching string in the image, only those strings that lie totally within the specified column bounds will be found. Partial matches to a column boundary will be ignored. In subsequent examples, references to columns c and d pertain to the column boundaries given in the CL, SE, SS, or ST command.

IF If (conditional command execution)

IF causes Edit to evaluate an sse (string selection expression). If the sse is true, the commands immediately following the IF on the command line are executed. If the see is false, command execution resumes with the next EL (else if) or EI (end if) command at the same nesting level as the IF.

The IF command has the format shown below.

```
[. . .;]IF sse [,c [,d] ]; . . .
```

If column numbers (c and d) are specified on the IF command, they only affect the evaluation of the sse; they have no effect on following commands.

The following example changes all occurrences of "A" to "B" between columns 10 and 20 on records that have an "*" in column 1.

```
SE0-9999, 10,20; IF/*/,1, 1; 0/A/S/B/;EI
```

The following example types all continuation lines in an ANS FORTRAN program.

```
IF NOT/*/AND, NOT/C/,1, 1;

IF NOT/0/AND, NOT/b̸/,6, 6; TY
```

EI End IF block

EI terminates an IF block (see IF). If the end of the command line is reached and there are more IFs than EIs, additional EIs are assumed to match the number of IFs.

The format of EI is shown below.

```
. . .;EI[; ...]
```

The following examples produce the same changes.

```
SE  0-9999,10,20;   IF/*/,1,1; 0/A/S/B/; EI

SE  0-9999,10,20;   IF/*/,1,1; 0/A/S/B/
```

The following example produces the same changes, and also types each record in the range after any substitutes are made.

```
SE  0-9999,10,20;  IF/*/,1,1;  0/A/S/B/;EI; TY
```

The following example produces the same changes, except only those records with an "*" in column 1 are typed after any substitutions are made.

```
SE  0-9999,10,20;  IF/*/,1,1;  0/A/S/B/;  TY
```

EL Else If

The EL command is executed only when all previous IF or EL commands at the same nesting level have evaluated false sse's. EL's function is otherwise the same as IF.

The format of the EL command is shown below.

```
. . .;EL[sse[,c[,d]]]; . . .
```

If column numbers (c and d) are specified on the EL command, they affect only the evaluation of the sse; they have no affect on the following commands.

The following example changes all occurrences of "A" to "B" in records that have an "*" in column 1. If column 1 does not contain an "*", all occurrences of "C" are changed to "D".

```
IF/*/1,1; 0/A/S/B/;  EL;  0/C/S/D
```

In the following example, if there is an "*" in column 1, all occurrences of "A" are changed to "B". If column 1 does not contain an "*" and there is an "X" anywhere in the record, all occurrences of "C" are changed to "D". If column 1 does not contain an "*" and there is no "X" in the record, all occurrences of "E" are changed to "F".

```
IF/*/,1,1;   0/A/S/B;   EL/X/;   0/C/S/D/;   EL;
0/E/S/F/
```

In the following example, if the record contains an "*" and an "A", the record is deleted. If the record contains an "*" and the "B", the record is typed. If the record does not contain an "*", all occurrences of "C" are changed to "D".

```
IF/*/;  IF/A/;  DE;  EL/B/;  TY;  EI;  EL;
0/C/S/D/;  EI
```

CL Change column limits

The CL command allows modification of the column limits on string substitutions. The format is

```
[. . .;] CL [c [,d] ]; . . .
```

c is the new first column and d is the new last column. c and d may be integers, or strings in the standard Edit string format.

If c is a string, the new first column is set to the beginning of that string. If d is a string, the new last column is set to the end of that string. If c or d is a string that does not exist, the columns are set such that no string substitution occurs.

If more than one record is being processed, the column limits are reset to the original limits specified on the SE command each time a new record is processed. If no limits are specified on the SE command, the column limits are reset to include the entire record.

RL Repeat Command Line

If any changes have been made to the record open for editing, execution of the RL command causes Edit to return to the beginning of the command line and reprocess the command line. If the RL command appears on a continuation command line, Edit returns to the first command line inthe series of lines. The record open for editing is not written to the file or read back from the file as a result of the RL command; the record is written to the file when the end of the command line(s) is encountered (unless NO or DE is executed). The format is shown below.

```
. . .; RL [; . . . ]
```

The following example right-justifies to column 5 the statement labels in an ANS FORTRAN program.

```
IF/*/OR/C/,1,1; EL/ /,5,5; IF NOT/ /,

1,4; RF; 1R1; 8L1; RL
```

S Substitute String

S causes Edit to locate a specified string (string$_1$) between columns specified by an SE, SS, or ST command and replace it with another string (string$_2$). The S command has the format shown below.

```
[i]/string1/S/string2/
```

The image to the right of string$_1$ is adjusted right or left as required, if the lengths of string$_1$ and string$_2$ differ. String$_2$ may extend past column d if d < 140.

If j = 0, all occurrences of string$_1$ between columns c and d are replaced by string$_2$. Otherwise, only the jth occurrence is replaced. If j is missing, the default value is 1.

Example:

Command	Effect		
*/LW/S/CW/		LW,R5 ALPHA+2 CW,R5 ALPHA+2	old new
*/10/S/5/		LW,R10 B LW,R5 B	old new
*/$10/S/ENTRY/	$10 ENTRY	LW,R5 ALPHA LW,R5 ALPHA	old new
*/ALPHA/S/B/		LW,R5 ALPHA+2,R6 LW,R5 B+2,R6	old new
*2/5/S/55/	15	C=DSQRT(TEMP +2.5*BASE)	old
	15	C=DSQRT(TEMP +2.55*BASE)	new

D Delete String

D causes Edit to locate a given occurrence of an indicated string, between columns specified by an SE, SS, or ST command, and delete it. The D command has the format shown below.

```
[i]/string/D
```

If j = 0, all occurrences of the string between c and d are deleted. Otherwise, only the jth occurrence is deleted. If j is omitted, the default value is 1.

Example:

```
*EDIT SOMEFILE (RET)
*TY 7 (RET)
    7.000    STW,4    ALPHA    ANSWER
*SE 7 (RET)
*/ANSWER/D (RET)
*TY 7 (RET)
    7.000    STW,4    ALPHA
```

P Precede String

P causes Edit to start before the first character of a given occurrence of a specified string (string$_1$) or column k and insert another string (string$_2$), pushing characters of the first string to the right as required to make room. The P command has the format shown below.

```
[i]/string1 /P/string2/
```

or

```
kP/string2/
```

String$_2$ may legally extend beyond column d if d < 140. The first character of string$_2$ will occupy the column vacated by the first character of string$_1$, etc.

If j = 0, Edit will insert string$_2$ before all occurrences of string$_1$ between columns c and d. However, after string$_1$ has been found once and string$_2$ inserted before it, scanning for the next occurrence resumes at the next character after string$_1$, as adjusted by the insertion. If j is not equal to zero, the command will only affect the jth occurrence of string$_1$. If j is omitted, the default value is 1.

Example:

```
*SE 17.69 (RET)              (set intrarecord mode)
*TS;0/AA/P/./;TS (RET)       (type; edit; type)
AAAAAAA                      (original record)
.AA.AA.AAA                   (edited record)
```

F Follow String

F causes Edit to start after the last character of a given occurrence of a specified string (string$_1$) or column k and insert another string (string$_2$), pushing everything from this column right as required to make room. The F command has the format shown below.

[j]/string$_1$/F/string$_2$/

or

kF/string$_2$/

The j specifies that the jth occurrence of string$_1$ between columns c and d (specified by an SE, SS, or ST commqnd) is to be followed by string$_2$. If j is omitted, the default value is 1. In the case where j = 0, Edit inserts string$_2$ at all occurrences of string$_1$ between columns c and d. Scanning for the next occurrence of string$_1$ resumes following the last character of string$_2$. If a given occurrence of string$_1$ is shifted beyond column d due to previous insertions, it will not be scanned.

String$_2$ may legally extend past column d if d < 140.

Example:

Command	Effect		
*/AB/F/+2/	LW,R6 LW,R6	AB,R2 AB+2,R2	old new

O Overwrite

O causes Edit to start at the column occupied by the first character of a given occurrence of a specified string (string$_1$) or column k and overwrite with another string (string$_2$). No blank preservation or other adjustment is done and all columns not overwritten remain unchanged. The O command has the form shown below.

[j]/string$_1$/O/string$_2$/

or

kO/string$_2$/

String$_2$ may overwrite beyond column d if d < 140. The j specifies that the jth occurrence of string$_1$ between affected columns is to be overwritten by string$_2$. If j is omitted, only the first occurrence is overwritten. If j = 0, all occurrences are overwritten. In the case where j = 0, string$_2$ is not scanned by Edit after string$_1$ is overwritten. Edit begins scanning with the column following string$_2$.

E Overwrite and Extend Blanks

E causes Edit to start at the column occupied by the first character of a given occurrence of a specified string (string$_1$) or column k and overwrite with another string (string$_2$). The E command has the format shown below.

j /string$_1$/E/string$_2$/

or

kE/string$_2$/

Blanks are extended from the end of string$_2$ through column d (where d is the upper limit of the column range selected by an SE, SS, or ST command). String$_2$ may overwrite beyond column d if d < 140, but blank extension only occurs through column d.

The j specifies that the jth occurrence of string$_1$ between affected columns is to be overwritten by string$_2$. If j is omitted, only the first occurrence is overwritten. The specification j = 0 may not be specified, since blank extension precludes multiple substitutions within the same record.

R and L Shift Record Image

R and L commands cause portions of the record image to be shifted right (R) or left (L). The R and L commands have the form shown below.

[j]/string/$\left\{\begin{matrix}R\\L\end{matrix}\right\}$s

or

k$\left\{\begin{matrix}R\\L\end{matrix}\right\}$s

The string must lie wholly within columns c and d specified by the current SE, SS, or ST command. The specified substring may contain embedded blanks, but the string to be shifted terminates with the first blank following the specified substring.

The j specifies that the jth occurrence of the specified substring between affected columns is to be shifted, together with all subsequent contiguous nonblank characters. If j is omitted, only the first such occurrence is shifted. Note that j = 0 may not be specified for this command.

L Shift Left

The jth field that begins with the indicated string (or column k) is shifted left s positions. If blank preservation (see the BP command) is ON, all of the fields to the right of the string are shifted left, intact, and the fields to the left of the string are overwritten (i.e., destroyed). If blank preservation is OFF, blanks are inserted to the right of the jth field, and the fields to the left of the string are overwritten The shift may legally overwrite below column c.

R Shift Right

The jth field that begins with the indicated string (or column k) is shifted right s positions. If blank preservation is ON, blanks are inserted to the left of the string and all of the fields to the right of the string are shifted right, intact. If blank preservation is OFF, blanks are inserted to the left of the string and are removed to the right. With blank preservation OFF, the image area to the right of the string may be compressed, but at least one blank will be left between each field; that is, overwriting does not occur in a shift right. The shift may legally push characters beyond column d, if d is less than 140.

In the following examples, blank preservation is OFF.

Command	Effect
*/L/R1	$10 LW,R6 B old $10 LW,R6 B new
*/L/R9	$10 LW,R6 B old $10 LW,R6 B new
*/L/L1	$10 LW,R6 B old $10 LW,R6 B new

A Align Specified Columns

The A command causes either a right or left shift of a part of the record being edited to align one specified column with another. The details of the shift operation are as described for the R and L commands. The form of the command is

```
[...;]cAd[;...]
```

where "c" specifies the column to be aligned and "d" specifies the column to which to align. The "c" and "d" items can be any of

k

an integer column number,

/string/

specifying the first column of the first occurrence of the string, or

n/string/

specifying the first column of the "n"th occurrence of the string. ("n" cannot be zero for this command.)

Examples:

```
*SE 1-3; TS (RET)
   -ONE
    XXXXX     -TWO, TWO
-THREE    THREE    THREE
```

```
*/-/ A 5; TS (RET)
    -ONE
    -TWO                TWO
    -THREE   THREE   THREE
```

Note the distinction between left-shift (TWO) and right-shift (THREE) behavior.

```
*SE 20; TS (RET)
*ONE    *TWO    *THREE
*3/*/A 2/*/; TS (RET)
*ONE    *THREE
```

DE Delete Records

DE causes Edit to delete the record currently open for editing under control of an SE, SS, or ST command. The DE command has the format shown below.

```
...; DE[;...]
```

If

1) DE is the only command on the line,

2) there is no sse (string selection expression) in effect,

3) the SE range includes more than one record,

and 4) Edit is not processing commands from an XEQ file,

then the following message is output.

```
--DELETE n to m?  (Y or N)?
```

n is the sequence number of the first record to be deleted and m is the sequence number of the last record to be deleted. If the user responds with "Y", the deletion occurs. If the response is "N", Edit prompts for the next command without deleting any records.

AD Add To Record, Sequence Number Included

AD causes Edit to type the contents of the record currently open for editing under control of an SE, SS, or ST command. The carriage will remain positioned at the end of the record. The user may then enter text to be placed after the end of the original record. The AD command has the format shown below.

```
[...;]AD[;...]
```

CP Copy Current Record, Protected

The CP intraline command causes the record(s) open for editing to be copied to another record position. The open record is not deleted. The format is shown below.

```
[...;] CP[n][,i][;...]
```

where

n specifies the lower limit (i.e., sequence number) of the position to which the records are copied. If n is omitted, the most recent sequence number specified on a CI or CP command is used.

i specifies the increment value to be used for renumbering records. If i is omitted, the most recent increment value specified on a CI or CP command is used. If no such commands have been given, the default value is 1.

The CP command terminates with an error message in the following cases:

1. An attempt Is made to write a record with a sequence number greater than 9999.999.

2. An attempt is made to write a record that exists.

3. An attempt is made to write a record and a record exists between the last destination record and the current destination record.

The following example copies all records in the range of 1 to 300 with an "*" in column 1 to the range starting with number 1000, incremented by 1 (1001, 1002, ...). Each copied record is typed. If there are 10 records in the range 1-300 with "*" in column 1, the destination range is 1000-1009, incremented by 1. If 1005 or 1006.2 exists, an error occurs.

```
SE1-300,/*/,1,1; CP 1000; TS
```

CI Copy Current Record, Interlacing

The CI intraline command causes the record(s) open for editing to be copied to another record position. CI allows over-writing existing records and interlacing records. When a record exists in the destination range with the sequence number that is to be assigned to a copied record, the record in the destination range is deleted (overwritten). When a record exists in the destination range with a sequence number that is not to be reassigned (that is, the specified increment is such that the sequence number is not reassigned to a copied record), the record is not deleted. This allows records to be interlaced. The format is shown below:

```
[...;]CI[n][,i][;...]
```

where n and i are the same as CP.

The following example copies all records in the range of 1 to 300 with an "*" in column 1 to the range starting with number 1000, incremented by 1 (1001, 1002, ...). Each copied record is typed. If there are 10 records in the range 1-300 with "*" in column 1, the destination range is 1000-1009, incremented by 1. If 1005 exists, it is overwritten; if 1006.2 exists, it remains within the range of new records.

```
SE1-300,/*/,1,1; CI1000; TS
```

RR Re-read Record, Sequence Number Included

RR causes Edit to type the contents of the record currently open for editing under control of an SE, SS, or ST command and then allow the user to modify the record using the editing functions described in the section "Typing Lines" of Chapter 2. A carriage return should be entered when the editing is complete. If the user enters a null line (e.g., types CONTROL X followed by a carriage return) the record will not be changed. (This command makes use of the COC routine's REREAD function.) The RR command has the format shown below.

```
[...;]RR[;...]
```

TC Type Record, Compress Blanks, Sequence Number Included

TC causes Edit to type the contents of the record currently open for editing under control of an SE, SS, or ST command. The record is typed in the same format as the record-editing TC command. (Unlike the record-editing version, the intrarecord version of TC does not allow column specification.) The TC command has the format shown below.

```
[...;]TC[;...]
```

(This page intentionally left blank)

TS Type Record, Sequence Number Not Included

TS causes Edit to type the contents of the record currently open for editing under control of an SE, SS, or ST command. (Unlike the record-editing version, the intrarecord version of TS does not allow column specification.)

The TS command has the format shown below.

```
[. . .;] TS[; . . .]
```

The three dots indicate that intrarecord commands may precede or follow the TS command.

Example:

```
*SE 5; TS
L1  LW,5 K
*190/KLB/; TS; 370/GET KLB/; TS
    (overwrite, type, overwrite, type)
L1  LW,5  KLB
L1  LW,5  KLB     GET KLB
```

Because all commands on a single input line are executed for the first record before the second record is processed, etc., TS will type each line in turn after all editing up to the TS command has been done.

Example:

```
*SE     10- 10.2
*2/A/F/,4/;TS
    DATA,4     X'FF'          (10.0)
    DATA,4     0.5            (10.1)
    DATA,4     GQX,X'0B'      (10.2)
```

TY Type Record, Sequence Number Included

TY is the same as TS, except that each line is printed with its sequence number. (Unlike the record-editing version, the intrarecord version of TY does not allow column specification.)

The TY command has the format shown below.

```
[. . .;] TY[; . . .]
```

TX Type Changed Records

TX is the same as TY, except that only those records within the Edit range (set by SE, SS or ST commands) which have been changed by the intrarecord command are typed. The TX command has the format shown below.

```
. . .; TX
```

Note that for TX to be effective, it must be given on the same line as the intrarecord command.

JU Jump

JU causes the SS or ST command to jump to a specified record and then continue stepping from that point. JU may only be used while in the "step" mode (i.e., while under the control of an SS or ST command). The JU command has the form shown below.

```
[...;] JU n
```

Record n may be forward or backward from the current sequence number at the time JU is given. The dots indicate that JU may be used on compound lines (i.e., a line with more than one command on it), but in such a case JU must be the last command on the line. If a range was specified on the SS or ST command, and the JU command causes a jump out of that range, the range is changed to include the entire file.

NO Make No Change

NO specifies that no editing is desired on the current active line under the set. The NO command has the format shown below.

```
[...;]NO[;...]
```

Example:

```
*ST 27.5
27.500                  LW,6      BLK
*NO
30.000                  STW,6     ALT
*/ALT/F/+ 19 ; TY ; JU 34
30.000                  STW,6     ALT + 19
34.000                  AI,F      X'91'
*
```

When using the SS and ST commands, linefeed may be substituted for NO . The following example copies the records in the SE range to new positions; the records are changed as they are copied. The NO command prevents the original records from being changed.

```
SE 10-20; /A/S/B/; CP30; NO
```

QR Quit Processing Current Record

When the QR command is executed, an unconditional skip is taken to the end of the command line. The next record in the SE range (if any) is then processed.

The format of the QR command is

```
...;QR;...
```

RF   Reverse Blank Preservation Mode

RF causes the current setting of the blank preservation mode (see the BP command) to be reversed temporarily. The RF command has the form shown below.

```
[...;]RF;...
```

The mode is reversed only for the duration of the input line in which RF appears and only for those commands which follow the RF command, and blank preservation is restored to its initial setting when a new input line is entered (i.e., at the time a new prompt character is given). Thus, to have any effect, RF must always be used as part of a compound input line and must be followed by other commands.

Example:

```
*SE    10;  TY(RET)
10.000  L5     LW,4  X  GET CURRENMT ADDR
*RF;/NM/S/N/;TY(RET)
10.000  L5     LW,4  X  GET  CURRENT ADDR
```

Without using RF in this case (assuming that BP OFF is the initial setting), one would get two blanks after CURRENT. In all cases, the BP mode is restored to the value it had before any RF commands were given.

MESSAGES

During the course of executing any command, Edit may communicate with the user through a variety of messages. Possible messages are summarized in Table 27. The following conventions are used in regard to message formats:

1. A message preceded by two periods is a comment on some system-oriented operation. For example,

   ```
   ..COPY DONE
   ```

2. A message preceded by two minus signs indicates the occurrence of some event (during the execution of a command) of which the user should be aware; the command is not aborted. For example,

   ```
   --EOF HIT AFTER xxxx.xxx
   ```

3. A message preceded by a single minus sign is an error message describing a condition that aborts the current command and causes any others on the same line to be skipped. For example,

   ```
   -P1:NO SUCH REC
   ```

Such a message is particularized as to cause by the following prefixes:

Prefix	Cause of Error
-Ck:	The kth command of the previous line caused the error.
-Pk:	The kth parameter of the first command on the previous line caused the error.
-CkPj:	The jth parameter of the kth command of the previous line caused the error.

EDIT COMMAND SUMMARY

Table 28 is a summary of Edit commands. The left-hand column gives the command formats. The right-hand column gives the command functions and options.

Table 27. Edit Messages

Message	Meaning
-BAD COL. NO. PAIR	The columns specified are not in the range 1 through 140, or c > d.
--Cn: 'ALL' IGNORED	The value 0 was specified for j. Since this value is not meaningful for the command, the value 1 has been assumed.
-Cn: CMND ILGL HERE	The nth command of the previous line is invalid and the intrarecord mode has been terminated.
-Cn: COL > LIMIT	The value specified for k is greater than d for the nth command.
-Cn: COL < LIMIT	The value specified for k is less than c for the nth command.
-Cn: ILGL SYNTAX	Invalid command syntax has been used.
-Cn: NO SUCH REC	The record specified in a JU command (the nth command) does not exist.
--Cn: OVERFLOW	The nth command of the previous line has caused characters to be shifted past column 140. Processing continues.
--Cn: UNDERFLOW	Characters were lost to the left of the record.
-Cn: UNKN CMND	The nth command specified is not one recognized by Edit.

Table 27. Edit Messages (cont.)

Message	Meaning
..COPY DONE	A COPY operation has been completed
..COPYING	A COPY operation has begun.
--CUTOFF AT x(y)	A specified operation could not be completed because of a conflict between an existing sequence number and a new one. The value x is the current sequence number of the last record affected (formerly record y).
..DELETED	A specified file has been deleted.
--DONE AT x	A specified operation has been completed. The value x is the current sequence number of the last record affected.
..EDIT STOPPED	The record editing mode has been terminated.
--EOF HIT AFTER xxxx.xxx	One or both sequence numbers specified are higher than the highest one in the file. xxxx.xxx is the last record in the file.
-FILE EXISTS: CAN'T BUILD	An existing file has the same name as that specified in a BUILD command.
-FILE OPEN FOR INPUT ONLY; CANNOT UPDATE	The user is attempting to edit a file in another account for which he has read access but not write access. Those commands which do not alter a file will be executed normally. Any command that alters a file will appear to be executed, but the file will not actually be changed.
-MAX.SEQ.NO.EXCEEDED	An attempt was made to insert a record with a record number greater than 9999.999.
-MERGE DESTINATION NOT KEYED	The destination file in a MERGE command is not keyed. The file must be copied with sequencing specified.
-MERGE SOURCE NOT KEYED	The source file in a MERGE command is not keyed. The file must be copied with sequencing specified.
..MERGE STARTED	A MERGE operation has begun.
-MISSING SE	No SE, SS, or ST command is currently in effect. The specified intrarecord task has been aborted.
-NO FILE NAMED	The command requires a file identification and none was given.
-NO SUCH FILE	A specified file does not exist.
--NONE	There are no records in the specified range containing the indicated string.
-NOT F/M/S	A parameter other than F, M, or S has been specified in a TA command.
-NOT ON/OFF	A parameter other than ON or OFF has been specified in a BP or CR command.
--NOTHING TO MOVE	No records (to be moved) were found in the specified range.
--OVERFLOW	More than 140 characters have been typed on a line or characters have been shifted past column 1 or 140. Excess characters are lost.
-P1: FILE NOT KEYED & P3 NULL	A file to be copied has no sequence numbers and no sequencing has been specified. The COPY operation has been aborted.
-P1: NO SUCH FILE	A COPY command has specified that a nonexistent file is to be copied.

Table 27. Edit Messages (Cont.)

Message	Meaning
-P1: NO SUCH REC	A specified record does not exist. The command has been aborted.
-P2: COL ERROR	Column c is greater than 140.
-P2: FILE EXISTS	A COPY ON command specified the name of an existing file.
-P2: REC EXISTS	A specified record already exists. The command has been aborted.
-P3: NOT INCR	An invalid increment has been used in a BUILD command.
-Pn: BAD FID	An invalid fid identification has been used.
-Pn: ILGL SEQ#	A required sequence number is missing or contains more than three fractional digits.
-Pn: ILGL STRG	The specified string is too long.
-Pn: ILGL SYNTAX	An invalid syntax has been specified for a parameter.
-Pn: NOT CNT	An invalid occurrence count (j) has been specified in an intrarecord command.
-Pn: NOT COL#	An invalid column number has been specified.
-Pn: NOT SEQ#	A required sequence number is invalid.
-Pn: NOT STRG	An invalid character string has been specified.
-Pn: NULL STRG	A null character string has been specified.
-Pn: PARAM MISSING	A required parameter has not been specified.
-Pn: SEQ2<SEQ1	The second sequence number is less than the first in a range specification.
xxxxxxx RECORDS DELETED	xxxxxxx specifies the number of records deleted as the result of either a DE, an MD, or an MK command.
xxxxxxx RECORDS MOVED	xxxxxxx specifies the number of records moved as the result of either an MD or MK command.
--RNG OVERLAP	Specified ranges of sequence numbers overlap. The command has been ignored.
x STRINGS CHANGED	As the result of an intrarecord command, x strings have been changed.

Table 28. Edit Command Summary

Command	Description
[...;]c**A**d[;...]	Align strings. c specifies the start of the string to be aligned and d specifies the column to align to. c and d may be a column number or a string of the format: j/string/. The string to be aligned will be shifted using the R and L shift command rules.
[...;] **AD** [;...]	Causes Edit to type the currently open record, leaving the carriage positioned at the end of the record. The user may then enter text to be appended to the record.
AD n[-m]	Causes Edit to type each record in the range of n to m, leaving the carriage positioned at the end of the record. The user may then enter text to be appended to the record.
BP {ON OFF}	Sets the blank preservation mode. When "on", all strings of blanks are preserved during intrarecord operations. When "off", blank strings are compressed to a single blank or expanded as required to retain column alignment of nonblank fields. The default mode is "off".
B[UILD] fid[,[n][,i]]	Enables the user to create a new file. Options: n is the sequence number at which the new file is to start. The default value is 1. i is the value by which the sequence numbers are to be incremented. The default value is 1.
CI [n][,i]	Causes Edit to copy record that is currently open for editing to new position, allowing interlacing and overwriting. Options: n is the destination sequence number. i is the destination sequence number increment.
CL [c [,d]]	Sets column limits. Options: c is the beginning column, and may be expressed as an integer or a string. d is the last column number, and may be expressed as an integer or a string.
CM n, c	Causes Edit to insert commentary (given by the user) into specified columns (starting at column number c) of each successive record beginning at the specified sequence number n.
CP [n] [,i]	Copies record that is currently open for editing to new position, preventing overwriting and interlacing. Options: Same as for CI.

Table 28. Edit Command Summary (cont.)

Command	Description
C[OPY] fid$_1$ $\left\{\begin{array}{l}\text{ON}\\ \text{OVER}\end{array}\right\}$ fid$_2$[,[n][,i]]	Copies a file. Fid$_2$ identifies the file to which the file identified by fid$_1$ is to be copied. Options: n is the starting sequence number for the new file. If omitted, the sequence numbers of fid$_1$ are retained in the copy. i is the sequence number increment for the new file. The default value is 1.
C[OPY] fid [,n [,i]]	Copies a file over itself where n is the starting sequence number for the new file and i is the sequence number increment. The default for both values is 1.
CR $\left\{\begin{array}{l}\text{ON}\\ \text{OFF}\end{array}\right\}$	Controls the inclusion of the carriage return character (X'15') at the end of each record in the user's output file. ON includes the X'15' terminator in the output file. OFF excludes the X'15' terminator from the output file and is the default setting.
CPRT [seed]	Causes the data encryption seed to be set or reset. Speed is 1 to 8 hexadecimal digits. If seed is omitted, data encryption is reset. See warnings under CRPT command description.
CT n, c	Causes Edit to type the record up to the specified column (column number c) and then to insert commentary (given by the user) into specified columns (starting at column number c) of each successive record beginning at the specified sequence number n.
[j] /string/**D**	Locates a given occurrence of the indicated string, between columns specified by an SE, SS, or ST command, and deletes it. Option: j specifies that only a particular occurrence (the jth occurrence) of the string in the specified columns is to be deleted. If j equals zero, all occurrences of the string in the specified columns are to be deleted. If j is omitted, the default value is 1.
[...;] **DE**	Causes the record currently open to be deleted.
DE n[-m]	Deletes all records whose sequence numbers lie in a specified range beginning at n. Option: m indicates the number of the last record to be deleted. If m is omitted, only record n will be deleted.
D[ELETE] fid	Deletes the file specified by fid from the log-on account.
DL n	Deletes the number of records specified by n during a DE, MK, or MD command.
[j] /string$_1$/**E**/string$_2$/ or k **E** /string$_2$/	Starts at a column occupied by the first character of a given occurrence of a specified string (string$_1$) or column k and overwrites with another string (string$_2$). Blanks are extended from the end of string$_2$ through column d (which is specified in an SE, SS, or ST command.) Option: j specifies that the jth occurrence of string$_1$ between affected columns is to be overwritten by string$_2$. If j is omitted, only the first occurrence is overwritten; j may not be zero.
ECHO $\left\{\begin{array}{l}\text{ON}\\ \text{OFF}\end{array}\right\}$	Sets the Edit XEQ echo mode.

Table 28. Edit Command Summary (cont.)

Command	Description
-[DIT] fid	Opens a file to be edited and enters the record editing mode.
EI	Ends an IF block.
EL [sse[,c[,d]]]	Starts an Else IF block in an IF block. Options: sse is a string selection expression. c specifies the lowest column to be searched. The default is 1. d specifies the highest column to be searched. The default is 256.
END	Closes all active files and returns control to the terminal executive language (TEL). This command is equivalent to the X command.
[j]/string$_1$/F/string$_2$/ or kF/string$_2$/	Starts after the last character of a given occurrence of a specified string (string$_1$) or column k and inserts another string (string$_2$), pushing everything from this column right as required to make room. Option: j specifies that the jth occurrence of string$_1$ between columns c and d (specified by an SE, SS, or ST command) is to be followed by string$_2$. If j is omitted, the default value is 1. If j equals zero, string$_2$ is inserted at all occurrences of string$_1$ between columns c and d.
FD n[-m], $\begin{Bmatrix}\text{/string/}\\ \text{sse}\end{Bmatrix}$ [,c[,d]]	Searches for the specified string or strings between specified columns in a specified range of records beginning at record n. If the search is satisfied, the record containing it is deleted from the file. Options: m specifies the sequence number of the last record to be searched. If omitted, only record n is searched. sse is a string selection specification as in the SE command. c specifies the lowest column number of the field to be searched. The default value is 1. d specifies the highest column number of the field to be searched. The default value is 140.
FS n[-m], $\begin{Bmatrix}\text{/string/}\\ \text{sse}\end{Bmatrix}$ [,c[,d]]	Searches for the specified string or strings between specified columns in a specified range of records beginning at record n. Each time the search is satisfied, the sequence number of the record is printed. Options: Same as for FD.
FT n[-m], $\begin{Bmatrix}\text{/string/}\\ \text{sse}\end{Bmatrix}$ [,c[,d]]	Searches for the specified string or strings between specified columns in a specified range of records beginning at record n. Each time the search is satisfied, the sequence number and the contents of the record are printed. Options: Same as for FD.

(This page intentionally left blank)

Table 28. Edit Command Summary (cont.)

Command	Description
IF sse [,c[,d]]	Starts an IF block. Options: Same as for EL.
IN [n] [,i]	Inserts new records into a file starting at record n. Edit prompts the user with the sequence number of each record to be inserted. Option: i specifies an increment amount for successive record numbers. If i is omitted, the increment size specified in the most recent record editing command is used. If no such command has been given, the default value is 1.
IP [n] [,i]	IP is identical in IN, except that if record n exists the command is immediately aborted.

Table 28. Edit Command Summary (cont.)

Command	Description
IS [n] [,i]	Inserts new records into a file starting at record n. Edit does not prompt with sequence numbers of the records to be inserted. Option: i specifies an increment amount for successive record numbers. If i is omitted, the increment size specified in the most recent record editing command is used. If no such commands have been given, the default value is 1.
[. . . ;] **JU** n	Causes the SS or ST command to jump to the specified record n and then continues stepping from that point. Option: The dots indicate that JU may be used on a line with more than one command on it, but in such a case JU must be the last command on the line.
L	Invokes the PCL LIST command.
MD n[-m],k[-p][,i]	Moves records within a file from a range beginning at n to a range beginning at k. The original records are deleted. Options: m specifies the sequence number of the last record that is to be moved. If omitted, only record n is moved. p specifies the upper limit of the range of records to be deleted. If omitted, only record k is deleted. However, records from the range n-m are still moved to record k and following. i specifies the increment value to be used for renumbering records. If omitted, the most recent value specified in a record edit command is used. If no such commands have been given, the default value is 1.
M[ERGE] fid$_1$[,n$_1$[-n$_2$]] INTO fid$_2$, n$_3$[-n$_4$][,i]	Merges records from fid$_1$ into fid$_2$. The records are numbered beginning at n$_3$ in fid$_2$. Options: n$_1$ specifies the number of the first record in fid$_1$ to be merged. If omitted, all records of fid$_1$ are to be merged. n$_2$ specifies the number of the last record in fid$_1$ to be merged. If omitted, only record n$_1$ is merged. n$_4$ specifies the number of the last record in fid$_2$ which is to be replaced by merged records. If omitted, only record n$_3$ is replaced by a merged record. i specifies an increment amount for resequencing from n$_3$. The default value is 1.
MK n[-m],k[-p][,i]	MK is identical to MD except that the records in the range n-m are not deleted as they are moved. Options: Same as for MD.
NO	Specifies that no editing is to be performed on the current active line.

Table 28. Edit Command Summary (cont.)

Command	Description
[j]/string$_1$/**O**/string$_2$/ or k**O**/string$_2$/	Starts at the column occupied by the first character of a given occurrence of a specified string (string$_1$) or column (k) and overwrites with another string (string$_2$). Option: j specifies that only a particular occurrence (the jth occurrence) of the string is to be overwritten. If j equals zero, all occurrences of the string are to be overwritten. If j is omitted, the default value is 1.
[j]/string$_1$/**P**/string$_2$/ or k**P**/string$_2$/	Starts before the first character of a given occurrence of a specified string (string$_1$) or column k and inserts another string, pushing characters of the first string to the right as required to make room. Option: Same as for the O command.
QR	Quit processing the current record, and continue with the next record.
[j]/string/$\left\{\begin{matrix}\mathbf{R}\\\mathbf{L}\end{matrix}\right\}$s or k$\left\{\begin{matrix}\mathbf{R}\\\mathbf{L}\end{matrix}\right\}$s	Shifts portions of the record right (R) or left (L) the number of positions indicated by s. The field to be shifted begins with the indicated string or column k. Option: j specifies that the jth occurrence of the specified substring between affected columns is to be shifted, together with all subsequent contiguous nonblank characters. If j is omitted, only the first such occurrence is shifted. Note that j = 0 may not be specified for this command.
[. . . ;]**RF** ; . . . or . . . ; **RF**[; . . .]	Causes the current setting of the blank preservation mode ("on" or "off") to be reversed temporarily (for the current line only). Option: The dots indicate that other commands are present on the line.
RN n,k	Renumbers a specified record from number n to number k.
RP $\left\{\begin{matrix}\text{ON}\\\text{OFF}\end{matrix}\right\}$	Sets the record size preservation mode. When ON, the size of the edited records is not changed. Trailing blanks are not deleted. When OFF, records are shortened or lengthened as necessary by the editing process. Trailing blanks are deleted. The default mode is OFF.
RR [n[-m]]	Types the contents of each record in the range n to m, and allows the user to modify each record using the editing functions described in the section "Typing Lines" of Chapter 2.
[...;]**RR**[;...]	Types the contents of the record currently open for editing under control of an SE, SS, or ST command and allows the user to modify the record using the editing functions described in the section "Typing Lines" of Chapter 2.
[j]/string$_1$/**S**/string$_2$/	Locates a specified string (string$_1$) between columns specified by an SE, SS, or ST command and replaces it with another string (string$_2$). Option: j specifies that only a particular occurrence (the jth occurrence) of string$_1$ is to be replaced. If j equals zero, all occurrences of string$_1$ are to be replaced. If j is omitted, the default value is 1.

Table 28. Edit Command Summary (cont.)

Command	Description
SE n[-m][sse][,c[,d]]	Causes Edit to accept successive lines of intrarecord commands to be applied to records beginning at record n. Options: m specifies the number of the last record to which the intrarecord commands are to be applied. If omitted, the intrarecord commands are only applied to record n. sse is a string selection specification as described within the chapter. It has the format: $[\text{NOT}]/\text{string}/\left[\begin{Bmatrix}\text{AND}\\ \text{OR}\\ \text{EOR}\end{Bmatrix}\begin{bmatrix},\text{NOT}\\ \not{b}\text{NOT}\end{bmatrix}/\text{string}/\right]\ \ldots$ c specifies the smallest column number of the range of columns to which the intrarecord commands are to be applied. The default value is 1. d specifies the largest column number of the range of columns to which the intrarecord commands are to be applied. The default value is 256.
$\textbf{SS}\ n\left[-m\right],\begin{Bmatrix}/\text{string}/\\ \text{sse}\end{Bmatrix}\left[,c\left[,d\right]\right]$	Causes Edit to start at a specified record (record n) and proceed to each record in succession, accepting one line of intrarecord commands to update the current record. Options: m specifies the last record to be selected. If m is not specified, records thru to the end of the file will be processed. c specifies the smallest column number of the range of columns to which the intrarecord commands are to be applied. The default value is 1. d specifies the largest column number of the range of columns to which the intrarecord commands are to be applied. The default value is 256.
$\textbf{ST}\ n\left[-m\right],\begin{Bmatrix}/\text{string}/\\ \text{sse}\end{Bmatrix}\left[,c\left[,d\right]\right]$	Causes Edit to start at a specified record (record n) and proceed to each record in succession, accepting one line of intrarecord commands to update the current record. The sequence number and contents of each record are typed prior to accepting a command. Options: Same as for the SS command.
$\textbf{TA}\begin{Bmatrix}\text{F}\\ \text{M}\\ \text{S}\\ \text{C}\end{Bmatrix}$	Causes Edit to set or reset the terminal tab stops. F implies FORTRAN and tabs set at column 7, 16 and 34. M implies Meta-Symbol and tabs set at columns 10, 19, and 37. S implies Meta-Symbol, short form, and tabs set at columns 8, 16, and 30. C implies COBOL and tabs set at columns 8, 12, and 36.
$\textbf{TABX}\begin{Bmatrix}\text{ON}\\ \text{OFF}\end{Bmatrix}$	Sets the tab expansion mode. If on, tab characters are replaced with blanks when read from a file, and blanks are replaced with tab characters when being written to a file. If off, tab characters are not changed.
[. . .;] **TC** [;. . .]	Causes Edit to type in blank-compression mode the record currently open for editing.

Table 28. Edit Command Summary (cont.)

Command	Description
TC n[-m][,c[,d]]	Types the sequence numbers and the contents of specified columns of one or more records beginning at record n. Any nonblank strings within the columns typed are shifted to the left to compress each blank string to a single blank. Options: m specifies the number of the last record to be typed. If omitted, only record n is typed. c specifies the smallest column number of the range of columns to be typed. The default value is 1. d specifies the largest column number of the range of columns to be typed. The default value is 140.
[· · · ;] **TS**[; · · ·]	Types the contents of the record currently open for editing under control of an SE, SS, or ST command. Option: The dots indicate that other commands may be present on the line.
TS n[-m][,c[,d]]	Types the contents of specified columns of one or more records beginning at record n. Options: Same as for the TC command.
. . . ; **TX**	Types the sequence number and contents of those records within the edit range (set by SE, SS, or ST commands) which have been changed by the preceding intrarecord command(s). TX must be specified on the same line as the intrarecord command(s).
[· · · ;] **TY**[; · · ·]	Types the sequence number and contents of the record currently open for editing under control of an SE, SS, or ST command. Option: The dots indicate that other commands may be present on the line.
TY n[-m][,c[,d]]	Types the sequence numbers and the contents of specified columns of one or more records beginning at record n. Options: Same as for the TC command.
X	Closes all active files and returns control to the terminal executive language (TEL). This command is equivalent to the END command.
XEQ fid	Causes Edit to obtain commands from the file specified by fid. The commands will be displayed as they are executed if ECHO is turned on.

7. DELTA

INTRODUCTION

Delta is designed to aid in the on-line debugging of programs at the assembly-language or machine-language level. It operates on object programs and tables of internal and global symbols used by the program but does not require that the tables be at hand. With or without the symbol tables, Delta recognizes computer instruction mnemonic codes and can assemble machine-language programs on an instruction-by-instruction basis. The main purpose of Delta, however, is to facilitate the activities of debugging by

1. Examining, inserting, and modifying such program elements as instructions, numeric values, and coded information (i.e., data in all its representations and formats).

2. Controlling execution, including the insertion of breakpoints into a program and requests for breaks on changes in elements of data.

3. Tracing execution by displaying information at designated points in a program.

4. Searching programs and data for specific values.

Although Delta is specifically tailored to machine language programs, it may be used to debug programs written in FORTRAN, COBOL, or any other language. Delta is designed and interfaced to the operating system in such a way that it may be called in to aid debugging at any time, even after a program has been loaded and execution has begun.

Warning: Using Delta to debug a program which has an associated shared library (such as FORTRAN libraries P0 and P1) has the following limitations:

- Data breakpoints are not monitored during execution of shared library code; therefore, changes to data breaks made from the library will not be detected.

- Transfer breakpoints are not followed into shared libraries, but continue after the return from the library routine.

- An attempt to display a location in a shared library will either display garbage (from Delta's code) or cause a 'MEMORY PROTECT' error.

- STEPMODE cannot be used to execute shared library instructions; and if attempted, STEPMODE acts as a ;G command.

The command language of Delta is cryptic and highly encoded, but is easily learned and used by the professional programmer. It is similar to the DDT (Dynamic Digital Debugging Tool) language family that has been used on a variety of machines for the last decade.

There are two versions of Delta:

1. A user version with codes and restrictions appropriate to multiple on-line users operating in the slave mode from on-line terminals.

2. An executive version for system debugging that operates in executive mode under control of one of the operator's consoles.

Differences in the language syntax of the two versions are few and are noted in this chapter. The main orientation of the chapter, however, is towards the user version of Delta. Instructions for calling the executive version of Delta are given in the UTS/Reliability and Maintainability Technical Manual, 90 19 90.

CALLING DELTA

The user version of Delta may be brought in at the time the user loads his program into core for execution or by direct call after execution begins. Delta also may be brought in without prior program loading for writing and checking short Meta-Symbol or machine language programs. In all cases, TEL commands are used to call Delta. The commands are outlined briefly below.

1. To bring in Delta at program load time, the user gives the command "START lmn UNDER DELTA". The user may also call Delta at program load time by preceding a START or lmn command with the command U. The U command causes the words UNDER DELTA to be inferred in the START or lmn command that immediately follows. When Delta is brought in at program load time, control goes to Delta and the user may examine and modify his program before passing control to it.

2. To bring in Delta after a program has started, the user returns to the TEL level by using the terminal command Y^c (by simultaneously pressing the control shift and the Y keys) and then gives the TEL command DELTA.

   Note: Attempting this approach may result in the message:

   A100 DON'T TRY TO DEBUG A SHARED PROCESSOR

   This means that execution of the user's program had not actually begun when the Y^c command was given and a processor such as LYNX was operational instead. If this happens, the user may either retry this approach and wait for execution of this program to actually begin or use the approach outlined in 1 (bringing Delta in at program load time).

3. To bring in Delta without prior program loading, the user simply gives the TEL command DELTA. Writing programs with Delta is discussed at the end of this chapter.

4. To bring in Delta after a program has aborted, the user simply gives the TEL command DELTA. The fatal condition can then be examined.

Delta responds to these commands by typing "DELTA HERE". In the user version it follows this by its prompt character, the bell. (The executive version of Delta does not have a prompt character, nor does the user version when connected to a keyboard display.) Delta is then ready to accept a command.

EXITING DELTA

The Delta command ;E terminates Delta and the program running under Delta and returns control to TEL. The Y^c control combination may also be used to return to TEL.

EXIT CONTROL

If an Exit Control routine has been established by the user program, it may be entered with the ;X command. When Exit Control is entered with the ;X command, registers 0 and 1 contain the addresses of the TCB and PSD respectively, but no other exit control registers will be set. Using the ;E command to exit DELTA will cause any user exit control routine to be bypassed. A Y^c, QUIT sequence will cause the Exit Control routine to be entered.

PREREQUISITES

There are three Delta restrictions the user must be aware of when he writes a program that will be run under Delta:

1. All assembly language mnemonics are reserved words in Delta and may not be used as symbols for instruction or data tags unless used only in address fields (e.g. B B).

2. The symbols used for the program should not exceed eight characters and must otherwise follow the rules for symbols in Meta-Symbol, except that the first seven characters of all the symbols should be unique. This is necessary because symbols are carried in Delta's symbol table as seven characters. Thus, symbols that originally were longer than seven characters are indistinguishable from each other if the first seven are the same. Symbols with eight or more characters in which the first seven are not unique must be accessed with 8 to 63 characters input. Delta prints that symbol with seven characters followed by *'s for the remaining characters.

3. A Meta-Symbol program should be assembled with the option SD if the user wishes to use internal symbolic references while debugging. (The SD option causes the assembler to produce debugging object code for internal symbols for use with Delta.) Also, at run time the user must request that the internal symbol table with this code be made available to Delta by using the Delta command s;S where s is the name of the file. (See the section "Symbol Table Control".)

SAVING PROGRAM MODIFICATIONS

When a user debugs a program under Delta, he may make modifications to the program code and symbol tables. These modifications only affect the core image of the program and are not saved unless the user returns control to TEL (by issuing the Y^c command or by depressing the break key four times) and then issues a SAVE command. (See SAVE command in Chapter 3.) A program that contains overlays cannot be saved as one intact program and therefore this approach to making the modifications permanent is not applicable.

OVERLAID PROGRAMS RUN UNDER DELTA

Overlaid load modules may be run under Delta with certain constraints. Since overlays are loaded fresh from the load module each time, modifications to overlays must be made each time the overlay is loaded (i.e., modifications to overlays are not permanent). Whenever a new segment is loaded and overlay stops are enabled (see the ;H command), control is passed to Delta which outputs the message

SEGMENT nnnn LOADED

where nnnn is the segment name and prompts for input. Modifications (see warning above) and instruction breakpoints may then be placed in the overlay. Instruction breakpoints may be associated with an overlay only if it is in core. Delta maintains the overlay number with the breakpoint information and reestablishes that breakpoint whenever the corresponding overlay is in core. Breakpoints may be added to the root (segment 0) at any time.

SYMBOL TABLE PLACEMENT

Symbol tables are loaded into different areas based upon whether the load module was formed by LOAD, SYMCON, LYNX, or by Link. For LINKed load modules the symbol tables are loaded at the end of the users virtual memory through descending virtual memory as needed. For LOADed or LYNXed load modules, the symbol tables are loaded from the end of the users pure procedure area through ascending virtual memory as needed. The space for the symbol tables is removed from the user's dynamic data area for LOADed or LYNXed load modules and thus does not interfere with M:GCP requests. Note that the number of available dynamic data pages will thus vary depending on whether the program is run under Delta or not.

PROGRAMS WHICH PERFORM M:LINK UNDER DELTA

Programs which are run under Delta may perform M:LINK CALS to other programs. Note that if DELTA is invoked via Y^c and typing ;DELTA, symbol tables must be fetched by M:GVP. If symbol tables must be used, the load module must be run with the "START...UNDER" command. The symbol tables and Delta context page are saved along with the load module. The called program is associated with a new copy of Delta's context page and the initial entry is made to Delta when the M:LINK is completed. Delta outputs the message 'DELTA HERE' and prompts for input. Symbol tables may be obtained for the linked program. If and when the called program returns to the calling program via an M:LDTRC CAL, the original program including Delta's symbol table and context page is restored and processing continues as before. There is no automatic call to Delta following the return to the calling program; however, all breakpoints, program modifications, etc., remain in effect.

CONVENTIONS

The following conventions are used in explaining the format of the commands typed by the user:

1. Special characters, numbers, and uppercase letters stand for themselves. Thus, in the command e;G the user actually types the semicolon and the G.
2. Lowercase letters are used to indicate places where the user has a choice of things to type. The letter e, alone or postscripted, is used to represent any expression consisting of symbols, special symbols, instruction mnemonics constants, the operators plus (+) and minus (-) and space ( ). At times, other lowercase letters are used to stand for expressions when some additional mnemonic content seems desirable (e.g., n, loc, val, m).
3. The letter f stands for one of the format characters. (The format codes are listed in Table 29.)
4. Abbreviations for user keystrokes are as follows:

Characters Used in Text	User Delta Keystroke	Executive Delta Keystroke
RET	RETURN	RETURN
LF	LINE FEED	EOM
↑	SHIFT and N	&
\	SHIFT and L	¢
TAB	CTRL and I	TAB
BRK	BREAK	Sigma INTERRUPT Switch

Table 29. Format Codes

Code	Meaning
F	Symbol table specified format.
X	Hexadecimal word.
I	Signed decimal integer.
C	EBCDIC characters.
R	Symbolic instructions with symbolic addresses.
A	Symbolic instructions with hexadecimal addresses.
S	Short floating-point number.†
L	Long floating-point number.†

†User version only; both have the format xxxxx E ±yy

COMMAND DELIMITERS

The characters listed below are used as end-of-message characters and, in most cases, as commands in the Delta language. Each has a particular meaning that will be discussed in detail with the commands to which it applies.

/

=

RET

LF (This character is represented by the EOM key in the executive version of Delta.)

↑ (This character is represented by the & key in executive Delta.)

TAB

With the exception of the slash (/) and equal (=) characters, which interact immediately within a single typed line, these characters cause a carriage return and a line feed.

More than one command can be input on a command line by separating the commands with spaces. The following line contains five commands:

```
PROG;S    TAG1;B    TAG2;B    ;B    ;D (RET)
```

However, it is important to note that any command that changes the contents of a cell should be the last command on a multiple-command line.

CORRECTING TYPING ERRORS

Correcting typing errors while using Delta requires special consideration because the = and / characters cause immediate interaction with the system without proceeding to a new line. The RUBOUT key will not affect a / or = character nor any information which precedes it on a line. Canceling a line by simultaneously pressing the CONTROL and X keys or sequentially pressing the ESC and X keys may only cancel a partial line. If a / or = character appears in the line, that character and all characters preceding it will not be canceled.

In the executive version of Delta, the question mark (?) cancels the command line and the at sign (@) is the rubout command.

EXPRESSIONS

Expressions are typed by the user for location value, for parameter value, and for assembly into an instruction. Expressions are composed of

Program symbols.
Special symbols (see Table 30).
Assembly language mnemonics.
Explicit constants.
The operators plus (+) and minus (-).
The shift operator (**).
The multiply operator (*).
Space ( ).

Examples:

```
;I
.1E07
A
A+3
A+3-B
AI,1  2
STW,7     *LOC
LW,7   TAB,5
CAL1,3    LIST
B*4
.5C2A0**-2
```

Table 30. Special Symbols

Symbol	Meaning
' $ or .	Last opened cell address.
;I	Instruction counter } As set by the last entry to Delta or as changed by the user.
;C	Condition code
;F	Floating controls
;M	Search mask.
;1	Lower search bound.
;2	Upper search bound.
;Q	Last quantity typed.
;H	PSD bits 8-11. (Available only in executive Delta.)

The locations referred to by all special symbols except $ and . may be set while in Delta by using a command e;s where s is the special symbol and e is an expression indicating the value to which the location is to be set.

The space character is used to introduce the address field in expressions to be assembled into instructions. Any necessary truncations in instruction expressions are performed as in SIG7FDP and the message *TR* is output to the user.

CONSTANTS

Constants must be input in the following formats:

1. Hexadecimal - hexadecimal numbers preceded by a period.

   ```
   .1C28
   .BE3
   .FFFFFFFF
   ```

2. EBCDIC - EBCDIC characters surrounded by single quotes. (EBCDIC text strings must consist of no more than four characters. If fewer than four characters are specified, the characters are right-justified and zero-filled.)

   ```
   '%ERR'
   'LOC2'
   'TOM'
   ```

3. Decimal - numerics only.

   ```
   1234
   -250
   10
   ```

Hexadecimal and decimal constants are output in the same format that they are input. EBCDIC constants are output as EBCDIC characters without the surrounding single quotes. Non-printing EBCDIC characters may also be output, including the EOT (end of transmission (04)) character, which will turn off some types of terminals.

DELTA COMMANDS

EXPRESSION EVALUATION: THE = COMMAND

Expressions consisting of program symbols, special symbols, assembly language mnemonics, explicit constants, and the operators plus (+) and minus (-), and space ( ) may be evaluated by use of the = command. When a program symbol is evaluated, the result is its absolute address. When a special symbol is evaluated, the result is the value that the symbol is set to. When an explicit constant is evaluated, the result is its numeric equivalent. When an assembly language instruction is evaluated, the result is its machine language equivalent.

The basic = command is

```
e =
```

Note that no carriage return is given. Delta responds immediately to the = character and evaluates the expression e.

In this command, no format is specified for typing the expression evaluation, so the default format is used. Usually the default format is X (hexadecimal), but the default can be changed by one of the variations of the = command (which will be discussed shortly).

Examples:

```
2+2 = .4
5+5 = .A
TOT = .C12E
AI,6  1 = .20600001
```

The user may temporarily change the output format with the following = command:

```
e(f =
```

where f specifies a particular format code selected from those listed in Table 29, "Format Codes". The temporary change only affects the = command in which it is given.

The default format for output can be changed by the command:

```
    (f; =
```

where f specifies a particular format code selected from those listed in Table 29, "Format Codes". The new default will be retained until another (f;= command is given. The original default setting of the output conversion format is X.

Example:

```
    5+5 = .A (RET)
    (I; =    6+7 = 13   (The default format is changed
                         to integer.)
```

The last expression typed by Delta may be evaluated simply by typing the = character. In the example below, Delta types the expression BAL,5 SUB as a result of the command ALPHA/ (which is discussed in the next section). Then the entire expression BAL,5 SUB is evaluated and the results are typed as a result of the = command.

```
    ALPHA/ BAL,5 SUB = .6A5006B3
```

MEMORY CELL OPENING AND DISPLAY: THE /, TAB, and \ COMMANDS

The slash (/) character is a command to Delta to open a memory cell and display its contents. There are several variations of the slash command and these will be discussed below. In each command the cell to be opened and displayed is indicated by an expression (designated by an e in the instruction formats)

The basic / command is

```
    e/
```

Note that no carriage return is given. Delta responds immediately to the / character and opens the cell, displaying its contents on the same line. In this command, no format is used. Usually this default format is F (symbol table specified format), but the default can be changed by one of the variations of the slash command (which will be discussed shortly). If the default is F, then the symbol table is searched to find a symbol at the same or at the closest smaller location (within a hexadecimal offset that may be specified by the Delta ;R command discussed later) than the indicated address and the data type associated with the symbol found is used to control output. If no symbol is found within the range, the default is R (symbolic instruction).

Examples:

```
    .C125/    .34
    A1/       BAL,6  ALPHA
    A+1/      STW,5  BETA
    BETA/     ABCD
```

The user may temporarily change the output format with the following slash command:

```
    e(f /
```

where f specifies a particular format code selected from those listed in Table 29, "Format Codes". The temporary conversion type is retained for all slash commands until the next RET command is given or another format change is specified. (The temporary conversion type is retained over any following LF,↑ , /, and TAB commands.)

Examples:

```
    X(X/    .C1     (hexadecimal conversion)
    X(C/    A       (EBCDIC character conversion)
    X(I/    193     (decimal integer conversion)
```

Below is an example which shows that temporary conversion types are retained until the next RET command is given:

```
    D(X/    .3230C122
    V/      .2030C122 (RET)   (D,V, and Z are locations
    Z/      AI,3   AA (RET)   of program instructions)
```

The default format for output can be changed by the command

```
    (f;/
```

where f specifies a particular format code selected from those listed in Table 29, "Format Codes". The new default will be retained until another (f;/ command is given. The original default setting of the output conversion format is F

Example:

```
    X/      .C1 (RET)
    (C;/    X/    A
```

The cell addressed by the last expression typed by Delta may be opened and displayed by typing a TAB (produced by simultaneously pressing the CTRL and the I keys for the user version or by pressing the TAB key for executive Delta).

In the example below, the cell DCT8 is opened and displayed.

```
    ALPHA/    LW,5    DCT8 (TAB)
    DCT8/  .32      (The carriage return was automatically
                     provided by Delta.)
```

The format for display is by default only and is the same default as for the slash command.

If the user types a slash by itself, the cell addressed by the last expression typed by Delta is displayed but not opened. In the example below, ALPHA remains the open cell even though the contents of cell DCT8 are displayed.

```
    ALPHA/    LW,5    DCT8    /    .32
```

Conversely, a cell may be opened without displaying its contents by the use of the \ command (produced by simultaneously pressing the SHIFT and the L keys for the user version or by using the ¢ key for executive Delta). The format of the command is

```
e\
```

In the example below, the cell SUM is opened but not displayed. Then SUM is set to zero.

```
SUM\     0 (RET)
```

Opening a cell without displaying its contents is convenient when the user wishes to insert new contents in memory and is not interested in the current contents.

If the user wishes to store data into a page that is not assigned to his program, the \ command will request that the monitor assign the page. (This is particularly useful when using Delta to write new programs.) For example:

```
.18000\
```

If the page at .18000 is not assigned to the program, the monitor will assign it (if possible). Also, the cell at .18000 will be opened.

More than one cell may be displayed by using the following command:

```
e1,e2/
```

where

e1 is an expression which identifies the lower address of a range of cells to be displayed.

e2 is an expression which identifies the upper address of a range of cells to be displayed.

Following the display of these cells, the upper limit cell is open for change. In the following example, ALPHA + 2 is open for change.

```
ALPHA,ALPHA+2/    BAL,4 SUB
ALPHA+.1/     STW,5     DCT2
ALPHA+.2/     AI,6       .100
```

Note that Delta types the word increments as hexadecimal numbers.

Temporary change in the output format may be added to the above command as shown below:

```
e1,e2 (f/
```

where f specifies a particular format code selected from those listed in Table 29, "Format Codes".

Example:

```
100, 101(X/     .58000100
101/            .68000200
```

If the user wishes to interrupt a display that is too long, he presses the BREAK key and the remaining output is discarded. The last displayed cell is opened. The INTERRUPT switch on the Sigma Processor Control Panel accomplishes this in the executive version of Delta.

MEMORY MODIFICATION: THE RET, LF, ↑, AND TAB COMMANDS

These four commands allow the user to store a typed expression for word value into the currently open memory location -- opened by /, \, or one of the modification commands LF, ↑, or TAB. If no expression precedes the command character, the action taken is as described, except that the open cell remains unchanged.

The RET command causes an expression to be assembled and stored in the open memory cell. Carriage return (RET) and new line (LF) characters are sent to the terminal, and temporary display modes are reset to default values. The format of the RET command is

```
e (RET)
```

Example:

```
A/    BAL,4  JWS  BAL,4  GEB (RET)
A/    BAL,4  GEB (RET)
```

(The expression BAL,4 GEB is assembled and stored into A).

```
JED/    EXU  LS  (X/  .68000643/  .78C (RET)
```

(The contents of JED are typed. Then the contents of LS are typed in hexadecimal. JED remains the open cell. Then the contents of .0643 are typed.)

```
./    EXU  LS
```

(The contents of JED are tvped. The . is a special symbol (see Table 30) that specifies the last opened cell address.)

Note that a temporary display format was established by the (X/ which carried over until reset by the carriage return (RET) command.

When the user terminates an expression with the new line (LF) command, the value of the expression is stored in the currently open cell, that cell is closed, a new line is produced at the terminal, and the cell with the next higher location value is opened. The type of command used for initial cell opening is preserved and carried forward on succeeding openings as is the display format.

The format of the new line (LF) command is:

```
e
```

Example:

```
A(I/      435  436 (LF)
A+.1/      763 (LF)
(A+.1 is displayed but remains unchanged.)
(A+.2/     7689  7000 (RET)
EM\      STM,4  ERS (LF)
EM+.1\      BAL,6  LP (LF)
EM+.2\      BGE   GAP (RET)
```

For the executive version the EOM (end-of-message) key replaces the LF key.

The e↑ command is the same as the LF command except that the cell with the next lower location value is opened. For the executive version, & replaces ↑.

Example:

```
EM+4/     0  B  GW↑
EM+.3/    0  AI,3   1 (RET)
```

The TAB command causes the typed expression to be stored in the currently open cell, and that cell is closed. Following output of a carriage return, the cell addressed by the most recently closed cell is opened and displayed (only the address is displayed in the \ mode). The effect is like that of a RET command followed by a ;Q/ command (see Table 29, Special Symbols). The format of the TAB command is:

```
e (TAB)
```

The TAB command is useful for patches:

```
A/     BAL,5  SUB (LF)
A+.1/     STW,6  BETA   B   PATCH (TAB)
(The carriage return is performed as part of the TAB
command.)
PATCH/      .0   AI,6   1 (LF)
PATCH+.1/      .0   STW,6   BETA (LF)
PATCH+.2/      .0   B   A+2 (RET)
```

SYMBOL TABLE CONTROL: THE ;U, ;K, ;S, !, AND<>COMMANDS

There are two types of symbol tables in Delta:

1. Constant (Internal to Delta).

2. User associated or defined.

The first type of table is always present in Delta and consists of the Meta-Symbol instruction mnemonics and a list of special symbols (see Table 30) associated with program debugging.

The second type of table consists of a set of global symbols (those defined by DEF directives) and a set of internal symbol tables, one for each ROM loaded (although some may be combined by Link). (See Chapter 8 on Link.) The internal symbol tables are filed under the name of the file from which the ROM was loaded.

The user must specify that the internal symbol table is to be loaded if he wishes to debug using internal symbolic tags. The command for specifying this is

```
s;S
```

where s is the name of the file from which the ROM was loaded. This command causes Delta to load the internal symbols from the program loaded from file s, and these internal symbols replace, for reference purposes, any previously selected internal symbol set. An example of the s;S command is

```
C=
?2
(Delta is confused because no internal symbol table
had been previously loaded. The internal symbol C
is not recognized.)
BIN;S (RET)
(The internal symbol table is loaded.)
C=.C125
(The symbol C is now recognized.)
```

The ;S command alone loads the global symbol table. The user must specify this command if he wishes to debug using global symbolic references.

The user may wish to release to the system the pages used for symbolic tables. The command ;K releases the pages containing the global and internal symbol tables. The command ;KG releases only pages containing the global symbol table and the command ;KI releases only pages containing the internal symbol table. Other uses of the ;K command are

s;K   prevents use of the symbol s in constructing output. The symbol is still recognized when typed in. Symbol s is returned to use if the user reloads the symbol table.

s;KD   prevents use of the symbols for any purpose. |

;K   removes all symbols from the symbol table. The lists of instruction mnemonics and special symbols are not erased. Individual internal symbol tables are recoverable using the s;S command. Global symbols are restored by ;S.

Undefined symbols in the loaded programs are printed by Delta when the ;U command is given. Undefined symbols within the range of an assembler LOCAL directive are lost. They are given a value of zero in the loaded code and do not appear when the ;U command is given. (In the executive version of Delta, the ;U command toggles the map bit in the current PSD. Thus it controls whether executive Delta references refer to mapped or unmapped mode.)

Symbols may be defined by the user at any time during his debugging session. Symbols so defined are added to the global symbols associated with the program load. Commands for adding symbols to the global symbol table are

s(f! adds the symbol s to the global symbol table with the location value of the currently open cell and format type f. (See Table 29, "Format Codes".) If format type f is omitted, symbolic instruction (R) type is assumed. For example,

```
C+1/  0  LOC!  LW,4  TAB (RET)
```

In this example, location C+1 is given the name LOC. Then the contents of LOC are changed to contain the assembled instruction LW,4 TAB.

Note that if a format code is specified for a slash command (e(f/), it is retained until the next carriage return and meanwhile the format specified is applied to any command.

Example:

```
INST/  CI,2  .4 (RET)
INST(X/  .21200004  SYM! (RET)
INST/  CI,2  .4  SYM/  .21200004
```

The cell INST was displayed in its default format. The instruction at INST was then displayed in hexadecimal format and then the cell INST was assigned an additional name, SYM, with the display format for SYM being X.

e(f<s>[K] adds the symbol s to the global symbol table with value defined by the expression e and format code f. In addition to the format codes of Table 29, the letter K may be used to indicate value is to be a constant. If f is omitted, R is assumed. If the final angle bracket is followed by a K, the symbol is flagged as a control section type symbol in the symbol table. K may not be used as the format code if K is specified following the final angle bracket.

SINGLE LINE MACROS

Since the symbol table definition gives a 32-bit value to constant symbols, it may be used as a macro-definition facility for single-word values.

Example:

LI,3 0(K<CLEAR> (RET)	The symbol CLEAR now represents the instruction LI,3 0.
AA\ CLEAR (RET)	The cell AA is opened and its content is set to the instruction LI,3 0.
MM\ CLEAR (RET)	The content of MM is also set to the instruction LI,3 0.

EXECUTION CONTROL: THE ;G, ;P, ;X, AND) COMMANDS

The four commands described in this section allow the user to begin execution of his program and to resume execution of the program if it is interrupted.

Execution is started by typing the [e];G command, where e is an expression which identifies the starting location. The expression e may be omitted, in which case execution will begin at the first instruction of the program.

Example:

```
BEGIN;G (RET)
;G (RET)
```

Execution can be stopped in four ways:

1. A breakpoint. (Breakpoints are discussed in the next section.)
2. A user interruption via the BRK key.
3. An error causing a machine trap (illegal instruction, memory protection violation, etc.).
4. A normal program exit.

In each case the values of ;I, ;C, and ;F (see Table 30 on Special Symbols) are set, the cause of the stop is reported by an appropriate message, and terminal control returns to the user.

Example:

```
BRK AT .5C3
PRIVIL INSTR AT .77B
;I = .77B
```

Proceeding from a stop condition is accomplished by typing the ;P or ;G command. Execution continues from the location specified by the current value of ;I (i.e., where execution left off). The ;P command has an optional special format for use with instruction breakpoints. This is discussed in the section on breakpoints. For user interruptions via the

BRK key, the ;P and ;G commands cause execution to continue as if the interruption had not occurred.

```
BRK at .68C
;P
```

Proceeding from a machine trap causes reexecution of the violating instruction and another trap.

```
MEM PROTECT FAULT AT .74B
;P
MEM PROTECT FAULT AT .74B
```

The e;X command assembles and executes the expression e. The expression e must be an assembly-language instruction.

Examples:

```
LH,3  TABLE+4;X
STB,6  *LOC;X
```

If the expression does not result in a legitimate instruction, an error message is typed.

In most cases the instruction is executed and then terminal control returns to the user. However, if the expression is a branch instruction, control goes to the user's program (or causes a memory violation). Thus, the commands B GO;X and GO;G are equivalent. If the expression is a subroutine jump, the subroutine is entered. If the subroutine returns normally (i.e., to the calling location plus 1, 2, or 3), control returns to Delta and terminal control returns to the user. If the return is to other than the calling location plus 1, 2, or 3, the results are unpredictable.

The) command controls step mode execution. It executes the instruction in the currently open cell and opens and displays the next program step. If the instruction executed by) causes a branch, the effective branch address specifies the location to be opened and displayed. By using the / command to open and display a location and repeatedly issuing the) command, the user can proceed step-by-step through his program. Step mode may not be used to execute instructions in the registers.

OVERLAID PROGRAM CONTROL: THE ;H COMMAND

The ;H command controls stopping in DELTA at the completion of segment loading. Each time the command is given it toggles the STOP/NOSTOP on segment loaded flag. This flag is initially set to NOSTOP. If overlay breakpoints are desired, the ;H command should be given so the user can be informed when segments have been loaded and are in-core (and thus available for breakpoint setting). Once the desired breakpoints have been defined, the ;H command may be given to reset the segment loaded flag back to NOSTOP. When an overlay is loaded and the flag is set to STOP, Delta outputs the message

```
SEGMENT 'segname' LOADED
```

and prompts for input. Either the ;G or ;P command may be given to return control to the user program.

BREAKPOINTS: THE ;B, ;T, ;D, AND ;Y COMMANDS

Delta provides the user with multiple breakpoints of three types:

1. Instruction breakpoints.
2. Data breakpoints.
3. Transfer breakpoints.

The BRK key also causes a break in execution and is discussed in this section.

Eight instructions and eight data breakpoints are available to the user. Transfer breakpoints are limited only by options within the transfer breakpoint command.

As each breakpoint is reached, a small amount of information is printed out, giving the breakpoint location and an associated value. An optional "trace" mode allows execution to continue automatically after the breakpoint report to provide a flow-trace of both execution control and variation of data values.

INSTRUCTION BREAKPOINTS

Instruction breakpoints allow the user to halt execution at specified locations in the logical flow of his program. Eight instruction breakpoints, numbered 1 to 8 may be set. The command has the format

```
e[,n][,loc][,val];B[r]
```

where

e specifies the location of an instruction. The breakpoint stop occurs just before execution of the instruction at e. e may not specify a register.

n specifies the number of the breakpoint. If n is not specified, Delta assigns the next available breakpoint. If all instruction breakpoints are used, the error message NONE is typed. The user may then release one of the eight instruction breakpoints he has set and try again. (Releasing breakpoints will be discussed shortly.)

loc specifies a location, the contents of which are to be displayed when the breakpoint is reached. Registers as well as core locations can be displayed.

val specifies a value that is compared to the value in loc. The parameters val and r must both be present if one is present. The parameter val is discussed further in the r description.

r specifies a relationship such as less than or equal to. When r and val are specified, a breakpoint will occur only whenever the contents of the memory location of loc is in relation r to val before the instruction at e is executed. If no r and val specifications are given, a breakpoint occurs for all executions of the instructions at e.

The letters used for r and their meanings are:

LS L	(loc) c<val	Contents of loc are less than value.
EQ E	(loc) c = val	Contents of loc are equal to value.
GR G	(loc) c>val	Contents of loc are greater than value.
GQ GE	(Loc) c≥val	Contents of loc are greater than or equal to value.
LQ LE	(loc) c≤val	Contents of loc are less than or equal to value.
NQ NE	(loc) c≠val	Contents of loc are not equal to value.

The following list shows the format of the command when various parameters are omitted:

```
e;B
e,n;B
e,,loc;B
```

(This page intentionally left blank)

The breakpoint stop occurs just before execution of the instruction at e. When the breakpoint is reached, Delta prints the number and type of breakpoint, its location, and optionally the contents of the location specified by loc.

Examples:

```
A+3,1;B   A;G (RET)
1;B>A+.3
A+8,1,FF;B   ;G (RET)
1;B>A+.8  FF/  .54
```

When stopped at a breakpoint, the user may examine and modify his program as appropriate and then continue from the point of interruption by giving the command

```
;G  or  ;P  or  n;P
```

If the command n;P is given, program execution resumes as with the ;P command but the breakpoint that caused the interrupt will be passed n times before the break occurs again.

Example:

```
PH+8,2,R2;B   PH;G (RET)
2;B>PH+8 R2/  .4   ;G (RET)
2;B>PH+8 R2/  .5   ;P (RET)
2;B>PH+8 R2/  .6   5;P (RET)
2;B>PH+8 R2/  .12
```

(The breakpoint was passed five times before it caused this interrupt.)

If the user wishes to trace a particular instruction, he may give any of the four forms of the breakpoint command and specify the trace mode with a T following the B. That is,

```
e,n;BT
e;BT
e,n,loc;BT
e,,loc;BT
```

In this mode when the instruction e is reached, the breakpoint reporting information is printed and execution continues automatically.

Example:

```
A+3,4,5;BT   A;G (RET)
4;B>A+3  5/  54
4;B>A+3  5/  -1
4;B>A+3  5/  -175
```

The trace mode may be set after a breakpoint occurs with the ;T command, which sets the trace mode at the current breakpoint instruction.

Instruction breakpoints may be removed by

1. Giving an instruction breakpoint command that specifies the same breakpoint number as the instruction breakpoint to be removed.

   Example:

   ```
   AA,2;B (RET)
   .
   .
   .
   FF,2;B (RET)         (There is no longer a breakpoint
                         at AA.)
   ```

2. Giving the command n;B that specifies that the nth instruction breakpoint is to be removed.

3. Giving the command 0;B that specifies that all instruction breakpoints are to be removed.

The current instruction breakpoints may be listed for inspection with the ;B command. The list has the following form for each established breakpoint:

```
n[T]loc [overlay][display]
```

where

n is the breakpoint number

T indicates that the trace mode is set for that breakpoint.

loc is the breakpoint location.

overlay is the overlay number associated with this breakpoint (if any).

display is the address to be displayed when the breakpoint occurs.

CALs, XPSDs, or LPSDs that depend on following calling sequences will not operate properly if they have an instruction breakpoint on them. BALs are not limited in this way, unless the return is other than to BAL+1.

DATA BREAKPOINTS

Data breakpoints allow the user to halt execution when a specified memory location changes value in a specified way. Eight data breakpoints (numbered 1 through 8) may be set. The command has the format

```
e[,n][,val][,m];D[r]
```

where

e specifies a memory location. When the contents of this location change, a break will occur (unless other optionally specified requirements are not met). e may not specify a register.

n specifies the number of the breakpoint. If n is not specified, Delta assigns the next available breakpoint. If all data breakpoints are used, the error message NONE is typed. The user may then release one of the eight data breakpoints he has set and try again. (Releasing breakpoints will be discussed shortly.)

val specifies a value that is compared with the value in e. The parameters val and r must both be present if either one is present. The parameter val will be discussed further when r is discussed.

r specifies a relationship such as less than or equal to. When r and val are specified, a breakpoint will occur only whenever the contents of the memory location at e is in relation r to val. If no r and val specifications are given, a breakpoint occurs for all changes in the data and if a mask m is specified, it is ignored.

The letters used for r and their meanings are

LS L	$(e)_c < \text{val}$	Contents of e under m are less than value.
EQ E	$(e)_c = \text{val}$	Contents of e under m are equal to value.
GR G	$(e)_c > \text{val}$	Contents of e under m are greater than value.
GQ GE	$(e)_c \geq \text{val}$	Contents of e under m are greater than or equal to value.
LQ LE	$(e)_c \leq \text{val}$	Contents of e under m are less than or equal to value.
NQ NE	$(e)_c \neq \text{val}$	Contents of e under m are not equal to value.

m specifies a mask. If m is specified, the contents of e are masked under m before being compared with val. The default mask is all one's.

Some specific variants of data breakpoint commands are given below.

e,n;D Sets data breakpoint n. Terminal control returns to the user after each change in the contents of e and printing of the data breakpoint message.

e;D Sets next available data breakpoint. If all data breakpoints are used, the error message NONE is typed. Terminal control returns to the user immediately after each change in the contents of e and printing of the data breakpoint message.

e,,val;Dr Sets next available data breakpoint with value, val, and relation, r. Terminal control returns to the user when the contents of e stand in relation r to the value val and the data breakpoint message has been printed.

e,,val,m;Dr Same as above except that the contents of e are masked by the mask m before being compared with val.

Some sample breakpoint settings are:

```
A,1,3;DGR

A+5,2,.FF,.FF;DEQ

AB,3;D

SDS,4,CSC;DGE
```

A T or trace parameter applies to all data breakpoint commands in the same way and with the same effects as described above for instruction breakpoints. For example,

```
A,1,3;DTGR
```

Also the command ;T may be given to set the trace mode at the current breakpoint (which just caused an interrupt).

The output resulting from a data breakpoint has the form

```
n;D > loc e/cont
```

where

n is the number of the breakpoint.

loc is the location of the data modifying instruction.

e is the data address in question.

cont is the new value as just modified.

Example:

```
4;D > ADD   SUM/.14
```

When stopped at a data breakpoint, the user may examine and modify his program as appropriate and then continue from the point of interruption by giving the command

```
;G
```

or

```
;P
```

These commands are discussed in the previous section, "Instruction Breakpoints". (For data breakpoints, the ;G command is effectively the same as the ;P command.)

Data breakpoints may be removed by

1. Giving a data breakpoint command that specifies the same breakpoint number as the data breakpoint to be removed.
2. Giving the command n;D that specifies that the nth data breakpoint is to be removed.
3. Giving the command 0;D that specifies that all data breakpoints are to be removed.

The current data breakpoints may be listed for inspection with the command ;D. The list has the following form for each established breakpoint:

```
n[T]loc   cond   value   mask
```

where

n is the breakpoint number.

T indicates that the trace mode is set.

loc is the breakpoint location.

cond is the breakpoint condition relation.

value is the breakpoint value.

mask is the mask under which the data is tested.

The data breakpoint will detect changes caused by M:READ calls, but other CALS which might cause a data breakpoint location to be altered (such as M:TIME or M:KEYIN) are not checked. Changes in a temp stack or a stack pointer doubleword are also detected.

TRANSFER BREAKPOINTS AND INTERPRETIVE EXECUTION

Transfer breakpoints allow the user to halt or trace execution when a branch instruction is encountered that branches when executed. This command differs from the other two breakpoint commands in that it initiates execution as soon as the command is decoded and processed. The format of the transfer breakpoint command is

[loc][,option_1][,option_2];Y

where

loc specifies a location at which to begin execution of the program. The default value is the value of the current location counter.

option_1 indicates whether or not an interrupt should be allowed to occur at the branches specified in the special action table (SAT) which will be described below. If $\text{option}_1 = 0$, then all branches except those specified in the SAT are to be processed as possible transfer breakpoints. If $\text{option}_1 = 1$, then only those branches specified in the SAT are to be processed as possible transfer breakpoints. If this option is omitted and the SAT contains no entries, then all branches are processed as possible transfer breakpoints. If the option is omitted and the SAT does contain entries, then the default value for the option is zero (so that all branches except those specified in the SAT are to be processed as possible transfer breakpoints).

option_2 indicates whether or not BDR and BIR branches are to be processed as possible transfer breakpoints. If $\text{option}_2 = 0$, then BDR/BIR branches are not to be processed as possible transfer breakpoints. If $\text{option}_2 = 1$, then BDR/BIR branches are to be processed as possible transfer breakpoints. The default value is 0.

The following list shows the format of the command when various parameters are omitted:

;Y

loc;Y

loc,option_1;Y

loc,,option_2;Y

,option_1;Y

,,option_2;Y

,option_1,option_2;Y

When a break occurs as the result of the transfer breakpoint command, the following message is output:

loc1 ⟶ loc2

where

loc1 is the address of the branch instruction that just branched.

loc2 is the address of the instruction to which the program branched.

Execution may be continued with the ;P or ;G command. The ;P command with a proceed count (n;P) is not meaningful in the transfer breakpoint mode.

If the user wishes to use the trace mode with the transfer breakpoint command, he may give any of the forms of the command and specify the trace mode with a T following the Y. For example:

;YT

loc,,option_2;YT

loc,option_1,option_2;YT

In this mode, when a breakpoint occurs, the breakpoint reporting information is printed and execution continues automatically.

The trace mode for all transfer breakpoints may be set after a transfer breakpoint break occurs. The command which sets the trace mode is ;T.

The transfer breakpoint mode may be turned off with the command:

0;Y

Special Action Table (SAT). The special action table lists up to eight locations in the user's program. These locations are meaningful only if they contain branch type instructions. The action to be taken depends on option_1 of the transfer breakpoint command.

The following command enables the user to set entries in the SAT:

loc1[,loc2[,loc3[,loc4]]];YS

The command enters the specified locations in the SAT if space is available.

The command

```
    loc1[,loc2[,loc3[,loc4]]];YR
```

releases specified locations from the SAT.

The command ;YR releases all SAT entries. The command ;YD displays the SAT.

BRK KEY BREAKPOINTS

At any time during program execution the user may halt his program by pressing the BRK key. A message is printed for the user, giving the location of the breakpoint. If the user hits the BRK key while his program is in execution, the message is:

```
    BRK  AT  loc
```

After such a breakpoint, the ;P or ;G command continues execution.

If the breakpoint occurs while Delta is executing, the message is

```
    BRK  IN  DELTA
```

The user may then give any of the Delta commands.

Since certain portions of Delta are sensitive to the break key, Delta maintains a flag indicating whether breaks are inhibited or not. If the inhibit flag is on, Delta does not honor the break until completion of the break inhibited code. If Delta is processing (or about to process) a trap or abort for the user, that processing takes precedence over the break key breakpoint. In this last case Delta will output the trap or abort message and prompt for input rather than honor the break key. The break key is always honored if the user is in control. Note that several lines of output (from a trace, for example) may be output prior to the break message.

MEMORY SEARCH AND MODIFICATION: THE ;W, ;N, ;M, AND ;L COMMANDS

There are two search commands, e;W and e;N. The e;W command searches for values which match the expression e and displays the location and contents of each cell containing the value. The e;N command searches for cells that do not contain the expression e and displays their location and contents.

The search is carried out between the limits determined by the symbol table values of ;1 and ;2. The special symbols ;1 and ;2 identify the lower and upper search bounds respectively. The initial value of ;1 is the lowest current user data area address, and the initial value of ;2 is the highest current user data area address. Usually the initial value of ;2 is greater than the last address of the user's program and this causes a trap to occur when a search is requested. Therefore, the user should always set limits on the area in which the search is to be done by using the e;1 and e;2 commands. The field e is an expression which specifies the bound location. An example is given below:

```
    AA;1  EE;2  'ABCD';W (RET)
```

In the example, each cell between AA and EE will be searched for the EBCDIC value ABCD. The location and contents of each cell containing that value will be displayed.

Both bounds may be set by one command, the ;L command. The format of the ;L command is

```
    e1,e2;L
```

where e1 specifies a value for ;1 and e2 specifies a value for ;2. The example above might also be written

```
    AA,EE;L  'ABCD';W (RET)
```

When the user sets the search bounds, they remain set at the specified value until they are reset by the user. The bounds do not revert back to their initial values after a search has been performed.

The search may examine entire cells or portions of cells. This is determined by a mask which is identified by the special symbol ;M. The initial value of ;M is all ones, so that entire cells will be examined. The mask ;M may be reset by the ;M command which has the format

```
    e;M
```

The expression e is used to set the bits of ;M to a particular pattern of ones and zeros. Only those bits corresponding to the one bits of the mask will be examined when a search is performed. For example:

```
    .FF000000;M
```

The mask will be set so that only the first eight bits of each cell will be examined to see if they match the value being searched for.

Like the search bounds, the value of the mask ;M will not be changed until another ;M command is given.

In the following example, only the last byte of the cells AA through EE will be examined. Those containing the EBCDIC value 'D' will be displayed.

```
    AA,EE;L  .000000FF;M  'D';W (RET)
```

(In the .000000FF;M command, the leading zeros are not required.)

The user may express values to be searched for in their assembly-language format or in their machine-language format. In the example below, all words between ABC and ABC+.100 with the last 17 bits equal to the address of the ERR will be displayed as shown.

```
.1FFFF;M    ABC,ABC+.100;L    ERR;W (cr)
ABC+.3/     BAL,4    ERR
ABC+.A/     BAL,4    ERR
ABC+.D/     BAL,4    ERR
ABC+.6A/    AWM,1    ERR
```

A second value may be specified in the ;W and ;N commands so that the formats of the commands are

```
e1,e2;W and e1,e2;N
```

The e2 field specifies a value which will be stored through the mask ;M into all locations that meet the specified condition (i.e., match or mismatch). Locations meeting the conditions will be displayed after the substitution has taken place. The following example is the same as the example above, except that the symbol OUT will be substituted for ERR. (OUT must be a defined symbol within the program.)

Example:

```
.1FFFF;M    ABC,ABC+.100;L    ERR,OUT;W (cr)
ABC+.3/     BAL,4    OUT
ABC+.A/     BAL,4    OUT
ABC+.D/     BAL,4    OUT
ABC+.6A/    AWM,1    OUT
```

The user may interrupt an in-progress search by pressing the BRK key. Delta halts the search and returns terminal control to the user.

MEMORY CLEARING: THE ;Z COMMAND

The ;Z command is basically used to clear (i.e., set to zeros) specified areas of memory. The basic format of the ;Z command is

```
e1,e2;Z
```

where expression e1 is the lower limit and expression e2 is the upper limit of the memory area to be cleared. An error results if the value of e2 is less than that of e1. Also e1 and e2 must not specify addresses outside of the user's area in memory.

A third field may be added to the ;Z command so that the format is

```
e1,e2,v;Z
```

The field v specifies a value to be stored into each of the memory cells in the area delimited by e1 and e2. In this way, the ;Z command may be used for purposes other than clearing memory.

Examples:

```
A,A+5;Z (cr)
.1CE0,.1CF0;Z (cr)
ALPHA,ALPHA+2,1;Z (cr)
```

(Stores the value 1 into the three memory cells ALPHA, ALPHA+1, and ALPHA+2).

DISPLAY MODES: THE ;A ;R ;RK AND ;AM COMMANDS

The ;R and ;A commands control the way in which Delta displays location values when typing the contents of cells. The mode display is either relative (;R) or absolute (;A). When in the relative mode, Delta looks up location values in the symbol table and displays the symbol if one corresponds exactly to the value. If no exact correspondence is found, Delta displays the symbol with the next smaller value followed by a word offset in hexadecimal. If the mode is absolute (;A), then location values are displayed as hexadecimal numbers. Note that these commands control the display of location values but not the display of the address portion of instructions contained in those locations. Examples of the ;R and ;A commands are shown below:

;R Display Example:

```
A,A+5/    LI,1     .10
A+.1/     CW,1     K45
A+.2/     BGE      ZZZ
A+.3/     AI,1     1
A+.4/     B        A17
ZZZ/      STW,2    BR13
```

;A Display Example:

```
A,A+5/    LI,1     .10
.5CD/     CW,1     K45
.5CE/     BGE      ZZZ
.5CF/     AI,1     1
.5D0/     B        A17
.5D1/     STW,2    BR13
```

The ;R command may be preceded by a value (n;R) that sets the maximum offset to be used in address output. If no symbol lies within "offset" of the value, the address is printed as absolute hexadecimal. Thus, 10;R causes Delta to display symbol plus relative offset only when a symbol lies within 10 locations of the display address.

The ;RK command sets relative address output mode, using only control section type symbols for output unless there is an exact match between the symbol value and output value (for a discussion of setting the control section type, see "Symbol Table Control: The ;U, ;K, ;S, !, and <> Commands" earlier in this chapter). If there are no control section symbols, the output is hexadecimal. Thus, output is "control section plus hexadecimal offset", "symbol", or "hexadecimal constant".

;RK Display Example:

```
A,A+5/    LI,1      10
.5CD/     CW,1      K45
.5CE/     BGE       ZZZ
.5CF/     AI,1      1
.5D0/     B         A17
ZZZ/      STW,2     BR13
```

The m;AM command sets the address mask for locations to the value m. The default for m is .1FFFF. This command exists for special applications such as the ANLZ and GENMD processors.

PRINTER OUTPUT: THE ;O AND ;J COMMANDS

These two commands provide for output (via symbionts) to the line printer. The ;O command produces hexadecimal dumps on the line printer, while the ;J command directs all Delta output to the line printer. This is particularly useful in the cases of large formatted displays and output from tracing breakpoints.

The printer and tape I/O routines are completely self-contained in the executive version with no dependence on system I/O routines. The executive version of Delta operates with all interrupts except console interrupts disabled. Examples of the ;O and ;J commands are

e1,e2;O[header] contents of memory from location e1 through location e2 are printed on the line printer, single-spaced, eight hexadecimal words with initial hexadecimal location value per line. Duplicate lines are suppressed. If any input follows the O, it is printed as a header. Each dump begins at the top of a fresh page with the contents of the general registers printed first.

;J toggles the output location switch that alternates between the terminal and the line printer each time the command is given. Output from the equal command, from nontracing breaks, from trap, abort, and error returns, and from syntax and other error conditions in Delta are always directed to the terminal. Examples are

```
A, 1;B

X,2,3;DLS     ;J     B;G
```

Note: The output that would have appeared here from data break 2 goes to the line printer.

CHANGING THE PROMPT IN USER DELTA: THE ;Vp COMMAND

The ;Vp command changes the prompt character in Delta from the bell to whatever character was specified by 'p' This character will then be used by Delta as the prompt this session (unless changed by a subsequent ;V command).

USER BREAK AND TRAP CONTROL

Users may set trap control with the M:TRAP CAL. Delta monitors traps to determine if they are Delta breakpoints, and if not, checks the trap mask to see if the user wishes control of the trap. If so, the user TCB is set up with the current user environment and control automatically goes to the M:TRAP routine specified in the CAL. Programs loaded with the (NOTCB) option may pass a trap stack address in R0 when issuing the M:TRAP CAL.

SIMULATED BREAKS n;V OR (CONTROL SHIFT 0)

Since Delta takes over break control, users with an M:INT routine may wish to simulate a break occurrence under Delta. Delta recognizes the unit-separator signal, (US), as a simulated user break. (This keystroke (US) varies on different terminals.) Delta sets up the user TCB and passes control to the M:INT routine specified in the CAL. n;V causes the next n breaks to pass directly to the user routine.

Note: Delta must be in control when this character is entered to simulate a break.

EXECUTIVE DELTA

Executive Delta does not honor the following commands:

;Y

;RK

;V

;AM

The following commands have different meanings for executive Delta:

;E Displays the last 32 words patched with Delta.

;U Toggles Delta's map bit.

;H Displays or sets bits 8-11 of the current PSD. (See Table 30.)

;S Manipulates snap dump specifications. (See Table 30.)

All other Delta commands may be used. Special executive Delta restrictions have been noted throughout this chapter.

In addition, executive Delta has the following command to provide a disk dump capability:

```
{da,ndd,ns;O}
{fda,,ns;O  }
```

where

da specifies the starting disk address, the format of which is device dependent.

fda specifies the file management disk address. The address is comprised of the DCT index in the first halfword and the relative sector number in the second halfword.

ndd specifies the actual device address. Note that this field is empty if the fda format is used.

ns specifies the number of sectors to dump.

Assuming a system RAD or disk pack address of .1F0 and a DCT index of 7, either of the following examples could be used to dump the first ten sectors of the HGPs contained in ALLOCAT:

```
8,.1F0, 10;O
.70008,, 10;O
```

WRITING PROGRAMS WITH DELTA

The user may write and check short Meta-Symbol or machine language programs using Delta. The following two commands are especially helpful for writing programs:

1. Symbolic tags may be defined at a specific address using the command

   e(f<s>[K]

   (See the section "Symbol Table Control".) Each symbolic tag should be defined with this instruction before it is used in the program. The range of addresses available to the user is .C000-.1BFFF.

2. Pages for the program may be requested from the monitor by using the command

   e\

   (See the section "Memory Cell Opening and Display".) This command also opens the specified cell so that the user may store an instruction or data into it.

   Example:

   .10000(R<BEGIN>K ⓁⒻ defines the tag BEGIN at location .10000.

   BEGIN\ LI,2 0 ⓁⒻ opens the cell at BEGIN and requests the page from the monitor (if it is not already assigned to the user). The instruction LI,2 0 is then stored into the cell at BEGIN.

ERRORS AND ERROR MESSAGES

Errors that result in machine traps are reported to the user, and console control is returned to the user to await further commands. Each message is accompanied by the location, symbolically if possible, of the offending instruction. The messages are

```
NONEXIST INSTR AT
NONEXIST MEM REF AT
PRIVIL INSTR AT
MEM PROTECT FAULT AT
STACK LIMIT FAULT AT
UNIMP INSTR AT
FIXED ARITH OVFLW AT
FLOAT FAULT AT
DECIMAL FAULT AT
MEM PROTECT FAULT ACCESSING
```

Syntax errors are reported by the message "?n", where n is the number of the character in the command line that Delta was processing when the error was detected. This message is sent to the user whenever Delta cannot understand the user's command syntax. Because the commands are brief (i.e., requiring few keystrokes) and most errors can be spotted easily by eye, only a few syntax errors are explicitly commented. Example errors and Delta's response to them are listed below:

'ABCDE'= ? 6	Constant value larger than one word.
ABC;K ⓁⒻ ? 5	Symbol not in symbol table.
.FF;M 100,XY;L .6B;W ⓁⒻ ?13	Symbo! value not found for XY. Remainder of command string ignored.
A,5;E ⓁⒻ ? 5	Command unknown.
LW*5 ALPHA= ? 3	Asterisk in the wrong plac .
.3ACR/ ? 5	Illegal character in a hexadecimal number.

(B;/ ? 2	Illegal format control character.
LOC,,3;DNG (RET) ? 10	Illegal relation.
;T (RET) ? 2	No break in an attempt to set trace mode on.
. A216< LI > (RET) ? 9	Symbol specified conflicts with one of Delta's op-code mnemonics.
.A\ LI,17 X'20' (RET) *TR*	Value specified is too large for the intended field and has been truncated.
'CAN'T GET PAGE'	A request was made requiring a page not available to Delta.

PROGRAM EXITS

Delta takes control of program exits and aborts. It reports execution of exit CALs with the message:

EXIT AT loc

for normal M:EXIT calls, along with either the proper message from the system error message file, or the message:

ABORT WITH CODE = xx SUBCODE = yy
LOCATION = loc

if no message file entry exists.

DELTA COMMAND SUMMARY

The Delta commands are summarized in Table 31. They are listed by groups according to the type of function they perform.

Table 31. Delta Command Summary

Command	Function
<u>Expression Evaluation</u>	
e=	Evaluates and types the value of the expression e in the most appropriate format.
e(f=	Evaluates and types the value of e in format f.
=	Following a display, evaluates and types the value of the last expression typed by Delta.
(f;=	Changes the default format for output for the = command to the format specified by f.
<u>Memory Cell Opening and Display</u>	
e/	Displays the contents of cell e in the most appropriate format, and opens the cell in preparation for change.
e(f/	Opens and displays the contents of cell e in format f.
e1,e2/ e1,e2(f/	Displays the contents of cell e1 through e2 in the most appropriate format or in the specified format f, and opens cell e2.
e\	Opens but does not display cell e. Also may be used to request pages from the monitor. (The \ command is replaced by ¢ in the executive version.)
/	Following a display, displays but does not open the last cell addressed by the display. The new display is in the default format.
(TAB)	Following a display, displays and opens the last cell addressed by the display.
(f;/	Changes the default format for output for the slash command to the format specified by f.
<u>Memory Modification</u>	
e	Assembles the expression specified by e, stores it in the currently open cell, and closes the cell.
e	Stores e in the currently open cell, closes it, and opens and displays the next higher addressed cell. (The LF is replaced by EOM in the executive version.)

Table 31. Delta Command Summary (cont.)

Command	Function
Execution Control (cont.)	
e ↑	Stores e in the currently open cell, closes it, and opens and displays the next lower addressed cell. (The ↑ is replaced by & in the executive version.)
e TAB	Displays and opens the cell addressed by the last quantity typed. If an expression precedes the TAB, the expression is stored in the open cell and that cell is closed.
Symbol Table Control	
s;S	Selects internal symbol table s.
;S	Loads global symbol table.
;U	Displays undefined symbols. (In executive Delta, ;U toggles the map bit in Delta's PSD.)
e(f<s> [K]	Assigns to symbol s the value e and the format f.
s(f!	Assigns to symbol s the value of the currently open cell and the format code f.
s;K	Flags symbol s in the symbol table. It will not be used in output expressions, but it can still be used in input expressions.
s;KD	Prevents use of the symbols for any purpose.
;K	Removes all symbols except instruction mnemonics and special symbols.
;KI	Removes the current internal symbol table.
;KG	Removes the global symbol table and any symbols defined from the console.
Execution Control	
e;G	Begins execution at e.
;G	Begins execution at the address specified by the current location counter value.
;P	Begins execution at the address specified by the current location counter value.
n;P	Proceeds with no output the next n times the current instruction breakpoint is encountered.
;X	Enter M:XCON Routine.
e;X	Executes the instruction e.
)	Executes the current instruction and displays the next one.
e;I e;C e;F e;H	Sets the indicated portion of the current PSD to e.

Table 31. Delta Command Summary (cont.)

Command	Function
Snap Dump Control (Executive Delta only)	
n,loc,nw;S	Sets dump n to perform the dump from loc to loc+nw.
n,da,ndd,ns;S	Sets dump n to perform a disk dump. (See ;O.)
n,fda,,ns;S	Alternate format of above command. (See ;O.)
	Note: If n is omitted in any of the preceding commands, the next available (unused) value for n is used.
n;S	Removes dump n.
0;S	Removes all snap dumps.
;S	Displays current snap dump specifications.
Overlaid Program Control	
;H	Toggles the switch to enable or disable the stop in Delta when an overlay is loaded. The switch is set initially to "no stop" and alternates between "stop" and "no stop" each time the command is given.
	BREAKPOINTS
Instruction Breakpoints	
e,n;B	Sets the nth instruction breakpoint at location e.
e,n;BT	Same as above, but the program automatically proceeds from the breakpoint after the breakpoint message is printed (trace mode).
e;B	Sets the next available breakpoint at location e.
e;BT	Same as above, but the program automatically proceeds from the breakpoint after the breakpoint message is printed (trace mode).
e,n, loc;B	Sets the nth instruction breakpoint at location e and causes the contents of loc to be displayed when the break occurs.
e,n, loc, val;Br	Same as above but stops only when the contents of loc are in relation r to val. (See Data Breakpoints for relation r).
e,n, loc;BT	Sets the nth instruction breakpoint at location e and causes the contents of loc to be displayed when the break occurs. The program automatically proceeds from the breakpoint after the breakpoint message is printed (trace mode).
e,n, loc, val;BTr	Same as above but stops only when the contents of loc are in relation r to val. (See Data Breakpoints for relation r).
e,, loc;B	Sets the next available breakpoint at location e and causes the contents of loc to be displayed when the break occurs.
e,, loc, val;Br	Same as above but stops only when contents of loc are in relation r to val. (See Data Breakpoints for relation r).
e,, loc;BT	Sets the next available breakpoint at location e and causes the contents of loc to be displayed when the break occurs. The program automatically proceeds from the breakpoint after the breakpoint message is printed (trace mode).
e,, loc, val;BTr	Same as above but stops only when contents of loc are in relation r to val. (See Data Breakpoints for relation r).

Table 31. Delta Command Summary (cont.)

Command	Function
Execution Control (cont.)	
;T	Sets the trace mode at the current breakpoint (which just caused a breakpoint interrupt) .
e,,,n;B	The addition of a fourth parameter to any of the above causes snap dump n to be produced when the break occurs. (Executive version only.)
n;B	Removes the nth instruction breakpoint.
0;B	Removes all instruction breakpoints.
;B	Displays all active instruction breakpoints.
Data Breakpoints	
e,n,val,m;Dr	Causes data break n to occur whenever the contents of cell e, masked by m, are in relation r to val. The relations are LS or L e < val GQ or GE e ≥ val EQ or E e = val NQ or NE e ≠ val GR or G e > val LQ or LE e ≤ val
e,,val,m;Dr	Same as above, but uses the next available data breakpoint number.
e,n,val,m;DTr e,,val,m;DTr	Same as the two above, but the program automatically proceeds from the breakpoint after the breakpoint message is printed (trace mode).
e,n;D	Causes data breakpoint n to occur whenever the contents of cell e are changed.
e;D	Same as above, but uses the next available data breakpoint number.
e,n;DT e;DT	Same as the two above, but the program automatically proceeds from the breakpoint after the breakpoint message is printed (trace mode).
e,,val;Dr	Sets the next available data breakpoint. A break will occur whenever the contents of e are in relation r to val.
e,,val,m;Dr	Same as above except that the contents of e are masked by the mask m.
e,,val;DTr e,,val,m;DTr	Same as the two above, but the program automatically proceeds from the breakpoint after the breakpoint message is printed (trace mode).
;T	Sets the trace mode at the current breakpoint (which just caused a breakpoint interrupt).
n;D	Removes the nth data breakpoint.
0;D	Removes all data breakpoints.
;D	Displays all active data breakpoints.
Transfer Breakpoints and Interpretive Execution	
;Y	Starts execution at the current location counter in the transfer breakpoint mode. Does not display branches specified in the SAT. Does not display BDR and BIR branches.

Table 31. Delta Command Summary (cont.)

Command	Function
Transfer Breakpoints and Interpretive Execution (cont.)	
loc;Y	Same as above except that execution begins at loc.
;YT loc;YT	Same as the two above, except that the trace mode is also set.
,option1;Y	Starts execution at the current location counter in the transfer breakpoint mode. Does not display branches specified in the SAT if option1 = 0. Displays only those branches specified in the SAT if option1 = 1. Does not display BDR and BIR branches.
loc,option1;Y	Same as above except that execution begins at loc.
,option1;YT loc,option1;YT	Same as the two above, except that the trace mode is also set.
,,option2;Y	Starts execution at the current location counter in the transfer breakpoint mode. Does not display branches specified in the SAT. Displays BDR and BIR branches if option2 = 1. Does not display BDR and BIR branches if option2 = 0.
loc,,option2;Y	Same as above except that execution begins at loc.
,,option2;YT loc,,option2;YT	Same as the two above, except that the trace mode is also set.
,option1,option2;Y	Starts execution at the current location counter in the transfer breakpoint mode. Does not display branches specified in the SAT if option1 = 0. Displays only those branches specified in the SAT if option1 = 1. Displays BDR and BIR branches if option2 = 1. Does not display BDR and BIR branches if option2 = 0.
loc,option1,option2;Y	Same as above except that execution begins at loc.
,option1,option2;YT loc,option1,option2;YT	Same as the two above, except that the trace mode is also set.
;T	Sets the trace mode for all transfer breakpoints.
0;Y	Turns off the transfer breakpoint mode.
loc1[,loc2[,loc3[,loc4]]];YS	Sets one to four entries in the SAT (Special Action Table).
loc1[,loc2[,loc3[,loc4]]];YR	Releases one to four entries in the SAT.
;YR	Releases all entries in the SAT.
;YD	Displays the SAT.
Memory Search and Modification	
Memory between the bounds specified in ;1 and ;2 (initially set to the lower and upper limits of memory assigned for user data) is searched under the mask in ;M (initially all ones). If field e2 is specified in the search command, the value in that field is stored through mask ;M into each location that meets the specified condition.	
e;W	Searches for and displays words that match e under the mask ;M.
e1,e2;W	Stores e2 through mask ;M in locations that match e1 through the mask.
e;N	Searches for and displays words that do not match e.
e1,e2;N	Stores e2 through mask ;M in locations that do not match e1 through the mask.

Table 31. Delta Command Summary (cont.)

Command	Function
Memory Search and Modification (cont.)	
e;1	Sets the memory search lower bound to e.
e;2	Sets the memory search upper bound to e.
e1,e2;L	Sets ;1 to e1 and ;2 to e2.
e;M	Sets the search mask to e.
Memory Clearing	
e1,e2;Z	Zeros memory from e1 through e2.
e1,e2,v;Z	Stores the value v in memory from e1 through e2.
Display Modes	
;R	Sets the display mode in memory addresses to symbol plus relative hexadecimal offset.
n;R	Same as above, but sets the maximum hexadecimal offset to n.
;RK	Displays addresses as control section type symbol plus any hexadecimal offset. If the value displayed is equal to that of any symbol, then the symbol is displayed. If there is no control section type symbol, then a hexadecimal constant is displayed.
;A	Sets the display mode for locations to hexadecimal numbers.
m; AM	Sets the address mask for locations to the value m. The default for m is .1FFFF.
Printer Output	
e1,e2;O [header]	Prints the contents of memory from location e1 through location e2 on the line printer in the standard core memory dump format. If any input follows the O, it is printed as a header.
n;O [header]	Dumps snap dump number n. (Executive Delta only.)
;J	Toggles the output location switch which alternates between the terminal and the line printer each time the command is given.
Disk Dumps	
{da,ndd,ns;O fda,,ns;O}	Prints the contents of the disk from the area specified. (This command is only available in executive Delta.)
Prompt Character Changing	
;Vp	Changes the prompt character for Delta to character specified by p.
Program Termination	
;E	Disassociates Delta and exits to TEL.
User Break Control	
;V or unit-separator (US)	Enters the user break control routine (M:INT) as though BRK had been depressed.
n;V	Same as above; also enters the user break routine the next n-1 times that BRK is depressed.

8. ON-LINE LOAD PROCESSORS

INTRODUCTION

There are three CP-V processors that can be used to control loading and execution of object programs: the Load processor, the LYNX processor, and the Link processor.

Load is a two-pass overlay loader. The first pass processes not only ROMs but previously formed load modules or a combination of both. (For example, Load processes dummy sections from library load modules as well as from ROMs.) The first pass also processes expressions for definitions and references (primary, secondary, and forward references). The second pass forms the actual core image and its relocation dictionary.

LYNX is a load processor that is available in both the on-line and batch modes. LYNX has the capabilities of the overlay loader, Load, and also provides some control over internal and global symbol table construction. LYNX is speed-competitive with the Link loader, and in many cases will run faster than Link. In addition, on-line load maps are formatted taking into account the platen width of the terminal.

Link is a loader that is now supplied only for compatibility with previous versions of CP-V. Although Link is described in full detail in this manual, it is recommended that LYNX be used.

LYNX and Link operate in both the on-line and batch modes and are described in this chapter. Load operates only in the batch mode and is described in the CP-V/BP Reference Manual, 90 17 64.

The LEMUR processor is also described in this chapter although it is not a loader. LEMUR (Library Editor and Maintenance Utility Routine) is a processor that builds and manipulates ROM and load module libraries. The libraries thus built are accessed by LYNX or Load when constructing user load modules.

LYNX

LYNX is a load processor that is available in both the on-line and batch modes. LYNX has the capabilities of the overlay loader, Load, and also provides some control over internal and global symbol table construction. LYNX is speed-competitive with the Link loader, and in many cases will run faster than Link. In addition, on-line load maps are formatted taking into account the platen width of the terminal.

LYNX may be viewed as a preprocessor for the Load loader. After it analyzes the user's commands, it constructs a table of loader control information which it then passes to the overlay loader. It is the Load loader which actually performs the loading process. (The Load processor is described in the CP-V/BP Reference Manual, 90 17 64.)

The LYNX processor recognizes three commands in the on-line mode: LYNX, RUN, and :TREE. The LYNX and RUN commands call the LYNX processor from the TEL level.

COMMAND CONTINUATION

The presence of a semicolon as the last character on an input line indicates that the command is to be continued. LYNX will perform another read of the SI device, prompting the user with a > character if SI is assigned to an on-line terminal.

COMMAND FILE INPUT

In order to have LYNX read its commands from a file, the following command should be given:

```
!LYNX  fid
```

where fid identifies the file.

LYNX will examine the indicated file to determine whether or not it is a ROM. If the file is not a ROM, it will be treated as input commands for LYNX. If the file is a ROM, it will be loaded, creating (as in the case of Link) a temporary load module file which can then be run using the

```
!START  $
```

command in the on-line mode.

LYNX COMMANDS

LYNX The LYNX command has a syntax which is generally compatible with that of the LINK command. This permits a LINK command to be run under the LYNX processor by simply changing the command name from LINK to LYNX. However, there are some restrictions. These are listed below:

1. ROM names may not be enclosed in parentheses to merge their internal symbol tables. If the construction of internal symbol tables is specified (via the I option), one table will be built for each ROM.

2. The D and ND options concerning the displaying of undefined symbols will not be meaningful. Undefined symbols will always be displayed.

3. The C and NC options concerning the displaying of conflicting internal symbols will not be meaningful since internal symbol tables cannot be merged. Conflicting (doubly defined) external symbols will always be displayed.

4. The options Ji, Pi, FDP, and NP for associating or not associating public libraries will not be necessary. The libraries will automatically be associated in the case of P0 and P1. For J0, J1, and J2, the load modules :J0, :J1, and :J2 from :SYS can be specified as element files. However, the presence of J0, J1, or J2 as an option will produce the desired results (i.e., loading of the appropriate library module with the other element files). Note that if a load module using any of these libraries is overlaid, the appropriate module name(s) must appear on the :TREE command.

The general format of the LYNX command is:

```
! LYNX ef[,ef]... ⎡ON       ⎤
                  ⎣OVER  lmn⎦ [options] ——┐
└——[;[libname] [.[libacct] [.password]]]...
```

where

ef may be the file identification (fid) of a ROM, a library load module, a DEFCOM-build load module, or a SYSGEN-built load module, or simply a dollar sign ($).

lmn (load module name) specifies where the load module is to be placed and may be a file identification (fid) or dollar sign. If lmn is omitted, the resulting load module is placed in a special file and is available for subsequent execution via the TEL START command or the batch mode version of the RUN command. TEL commands (including their abbreviated forms) cannot be used as load module names.

libname specifies the name of a library. :LIB.:SYS is the default library if no library name, account or password is specified and if the NL option is not specified. If libacct is specified, but libname is not specified, then the default libname is :LIB.

libacct specifies the account from which the library is to be obtained. If libname is specified but no libacct is specified, the default account is the log-on account.

password specifies the password for the library if one exists.

options specifies loading and linking options. These options are described below. Most options may be specified anywhere in the command except between a preposition and its object. For convenience they are shown immediately following lmn. All options must be specified within parentheses.

As with the LINK command, the options may actually appear anywhere in the command string and <u>each must be preceded by a left parenthesis or enclosed within parentheses</u>. The options are described below.

<u>Options that determine input to the loader</u>

BI specifies that the M:BI DCB is to be used to read unspecified relocatable object modules. Object modules will be loaded from the BI device until either two end-of-data codes (05) or one end-of-file code (06) is encountered.

L specifies that the system library is to be searched. (L is assumed by default if NL is not specified.)

NL specifies that the system library is not to be searched.

J0 specifies that :J0 (which contains all JIT definitions) is to be included as an element file.

J1 specifies that the monitor's REF/DEF stack is to be included as an element file.

<u>Options affecting future access to the load module file</u>

T specifies that the named load module is to be created as a temporary file.

LIB [,libname] specifies that a library load module is to be built (provided that the T option is not specified.

If LIB is specified, any external definitions or external references in the load module will be added to the library's table of external definitions and the load module will be inserted into the library (libname). If LIB is specified, the load module must consist of a single control section of uniform memory access type.

NDIC prevents modification of the library's dictionary tables. This option may only be used in conjunction with the LIB option.

RD[,value]... specifies the account numbers of those accounts that may read but not write the file. The value ALL may be used to specify that any account may read but not write the file (e.g., RD,ALL). The value NONE may be used to specify that no other account may read the file. If no value is specified, or if RD is omitted, ALL or NONE as specified in the user's authorization record is assumed by default. The total number of accounts explicitly specified in a RD specification may not exceed eight.

WR[,value]... specifies the account numbers of those accounts that may have both read and write access to the file. The values ALL and NONE may be used as described for the RD option above, except that NONE is assumed by default. If a conflict exists between RD and WR specifications, those of the WR option take precedence. The total number of accounts explicitly specified in a WR specification may not exceed eight.

EX,value[,value]... specifies the account numbers of those accounts that may execute the file. Up to eight account numbers may be specified. The value ALL may be used to specify that any account may execute the file. The value NONE may be used to specify that no other account may execute the file. In all of the above cases, RD, NONE is implied in the absence of any RD specification.

EXP, $\left\{\begin{array}{l}\text{mm,dd,yy}\\ \text{ddd}\\ \text{NEVER}\end{array}\right\}$ specifies either an explicit expiration date (mm,dd,yy), a life in days (ddd) or that the file is never to expire (NEVER). The default value is that in the user's authorization record. The value specified may not exceed the maximum expiration period authorized for the user. If the maximum expiration period is exceeded or if EXP is not specified, the default expiration period authorized for the user will be used.

Options affecting the location of the program at execution time

LB,value specifies the load bias (as a hexadecimal word location). If the value is not a page boundary, the next lower page boundary is used. If no bias is specified, the program will be loaded at location X'A000'.

CL specifies that when the load module is brought into core for execution, virtual core is to be allocated with the special shared processor area held in reserve. This permits the association of a core library at run time and linkage (via M:LINK/M:LDTRC) to another load module that is associated with a core library.

C1 specifies that the load module is to be formed with a protection type of 01, except for the TCB and blank COMMON (which have a code of 00) and except for any type 10 control sections input in load module form.

M10 specifies that each control or dummy section is to be loaded at the next greater multiple of 10_{16}.

M100 specifies that each control or dummy section is to be loaded at the next greater multiple of 100_{16}.

Options concerning the loader-built Task Control Block

TSS,size specifies (in hexadecimal) the maximum size, in words, of the load module's Temporary Storage Stack. If TSS is omitted, the maximum size is set at X'40' words. The greatest size that may be specified is limited to available core storage and may not exceed 7FFF words regardless of core size.

ERT,size specifies the size, in hexadecimal number of words, of the library error table. The default is ten words.

ERS,size specifies the size, in hexadecimal number of words, of the library error stack. The default is ten words.

NTCB specifies that no Task Control Block is to be created by the loader.

Options concerning symbol tables

I specifies that an internal symbol table is to be built for each ROM which was assembled or compiled to contain internal symbol tables.

NI specifies that internal symbol tables are not to be built. NI is the default if neither I nor NI is specified.

G specifies that a global symbol table is to be built for this load module. A global symbol table contains all symbols which were declared external (via a DEF) in one module to be referenced in another (via a REF). This is the default if neither G nor NG is specified.

NG specifies that a global symbol table is not to be built for this load module.

Options determining how overlay segments will be brought into core at execution time

$\left\{\begin{array}{l}\text{OS}\\ \text{SEG}\end{array}\right\}$ specifies that the overlay structure is to be set up for the segment loading mode. In this mode, it is the user's responsibility to explicitly load each segment from disk storage to core storage (e.g., by means of the M:SEGLD procedure) before it is referenced by the executing program. This mode is faster in operation than the reference mode (see below) but less convenient.

$\left\{\begin{array}{l}\text{OR}\\ \text{SEG}\end{array}\right\}$ [,num] specifies that the overlay structure is to be set up for the reference loading mode. In this mode, the execution of any instruction referencing an external definition in another segment on a lower overlay level will cause that segment and all its backward path (see the :TREE command) to be loaded if not already in core (even if the reference is an unsatisfied conditional branch). The external reference must not be in an instruction that may be changed or replaced during program execution.

The decimal value "num", if present, specifies the maximum number of interbranch references within the program. If "num" is absent or zero, the loader will reserve a total of 22 words per segment (four words are required for each interbranch reference) in its reference loading table.

{OB, BREF} [,num] specifies that the overlay structure is to be set up for the branch referencing loading mode. In this mode, any permissible branching reference (in another segment of the program) to an external definition within a given segment will cause that segment and all its backward path to be loaded, if it is not already in core storage. If a nonbranch reference is made to an external definition within a given segment, the OB mode will assume that segment to be in core. OB should be used for all overlaid FORTRAN or COBOL programs. A branch reference causes register 0 to be changed.

The optional value "num" has the same meaning as for the reference loading mode (see OR above). If "num" is absent or zero, a total of 11 words per segment are reserved in the reference loading table (two words per reference).

One of these options must be specified if the load module being formed is to be overlaid. The presence of one of these options in the command string will cause LYNX to read the SI device one more time following the end of the LYNX command string, looking for a :TREE command.

dditional options

A specifies that no relocation dictionary is to be formed for the load module (i.e., the load module is absolute).

R specifies that a relocation dictionary is to be formed for the load module, and the load module will be treated as semiabsolute (i.e., executable but capable of being relocated). If neither A nor R is specified, A is assumed.

M[N] specifies that a load map is to be output on the LL device and that the DEFs within each segment are to be sorted by name.

MV specifies that a load map is to be output on the LL device and that the DEFs within each segment are to be sorted by value.

{MVN, MNV} specifies that a load map is to be output on the LL device and that the DEFs within each segment are to be sorted by name and value.

NM specifies that no load map is to be output. NM is assumed if neither MN, MV, nor MNV is specified.

MO [N] specifies that only a map of an existing load module is to be produced. The map is to be sorted by name.

MOV specifies that only a map of an existing load module is to be produced. The map is to be sorted by value.

{MONV, MOVN} specifies that only a map of an existing load module is to be produced. The map is to be sorted by name and value.

LDEF is used in conjunction with the M or MO option and requests that a listing be produced that includes all the used library DEFs for the load module.

UDEF is used in conjunction with the M or MO option and the LDEF option and requests that a listing be produced that includes all the library DEFs defined in the load module.

RDEF specifies that all unused DEFs are to be removed from the load module's REF/DEF stack. A shortened REF/DEF stack is created for the load module.

SS specifies that a size summary for each segment detailing the memory allocation for each protection type is to be output. SS is assumed if any type of load map is requested.

SL,value specifies the error severity level that will be tolerated by the loader in forming a load module. The value may range from 0 to F. The default is 4.

PA specifies that those portions of the load module that will be loaded into core at execution time are to be developed in page-size records. The load module formed is called a paged load module. The load module is formed in extended memory mode. More time is required to form the load module, but since uninitialized pages do not get written as part of the load module, programs that have large areas of uninitialized data will occupy fewer granules.

NBS specifies that the loader is not to use a sort table to speed up stack searches. The core required for this table is then available for creating very large core images (40K) without using extended memory mode.

OSP specifies that any control sections of protection type 00 in an overlay segment should be forced to the root of the load module. This option is intended primarily for loading overlaid shared processors written in FORTRAN and is only valid for programs having one level of overlay structure.

DREF when used in conjunction with the LIB option, causes all dummy section definitions to be changed to PREFs. This allows a library to be built in which all references to a particular named DSECT will be linked to a single copy of that DSECT (e.g., a FORTRAN BLOCK DATA subprogram). Such initialized dummy sections should be contained in the only library load module loaded <u>without</u> the DREF option.

PRIV[, P][, J][, M][, X] sets the privileged processor flags for the load module. One to four flag letters may be specified in any order. The flag letters have the following meanings:

P - processor accounting. (Execution time is to be tallied as processor rather than user execution time in the accounting record.)

J - special JIT access.

M - maximum memory protection.

X - execute M:SYS CALs.

These flags have no meaning unless the load module resides in the :SYS account.

LDR,name [.[account] [.password]] directs LYNX to be a preprocessor for a loader other than LOADER.:SYS. The default account is :SYS. The function performed by this option can also be performed by assigning the F:LOADER DCB to the desired loader.

NASN instructs the loader to ignore any F:number DCB assignments specified via !SET or !ASSIGN commands when constructing DCBs for the load module being built; i.e., any such DCBs will not be included in the load module if NASN is specified.

RUN The on-line RUN command is a combination of the LYNX and (TEL) START commands. It calls the LYNX loader and instructs it to link, load, and start execution of the designated module. The RUN command has the same format as the LYNX command (except that the command verb is RUN rather than LYNX), may include any option available in the LYNX command, and may be continued in the same manner as the LYNX command.

MAPPING EXISTING LOAD MODULES

In order to produce a map of an existing load module, the format of the LYNX command must be:

$$\text{!LYNX} \quad \text{fid} \left\{\begin{matrix}\text{(MO)}\\ \text{(MOV)}\\ \text{(MONV)}\end{matrix}\right\}$$

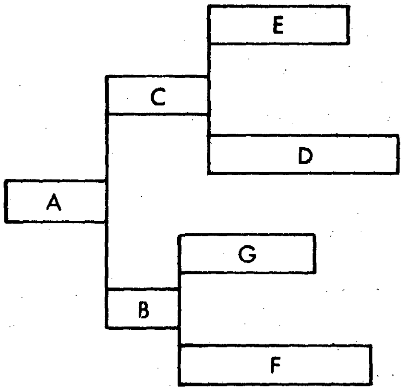
where fid specifies the file identification of the load module. The LDEF and UDEF options are also valid in this context. All other options will be ignored.

:TREE If a program is to be overlaid, a :TREE command must be the next command following the LYNX command. It must specify the overlay structure of the load module to be formed, so that the logical segments of the program will be loaded from secondary storage into core storage as required. It is the user's responsibility to plan the relationship of these segments. If BI relocatable object modules (ROMs) are to be loaded from the C-device, they must be placed after the LYNX command and must precede the :TREE command.

The relationship of the segments that comprise an overlay program can be represented graphically by means of a tree diagram, as in the example shown below. The horizontal coordinate of the diagram denotes increasing core storage (address) allocation, from left to right. The vertical coordinate denotes overlays. The leftmost segment, or "root", is that portion of the program that resides in core storage through program execution. A "path" of an overlay consists of those segments that may occupy core storage at the same time. The portion of a path that extends from the start of the program (i.e., the root) to a given segment is termed the "backward path" of that segment.

The following example consists of four paths, any one of which may be present in core storage at any given time. Segment A, below, is the root of the program and is never overlaid by another segment. Any path may be loaded into core storage and overlaid as many times as required by the program. All segments of the load module are saved in disk storage and, when a segment that has been overlaid is called again by the executing program, the original copy is loaded from the disk. Therefore, any communication between two overlay segments (e.g., D and E, below) must be done in a part of the backward path common to both.

Example:



The form of the :TREE command is

```
:TREE specification
```

where specification specifies the tree structure by use of the symbology given below.

name specifies the name of an element file (EF). The name (1-10 characters) must not contain any special delimiters (e.g., -) embedded in it.

- indicates that two named relocatable object modules are to be contiguous in core storage.

, indicates that two segments are to overlay one another (i.e., begin at the same core storage location).

( ) indicates a new (lower) level of overlay.

No two segments may begin with the same EF name, since the name of the first EF becomes the name of the segment.

Example:

```
:TREE A - (C - (E, D), B - (G, F))
```

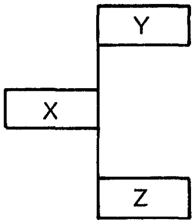
The above example is a symbolic representation of the overlay structure of the preceding graphic example.

LYNX EXAMPLE

The following is an on-line example of LYNX usage.

```
!LYNX X, Y, Z OVER LMS(M) (I) (G);
≥(OS);.ACCNT1
≥:TREE X-(Y, Z)
```

This example specifies that an overlaid load module 'LMS' is to be produced from element files X, Y, and Z in the running account. A map sorted by name is desired, internal and global symbol tables are to be built, and overlaying will be done explicitly within the program via M:SEGLD CALs. The default library in ACCNT1 will be searched to satisfy any primary external references (PREFs). The load module will have the tree structure:



The load module can be executed by one of the following two commands in the on-line mode:

```
!START LM5
!LM5.
```

It can be executed by the following command in the batch mode:

```
!RUN (LMN, LM5)
```

ERROR MESSAGES

Error messages are output on the terminal in the on-line mode and on the LL device in the batch mode. They are preceded by a portion of the command line, ending at the point of error detection. The LYNX error messages are listed in Table 32.

Table 32. LYNX Error Messages

Message	Description
*** BAD :TREE COMMAND	LYNX is completely unable to make sense of the :TREE command, or the :TREE command is missing but an overlay option was specified in the LYNX or RUN command.
*** CONFLICTING OPTIONS	The user specified two conflicting options (e.g., I,NI) or the same option twice.

Table 32. LYNX Error Messages (cont.)

Message	Description
*** ELEMENT FILES IN E. F. LIST NOT IN TREE	The user specified element files in the LYNX command which did not appear anywhere in the :TREE command.
*** ELEMENT FILE IN TREE NOT IN E. F. LIST	An element file appeared in the :TREE command which was not specified in the element file list.
*** FILE NAME IS TOO LONG	An element file name must be no more than 10 characters in length.
*** ILLEGAL DECIMAL NUMBER	An illegal decimal digit was detected in one of the LYNX options.
*** ILLEGAL HEXADECIMAL NUMBER	An illegal hexadecimal digit was detected in one of the LYNX options.
*** INSUFFICIENT MEMORY AVAILABLE	The user's core allocation is so low that LYNX is unable to obtain the memory it requires for constructing tables.
*** NOT BACK TO LEVEL 0 OF TREE	At the conclusion of scanning the :TREE command, it was apparent that the overlay structure has not been completely defined. The user probably omitted a closing parenthesis somewhere.
*** NUMBER TOO LARGE	The numerical value specified on an option is beyond the legal range.
*** 'ON' ILLEGAL – LOAD MODULE EXISTS	The user attempted to use the ON preposition to build a load module which already exists.
*** SYNTAX ERROR	The user made a syntactical error in the LYNX command about which LYNX is unable to be more specific.
*** TOO MANY ACCESS ACCOUNTS	More than eight read accounts, write accounts, execute accounts or libraries have been specified.
*** UNABLE TO COPY BI INPUT	An error other than end-of-data or end-of-file has occurred while reading M:BI for the BI option.
*** UNBALANCED PARENTHESIS – BAD TREE STRUCTURE	The user probably supplied an unexpected or superfluous closing parenthesis.
*** UNEXPECTED END OF COMMAND	A closing parenthesis is absent, or an expected final field in the LYNX command is missing.
*** UNRECOGNIZED OPTION	The user specified an option which LYNX is unable to identify.

LINK

Link constructs a single entity called a load module (LM), which is an executable program formed from relocatable object modules (ROMs). Link also provides the necessary data space and program linkages for the association of public libraries.

The final program resulting from a linking operation has three protection types, one for data, one for pure procedure, and one for DCBs. Static data and nonaccess information, if specified, are loaded with the pure procedure.

The access protection types provided by Sigma 6, 7, or 9 hardware are

00	read, write, and execute access permitted (data).
01	read and execute access permitted (pure procedure).
02	read access permitted (static data).
03	no read or write permitted (no access).

Load modules produced by Link may be restricted to execution only by selected accounts. Only those users running in the specified accounts may execute the output load module.

LOAD MODULE STRUCTURE

A load module formed by Link is composed of three parts: program, global symbol table, and internal symbol table. Each of these parts is described in the following sections.

PROGRAM

A program may be sectioned into six parts: pure procedure, data, common, DCBs, public libraries, system library.

1. Pure Procedure

   This section of code contains machine instructions and is generated by compilers and assemblers with protection type 01 (read and execute access). Sections with a nondata protection type (static data and no access) are also included here.

2. Data or Program Context

   This section is generated by the compilers and assemblers with protection type 00 (read, write, and execute access).

3. Common

   This blank common storage is generated by compilers and assemblers as a dummy section with the name F4:COM. The size of blank common storage is determined by the first size declared. All subsequent F4:COM declarations must be less than or equal to that size.

4. DCBs

   A data control block (DCB) is a table containing the information used by the monitor in performance of an I/O operation. At the end of a link operation, Link constructs a DCB corresponding to each outstanding external reference with names beginning with F: and M:.

The M:UC DCB, which is the DCB most commonly used for terminal I/O, is supplied as a portion of the user's JIT (job information table); any M:UC reference is automatically satisfied thereby. The default assignment of M:UC to the user's terminal is unalterable. (Output operations via M:UC are treated specially by the monitor; see Chapter 11.) If the program being linked does not contain a reference to M:DO, a reference to it is supplied by Link, since diagnostic output is generally written via this DCB. If the user does not want this DCB to be constructed, due to space considerations, he can explicitly reference M:DO and satisfy the reference (vacuously) within his program. (Some diagnostic output is likely to be lost.)

A DCB name of the form M:ab, where ab corresponds to an operational label, is considered a reference to a standard system DCB. The standard system DCBs are discussed in terms of operational labels and default assignments in CP-V/BP Reference Manual, 90 17 64.

DCBs constructed by Link are 51 words long and consist of

a. A 22-word standard initial segment, containing a standard default operational label if the DCB is one of the system DCBs.

b. Five variable length items including a control word for each, with space for

   - A three-word file name.
   - A two-word account number.
   - A two-word password.
   - A three-word block for three input serial numbers.
   - A three-word block for three output serial numbers.
   - A two-word block for expiration date.

c. An eight-word key buffer.

The standard system DCBs also exist in ROM form on files in the system account; in this form they differ from Link-constructed DCBs in size and composition, as described in CP-V/BP Reference Manual, 90 17 64. These ROMs can be explicitly named in a LINK or RUN command to satisfy corresponding references.

While allocating, constructing, and combining DCBs, Link guarantees that each DCB is contained within a page. This allows the operating system to access DCBs in either mapped or unmapped mode. User-supplied DCBs (i.e., DSECTs with names beginning M: or F:) are placed in the DCB record, in user-context space, together with those constructed by Link. All are given protection type 02.

5. Public Libraries

Any CP-V installation can define a set of subroutines that constitute a public library. The installation may specify several different public libraries containing collections of routines that are useful in various environments. Only one library of type 'P' and one of type 'J' may be associated with an executing program. DEF stacks for public libraries are stored under special names in the system account and are used to link programs to them. See the CP-V/SP Reference Manual, 90 31 13, for more detailed information on the structure and creation of public libraries.

Only one block of core memory is required for the public library no matter how many users are using it. However, use of just one routine in the public library requires core for the entire package. The reentrant portion of each library is shared among users (on-line and batch), thus saving physical core memory and allowing for more efficient system operation. User-dependent data storage for each library routine is allocated by Link at a fixed virtual address. Thus, each public library is constructed in two parts: reentrant procedure and direct access data. By forming the library in this manner, a speed advantage of from 5 to 20 percent over push-down storage reentrancy is obtained.

CP-V provides four public libraries: P0, P1, P4, and J0 (only the first three are of general interest). Library P1 contains the most commonly required routines from the Extended FORTRAN IV run-time and mathematical library (about 60 routines). Library P0 includes library P1 plus the FORTRAN Debug Package (FDP). Library P4 includes library P1 plus the FORTRAN real-time features. These three libraries will satisfy the requirements of the majority of users for program execution, debugging, and real-time services, repectively. (The remainder of the run-time and mathematical routines comprising the entire Extended FORTRAN IV subprogram library reside on the system library, described below.) Public library J0 contains the user-JIT Definition Package. (See the CP-V/SP Reference Manual, 90 31 13, also for more detailed descriptions of libraries P0, P1, P4, and J0.) Additional public libraries created by a user-installation may be names P2, P3, or P5 through P9.

Use of the real-time public library, P4, requires specification on the LINK (or RUN) command of the file :BLIB in the real-time system account (e.g., :SYSRT) as a library file identification. This library file will be searched before the public library is searched.

6. System Library

The system library consists of approximately 190 FORTRAN IV library routines in ROM form, in file :BLIB in the :SYS account. Searching of this library is implied by the default library-search code L in a LINK or RUN command. This library is always searched last if any unsatisfied references remain unless the NL option is specified. Routines that are obtained from the system library become part of the user program and are not shared. Thus, core is required for each system library routine. The speed advantage is still maintained since each routine includes any necessary data.

GLOBAL SYMBOLS

While performing the linking process, Link constructs a global symbol table. This table is a list of correspondences between symbolic identifiers (labels) used in the original source program and the values or virtual core addresses that have been assigned to them by Link. The global symbols define (DEF) objects within a module that may be referenced (REF) in other modules. This table is available to Delta for use in debugging.

INTERNAL SYMBOLS

An internal symbol table is a list of correspondences similar to the global symbol table but applies only to symbols defined within the module. Each internal symbol table constructed by Link is associated with a specific input file and is identified by its name. This table is also available to Delta for debugging.

When an internal symbol is equated to an external symbol with an addend, and the module containing the external definition is in a different file from the module containing the external reference, the file containing the definition must appear on the LINK or RUN command before the file containing the external reference. Furthermore, an internal symbol should not be equated to an external reference with an addend satisfied from a library.

No internal symbol table is generated for a named library (one with a fid).

SYMBOL TABLES

Delta makes it possible to reference both global and internal symbols at the time programs are debugged. Programs formed by loaders, together with the tables of global and internal symbols, are operated on in a code similar to assembly language symbolic code.

Global and internal symbol tables, as formed by Link and used by Delta, consist of three word entries. Symbolic identifiers (labels) are limited to seven characters. Symbols originally longer than seven are truncated, leaving the initial seven characters, although the original count is retained.

Thus, symbols that are identical in their first seven characters and are of equal length occupy one position in the symbol table. The value retained for multi-defined symbols is the first one encountered during the linking process. Each symbol entered into the table has an internal resolution and a type classification. Internal resolutions are: byte, halfword, word, doubleword, and constant. Symbol types are: instruction, integer, EBCDIC text, short floating-point, long floating-point, decimal, packed decimal, and hexadecimal. Object language code produced by CP-V assemblers and compilers provides internal symbols with internal resolution and type classification. CP-V loaders retain this information in processing object language code.

CONVENTIONS

The terminal and language conventions for Link are the same as for TEL except the function of the BREAK key. If the BREAK key is depressed while a Link command is being entered, the command is ignored and a new command must be typed (as if X^c had been pressed).

LINK COMMAND

The Link processor is called by a LINK command given at TEL level. The command is described in Table 33.

Table 33. Link Command Summary

Command	Description
LINK [options]rom[,rom]... $\left[\begin{matrix}\text{ON}\\\text{OVER}\end{matrix}\ \text{lmn}\right]$ [;lid [,lid]...] [UNDER FDP]	Forms the load module as specified. Options: library search: (L) search system library (NL) do not search system library default: (L) (Ji) or (Pi) associate ith public library where i = 0-9 (FDP) associate public library P0 (NP) do not associate any public library default: P1 display: (D) display undefined internal and external symbols (ND) do not display undefined internal and external symbols (C) display conflicting internal and external symbols (NC) do not display conflicting internal and external symbols (M) display load map (NM) do not display load map default: (D), (C), (NM) symbol table: (I) include symbol table with LM (NI) do not include symbol table with LM default: (I)

Table 33. Link Command Summary (cont.)

Command	Description
LINK [options]rom[,rom]... $\left[\begin{matrix}\text{ON}\\\text{OVER}\end{matrix}\ \text{lmn}\right]$ [;lid [,lid]...] [UNDER FDP] (cont.)	execute accounts: (EX,acct[,acct]...) specifies those accounts which may execute this lmn. Up to 8 accounts may be specified. The value ALL may be used to specify that any account may execute the lmn. (This is the default if no EX option is specified). The value NONE may be used to specify that no other account may execute the lmn. rom may be fid or $; the name portion of the fid may consist of from 1 to 10 alphanumeric characters, except for shared processor names which may only have up to 8 alphanumeric characters; parentheses enclosing roms cause merge of symbol tables. lid must name a file containing one or more roms. The name may consist of from 1 to 10 alphanumeric characters, except for shared processor names which may only have up to 8 alphanumeric characters.

Examples:

1. Assume there are two relocatable object modules. The internal symbols for the first module (MFL1) are to be left out of the resulting load module, but the internal symbols for the second module (MFL2) are to be included. The resulting load module is called LM1.

```
!LINK (NI) MFL1, (I) MFL2 ON LM1 RET
!
```

If Link needs additional information, it will identify the problem, and then prompt (:) for input.

2. Assume the same example as above except that Link cannot find MFL2 because it was supposed to be MFL3.

```
! LINK (NI) MFL1, (I) MFL2 ON LM1 RET
CANT FIND: RETYPE MFL2
: MFL3 RET
!
```

Note that the ROM specification indicated as unfound (e.g., MFL2) can, alternatively, be bypassed by responding with carriage-return only.

Link commands may be continued by ending the command line with "<". This symbol cannot be embedded in words or between a preposition and its object.

3. Assume that modules A and B are to be linked, with merging of internal symbol tables, to form output module C. In the linking process, one double definition (Z) and one internal unsatisfied definition (Y) are found.

```
! LINK (A,B) < RET
: ON C RET
          LINKING A
          LINKING B
IDDEF          Z (internal double definition)
IUSAT          Y (internal unsatisfied reference)
```

ERROR MESSAGES

Whenever an error occurs during a linking operation, Link sends an error message to the terminal. Some of these messages are for syntax errors, others are for errors arising out of the linking operation. They are listed in Table 34. Most of these errors terminate the linking operation prematurely.

Table 34. Link Error Messages

Message	Description
CANT FIND :RETYPE rom	The specified relocatable object module cannot be found.
CARD CKS/COMPUTED CKS/cd/cp/	This message is sent to the terminal along with the CHECKSUM ERROR message. It specifies the card checksum (cd) and the computed checksum (cp).
CHECKSUM ERROR	A checksum error has occurred. The CARD CKS/COMPUTED CKS/cd/cp/ message specifies the difference.
CORE LIBRARY OVERLAPS PURE PROCEDURE	There is insufficient virtual memory to contain the pure procedure and the core library REF/DEF stack.
DATA LIMIT EXCEEDED	The data area is so large that it overlaps the pure procedure.
DONT TRY TO USE TWO J OR TWO P LIBRARIES AT ONCE	Only one library of each type is allowed.
DUMMY SECTION LARGER THAN PREVIOUS DEF	The dummy section initially defined was not the largest dummy section.
GLOBAL SYMBOL TABLE OVERLAPS PURE PROCEDURE	There is insufficient virtual memory to contain the pure procedure and the symbol tables.
ILLEGAL DATA FORMAT	Input modules did not contain ROM data.
ILLEGAL LOAD ADDRESS	An attempt was made to load outside the limits of the program.
ILLEGAL LOAD ITEM TYPE	ROM input data is illegal (e.g., it is load module data instead).
INSUFFICIENT PHYSICAL MEMORY TO CONTINUE	A request for a memory page has been refused.
I/O ERROR LINKING SYSTEM LIBRARY	This message usually indicates there is no system library.
I/O ERROR OPENING OUTPUT FILE	An I/O error occurred during the opening of an output file.
I/O ERROR READING ASSIGN MERGE RECORD	This message usually indicates there is no assign/merge record.
I/O ERROR READING CORE LIBRARY	This message usually indicates there is no core library.
MODULE# /SEQUENCE#/md/sq/	This message accompanies most other messages. It identifies the module number (md) and sequence number (sq) of the last card before the error. Both numbers start at zero.
MORE THAN 2 PAGES REQUESTED FOR DCBS	This message indicates that the limit of two pages for DCBs has been exceeded.
NO PROGRAM START ADDRESS	The program has no start address.
'ON' FILE ALREADY EXISTS	ON was specified and the output file already exists.
SEQUENCE ERROR	A sequence error has occurred.
SEVERITY x	The severity level associated with the ROM is specified by x.
STACK OVERFLOW	An internal storage overflow has occurred.
UNEXPECTED END OF ROM DATA	EOF encountered before last card of ROM.

<u>Note:</u> All errors, except CANT FIND, cause abnormal termination of Link.

LEMUR

LEMUR (Library Editor and Maintenance Utility Routine) is a processor that builds and manipulates ROM and load module libraries. The libraries thus built are accessed by LYNX or Load when constructing user programs (load modules) which require library routines. LEMUR is available in both on-line and batch modes.

LEMUR allows the user to

- Construct a library ROM module out of specifiedROMs.
- Construct a library load module out of specified ROMs. (A library load module must be of one protection type.)
- Have more than one library per account.
- Delete a specified portion of a library and all references to that portion in the dictionary.
- Delete a library.
- Copy a library module from library to another.
- Copy a library to another library.

CALLING LEMUR

LEMUR is invoked in the on-line mode by the TEL command

```
! LEMUR
```

When invoked, LEMUR identifies itself and prompts with a greater than (>) character. All commands are read through the M:SI DCB and output is through the M:LL DCB.

Commands are relatively free-format; i.e., blanks are ignored except as delimiters. If a semicolon is encountered in a command line, all subsequent characters in that line are ignored and the next input line is treated as a continuation line. A command line beginning with an asterisk (*) is treated as a comment.

LEMUR CONCEPTS

The following conventions are used in LEMUR:

1. Names of library modules and DEFs consist of a string of any of the following characters:

   A-Z a-z 0-9 _ $ * % : @ #

   They may also consist of a string of the above characters enclosed within single quotes. A library load module name cannot exceed 11 characters. A DEF cannot exceed 14 characters.

2. A file identification has the standard format with the exception that the name, account, or password may be a string of characters enclosed within single quotes. The name portion of a file identification cannot exceed 10 characters if it identifies a ROM which is to be part of a library load module.

3. A rom-id is the file identification of a ROM.

4. A lib-id is the file identification of a library.

5. The term "destination library" is defined to be the library specified by the LIBRARY command. This is the library on which the user wishes to work.

6. The term "default library" implies the :LIB library. If library name is missing from a command in which lib-id is optional, then :LIB is assumed by default. Also, if the LIBRARY command is not used in a LEMUR session, the default and the destination library are the same (:LIB).

7. The term "library module" refers to a named collection of one or more ROMs or a load module which has been entered into the library via a BUILD or a CARRY command. The module gets its "name" when it is entered in this manner.

LEMUR COMMANDS

LIBRARY The library command specifies the destination library (i.e., the library on which the user wishes to work). The format of the command is

```
LIBRARY [name][.[account][.password]]
```

where name, account and password have their usual meanings. If name is omitted, the default is :LIB. If account is omitted, the default is the user's account.

BUILD The BUILD command constructs a library module and enters it into the destination library. The library module constructed may either be a load module or a ROM module, depending pn specifications within the command. The format of the command is

```
    BUILD name FROM rom-id[,rom-id]...┐
└ [(option)[.(option)]...]
```

where

name specifies the name of the library modules to be constructed. If the module name already exists in the destination library, it is deleted and the new version is constructed and entered.

rom-id specifies the name of a ROM to be used in the construction of the load module or ROM library module.

If the name already exists, all old dictionary entries which point to it are deleted. New dictionary entries are then made for each symbol within the new version of the module specified by name. If a dictionary entry for a symbol defined in name already exists in the dictionary (because it is DEFed in some other module with a different name), the entry is changed to point to name.

A library module is either a ROM module (one or more ROMs) or a library load module. The option (ROM) or its absence specifies the type. If the (ROM) option is specified, the module being constructed will consist of the ROMs specified by the rom-id's in ROM form. This allows subroutines with more than one protection type to be included in the library accessed by the loader. Omission of the (ROM) option implies that the library module is to be a load module. In this case, LEMUR invokes the loader to perform the load using the specified ROMs as element files.

Options for the BUILD command

ROM module options: The following options are used if the module being constructed is to be a ROM module. (Load module options have no meaning for ROM modules and will cause an error message if used.)

(ROM) specifies that the module is to be a ROM module consisting of the ROMs specified on the BUILD command.

(M) or
(MN) produces a list of REFs and DEFs in the module, sorted by name.

(SL,value) specifies the ROM severity level that is to be tolerated by LEMUR in forming the library module. The value may range from 0 to F. The default is 7. (The severity level is presented by the ROM.)

Load module options: The following options are used if the module being constructed is to be a library load module.

(C1) specifies that the library load module is to be formed with protection type 01, regardless of the protection type specified in the ROM.

(M) or
(MN) specifies that a load map is to be output on the LL device and that the DEFs are to be sorted by name.

(MV) specifies that a load map is to be output on the LL device and that the DEFs are to be sorted by value.

(MNV) specifies that a load map is to be output on the LL device and that the DEFs are to be sorted by both name and value.

(SS) specifies that a size summary detailing the amount of memory allocated is to be output.

(SL,value) specifies the ROM severity level that is to be tolerated by LEMUR in forming the library load module. The value may range from 0 to F. The default is 7. (The severity level is presented by the ROM.)

(DREF) specifies that all dummy section definitions should be changed to PREFs. This allows a library to be built in which all references to a particular named DSECT will be linked to a single copy of that DSECT (e.g., a FORTRAN BLOCK DATA subprogram). Such initialized dummy sections should be contained in a library ROM module or in a library load module loaded without the DREF option.

(X) specifies that LEMUR should abort if an error is detected in creating the load module. If (X) is not specified in the batch mode, a warning message is issued and LEMUR executes the next command. (Note: The (X) option is meaningful only for running LEMUR in the batch mode. If specified in the on-line mode, the (X) option is ignored.)

Examples:

1. Assume that the user is logged on in account A55 and that the user wishes to

   - Create a new library called LIB5 in account A55.
   - Include R1 as a ROM module.
   - Include R2 and R3 as one ROM module.
   - Include R4 as a load module.
   - Include R5 and R6 as one load module.

   (All these modules are in account A55.)

```
!LEMUR(RET)
>LIBRARY LIB5(RET)
>BUILD LR1 FROM R1(ROM)(M) (RET)
>BUILD LR2 FROM R2,R3(ROM)(RET)
>BUILD LR3 FROM R3(SL,4)(C1)(RET)
>BUILD LR4 FROM R5,R6(SL,4)(M)(RET)
>END(RET)
```

2. Assume that the user is logged on in account A55 and that the user wishes to replace LR3 in the examp'e above with a load module built from R4.OTHERACT.

```
!LEMUR(RET)
>LIBRARY LIB5(RET)
>BUILD LR3 FROM R4.OTHERACT(SL,4)(C1)(RET)
>END(RET)
```

DELETE The DELETE command is used to delete either the destination library (i.e., the library named on the LIBRARY command or :LIB by default) or to delete one or more named modules from the destination library. In the latter case, all entries in the dictionary which point to the deleted module are removed. The format of the command is

```
    DELETE [name[,name]]...
```

where name specifies the name of a library module. If no name is specified, the entire destination library is deleted.

Examples:

1. Assume that the user is logged onto account A55 and wishes to delete modules X, Y, and Z from the library :LIB.A55 and to delete module A from the library LIB6.A55.

```
        !LEMUR(RET)
        >DELETE X,Y,Z(RET)
        >LIBRARY LIB6(RET)
        >DELETE A(RET)
        >END(RET)
```

2. Assume that the user is logged onto account A55 and wishes to delete the library :LIB.A55.

```
        !LEMUR(RET)
        >DELETE(RET)
        >END(RET)
```

COPY The COPY command copies the source library to the destination library. The format of the command is

```
    COPY  lib-id
```

where lib-id specifies the source library. (The destination library was either specified on a LIBRARY command or is :LIB by default.)

The source and destination libraries must be different (i.e., different names or different accounts or different library names in the same account).

Examples:

1. Assume that the user is logged onto account A55 and wishes to copy the library LIBA from account 1234 to library LIBB in account A55.

```
        !LEMUR(RET)
        >LIBRARY LIBB(RET)
        >COPY LIBA.1234(RET)
        >END(RET)
```

2. Assume that the user is logged onto account A55 and wishes to copy library :LIB from account B36 to the :LIB library in account A55.

```
        !LEMUR(RET)
        >COPY .B36(RET)
        >END(RET)
```

CARRY The CARRY command copies a library module from one library (source library) to another library (destination library). The format of the command is

$$\text{CARRY name}_1\ \text{FROM lib-id[/name}_2\text{]}$$

where

name$_1$ specifies the module name in the destination library.

lib id specifies the source library. (The destination library was either specified on a LIBRARY command or is :LIB by default.)

name$_2$ specifies the module name in the source library. If it is omitted, the source module is assumed to be the same as name$_1$ by default.

The source and destination libraries must be different (i.e., different accounts or different library names in the account).

If a module with the name specified by name$_1$ already exists in the destination library, then the original name$_1$ module records and dictionary records which point to it are deleted from the destination library, with all name$_2$ module and dictionary records being copied from the source library and entered into the destination library.

If a symbol in the name$_2$ module already exists as a dictionary entry in the destination library and it points to a module other than name$_2$, it will be replaced by the new entry pointing to name$_1$.

Examples:

(In all these examples, assume that the user is logged onto account A55.)

1. The user wishes to carry module ZAP from NEWLIB.:SYS to ZAP in NEWLIB.A55.

```
        !LEMUR(RET)
        >LIBRARY NEWLIB(RET)
        >CARRY ZAP FROM NEWLIB.:SYS(RET)
        >END(RET)
```

(Omission of the source module name (name$_2$) implies that ZAP is the source module name.)

2. The user wishes to carry module ZAP from LIB2.ACN2 to module MAP in LIB2.A55.

```
        !LEMUR(RET)
        >LIBRARY LIB2(RET)
        >CARRY MAP FROM LIB2.ACN2/ZAP(RET)
        >END(RET)
```

3. The user wishes to carry module SQRT from :LIB.:SYS to SQRT in :LIB.A55.

```
        !LEMUR(RET)
        >CARRY SQRT FROM .:SYS(RET)
        >END(RET)
```

(Omission of a LIBRARY command implies that the destination library is to be :LIB.A55 by default. Omission of the source library name implies :LIB by default.)

4. The user violates the rule that the source and destination libraries must be different.

```
   ! LEMUR(RET)
   > CARRY ZAP FROM :LIB/MAP(RET)
   > END(RET)
```

An error message is issued and the command is aborted.

END The END command terminates LEMUR and returns control to TEL. The format of the command is

```
END
```

ERROR MESSAGES

Table 35. LEMUR Error Messages

Message	Meaning
")" MISSING AFTER OPTION	Self-explanatory.
ACCOUNT NAME TOO LONG	The account name exceeds eight characters.
BAD FILE I.D.	Self-explanatory.
BAD QUOTE STRING	An illegal character occurred within a string.
CAN'T CREATE LIBRARY	An I/O error occurred when trying to create a new library.
CAN'T OPEN FILE	Either the ROM id doesn't exist, the module doesn't exist (DELETE), or the source module doesn't exist (CARRY).
CAN'T OPEN LIBRARY	An I/O error occurred when trying to open an existing library.
COMMAND TOO LONG	A command (including continuations) is too long for LEMUR's command buffer of 256 characters.
EH?	A command is malformed.
FILE NAME TOO LONG	A file name exceeds ten characters.
GARBAGE AT END OF LINE	A command contains unrecognizable characters.
I/O ERROR	Self-explanatory.
ILLEGAL CONTINUATION LINE	Self-explanatory.
ILLEGAL LIBRARY FORMAT	A reference to a file which is supposed to contain a library was made in a LEMUR command, but the file is not in library format.
ILLEGAL OPTION FOR THIS COMMAND	Self-explanatory.
ILLEGAL ROM LANGUAGE	The ROM is malformed.
ILLEGAL ROM RECORD HEADER	The ROM is malformed.
ILLEGAL ROM RECORD LENGTH	The ROM is malformed.
LIBRARY NAME MISSING	A required library name is missing in a command.

Table 35. LEMUR Error Messages (cont.)

Message	Meaning
LIBRARY NAME TOO LONG	A library load module name exceeds 11 characters.
MALFORMED OPTION	Self-explanatory.
MAXIMUM SEVERITY LEVEL EXCEEDED	The severity level specified by the SL option has been exceeded.
MISSING FILE NAME	A required file name is missing in a command.
MODULE NAME MISSING	A required module name is missing in a command.
NOT ENOUGH CORE	There is insufficient common or virtual memory to satisfy the requirements for I/O buffers used in the COPY and CARRY commands.
NOT ENOUGH SYMBOL SPACE	The space required by LEMUR to construct the dictionary is insufficient.
PASSWORD TOO LONG	The password exceeds 8 characters.
SOURCE SAME AS DESTINATION LIBRARY	The requirement that the source and destination libraries be different on the COPY and CARRY commands has been violated.
UNEXPECTED END OF ROM	The ROM is malformed.
UNKNOWN COMMAND	Self-explanatory.
UNKNOWN OPTION	Self-explanatory.
YOU USED THE SAME OPTION TWICE	Self-explanatory.

COMMAND SUMMARY

The LEMUR commands are summarized in Table 36.

Table 36. LEMUR Command Summary

Command	Function
BUILD name FROM rom-id [,rom-id]... [(option)[(option)]...	Creates and enters a ROM module or library load module into the destination library.
CARRY name_1 FROM lib-id[/name_2]	Copies a library module from a source library to the destination library.
COPY lib-id	Copies a source library to the destination library.
DELETE	Deletes either the entire destination library or one or more modules from the destination library.
END	Terminates LEMUR.
LIBRARY [name][.[account][.password]	Defines the destination
*	Indicates that the line is a comment line.

9. BATCH PROCESSOR

INTRODUCTION

The Batch processor is used to submit a file or a series of files to the batch queue for execution. Through Batch processor commands, the following capabilities are available:

1. Files may be inserted into a file being sumbitted for execution, thus bringing together more than one file to create a single job.
2. Selected strings and fields existing in files being submitted for execution may be replaced by new strings and fields.
3. The results of string and field replacements can be examined before the job is submitted to the batch stream.
4. Files to be submitted for execution may reside on tape or private disk pack.
5. Jobs may be submitted to run in an account other than the account from which the job is submitted.

The Batch processor may be called in either the on-line, ghost, or the batch mode. The file to be submitted must include all appropriate batch control commands that would be needed for normal batch job submission. However, the specification field of the JOB control command may optionally be left blank and the batch processor will supply the missing subfields before submitting the job to the batch queue. Each record in the file must not exceed 80 characters.

Any user with at least C0 privilege may enter jobs to run in accounts other than the account through which the job is submitted.

When a job is submitted through the Batch processor, the system responds by assigning the job a job identification (jid) and sending one of the following messages to the terminal or printer (via M:LL):

```
ID = jid SUBMITTED time-date
WAITING: n TO RUN
```

or

```
ID = jid SUBMITTED time-date
RUNNING
```

If the user is an on-line user, he may check the status of the job by using the JOB command or may cancel the job using the CANCEL command. These two commands are described in Chapter 3.

DATA REPLACEMENT

There are five Batch processor commands. Three of the five commands allow the user to request data replacements. As each record from the input file is read, it is examined to see if any data replacement requests apply to it. If so, the appropriate substitutions are made and the resulting record is placed in the job stream (except when the "test" mode has been requested).

Data replacement requests have the same format regardless of which command they appear in. The general format of data replacement specifications is discussed in the following paragraphs. The specific effect of data replacement requests is discussed in the descriptions of the individual commands.

There are two types of data that may be replaced: fields and strings.

A field is defined to be a contiguous set of nondelimiters bounded on either side by a delimiter or by the left or right record boundary. The nondelimiters are:

```
A-Z
0-9
#
@
:
$
```

In the following two lines, the fields are underscored.

```
AB+44+(XYZ, :ABC) IS THE FABIT#
!ASSIGN F:INPUT, (LABEL, MYTAPE, ACCT#6)
     (SN, IN)
```

A string is defined to be part of a field or of a set of contiguous fields. Any part of a record may be treated as a string. In fact, the entire record may be treated as one string. The only limitation on string replacement is that the string may not contain a quote character (because a quote character is used to specify a string in a data replacement specification).

The general format of a data replacement specification is:

$$\begin{Bmatrix}\text{field}\\ \text{'string'}\end{Bmatrix} = \begin{Bmatrix}\text{field}\\ \text{'[string]'}\end{Bmatrix}$$

The left side specifies what is to be replaced and the right side specifies the replacement. The format allows fields and strings to replace each other interchaneably. It also allows a replacement string to be a null string.

Examples:

In the following examples, the replacement specification will be applied to the following record:

```
!ASSIGN F:IN, (LABEL, A123, ACCT#6)
```

Each example is to be regarded as independent of the other examples.

Replacement Specification	Result
a. A123 = B456	!ASSIGN F:IN, (LABEL, B456, ACCT#6)
b. 'IN' = 'INPUT'	!ASSIGN F:INPUT, (LABEL, A123, ACCT#6)
c. 'A123, ACCT#6'=NEWTAPE	!ASSIGN F:IN, (LABEL, NEWTAPE)
d. ', ACCT#6' = ''	!ASSIGN F:IN, (LABEL, A123)

The last example illustrates that string data replacement requests can be used to eliminate characters.

Note that the user must specify data replacement requests very carefully. For example, the specification 'A' = 'C' would have the following effect:

```
!CSSIGN F:IN, (LCBEL,C123,CCCT#6)
```

The request ACCT=ACCOUNT would have no effect because in this example ACCT is not a field by itself. To change ACCT to ACCOUNT, the specification might be 'ACCT'='ACCOUNT'.

The following restrictions are placed on data replacement specifications. No more than 50 data replacement requests may be made per file. There may be no more than 470 characters in the data replacement requests per file (including the left and right side, the equal sign, and quote characters).

Precedence of data replacement requests is in the order of appearance within the Batch processor commands. When replacement of the same field or string is requested more than once, only the first request is honored.

COMMAND CONTINUATION

Due to data replacement specifications, Batch processor commands can sometimes be quite lengthy. Any Batch processor command can be continued from one card or line to the next simply by using a semi-colon at the end of the card or line to be continued. If a semi-colon is present on a card or line, the first character of the next card or line effectively overlays the semicolon. A command cannot exceed 255 characters in length.

When a command is continued in the on-line mode, the Batch processor prompts for a continuation line with a dollar sign ($).

Example:

```
!BATCH FILE1, PILE2  ;(RET)
$P1=224,;(RET)
$'XXX, VVV'=FFF(RET)
:
```

As will be seen later, the blank after FILE2 is mandatory. The user must ensure that such blanks are not left out when continuing a command from one line to the next.

BATCH COMMANDS

There are five Batch processor commands. They are:

BATCH
DEFAULT
EOF
EXEC
EOF EXEC

The BATCH command is a control command that (among other things) calls the Batch processor. The remaining commands must be embedded within the file being submitted for execution. Their location within the file determines what portion of the file they affect.

All Batch processor commands begin with an exclamation point, even those that appear within the input deck.

BATCH The BATCH command calls the Batch processor, specifies the files that are to be submitted for execution, specifies Batch processor options to be used, and specifies data replacement. The format of the command is:

```
!BATCH[([P][E][S][T])][fid[,fid]...][rep[,rep]...]
```

where

P specifies the "print" mode. In this mode, every record that is submitted for execution is printed through the F:BATCH DCB. (The F:BATCH DCB is discussed briefly in the section "Batch Error Messages".)

E specifies that EXEC commands are to be honored. An EXEC command is a Batch processor command and is described below. If E is not specified, EXEC commands are treated simply as data records.

S specifies that the input file is not named on the BATCH command. Instead, the user has issued a SET or ASSIGN command that has assigned the M:EI DCB to the input file. For example:

```
!SET M:EI/INFILE (on-line mode)

!ASSIGN M:EI,(FILE, INFILE) (batch mode)
```

T specifies the "test" mode. In this mode, the Batch processor prints (through the F:BATCH DCB) each record it alters because of data replacement requests and does not submit the job to the batch queue for execution. This allows the user to examine the effects of data replacement requests before submitting the job for execution. The original file is not modified, thus allowing the user to experiment.

fid identifies a file in one of the formats below

name

name.account

name..password

name.account.password

rep is a data replacement specification in the format described previously.

Example:

Assume that the following file (FILEA) exists in swap storage:

```
!JOB MYNAME, MYACCT(SUBACCT#88)

!ASSIGN M:LO, (DEVICE, LP), (VFC)
.
.
.
```

and that the following BATCH command is used to submit FILEA:

```
!BATCH FILEA '88'='89', VFC=NOVFC
```

The following changes would be made:

```
!JOB MYNAME, MYACCT(SUBACCT#89)

!ASSIGN M:LO, (DEVICE, LP), (NOVFC)
.
.
.
```

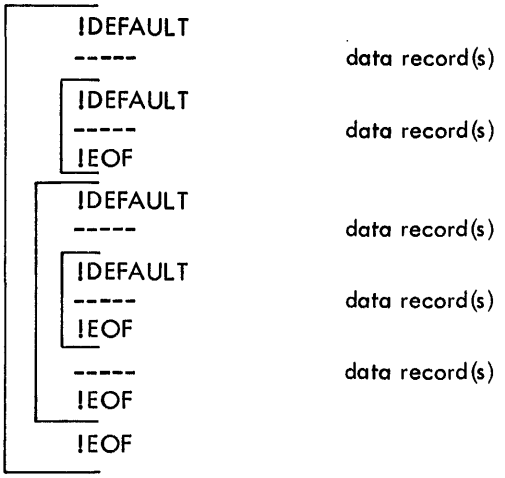
DEFAULT The DEFAULT command allows data replacement requests to be made within the input file. The DEFAULT command may appear any number of times and anywhere within the file being submitted and is effective on subsequent records of that file. If a data replacement request on a DEFAULT command is made for a field or string for which a data replacement request was also made on the BATCH command, the BATCH request overrides the DEFAULT request. The format of the DEFAULT command is

```
!DEFAULT rep[,rep]...
```

where rep is a data replacement specification in the format described previously.

EOF The EOF command specifies that all data replacement requests made on the previous DEFAULT command are not to be effective on subsequent records of the file.

The DEFAULT and EOF command functions may be considered to operate in pairs. This is shown schematically as follows:



```
!DEFAULT
-----                  data record(s)
!DEFAULT
-----                  data record(s)
!EOF
!DEFAULT
-----                  data record(s)
!DEFAULT
-----                  data record(s)
!EOF
-----                  data record(s)
!EOF
!EOF
```

The EOF command does not affect data replacement requests that were made on the BATCH command. The format of the EOF command is:

```
!EOF
```

EXEC The EXEC command allows the user to insert one file within another file. The EXEC command has the following format:

```
!EXEC fid [rep[,rep]...]
```

where

fid identifies the file to be inserted in one of the formats below:

name
name.account
name..password
name.account.password

rep is a data replacement specification in the format described previously.

The EXEC command is replaced by the entire file named on the EXEC command. The EXEC command can appear any number of times and anywhere within the user's file. If the E option is not specified on the BATCH command, the EXEC commands are treated as ordinary data records and are moved to the job stream. EXEC commands within EXEC files are also treated as ordinary data records and are moved to the job stream; however, their presence in the file will cause an error at a later time.

The data replacement requests on the EXEC command apply only to the EXEC file. All previous data replacement requests on the BATCH command or on DEFAULT commands do not apply to the EXEC file. (Such data replacement requests resume their effect after the EXEC file has been completely inserted.) However, it is important to note that an EXEC command is subjected to data replacements specified on the BATCH command and on previous DEFAULT commands before the EXEC command is processed.

DEFAULT and EOF commands within the EXEC file apply only to that file and function as previously described.

EOF EXEC The EOF EXEC command specifies that all data replacement requests made on either the BATCH command or an EXEC command (if the EOF EXEC command appears within an EXEC file) are not to affect subsequent records of the file. The EOF EXEC command may appear anywhere within the user's file. (It does not affect requests that were made on a DEFAULT command.) The format of the command is:

```
!EOF EXEC
```

BATCH ERROR MESSAGES

Error conditions that may be encountered and reported by the Batch processor are listed in Table 37. These messages are output through the F:BATCH DCB. F:BATCH is set to the terminal for on-line users and the line printer for batch users. In addition to these error messages, there are several self-explanatory messages which may be issued by the monitor's file management routines to report such things as the file does not exist or the file has a password which was not specified.

BATCH PROCESSOR COMMAND SUMMARY

The Batch processor commands are summarized in Table 38. The left-hand column gives the command formats. The right-hand column describes the command functions.

Table 37. Batch Processor Error Messages

Message	Description
BATCH QUEUE FULL	No more symbiont space is available or the queue is full.
BATCH WHAT?	No file was specified on the BATCH command and the M:EI DCB was not assigned to a file.
BLANK NOT ALLOWED IN XACCT FIELD	A blank is not allowed in the extended accounting field on the JOB command.
***** CAN'T GET DYNAMIC PAGE	There is a problem in the system. Notify the system analyst.
COMMAND REJECTED	The file contains a BIN or FIN control command. The BIN or FIN command was ignored.
***** COMMAND TOO LONG	A BATCH, DEFAULT, or EXEC command (with its continuations) has exceeded 400 bytes. The job is aborted.
COMPLETED OR NOT INPUT	The user attempted to CANCEL a job that is not an input file or has already been processed.
DATA LOST ON RECORD nnnn	The job expects card image input: 80 characters-per-record maximum, EBCDIC; 120 characters-per-record maximum, binary.
EH? @ n	A syntax error exists at character n.
ILLEGAL ACCOUNT	The account on the JOB control command must match the user log-on account.
ILLEGAL NAME	The name on the JOB control command must match the user log-on name.
ILLEGAL PRIORITY	The terminal-batch job priority may not exceed the user's maximum on-line priority. This maximum value is contained in the user's job-information-table (JIT).
***** JOB ABORTED	Due to syntax errors listed previously, the job was aborted.
***** JOB NOT SUBMITTED BECAUSE OF ERRORS	Due to syntax errors listed previously, the remainder of the file was processed for syntax errors (without data replacement) but the job was not submitted, for execution.
MISSING JOB COMMAND	The first record of the job must be a JOB control command.

Table 37. Batch Processor Error Messages (cont.)

Message	Description
***** MODIFIED DATA RECORD EXCEEDS 80 BYTES	Data replacement for the record listed below this message has caused the length of the record to exceed 80 bytes. The remainder of the file is processed for syntax errors (without data replacement) but the job is not submitted for execution.
NO REPLACEMENT MADE	The user specified replacement requests but no matches were found. The job is submitted for execution unless the "test" mode was specified.
NOT YOUR FILE	The user attempted to CANCEL a job that was submitted under another account.
***** SYNTAX ERROR IN ABOVE LINE	Self-explanatory. The remainder of the file is processed for syntax errors (without data replacement) but the job is not submitted for execution.
***** TOO MANY REPLACEMENT REQUESTS	Either more than 50 data replacement requests have been made for one file or the number of replacement requests for one file exceeds 470 characters.
***** WHILE PROCESSING FILE filename	The above errors occurred while processing this file.
XACCT FIELD NOT TERM. BY RT. PAREN.	Either a comma or a left parentheses occurred before the right parentheses in an extended accounting field. (The extended accounting field must be terminated by a right parentheses or the end of the command.)

Table 38. Batch Processor Command Summary

Command	Description
!BATCH ([P][E][S][T]) [fid[,fid]...][rep[,rep]...]	Calls the Batch processor, specifies the files that are to be submitted for execution, specifies Batch processor options to be used, and specifies data replacement requests. Options: P - "print" mode is to be used. E - EXEC commands are to be honored. S - user has assigned the M:EI DCB to the input file. T - "test" mode is to be used. rep - specifies a data replacement request and has the form: $\left\{\begin{matrix}\text{field} & = & \text{field}\\ \text{'string'} & & \text{'[string]'}\end{matrix}\right\}$
!DEFAULT rep[,rep]...	Specifies data replacement requests within the input file. The rep specification is the same as for the BATCH command.
!EOF	Specifies that all data replacement requests made on a previous corresponding DEFAULT command are not to be effective on subsequent records of the file.
!EOF EXEC	Specifies that all data replacement requests made on either the BATCH command or an EXEC command (if the EOF EXEC command appears within an EXEC file) are not to affect subsequent records of the file.
!EXEC fid [rep[,rep]..]	Specifies that the entire file named on the command is to be inserted at the location of the EXEC command and specifies data replacement requests for that file. The rep specification is the same as for the BATCH command.

10. MONITOR SERVICES TO USER PROGRAMS

INTRODUCTION

All monitor services available to batch and on-line programs are described in the CP-V/BP Reference Manual, 90 17 64. Those services that are available only to on-line programs are discussed in this chapter. In addition, a description of differences between on-line and batch responses to certain procedures is provided.

ON-LINE CP-V SERVICE CALLS

SET PROMPT CHARACTER

M:PC When control is turned over to an on-line program, a null prompt character is assigned unless the user has set a default prompt character via the TEL PROMPT command. The PC routine allows an on-line program to set a different prompt character. This character, if non-null, is typed (usually at the left margin) whenever input is requested from the terminal (UC device). If M:PC is used in a batch program, it is ignored.

The procedure call is of the form

M:PC 'character'

where character specifies the EBCDIC prompt character that is to be associated with the user program (an EBCDIC 00 or null character means that no prompt character is desired.)

Calls generated by the M:PC procedure have the form

CAL1, 1 fpt

where fpt points to the FPT shown below.

X'2C'	0——————————0	EBCDIC Prompt Character
0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31

CHANGE TERMINAL TYPE

M:CT The CT routine allows an on-line program to switch among the terminal translations provided by the COC I/O routines. Tables related to each terminal type control the translation of characters transferred between the computer and the terminal.

The CT routine also affects other functions treated differentially by terminal type. These functions include certain line editing and terminal control functions.

The procedure call is of the form

M:CT number

where number specifies the number of the desired table.

Calls generated by the M:CT procedure have the form

CAL1,8 fpt

where fpt points to the FPT shown below.

X'06'	Terminal Type
0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

The current tables translate for Models 33, 35, and 37 Teletypes, the Xerox 7015 Keyboard/Printer, and the IBM 2741. Available terminal type numbers are listed in Table 39.

Table 39. Terminal Type Numbers

Number	Meaning
0	Teletype Model 33.
1	Teletype Model 35.
2	Teletype Model 37.
3	Xerox Model 7015 Keyboard/Printer.
4(-5)[†]	IBM 2741 Terminal EBCD Standard.
6(-7)[†]	IBM 2741 Terminal EBCD APL.
8(-9)[†]	IBM 2741 Terminal Selectric Standard.
10(-11)[†]	IBM 2741 Terminal Selectric APL.

[†] For M:TS, the terminal number is even if the terminal is in lowercase and odd if the terminal is in uppercase.

CC1 is set if there is an illegal type code or M:CT is not in an on-line program.

SET TERMINAL ATTRIBUTES

M:STA The STA routine allows an on-line program to set the terminal attributes. The procedure call is of the form

M:STA (option)[,(option)]...

where the options are described below. If the option specifies a flag, ON or OFF may be specified; otherwise, an integer must be specified, where a value of 1 indicates ON and 0 indicates OFF. If an integer is specified, or if the option calls for a value, it must not be a forward reference.

ACS,value specifies the activation character set to be used in determining the end of the input message. Value must be in the range of 0 to 3. See M:CAC.

AFTER,[*]value specifies the number of blank lines to be output after a page header is printed. Value must be in the range of 0 to 15.

ALGORITHM,value specifies the timing algorithm to be used in computing the number of idles to send before and after carriage return characters, and after tab characters. Value must be in the range of 0 to 7. See Table 10.

BEFORE,[*] value specifies the number of blank lines to be output before a page header is printed. Value must be in the range of 2 to 15.

BSEDIT,flag specifies whether backspace edit mode is enabled. See ESC O and O ATTN.

BRKCNT,value specifies the number of contiguous breaks received. BRKCNT is normally used only to reset the break count to avoid having a CONTROL Y simulated because of four contiguous breaks.

COUPLE,flag specifies whether terminal coupling is enabled. See the TEL COUPLE command.

CPOS,[*] value specifies the current position of the terminal cursor.

CRTTYPE,value specifies the type of pagination to be used in end-of-page situations. Value must be in the range of 0 to 7, and is described in Table 45.

DONTSEND,flag specifies whether the operator SEND key-in is to be inhibited. See the TEL DONT SEND command.

ECHOPLEX,flag specifies whether the line is to operate in the ECHOPLEX mode. See ESC E.

FDPTAPE,flag specifies whether the line is to operate in the full duplex paper tape mode. See X-ON under PAPER TAPE INPUT in Chapter 2.

HANGUP,flag specifies whether the line is operating in the hang up mode. When set, reads are disallowed, and termination of queued output will cause the associated dataset to be disconnected. This flag can only be set and reset if special JIT access is set, which is not the case for user programs.

HDPTAPE,flag specifies whether the line is to operate in the half duplex paper tape mode. See ESC P.

IGNORE,flag specifies whether the line is to operate in the input-ignore mode. See ESC Z.

LCSHIFT,flag specifies whether upper case characters are to be shifted to lower case on input. See ESC) and ESC (.

LENGTH,[*]value specifies the page length', and must be in the range of 0 to 255. See the TEL PLATEN command.

PARITYCHECK,flag specifies whether parity is to be checked on ASCII lines on input. Even parity is considered to be normal. Parity is always checked on IBM 2741 lines.

SPACEINSERT,flag specifies whether, when processing tab characters on input, spaces are to be substituted when transferring the input stream to the user's buffer. See ESC S.

TABREL,flag specifies whether carriage positioning when handling tab characters is to be relative to the position at the start of the read, or is to be an absolute column position. See ESC C.

TABSIM,flag specifies whether tabbing is to be simulated by positioning the carriage with space characters. See ESC T.

TRANSLATION,[*]value specifies the translation table to be used for translating the input and output characters to and from EBCDIC. Value may be a number listed in Table 39, or may be a four-character left-justified character string enclosed in quotes and listed under type in Table 9.

UCSHIFT,flag specifies whether lower case characters are to be shifted to upper case on input. See ESC U.

WIDTH,[*]value specifies the page width, and must be in the range of 0 to 255. See the TEL PLATEN command.

Calls generated by the M:STA procedure have the form

```
CAL1,8    fpt
```

where fpt points to word 0 of the FPT shown below.

word 0

X'06'	2	0 ——————————— 0
0 1 2 3 4 5 6 7	8 9 10 11	12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

word 1

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	0 ——————————— 0
0	1	2	3	4	5	6	7	8	9	10	11	12	13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

The first three bits of word 1 indicate which of the following words will follow in the FPT. For example, if P_1 and P_3 are set to 1 and P_2, P_4, P_5, P_6, P_7, P_8, P_9, and P_{10} are set to zero, then the MODE and MODE3 words will be words 2 and 3 of the FPT.

Set MODE (P_1)

*	Bit values	Selection mask
0	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Set MODE2 (P_2)

*	Bit values	Selection mask
0	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Set MODE3 (P_3)

*	Bit values	Selection mask

Set MODE4 (P_6)

*	Bit values	Selection mask

Set MODE5 (P_8)

*	Bit values	Selection mask

Set MODE6 (P_9)

*	Bit values	Selection mask

The "selection mask" indicates which bits in the specified table (MODE, MODE2, MODE3, MODE4, MODE5, or MODE6) are to be set. (The bits in the tables control and/or reflect the terminal attributes.) The "bit values" specify the desired setting for the bits selected by the "selection mask". A one means the attribute should be turned "on"; a zero means the attribute should be turned "off".

Only the attributes preceded by an asterisk in Tables 40, 41, 42, 43, and 44 may be set by the user. Any other attributes specified are ignored by the monitor. (The other attributes are listed for reference in the discussion of the M:TS procedure.) The user specifies which attributes he would like to set (i.e., turn on or off) by setting the appropriate bit in the selection mask to one.

For MODE5 terminal attributes only Selection mask bit 26 (with hexadecimal value 20) can be set by the user. If it is set to one, operator messages sent via the operator SEND key-in will be deferred.

Set Platen Width (P_4)

*	0 —————————— 0	width

Width is the maximum number of characters to be written per line on a terminal and must be in the range of 0-255. If more characters are written for a line, a line feed and carriage return sequence is inserted. If the width is 11 or less, no line feed and carriage return sequence is supplied.

Set Page Length (P_5)

*	0 —————————— 0	length

Length is the maximum number of lines per page of terminal output and must be within the range 0-255. If the length is 11 or less, no heading is produced and the page length is unlimited.

Set Terminal Type (P_7)

*	0 —————————— 0	Terminal type number

See the M:CT call description for an explanation of this parameter. If indirect addressing is specified, the word that is addressed may contain either a terminal type number, or a terminal name composed of four characters of left-justified text. Terminal names are listed in the TEL TERMINAL STATUS command description. The terminal type word in the FPT may also consist of up to four characters, left justified in a word and blank-filled to the right (if necessary).

Set COCOPT2 (P_{10})

*	0 —————————— 0	Z	0—0	PAGE TYPE

where

Z is the ESC Z mode flag. If set, input is ignored.

Page type is the pagination type as described in Table 45.

Set Lines Before Header (P_{11})

*	0 —————————— 0	lines

Lines is the number of blank lines to be output before a page header is printed, and must be in the range 2-15.

Set Lines After Header (P_{12})

*	0 —————————— 0	lines

Lines is the number of blank lines to be output after a page header is printed, and must be in the range 0 to 15.

Set Cursor position (P_{13})

*	——————————	CPOS

CPOS is the current position of the terminal cursor. Note that setting CPOS has no effect on the physical cursor. This option is meant to be used to inform the system of the physical cursor's actual position if special terminal I/O has occurred (e.g., transparent mode, graphics, special CRT cursor positioning).

Table 40. MODE Terminal Attributes

Selection Mask Bit	Hexadecimal Value	Attribute
*24	80	Echoplex (full-duplex; not for 2741s).
25	40	Escape (ESC) sequence in progress.
26	20	Transparent mode.
27	10	Read pending (0 read ahead).
*28	08	Tab simulation.
*29	04	Restrict code to upper case.
*30	02	Break count.
*31	01	

Table 41. MODE2 Terminal Attributes

Selection Mask Bit	Hexadecimal Value	Attribute
24	80	Line reported off.
*25	40	Full-duplex paper tape mode.
*26	20	Space insertion.
27	10	2741 line.
*28	08	Shift to lowercase.
*29	04	Check parity mode.
*30	02	Break set.
*31	01	

(This page intentionally left blank)

Table 42. MODE3 Terminal Attributes

Selection Mask Bit	Hexadecimal Value	Attribute
*24	80	Tab relative to beginning of input.
*25	40	Half-duplex paper tape mode.
*26	20	Backspace edit.
27	10	2741 keyboard locked.
28	08	Lost input (insufficient buffers).
29	04	Number of lines upspaced during input.
30	02	
31	01	

Table 43. MODE4 Terminal Attributes

Selection Mask Bit	Hexadecimal Value	Attribute
*24	80	Accept couple, or coupled to line.
25	40	Couple rejected.
26	20	Insert mode active.
27	10	Unused.
28	8	Unused.
*29	4	Timing algorithm number.
*30	2	
*31	1	

Table 44. MODE6 Terminal Attributes

Selection Mask Bit	Hexadecimal Value	Attribute
24	80	Half-duplex line.
25	40	Half-duplex line in input mode.
26	20	Read with user-controlled timeout.
27	10	Output halt mode.
28	8	Half-duplex line turnaround, phase 1.
29	4	Half-duplex line turnaround, phase 2.
*30	2	Unused.
31	1	Hardwired line.

Example:

If word 1 of the FPT contains

```
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
```

and word 2 contains

```
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
```

then the echoplex mode will be turned on and the tab simulation mode will be turned off.

Table 45. Pagination Types

Pagination Type	Usage	Action at End of Page
0	Any	Normal pagination.
1	Tektronix CRTs	1. Set output half mode (see ESC H). 2. After output halt has been cleared by the user, send page erase/home sequence (ESC FORMFEED). 3. Wait from .875 to 2.4 seconds. 4. Resume normal processing.
2	Teletype Model 40 CRT	1. Set output halt mode (see ESC H). 2. After output halt has been reset by the user, send page erase/home sequence (ESC H ESC J). 3. Wait .25 seconds. 4. Resume normal processing.
3-7	Most CRTs	1. Set output halt mode (see ESC H). 2. After output halt has been reset by the user, resume normal processing.

OBTAIN TERMINAL STATUS

M:TS The TS routine provides an on-line program with the current status of data used by the COC I/O routines to control the functional characteristics of the terminal.

The procedure call is of the form

M:TS

The call generated by the M:TS procedure has the form

CAL1,8 fpt

where fpt points to the FPT shown below.

X'06'	4	0 ———————————————— 0
0 1 2 3 4 5 6 7	8 9 10 11	12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Upon return to the caller, registers 8 and 9 contain the following

Register 8

COCTERM	MODE	MODE2	MODE3
0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31

Register 9

CPOS	COCOC	BUFCNT	LB:UN
0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31

The meanings of the values in COCTERM, MODE, MODE2, and MODE3 are listed in Tables 39, 40, 41, and 42 respectively. CPOS contains the current carrier position. COCOC contains the current number of characters remaining for output. If the count is greater than 255, 255 will be returned. BUFCNT contains the current number of COC buffers in use for input. LB:UN contains the user number associated with this line.

An extended version of the M:TS call is also available. The procedure call is of the form

M:TS2

The call generated by the M:TS2 procedure has the form

CAL1, 8 fpt

where fpt points to the FPT shown below.

X'06'	6	0 ———————————————— 0
0 1 2 3 4 5 6 7	8 9 10 11	12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

This CAL returns the same information in the same registers as the standard M:TS CAL. In addition, the following information is returned:

Register 10

MODE4	CPI	▨	I	TIE
0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22	23	24 25 26 27 28 29 30 31

Register 11

MODE5	MODE6	Platen Width	Page Length
0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31

MODE4, platen width, and page length are described in the terminal mode change CAL description.

CPI is the carriage position at the start of the last M:READ. If I is set, then input buffers (type ahead) exists.

If the X'40' bit of MODE4 is set, TIE contains the line number of the user attempting to couple to the line. If the X'80' bit of MODE4 is set, TIE contains either the user's line number or the line number to which the user is coupled.

Another extension of the M:TS call is also available. The procedure call is of the form

M:TS3

The call generated by the M:TS3 procedure has the form

CAL1,8 fpt

where fpt points to the FPT shown below.

X'06'	X'B'	0 ———————————————— 0
0 1 2 3 4 5 6 7	8 9 10 11	12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

This CAL returns the same information in the same registers as M:TS2. In addition, the following information is returned:

Register 12

Terminal translation type (in left-justified TEXT format)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Register 13

MODE 4INIT	COCOPT2	LBPH	LAPH
0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31

where the meaning of the hexadecimal value in MODE4INIT is described in Table 46; the COCOPT2 byte has the same meaning as COCOPT2 of the M:STA procedure and is described in Table 45; LBPH and LAPH indicate the number of blank lines output before and after (respectively) a page header is printed at the terminal.

Registers 14 and 15 are reserved for future use.

Table 46. MODE4INIT Terminal Attributes

Hexadecimal Value	Attribute
80	Unused.
40	Unused.
20	Initial timing algorithm number.
10	
8	
4	Line speed indicator.
2	
1	

PURGE TERMINAL I/O BUFFERS

M:PURGE This procedure allows the user program to release current unprocessed terminal I/O. Input buffers, output buffers, or both input and output buffers may be purged.

Input and/or output may also be purged with the M:READ and M:WRITE procedure calls.

The procedure call has the format

M:PURGE (option)[, (option)]

The options are:

WRITE specifies that any output remaining in the terminal line's output buffers is not to be output to the terminal. The output line buffers will be released to the system and the write operations associated with them will be treated as having completed normally.

READ specifies that any read-ahead data accumulated in the line's input buffers is not to be moved to the user's buffer when the next read request is made. The input line buffers will be released to the system.

Calls generated by the M:PURGE procedure have the form

CAL1, 8 fpt

where fpt points to the FPT shown below.

X'06'	7	▨	W	R
0 1 2 3 4 5 6 7	8 9 10 11	12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29	30	31

where

R = 1 if READ was specified.

W = 1 if WRITE was specified.

CHANGE ACTIVATION CHARACTERS

M:CAC The M:CAC procedure allows the calling program to choose among three sets of message-terminating, or activation, characters for terminal input. The normal set of activation characters is: CR, LF, FF, FS, RS, US, GS, EOT, SUB, and ESC F.

Three additional activation sets are available that augment the normal activation set. They are:

1. "All" special graphics and control characters.

2. "All" control characters.

3. "Delta" activation characters.

Character-count-satisfied is also an activation condition for all sets. (Activation on every character can be achieved by requesting one-character read operations.)

The procedure call is of the form

M:CAC number

where number specifies the activation set. (Number may be any value between 0 and 3.)

Calls generated by the M:CAC procedure have the form

CAL1, 8 fpt

where fpt points to the FPT shown below.

X'06'	1	0——0	n	0——0
0 1 2 3 4 5 6 7	8	9 10 11 12 13	14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

where

n = 0 for normal activation set.

n = 1 for the special graphics and Teletype control characters defined below.

n = 2 for the Teletype control characters defined below and for EOT activation on 2741s.

n = 3 for Delta activation and no prompting.

The special graphics characters are

] [} { \ " = ' @ # : ? > _ % , ^ / - ¬ ;) * $! &
| + (< . ¢

The control characters are

SOH, STX, ETX, HT, ACK, BEL, BS, ENQ, NAK, VT, SO, SI, DLE, DC2, DC4, SYN, ETB, CAN

All characters are transmitted to a reading program in their Xerox Standard EBCDIC value (see Appendix A). Note that the control characters EM, ESC NUL (ignore), and DEL (RUBOUT) are not included in any set.

The activation character set may be temporarily changed with the OACS option on the M:READ procedure.

TERMINAL COUPLING

The following four procedures allow the user to control terminal coupling directly in his program. (Terminal coupling is described in Ch^pter 3.) If the CALs associated with these procedures are issued by a program that is not on-line, no action takes place and CC1 is set to 1 on return.

M:ACPL This procedure puts the terminal in a mode such that it will accept coupling from another terminal. The procedure call is of the form

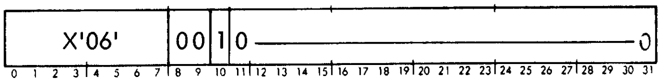
M:ACPL

Calls generated by the procedure have the form

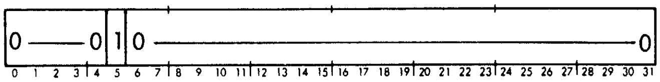
CAL1,8 fpt

where fpt points to word 0 of the FPT shown below.

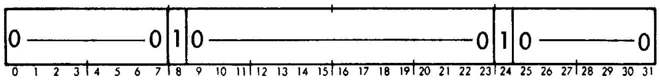
word 0



word 1



word 2



M:RCPL This procedure puts the terminal in a mode such that attempts to couple to it will be rejected. (The procedure does not release any current coupling.) The procedure call is of the form

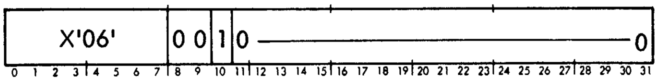
M:RCPL

Calls generated by the procedure have the form

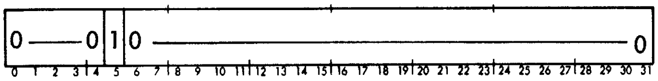
CAL1,8 fpt

where fpt points to word 0 of the FPT shown below.

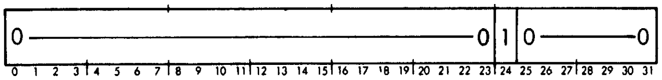
word 0



word 1



word 2



Internally, the action of the two CALs above is to set or reset the high order bit of the byte table MODE4. Both also reset the 40 bit of MODE4, a flag which signals the existence of a couple attempt and the presence of a line number in a line associated table, TIE.

M:COUPLE This procedure allows the user program to couple its associated terminal with another terminal. The procedure call is of the form

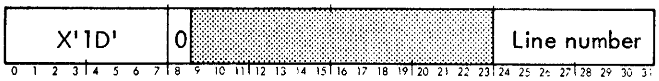
M:COUPLE line

where line specifies the line number of the terminal to which the program's terminal is to be coupled.

Calls generated by the procedure have the form

CAL1,8 fpt

where fpt points to the FPT shown below.



If the operation is successful, CC1 is set to zero and the message

COUPLE FROM line#

is sent to both terminals after the couple is complete. The line number is the number of the issuer's line. If the object of an M:COUPLE is currently in reject mode, the couple does not take place and CC1 is set to 1. CC2 is set to 1 if the line is not on, is undefined, or is an IBM 2741.

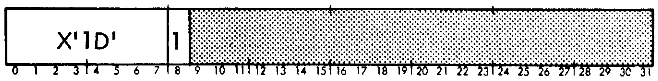
M:DECOUPLE This procedure allows a program to decouple its terminal from a previously established connection. The procedure call is of the form

M:DECOUPLE

Calls generated by the procedure have the form

CAL1,8 fpt

where fpt points to the FPT shown below.

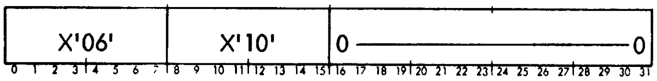


RESET BREAK COUNT

If the BREAK (or equivalent) key is hit 4 times with no intervening characters, a CONTROL Y is simulated. The user can prevent the CONTROL Y by resetting the BREAK count with the following CAL:

CAL1,8 fpt

where fpt points to the FPT shown below.



ON-LINE AND BATCH DIFFERENCES

The CP-V monitor responds differently to certain procedures depending on whether an on-line or a batch program issued the call. These differences are outlined below. (The procedures are discussed in the CP-V/BP Reference Manual, 90 17 64.)

EXIT RETURN (M:EXIT)

Batch: The monitor performs any PMDI dumps that have been specified for the program. It then reads the C device, ignoring everything up to the next control card.

On-line: The monitor returns control to the on-line executive program (TEL) and, after sending a message, sends a prompt (!) character to the terminal. It then awaits additional commands.

ERROR RETURN (M:ERR)

Batch: The monitor lists the message

```
!! JOB id ERRORED BY USER AT xxxx
```

where xxxx is the address of the last instruction executed in the program. The message plus the contents of the current register block and program status doubleword (PSD) are listed on the LL and DO devices. Postmortem dumps are performed and the C device is read; everything up to the next control command is ignored.

On-line: The monitor lists the message

```
A800 YOU ISSUED AN ERROR OR ABORT CAL
```

The monitor then returns control to the on-line executive (TEL), which sends a prompt character (!) to the terminal and awaits commands.

ABORT RETURN (M:XXX)

Batch: The monitor lists the message

```
!! JOB id ABORTED BY USER AT xxxx
```

where xxxx is the address of the last instruction executed. This message plus the contents of the current register block and program status doubleword (PSD) are listed on the LL and DO device.

When a job is aborted, all specified postmortem dumps are performed but no further control commands are honored until a JOB or FIN control command is encountered.

On-line: The monitor lists the message

```
A800 YOU ISSUED AN ERROR OR ABORT CALL
```

This message is listed on the UC device. The monitor then returns control to the on-line executive, which sends a prompt character (!) to the terminal and awaits additional commands.

TYPE A MESSAGE (M:TYPE)

Batch: The monitor lists the specified message on the OC device.

On-line: The monitor lists the specified message on the UC device.

A variant of M:TYPE is M:MESSAGE which unconditionally lists a message on the operator's console (OC device). The format of M:MESSAGE is identical to that of M:TYPE except for the FPT code which is zero.

REQUEST A KEY-IN (M:KEYIN)

Batch: The monitor lists the specified message on the OC device and enables the operator's reply to be returned to the user program. The ECB flag is set to zero when the reply is completed.

On-line: The monitor lists the specified message on the UC device and enables the user's reply to be returned to the user program. If the OC option of the procedure is specified, the message is listed on the OC device and the reply is received from the OC device. The ECB flag is set to zero when the reply is completed.

CONNECT TO INTERRUPT OR BREAK KEY (M:INT)

Batch: The purpose of this procedure is to set the address of a routine to be entered when an interrupt is simulated at the operator's console. When control is given to the INT routine as a result of an unsolicited INT key-in (an operator key-in), the monitor pushes the PSD and general registers into a 19-word block of user's memory (the user's TCB) on a doubleword boundary and places a pointer to word 0 of the PSD in register 1. The TRTN routine may be used to restore control to the user program.

On-line: The purpose of this procedure is to set the address of a routine to be entered when an interrupt is generated at an on-line terminal. When the BREAK key is depressed, the monitor pushes the PSD and general registers into a 19-word block of user's memory (the user's TCB) on a doubleword boundary and places a pointer to word 0 of the PSD in register 1. The TRTN routine may be used to restore control to the user program.

11. COMMUNICATIONS SERVICES TO USER PROGRAMS

INTRODUCTION

Communication services are the functions performed by character-oriented communication (COC) routines for user programs. COC routines control the operation of input/output terminals, such as Teletype and 2741 terminals, that communicate with the computer a character at a time. The functions performed by COC routines include

1. Device handling for Xerox Model 7611 Character-Oriented Communication hardware.

2. Character translation (unless suppressed) to and from internal EBCDIC codes and the external codes of the various types of terminals that may be attached to the system. Terminal types include: Teletype Models 33, 35, 37, and 38; Xerox Model 7015 Teletypewriters; IBM 2741, Tektronix Models 4010 and 4013, Datapoint 3300, and any others compatible with any of the above.

3. Parity generation and detection by character for those terminals requiring it.

4. Division of input character strings into messages as defined by receipt of activation characters (usually carriage return, line feed, form feed, and count complete, but other sets are specially available).

5. Communication with the system scheduler on break, read, read complete, output blocked, output unblocked, and other events that effect swap and execution scheduling.

6. Special interpretation of certain characters for intraline editing and software control functions.

Input and output from COC terminals is stored in four-word blocks, each containing 14 characters plus a halfword link to the next related block. After a read operation is complete, the input message is moved from these buffers directly to the user's buffer area (BUF in M:READ). The actual number of characters received is reported in ARS (actual record size) of the DCB. On a write operation, the user output message (BUF in M:WRITE) is moved to COC buffers to await transmission. Unused COC buffers are held in an available pool. The user program is blocked appropriately when required buffers are not available for output and is restarted when they become available.

WRITE OPERATIONS

Records are written on a COC terminal using the M:WRITE procedure call. The WRITE routine moves the specified number of bytes from the user's buffer to a buffer in the COC routines. The write operation is always a "wait" operation. This means that control is returned to the user program after the character string has been transferred to the COC buffer but before it has been completely transmitted to the terminal. If record keys are specified, they are ignored.

Output in excess of 140 bytes from a single write CAL is ignored. If the specified record size is zero, no action is taken and no characters are transmitted. If more than three trailing blanks occur in an output record, all are suppressed unless DRC is set in the DCB.

If the output contains a NUL character (X'00') the write operation is terminated at that point; i.e., the zero byte and all remaining characters in the record are ignored. NUL characters can be output via a X'B6' character.

Characters are transmitted to the terminal exactly as supplied, with the following exceptions. Certain characters such as FF and SUB are modified (see Table A-4). Whenever either a carriage return or line feed character is detected, the appropriate character pair (carriage return and line feed) is sent to the terminal to return the carrier.

If the write operation is through a DCB other than the M:UC DCB, say the M:LO or M:DO DCB, the COC routines automatically supply carriage return and line feed characters at the end of the character string unless a carriage return, SYNC (which is not transmitted), or line feed were the last characters in the buffer (see VFC in "Device and DCB calls" for special format control). This means that the number of bytes specified in the function parameter table is moved from the user's buffer area to COC buffers and the carriage return and line feed characters are appended in the COC buffers.

If the write is through the M:UC DCB, the carriage return and line feed characters are not automatically supplied. The user may therefore make up single lines through a series of writes (without carriage return characters) or may produce several lines at the terminal with a single write (by inserting several carriage return characters in the buffer).

For all write operations, a count of characters between carriage returns is maintained. This count is compared with the maximum for the physical terminal as specified with the PLATEN command. If the line is too long, additional carriage return and line feed characters are inserted to break the line unless the platen width is less than 12 characters. Line length is retained in the job information table (JIT). It may be altered with the TEL PLATEN command. A count of the lines on a page is also maintained and a page heading line is supplied to the terminal as outlined in the section "Page Control and Page Headings".

READ OPERATIONS

Records are read from a COC terminal using the M:READ procedure call. The READ routine causes the COC routines to accept input characters from the terminal. If a prompt character has been specified, it is sent to the terminal to signal that the COC routines are ready to accept input characters. If characters have been typed ahead, they are echoed after the prompt is issued.

The read operation is always a "wait" operation. This means that the complete input message is transferred to the user's buffer area before control passes to the next instruction. Messages are completed on receipt of

1. The number of characters requested.
2. A carriage return character.
3. A line feed character.
4. A form feed character.
5. The FS, RS, GS, and US codes (L^{cs}, M^{cs}, N^{cs}, and O^{cs} keys).
6. The EOT and SUB codes (D^{c} and Z^{c} keys).
7. The end-of-file convention, ESC F.

The activation character (any item in 2-6 above) is the last character in the buffer. Additional special activation or termination characters are supplied when Delta initiates a read operation. They are

tab

^

,

=

/

The actual number of characters in the message received, including the activation character, is returned in word 4 (ARS) of the DCB. If RUBOUTS were entered while backspace edit mode was active, character positions beyond the activation character may have been modified. However, no more characters than specified in the M:READ functional parameter table are transferred to the user's buffer area. Read requests for zero bytes yields an abnormal code of ID. The response of COC routines to receipt of various end-of-message characters from a terminal is as follows:

Characters	Response
Carriage return or line feed	The appropriate characters are sent to the terminal to ensure a carrier return. However, the actual character received is placed in the buffer.
Form feed	The code FF (EBCDIC 0C) is placed in the buffer, a carriage return and line feed character pair is sent to the terminal, followed by page heading output.
FS,RS,GS,US,EOT	The carrier is not moved. The character is placed in the buffer, and the message is terminated.
Break	An underscore (left arrow on TTYs) is sent to the terminal, the carrier is returned, the message is deleted, and the break entry of the program, if any, is taken.
ESC F	The end-of-file exit from the read CAL is taken. Any characters preceding the ESC F are delivered to the reading program and appended with a carriage return character.

Other characters may act as message terminators if special activation sets are requested; see Change Activation Characters, Chapter 10.

Characters received with parity errors for terminals in the parity checking mode are identified by the SUB code (EBCDIC 1A) which is placed in the buffer. For these characters, a number character, #, is returned to the terminal.

Bad information, such as a character parity error, is reported via the lost-data (07) code to the abnormal CAL exit, if it exists. If no abnormal exit is specified, then the bad information is not reported.

In addition to the line cancel, which may be initialized by the ESC X keys, individual characters may be deleted by the RUBOUT key. A number of characters, n, may be deleted by typing the rubout character n times. If the first character of a line is deleted, the response is as if ESC X were received.

The user program or processor may set up a prompt character to be delivered to the terminal just prior to each read. The prompt character is set by using the M:PC procedure call described in Chapter 9. Any valid EBCDIC character may be specified. A null character (EBCDIC 00) turns off the prompt action. Also, the TEL PROMPT command may be used to set a prompt character.

ERROR AND ABNORMAL CONTROL

Error returns occur in the following cases:

1. Bad DCB address (CAL error return)

2. Bad buffer address (DCB error return)

CAL abnormal returns are taken for

1. Lost data (TYC=2) - parity errors in received message or insufficient COC buffers.
2. Beginning-of-tape (TYC=3) - CAL not read or write, bad line number, or zero byte count.
3. End-of-file (TYC=7) - ESC F character pair received.
4. User controlled read timed out (TYC = X'10').
5. Conditional read issued with no type ahead (TYC = X'11').

BREAK CONTROL

Action on receipt of the break character depends on whether the terminal is reading or not. If reading, the carrier is returned and the message, if any, is deleted. The current read operation is terminated.

Whether reading or writing, control goes to an alternate address associated with the user program, and the user program status doubleword (PSD) and registers, as of the point of interrupt, are placed in the users task control block (TCB) temporary stack. The program may be continued from the point of interrupt by giving a trap return (M:TRTN or CAL1,9 5). The actual alternate address used depends on the user program and associated processors in the following order:

1. If Delta is associated with the program, then control goes to Delta.
2. If the user has issued a M:INT CAL and Delta is not associated with the program the address specified by that CAL is used. A zero or invalid address resets break control.
3. If neither 1 nor 2 apply, then control goes to TEL. A message is typed and TEL issues a request for commands from the terminal.

Break signals are counted by the COC handler. This is done to provide fail-safe operation against program errors in the user break handling routine, to allow special subprocessor action on multiple break signals and to provide compatible operation with future communication equipment that does not have full-duplex lines. If four break signals are received from a terminal without intervening characters, control is given to TEL as if a monitor escape (Y^c) character had been received. The monitor escape simulation may be avoided by using the M:STA procedure.

MONITOR ESCAPE

A terminal may always be put in communication with TEL by input of the Y^c character unless a transparent mode read is pending. Input and output is canceled (characters for a left-facing arrow, a carriage return, and a line feed are sent and the carrier is returned) if the terminal is in read status. If the user program is restarted (via the CONTINUE or GO command) from the point of escape and the terminal was previously reading, the read is reissued.

SET AND DEVICE DCB CALs

The M:SETDCB CAL may be used to set abnormal and error addresses in a DCB associated with a terminal. Error codes and other information communicated to the user program is as specified in Appendix B. If no error address is specified in the DCB, control is transferred to TEL and a message is sent to the terminal.

Only certain M:DEVICE CALs are acknowledged by the COC routines. These CALs are listed in Table 47. All other CALs that set parameters in a DCB associated with a COC terminal are ignored without comment. In general, any CAL may be used and will result in the specified modification to the DCB but only the parameters listed in Table 47 are used by COC routines.

PAGE CONTROL AND HEADINGS

COC routines count the lines transmitted to and from a terminal. Whenever a read or write operation is initiated, this line count is compared with the limit for the terminal. If the maximum has been exceeded, a new page heading is produced. (The maximum may be exceeded by several lines if several input lines have been canceled via the X^c keys at the bottom of the page before the next read or write call is issued. If this occurs an appropriate adjustment is made in the heading.)

Page headings are also produced whenever an M:DEVICE call specifying PAGE is issued by a user program or the characters "FF" (L^c) are entered into the terminal. This case is similar to page overflow in that heading information is not produced until the associated user program or processor issues its next read or write call.

Two kinds of page headings are produced:

1. The standard page heading (or CRT pagination as described under "CAL Control of Terminal Mode").
2. A user heading as specified by HEADER and COUNT in a device call.

Heading information is taken from the DCB associated with the read or write call. Thus, if write calls are issued through several DCBs, the heading printed will depend on the DCB associated with the call that produced the page overflow.

The standard page heading includes current time, date, user account number, scheduler's job identification and line number, page number, and possibly an administrative message. The heading is typed on the top line of the form just under the fold (if any). The heading information is preceded by six blank lines (fewer if excess lines were printed

Table 47. M:DEVICE Parameters Acknowledged by COC Routines

Parameter Set by M:DEVICE CAL	COC Action
PAGE	Page heading is typed on the terminal (see "Page Control and Page Heading").
LINES	Number of printable lines per page is set.
NLINES	The current number of lines on the terminal page is contained in the JIT (byte JB:LC).
SIZE	Record size (in bytes) used by read and write CALs for which no size is specified. If record size is not specified in either the CAL FPT or the DCB, no characters are transmitted and return is immediate.
SPACE	Number of indicated spaces minus one are inserted before each write if VFC is not on and SPACE is set. Counts 0 and 1 result in single spacing (no spaces are inserted before each write).
VFC	COC routines simulate the printer's vertical format control as specified in the first character of the text line if VFC is set. The simulation is limited to the following cases: Hex Code — Action C1 - CF — COC routines insert 1-15 lines before the print line. (Page check on each insert.) D1 — COC routines skip to top of page, print the heading information, and output the print line without a following upspace. E1 - EF — COC routines insert 1-15 blank lines and output the print line without a following upspace. F1 — COC routines skip to top of page and print the heading information followed by the print line. 60, E0 — COC routines insert a carriage return character after the print line. A line feed character is not inserted after the print line. In all cases except the latter, the print line is followed by a carriage return and line feed characters and a check for page overflow.
DRC/NORDC	Used to inhibit automatic page heading and the stripping of trailing blanks on writes if the mode is BCD. Used to control transparent mode if BIN is specified. (See "Transparent Mode" section.)
COUNT	See Page Control and Page Headings section.
HEADER	See Page Control and Page Headings section.
TABS	See TABS section.

on the preceding page). It is followed by five blank lines. With 54 printed lines to a page, this spacing produces 11-inch pages with one-inch margins at top and bottom. The standard heading line may be omitted, if desired, by setting DRC in the DCB or by setting the page length less than 11 lines.

Example:

```
12:01  12/12/69 ACCT  1A-03[36] Administrative Message

  1        2       3     4  5               6
```

1. Time the page heading was issued (24-hour clock).
2. Current date.
3. Log-on account.
4. Scheduler's job identification (ID) and line number of COC line.
5. Page number, enclosed in brackets, centered for a platen 72 characters wide.
6. Administrative message (limited to 64-characters) supplied to all terminals by system operator via this mechanism.

User headings, which are specified in the DCB of the read or write call, are provided following the automatic heading. The position, text, and page numbers of these headings are as specified in the CP-V/BP Reference Manual, 90 17 64. The page count in this heading is that carried in the DCB and and is reset with each COUNT device call while page count for the standard heading is carried in the JIT and is never reset.

TAB SIMULATION

TAB stops that are set in output DCBs by a device call specifying TAB, by a SET command, or by the TEL command TABS, cause spaces to be sent to the terminal. These spaces bring the current position of the carrier to that indicated by the next higher tab stop in the DCB. The platen width test is still in effect and the carrier is returned if the count-on-line exceeds the platen width. If tab simulation is not in effect, the tab character is sent directly to the terminal. If tab simulation is on but no tab stops are set, one space is sent for each tab character.

Tabs received in the input stream are handled similarly, except that a tab is always echoed by at least one space (if echoplexing is on).

Three things are necessary for tab simulation to take effect:

1. Tab simulation must be on (ESC and T control).
2. Tab stops must be set in the M:UC DCB or the DCB controlling read or write.
3. Tab characters must be sent or received.

Simulation of tab stops is turned off and on by the user via the character pair ESC T. These characters are not transmitted to the reading program and each pair switches tab simulation flag from on to off or vice versa. When the flag is on and a tab character (ANSCII 09) is received, enough blanks are sent to the terminal to move the carrier to the next higher tab position. When reading, the tab character is replaced by one or more spaces, as appropriate, in the input buffer if space-insertion mode is on; if off, the tab character is placed in the input buffer for the reading program. Space-insertion mode is toggled by the ESC S character pair. Carriage returns are not inserted to split extra long input lines created this way.

When in effect, the tab stops used for simulation are obtained in the following order:

Output

1. If tab stops are set in the calling DCB, they are used.
2. If tab stops are set in M:UC DCB, they are used.
3. Tabs are replaced with a single space.

Input

1. If tab stops are set in the M:UC DCB, they are used.
2. A single space is echoed for each tab.

In all cases in which tabs are set but the current carrier position is beyond any tab stop that is set, the tab is replaced with a single space.

TRANSPARENT MODE

The transparent mode for input or output is controlled by setting the DRC and BIN mode flags in the DCB. If DRC and BIN are set, the transparent mode is in effect. This will cause all input and output through that DCB to be passed literally (i.e., no translation or interpretation will be done). The transparent mode may be escaped from by depressing BREAK. This mode of operation is not allowed for 2741 terminals.

APPENDIX A. XEROX STANDARD SYMBOLS, CODES AND CORRESPONDENCES

XEROX STANDARD SYMBOLS AND CODES

The symbols listed here include two types: graphic symbols and control characters. Graphic symbols are displayable and printable; control characters are not. Hybrids are SP (the symbol for a blank space), and DEL (the delete code) which is not considered a control command.

Two types of code are also shown: (1)the 8-bit Xerox Standard Computer Code, i.e., the Xerox Extended Binary-Coded-Interchange Code (EBCDIC); and (2) the 7-bit American National Standard Code for information Interchange (ANSCII), i.e., the Xerox Standard Communication Code.

XEROX STANDARD CHARACTER SETS

1. EBCDIC

   57-character set: uppercase letters, numerals, space, and & - / . < > ( ) + | $ * : ; , % # @ ' =

   63-character set: same as above plus ¢ ! _ ? " ¬

   89-character set: same as 63-character set plus lowercase letters

2. ANSCII

   64-character set: uppercase letters, numerals, space, and ! " $ % & ' ( ) * + , - . / \ ; : = < > ? @ _ [ ] ^ # | ¬

   95-character set: same as above plus lowercase letters and { } ¦ ~ `

CONTROL CODES

In addition to the standard character sets listed above, the Xerox symbol repertoire includes 37 control codes and the hybrid code DEL (hybrid code SP is considered part of all character sets). These are listed in the table titled CP-V Symbol-Code Correspondences.

SPECIAL CODE PROPERTIES

The following two properties of all Xerox standard codes will be retained for future standard code extensions:

1. All control codes, and only the control codes, have their two high-order bits equal to "00". DEL is not considered a control code.
2. No two graphic EBCDIC codes have their seven low-order bits equal.

Table A-1. CP-V 8-Bit Computer Codes (EBCDIC)

			Most Significant Digits															
	Hexadecimal		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
		Binary	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Least Significant Digits	0	0000	NUL	DLE	LF only	ESC F	SP	&	-	∧				FF	SP		-	0
	1	0001	SOH	X-ON	FS	CAN	[7]ESC J		/	··	a	j		\ [1]	A	J		1
	2	0010	STX	DC2	GS	ESC X	⊥	ESC D	⌈	¯	b	k	s	{ [1]	B	K	S	2
	3	0011	ETX	X-OFF	RS	ESC P	ESC LF	⎕			c	l	t	} [1]	C	L	T	3
	4	0100	EOT	DC4	US	ESC U	⌊	ESC Z	↓	≤	d	m	u	[[1]	D	M	U	4
	5	0101	HT	LF NL	EM	ESC (	∊	⊤			e	n	v	] [1]	E	N	V	5
	6	0110	ACK	SYN [8]	/	ESC)		∘	⍵	≥	f	o	w	NUL	F	O	W	6
	7	0111	BEL	ETB	∧	ESC T			⊃		g	p	x		G	P	X	7
	8	1000	EOM BS	CAN	=	ESC S	Δ				h	q	y		H	Q	Y	8
	9	1001	ENQ	EM	CR only	ESC E	⍳			∨	i	r	z		I	R	Z	9
	A	1010	NAK	SUB	EOT	ESC C	¢ [2]	!	^ [1]	:					[7]			×
	B	1011	VT	ESC	BS	ESC O	.	$	,	#								÷
	C	1100	FF	FS	)	X-ON	<	*	%	@				[[6]				→
	D	1101	CR	GS	HT	X-OFF	(	)	_	'				] [6]				←
	E	1110	SO	RS	LF only	ESC R	+	;	>	=				Lost [6] Data				
	F	1111	SI	US	SUB	ESC CR	| [2]	¬ [2]	?	"			[6] |	¬ [6]				DEL
			3		6			4,7				5						

Notes:

1 The characters ^ \ { } [ ] are ANSCII characters that do not appear in any of the Xerox EBCDIC-based character sets, though they are shown in the EBCDIC table.

2 The characters ¢ | ¬ appear in the Xerox 63- and 89-character EBCDIC sets but not in either of the Xerox ANSCII-based sets. However, Xerox software translates the characters ¢ | ¬ into ANSCII characters as follows:

EBCDIC	= ANSCII
¢	` (6-0)
|	¦ (7-12)
¬	~ (7-14)

3 The EBCDIC control codes in columns 0 and 1 and their binary representation are exactly the same as those in the ANSCII table, except for two interchanges: LF/NL with NAK, and HT with ENQ.

4 Characters enclosed in heavy lines are included only in the Xerox standard 63- and 89-character EBCDIC sets.

5 These characters are included only in the Xerox standard 89-character EBCDIC set.

6 The EBCDIC codes in column 3 are used by COC to perform special functions. The EBCDIC codes in column 2 and positions AF and BC through BF are used by COC for output only.

7 APL characters (and some ESC sequences) are assigned EBCDIC values that fall within the shaded area of the CP-V code set. These assignments are for APL internal use and are only reflected in 2741-APL translation tables.

8 Placing a SYN code as the last position of a nontransparent message will prevent the transmission of the SYN and the normal message appendage of the CR/LF pair. This allows a user to continue writing more than one message on the same line without affecting the carrier position. The EBCDIC SYN code is translated to an idle (IL) on output to 2741 terminals.

Table A-2. CP-V 7-Bit Communication Codes (ANSCII)

			Most Significant Digits							
	Decimal (rows)	(col's.)→	0	1	2	3	4	5	6	7
	↓	Binary	x000	x001	x010	x011	x100	x101	x110	x111
Least Significant Digits	0	0000	NUL	DLE	SP	0	@	P	`	p
	1	0001	SOH	DC1	! [5]	1	A	Q	a	q
	2	0010	STX	DC2	"	2	B	R	b	r
	3	0011	ETX	DC3	#	3	C	S	c	s
	4	0100	EOT	DC4	$	4	D	T	d	t
	5	0101	ENQ	NAK	%	5	E	U	e	u
	6	0110	ACK	SYN	&	6	F	V	f	v
	7	0111	BEL	ETB	'	7	G	W	g	w
	8	1000	BS	CAN	(	8	H	X	h	x
	9	1001	HT	EM	)	9	I	Y	i	y
	10	1010	LF NL	SUB	*	:	J	Z	j	z
	11	1011	VT	ESC	+	;	K	[4] [[5]	k	{
	12	1100	FF	FS	,	<	L	\	l	¦
	13	1101	CR	GS	-	=	M	[4]] [5]	m	} [4]
	14	1101	SO	RS	.	>	N	[4] ^ [5]	n	~ [4]
	15	1111	SI	US	/	?	O	_ [4]	o	DEL
			2		3					

Notes:

1 Most significant bit, added for 8-bit format, is either 0 or an even-parity bit for the remaining 7 bits.

2 Columns 0-1 are control codes.

3 Columns 2-5 correspond to the Xerox 64-character ANSCII set.
Columns 2-7 correspond to the Xerox 95-character ANSCII set.

4 On many current teletypes, the symbol

^ is ↑ (5-14)
_ is ← (5-15)
~ is ESC or ALTMODE control (7-14)
} is ESC or ALTMODE control (7-13)

and none of the symbols appearing in columns 6-7 are provided. Except for the four symbol differences noted above, therefore, such teletypes provide all the characters in the Xerox 64-character ANSCII set. (The Xerox 7015 Remote Keyboard Printer provides the 64-character ANSCII set also, but prints ^ as Λ. It also interprets the [ ] characters as | ¬ .)

5 On the Xerox 7670 Remote Batch Terminal, the symbol

! is | (2-1)] is ! (5-13)
[is ¢ (5-11) ^ is ¬ (5-14)

and none of the symbols appearing in columns 6-7 are provided. Except for the four symbol differences noted above, therefore, this terminal provides all the characters in the Xerox 64-character ANSCII set.

Table A-3. CP-V Symbol-Code Correspondences

EBCDIC[t]						
Hex.	Dec.	Symbol	Card Code	ANSCII[tt]	Meaning	Remarks
00	0	NUL	12-0-9-8-1	0-0	null	00 through 1F are control codes.
01	1	SOH	12-9-1	0-1	start of header	On 2741 terminals, SOH is PRE.
02	2	STX	12-9-2	0-2	start of text	On 2741 terminals, STX is BY.
03	3	ETX	12-9-3	0-3	end of text	On 2741 terminals, ETX is RES.
04	4	EOT	12-9-4	0-4	end of transmission	
05	5	HT	12-9-5	0-9	horizontal tab	00, 06, 07, 09-0B, and 0E-0F
06	6	ACK	12-9-6	0-6	acknowledge (positive)	are idles for 2741 terminals.
07	7	BEL	12-9-7	0-7	bell	
08	8	BS or EOM	12-9-8	0-8	backspace or end of message	EOM is used only on Xerox Keyboard/
09	9	ENQ	12-9-8-1	0-5	enquiry	Printers Models 7012, 7020, 8091,
0A	10	NAK	12-9-8-2	1-5	negative acknowledge	and 8092.
0B	11	VT	12-9-8-3	0-11	vertical tab	
0C	12	FF	12-9-8-4	0-12	form feed	
0D	13	CR	12-9-8-5	0-13	carriage return	CR outputs CR and LF.
0E	14	SO	12-9-8-6	0-14	shift out	
0F	15	SI	12-9-8-7	0-15	shift in	
10	16	DLE	12-11-9-8-1	1-0	data link escape	
11	17	DC1	11-9-1	1-1	device control 1	On Teletype terminals, DC1 is X-ON.
12	18	DC2	11-9-2	1-2	device control 2	On 2741 terminals, DC2 is PN.
13	19	DC3	11-9-3	1-3	device control 3	DC3 is RS on 2741s and X-OFF on
14	20	DC4	11-9-4	1-4	device control 4	Teletypes.
15	21	LF or NL	11-9-5	0-10	line feed or new line	On 2741 terminals, DC4 is PF.
16	22	SYN	11-9-6	1-6	sync	LF outputs CR and LF.
17	23	ETB	11-9-7	1-7	end of transmission block	On 2741 terminals, ETB is EOB.
18	24	CAN	11-9-8	1-8	cancel	
19	25	EM	11-9-8-1	1-9	end of medium	
1A	26	SUB	11-9-8-2	1-10	substitute	Replaces characters with parity error.
1B	27	ESC	11-9-8-3	1-11	escape	
1C	28	FS	11-9-8-4	1-12	file separator	
1D	29	GS	11-9-8-5	1-13	group separator	10, 11, 16, 18, 19, and 1B-1E are
1E	30	RS	11-9-8-6	1-14	record separator	idles for 2741 terminals.
1F	31	US	11-9-8-7	1-15	unit separator	
20	32	LF only	11-0-9-8-1	1-5	line feed only	20 through 2F are used by COC for
21	33	FS	0-9-1	1-12		output only. These codes are
22	34	GS	0-9-2	1-13		duplicates of the label entries
23	35	RS	0-9-3	1-14		that caused activation. The
24	36	US	0-9-4	1-15		20-2F entries output a single code
25	37	EM	0-9-5	1-9		only and are not affected by any
26	38	/	0-9-6	2-15		special COC functional processing.
27	39	↑	0-9-7	5-14		
28	40	=	0-9-8	3-13		
29	41	CR only	0-9-8-1	0-13	carriage return only	
2A	42	EOT	0-9-8-2	0-4		
2B	43	BS	0-9-8-3	0-8		
2C	44	)	0-9-8-4	2-9		
2D	45	HT	0-9-8-5	0-9	tab code only	
2E	46	LF only	0-9-8-6	1-5	line feed only	
2F	47	SUB	0-9-8-7	1-10		
30	48	ESC F	12-11-0-9-8-1		end of file	30 through 3F cause COC to perform
31	49	CANCEL	9-1		delete all input and output	special functions.
32	50	ESC X	9-2		delete input line	
33	51	ESC P	9-3		toggle half-duplex paper tape mode	
34	52	ESC U	9-4		toggle restrict upper case	
35	53	ESC (	9-5		upper case shift	
36	54	ESC)	9-6		lower case shift	
37	55	ESC T	9-7		toggle tab simulation mode	
38	56	ESC S	9-8		toggle space insertion mode	
39	57	ESC E	9-8-1		toggle echo mode	
3A	58	ESC C	9-8-2		toggle tab relative mode	
3B	59	ESC O	9-8-3		toggle backspace edit mode	
3C	60	X-ON	9-8-4		start paper tape	
3D	61	X-OFF	9-8-5		stop paper tape	
3E	62	ESC R	9-8-6		retype	
3F	63	ESC CR	9-8-7		line continuation	

[t] Hexadecimal and decimal notation.

[tt] Decimal notation (column-row).

Table A-3. CP-V Symbol-Code Correspondences (cont.)

EBCDIC†						
Hex.	Dec.	Symbol	Card Code	ANSCII††	Meaning	Remarks
40	64	SP	blank	2-0	blank	
41	65	ESC J	12-0-9-1		toggle insert mode	46 and 47 are unassigned.
42	66	⊥	12-0-9-2		decode	
43	67	ESC LF	12-0-9-3		line continuation	
44	68	⌊	12-0-9-4		minimum	42, 44, 45, 48, and 49 are APL characters
45	69	ϵ	12-0-9-5		epsilon	
46	70		12-0-9-6			
47	71		12-0-9-7			
48	72	Δ	12-0-9-8		delta	
49	73	⍳	12-8-1		index	
4A	74	¢ or ‘	12-8-2	6-0	cent or accent grave	Accent grave used for left single quote. On Model 7670, ‘ not available, and ¢ = ANSCII 5-11.
4B	75	.	12-8-3	2-14	period	
4C	76	<	12-8-4	3-12	less than	
4D	77	(	12-8-5	2-8	left parenthesis	On 2741 APL, ¢ is ⊂ (subset).
4E	78	+	12-8-6	2-11	plus	
4F	79	| or ¦	12-8-7	7-12	vertical bar ,or broken bar	On Model 7670, ¦ not available, and | = ANSCII 2-1.
50	80	&	12	2-6	ampersand	On 2741 APL, & is ∩ (intersection). 51, 57, 58, and 59 are unassigned.
51	81		12-11-9-1			
52	82	ESC D	12-11-9-2		request re-read	53, 55, and 56 are APL characters.
53	83	⎕	12-11-9-3		quad	
54	84	ESC Z	12-11-9-4		toggle input ignore mode	
55	85	⊤	12-11-9-5		encode	
56	86	○	12-11-9-6		circular	
57	87		12-11-9-7			
58	88		12-11-9-8			
59	89		11-8-1			
5A	90	!	11-8-2	2-1	exclamation point	On Model 7670, ! is |. On 2741 APL, ! is ° (degree). On 2741 APL, $ is U (union).
5B	91	$	11-8-3	2-4	dollars	
5C	92	*	11-8-4	2-10	asterisk	
5D	93	)	11-8-5	2-9	right parenthesis	
5E	94	;	11-8-6	3-11	semicolon	
5F	95	~ or ¬	11-8-7	7-14	tilde or logical not	On Model 7670, ~ is not available, and ¬ = ANSCII 5-14.
60	96	-	11	2-13	minus, dash, hyphen	
61	97	/	0-1	2-15	slash	
62	98	⌈	11-0-9-2		maximum	62, 64, 66, and 67 are APL characters.
63	99		11-0-9-3			
64	100	↓	11-0-9-4		down arrow	
65	101		11-0-9-5			
66	102	ω	11-0-9-6		omega	63, 65, 68, and 69 are unassigned.
67	103	⊃	11-0-9-7		superset	
68	104		11-0-9-8			
69	105		0-8-1			
6A	106	^	12-11	5-14	circumflex	On Model 7670 ^ is ¬. On Model 7015 ^ is ∧ (caret). On 2741 APL, ^ is ↑. On 2741 APL, % is ρ.
6B	107	,	0-8-3	2-12	comma	
6C	108	%	0-8-4	2-5	percent	
6D	109	_	0-8-5	5-15	underline	Underline is sometimes called "break character"; may be printed along bottom of character line.
6E	110	>	0-8-6	3-14	greater than	
6F	111	?	0-8-7	3-15	question mark	
70	112	∧	12-11-0		APL	70-72, 74, 76, and 79 are APL characters.
71	113	¨	12-11-0-9-1		APL quote mark	
72	114	¯	12-11-0-9-2		overscore	
73	115		12-11-0-9-3			
74	116	≤	12-11-0-9-4		less than or equal	73, 75, 77, and 78 are unassigned.
75	117		12-11-0-9-5			
76	118	≥	12-11-0-9-6		greater than or equal	
77	119		12-11-0-9-7			
78	120		12-11-0-9-8			
79	121	∨	8-1		down delta	
7A	122	:	8-2	3-10	colon	
7B	123	#	8-3	2-3	number	
7C	124	@	8-4	4-0	at	
7D	125	'	8-5	2-7	apostrophe (right single quote)	
7E	126	=	8-6	3-13	equals	
7F	127	"	8-7	2-2	quotation mark	

† Hexadecimal and decimal notation.

†† Decimal notation (column-row).

Table A-3. CP-V Symbol-Code Correspondences (cont.)

EBCDIC[t]		Symbol	Card Code	ANSCII[tt]	Meaning	Remarks
Hex.	Dec.					
80	128		12-0-8-1			80 is unassigned.
81	129	a	12-0-1	6-1		81-89, 91-99, A2-A9 comprise the lowercase alphabet. Available only in Xerox standard 89- and 95-character sets.
82	130	b	12-0-2	6-2		
83	131	c	12-0-3	6-3		
84	132	d	12-0-4	6-4		
85	133	e	12-0-5	6-5		
86	134	f	12-0-6	6-6		
87	135	g	12-0-7	6-7		
88	136	h	12-0-8	6-8		
89	137	i	12-0-9	6-9		
8A	138		12-0-8-2			8A through 90 are unassigned.
8B	139		12-0-8-3			
8C	140		12-0-8-4			
8D	141		12-0-8-5			
8E	142		12-0-8-6			
8F	143		12-0-8-7			
90	144		12-11-8-1			
91	145	j	12-11-1	6-10		
92	146	k	12-11-2	6-11		
93	147	l	12-11-3	6-12		
94	148	m	12-11-4	6-13		
95	149	n	12-11-5	6-14		
96	150	o	12-11-6	6-15		
97	151	p	12-11-7	7-0		
98	152	q	12-11-8	7-1		
99	153	r	12-11-9	7-2		
9A	154		12-11-8-2			9A through A1 are unassigned.
9B	155		12-11-8-3			
9C	156		12-11-8-4			
9D	157		12-11-8-5			
9E	158		12-11-8-6			
9F	159		12-11-8-7			
A0	160		11-0-8-1			
A1	161		11-0-1			
A2	162	s	11-0-2	7-3		
A3	163	t	11-0-3	7-4		
A4	164	u	11-0-4	7-5		
A5	165	v	11-0-5	7-6		
A6	166	w	11-0-6	7-7		
A7	167	x	11-0-7	7-8		
A8	168	y	11-0-8	7-9		
A9	169	z	11-0-9	7-10		
AA	170		11-0-8-2			AA through AE are unassigned.
AB	171		11-0-8-3			
AC	172		11-0-8-4			
AD	173		11-0-8-5			
AE	174		11-0-8-6			
AF	175	|	11-0-8-7		logical and	AF is used by COC for output of an ANSCII 7-12 code only.
B0	176	FF	12-11-0-8-1	0-12	form feed	
B1	177	\	12-11-0-1	5-12	backslash	
B2	178	{	12-11-0-2	7-11	left brace	On 2741 terminals, { is output as (.
B3	179	}	12-11-0-3	7-13	right brace	On 2741 terminals, } is output as).
B4	180	[	12-11-0-4	5-11	left bracket	On Model 7670, [is ¢. On Model 7015, [is I.
B5	181	]	12-11-0-5	5-13	right bracket	
B6	182	NUL	12-11-0-6	0-0	null	On Model 7670,] is !. On Model 7015,] is¬.
B7	183		12-11-0-7			
B8	184		12-11-0-8			B0 and B7through BB are unassigned.
B9	185		12-11-0-9			
BA	186		12-11-0-8-2			
BB	187		12-11-0-8-3			
BC	188	[	12-11-0-8-4		left bracket	BC, BD, and BF are used by COC for output of ANSCII 5-11, 5-13, and 7-14, respectively.
BD	189	]	12-11-0-8-5		right bracket	
BE	190	lost data	12-11-0-8-5		lost data	
BF	191	¬	12-11-0-8-7		logical not	On 2741 Selectric and EBCD Standard Keyboards, [is output as (and] is output as).

[t] Hexadecimal and decimal notation.

[tt] Decimal notation (column-row).

Table A-3. CP-V Symbol-Code Correspondences (cont.)

EBCDIC†						
Hex.	Dec.	Symbol	Card Code	ANSCII††	Meaning	Remarks
C0	192	SP	12-0	2-0	blank	Output only.
C1	193	A	12-1	4-1		C1-C9, D1-D9, E2-E9 comprise the uppercase alphabet.
C2	194	B	12-2	4-2		
C3	195	C	12-3	4-3		
C4	196	D	12-4	4-4		
C5	197	E	12-5	4-5		
C6	198	F	12-6	4-6		
C7	199	G	12-7	4-7		
C8	200	H	12-8	4-8		
C9	201	I	12-9	4-9		
CA	202		12-0-9-8-2			CA through CF are unassigned.
CB	203		12-0-9-8-3			
CC	204		12-0-9-8-4			
CD	205		12-0-9-8-5			
CE	206		12-0-9-8-6			
CF	207		12-0-9-8-7			
D0	208		11-0			D0 is unassigned.
D1	209	J	11-1	4-10		
D2	210	K	11-2	4-11		
D3	211	L	11-3	4-12		
D4	212	M	11-4	4-13		
D5	213	N	11-5	4-14		
D6	214	O	11-6	4-15		
D7	215	P	11-7	5-0		
D8	216	Q	11-8	5-1		
D9	217	R	11-9	5-2		
DA	218		12-11-9-8-2			DA through DF are unassigned.
DB	219		12-11-9-8-3			
DC	220		12-11-9-8-4			
DD	221		12-11-9-8-5			
DE	222		12-11-9-8-6			
DF	223		12-11-9-8-7			
E0	224	–	0-8-2	2-13	minus	Output only. E1 is unassigned.
E1	225		11-0-9-1			
E2	226	S	0-2	5-3		
E3	227	T	0-3	5-4		
E4	228	U	0-4	5-5		
E5	229	V	0-5	5-6		
E6	230	W	0-6	5-7		
E7	231	X	0-7	5-8		
E8	232	Y	0-8	5-9		
E9	233	Z	0-9	5-10		
EA	234		11-0-9-8-2			EA through EF are unassigned.
EB	235		11-0-9-8-3			
EC	236		11-0-9-8-4			
ED	237		11-0-9-8-5			
EE	238		11-0-9-8-6			
EF	239		11-0-9-8-7			
F0	240	0	0	3-0		
F1	241	1	1	3-1		
F2	242	2	2	3-2		
F3	243	3	3	3-3		
F4	244	4	4	3-4		
F5	245	5	5	3-5		
F6	246	6	6	3-6		
F7	247	7	7	3-7		
F8	248	8	8	3-8		
F9	249	9	9	3-9		
FA	250	X	12-11-0-9-8-2		multiply	FA through FF are APL characters
FB	251	÷	12-11-0-9-8-3		divide	
FC	252	→	12-11-0-9-8-4		right arrow	
FD	253	←	12-11-0-9-8-5		left arrow	
FE	254		12-11-0-9-8-6			FE is not assigned.
FF	255	DEL	12-11-0-9-8-7		delete	Special – neither graphic nor control symbol.

†Hexadecimal and decimal notation.

††Decimal notation (column-row).

Table A-4. ANSCII Control-Character Translation Table

Input					Output	
ANSCII	TTY Key	Echoed	Prog. Receives (EBCDIC)	Process	EBCDIC	Transmitted (ANSCII)
NUL (00)	p^{cs}	None	None	None	NUL (00)	Nothing (end of output message)
SOH (01)[†]	A^{c}	SOH	SOH	None	SOH (01)	SOH
STX (02)[†]	B^{c}	STX	STX	None	STX (02)	STX
EXT (03)[†]	C^{c}	ETX	ETX	None	ETX (03)	ETX
EOT (04)[†]	D^{c}	EOT	EOT	Input Complete.	EOT (04)	EOT
ENQ (05)[†]	E^{c}	ENQ	ENQ (09)	None	HT (05)	Space(s) if tab simulation on, or HT (09) if not.
ACK (06)[†]	F^{c}	ACK	ACK	None	ACK (06)	ACK
BEL (07)	G^{c}	BEL	BEL	None	BEL (07)	BEL
BS (08)	H^{c}	BS	BS	None	BS (08)	BS
HT (09)	I^{c}	Space to tab stop if tab simulation on, or 1 space if not.	Spaces to tab stop, or one space, or tab (05) depending on space insertion mode.	None	ENQ (09)	ENQ (05)
LF/NL (0A)	NL	CR and LF	LF (15)	Input Complete.	NAK (0A)	NAK (15)
VT (0B)	K^{c}	VT	VT	None	VT (0B)	VT
FF (0C)	L^{c}	None	FF	Page Header and Input Complete.	FF (0C)	Page Header
CR (0D)	CR	CR and LF	CR (0D)	Input Complete.	CR (0D)	CR and LF (0A)
SO (0E)	N^{c}	SO	SO	None	SO (0E)	SO
SI (0F)	O^{c}	SI	SI	None	SI (0F)	SI
DLE (10)[†]	P^{c}	DLE	DLE	None	DLE (10)	DLE
DC1 (11)	Q^{c}	DC1	None	Paper Tape On.	DC1 (11)	DC1
DC2 (12)	R^{c}	DC2	DC2	None	DC2 (12)	DC2
DC3 (13)	S^{c}	DC3	None	Paper Tape Off.	DC3 (13)	DC3
DC4 (14)[†]	T^{c}	DC4	DC4	None	DC4 (14)	DC4
NAK (15)[†]	U^{c}	NAK	NAK (0A)	None	LF/NL (15)	CR and LF (0A)

[†] These characters are communication control characters reserved for use by hardware. Any other use of them risks incompatibility with future hardware developments and is done so by the user at his own risk.

Table A-4. ANSCII Control-Character Translation Table (cont.)

Input					Output	
ANSCII	TTY Key	Echoed	Prog. Receives (EBCDIC)	Process	EBCDIC	Transmitted (ANSCII)
SYN (16)[†]	V^{c}	SYN	SYN	None	SYN[†] (16)	SYN (not transmitted if last character in user's buffer).
ETB (17)[†]	W^{c}	ETB	ETB	None	ETB (17)	ETB
CAN (18)	X^{c}	Back-arrow and CR/LF	None	Cancel input or output message.	CAN (18)	CAN
EM (19)	Y^{c}	Back-arrow and CR/LF	None	Monitor Escape/ Control to TEL	EM (19)	EM
SUB (1A)	Z^{c}	SUB	SUB	Input Complete	SUB (1A)	# (A3)
ESC (1B)	K^{cs} ESC PREFIX	None	None	Initiate escape sequence mode.	ESC (1B)	ESC
FS (1C)	L^{cs}	FS	FS	Input Complete	FS (1C)	FS
GS (1D)	M^{cs}	GS	GS	Input Complete	GS (1D)	GS
RS (1E)	N^{cs}	RS	RS	Input Complete	RS (1E)	RS
US (1F)	O^{cs}	US	US	Input Complete	US (1F)	US
} (7D)	ALT-MODE	} or None	} or None	} if model 37; as ESC if model 33, 35, or 7015.	} (B3)	} (7D)
~(7E)	ESC (7015)	~or None	~or None	~if model 37; as ESC if model 33, 35, or 7015	¬(5F)	~(7E)
DEL (7F)	Rubout	\	None	Rubout last character.	DEL (FF)	None
All ANSCII upper and lower case alphabetics are translated on input into the corresponding EBCDIC graphics as shown in Tables C-1 and C-2. All special graphics map as shown, allowing for Table C-1, Note 2, and the exceptions above for model 33 and 35. Lower case alphabetics map into corresponding EBCDIC upper case if the ESC U mode is set. Upper case alphabetics map into corresponding EBCDIC lower case if ESC) is set.					Alphabetic and symbol output translation is also as shown in Tables C-1 and C-2; for Models 33 and 35, and 7015 terminals, however, lowercase alphabetics are automatically translated to upper case.	

[†] These characters are communication control characters reserved for use by hardware. Any other use of them risks incompatibility with future hardware developments and is done so by the user at his own risk.

Table A-5. Substitutions for Nonexistent Characters on 2741 Keyboards

EBCDIC Character	APL Keyboard	Selectric Keyboard	EBCD Keyboard
>	>	, (upper case)	>
<	<	. (upper case)	<
^	↑	¢	¢
|	|	° (degree)	|
¬	~	±	¬
#	≠	#	#
%	ρ	%	%
¢	⊂	¢	¢
@	α	@	@
"	∇	"	"
!	∘	!	!
&	∩	&	&
$	∪	$	$

APPENDIX B. MONITOR ERROR MESSAGES

INTRODUCTION

Four groups of monitor error codes are defined in this section. They are I/O error and abnormal codes (Tables B-1 through B-4), other monitor codes (Table B-5), and Enqueue/ Dequeue abnormal and error codes (Table B-6 and B-7). In all cases, a message is printed only if the monitor has control. If the user asks for control, the error codes are returned to him. Otherwise, the monitor takes unilateral action and prints the message corresponding to the code or the code itself if no message is in the ERRMSG file. Users who have taken control may return it for monitor disposition by using M:MERC.

The error and abnormal addresses specified in a function parameter table (FPT) for a Read, Check, or Write function are temporary and are not retained by the monitor between calls. Those addresses specified in an FPT for an Open function are retained in the specified data control block (DCB).

I/O error and abnormal conditions fall into two general categories:

1. Those associated with insufficient or conflicting information.

2. Those associated with device failures or end-of-data conditions.

The monitor responds to conditions of the first category by honoring the error and abnormal addresses in the associated DCB. The monitor responds to conditions of the second category by honoring the error and abnormal addresses in the FPT for the associated Read, Check, or Write functions.

The error and abnormal codes for insufficient or conflicting information are listed in Tables B-1 and B-3. Those for device failure or end-of-data are listed in Tables B-2 and B-4.

The monitor communicates the error or abnormal code and the DCB address in SR3, and the address following the instruction which caused the CAL1 trap is in SR1. The code is contained in byte 0 of the word in SR3, a subcode is contained in bits 8-14, and the DCB address is contained in the rightmost 17 bits.

SR3

Error Code	Subcode	DCB Address
0 1 2 3 4 5 6 7	8 9 10 11 12 13 14	15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Note that the subcode field contains seven bits and an error code of 75/13 would appear as X'7526' in bits 0-15. (The first digit of the subcode is contained in bit positions 8, 9, and 10. Hence, it may have a value of 0-7.) The previous contents of SR1 and SR3 are lost. The meaning of each error and abnormal code is shown in Tables B-1 to B-4.

Certain errors are also reported in the TYC field of the DCB. The correspondence between error/abnormal codes and TYC codes is given in Appendix A.

Table B-1. Abnormal Codes - Insufficient or Conflicting Information

Abnormal Code	Subcode	Originating Monitor Routine	Meaning of Code
01	00	OPEN	An attempt was made to open a DCB with insufficient information.
01	0B	OPEN	A number of contiguous granules (in random files) has been requested, but they are not available.
02	00	OPEN	An attempt was made to open the next file with NXTF specified in the DCB but there are no more files.
02	01	OPEN	The end of all accounts has been encountered, and NXTA is specified in the DCB.
03	00	OPEN	The input or update file does not exist.
08	00	OPEN	An attempt was made to open the next file but the name of the next file is a synonym for the primary name of the file.
09	00	RDERLOG	An attempt was made to close and return a device which was not partitioned or a device within a partitioned controller.
09	01	RDERLOG	The device referenced in the Diagnostic DCB is a nonexistent device.
09	02	RDERLOG	The device referenced in the Diagnostic DCB is currently in use.

Table B-1. Abnormal Codes – Insufficient or Conflicting Information (cont.)

Abnormal Code	Subcode	Originating Monitor Routine	Meaning of Code
09	03	RDERLOG	The device referenced in the Diagnostic DCB is currently in use by a symbiont.
09	04	RDERLOG	The Diagnostic DCB does not contain a command list.
09	05	RDERLOG	The command list was invalidated by a swap.
09	06	RDERLOG	There are more than 12 I/O command doublewords (IOCDs).
09	07	RDERLOG	The I/O command list is invalid. This includes invalid flags, invalid TIC address, invalid command list address specified by user, or insufficient room in the DDCB for the command list.
09	08	RDERLOG	Error during BLIST CAL. An invalid page found during PTV or VTP conversion, the status address is in error, the byte count is illegal in the IOCD, or an IOCD overlaps a page boundary.
09	09	RDERLOG	A buffer crosses a page boundary.
09	0A	RDERLOG	The user's ID does not match the ID specified on the last operator DIAG key-in or the user privilege level was less than A0.
09	0B	RDERLOG	The amount of core is not sufficient to allow the diagnostic program to lock itself in core.
09	0C	RDERLOG	The requested controller is not partitioned.
09	0D	RDERLOG	The device specifically requested on open is not partitioned.
09	0E	RDERLOG	A MAP CAL error due to an invalid page number during a PTV or VTP conversion.
09	0F	RDERLOG	Cannot get MPOOL for use in processing command list or MPOOL is less than 13 words long.
09	10	RDERLOG	A TIO, TDV, or HIO was requested with an invalid FPT.
09	11	RDERLOG	A CHAN option on an M:OPEN to a device type or op label is illegal.
0A	00	CLOSE	An attempt was made to close a DCB that is already closed.
0A	01	CLOSE	Illegal VLP code on M:CLOSE CAL.
0A	02	CLOSE	Not enough room in FIT for requested change.
0A	08	CLOSE	Illegal file name.
0A	09	CLOSE	New file name already exists.
0A	0A	CLOSE	Can't modify a synonymous file.
0B	00	OPEN, READ CVOL	Unrecognized sentinel on labeled tape.
0C	00	OPEN	Illegal SYNON operation.
0D	00	OPEN	Insufficient room exists in the variable length parameter section of the DCB for the private pack serial number.

Table B-1. Abnormal Codes – Insufficient or Conflicting Information (cont.)

Abnormal Code	Subcode	Originating Monitor Routine	Meaning of Code
0D	01	OPEN	The private pack serial number list cannot be moved to the DCB because of an I/O error.
0E	00	OPEN	127 DCBs are open to the file. Access is denied.
13	00	DELREC or WRITE	The specified key was not found for an update file and the option is not NEWKEY.
14	00	OPEN	Access has been denied for one of the following reasons: (1) password missing or incorrect, (2) the file is execute-only and the wrong execute vehicle is accessing it, (3) there is a read or write account restriction, (4) a tape or private pack is being accessed with the wrong account in the DCB, (5) an attempt is being made to create a file in an account different from the log-on account, (6) an open OUT or OUTIN was attempted for an existing file on a private disk pack and the organization of the file is different from the organization in the open FPT or DCB, or (7) the first non-input open to tape did not occur at load point.
14	01	OPEN	An attempt was made to open a file for output and another user or DCB has the file open for input or output.
14	02	OPEN	Bad FPARAM location.
14	03	OPNL	The BREAK key was depressed or CONTROL Y was entered while waiting for a mount to be completed. The open was not performed.
14	04	MOVECAL (RDL)	User escape from random file cleaning operation on a M:MOVE CAL.
14	05	OPND	Invalid op label in DCB.
14	06	OPNF	Conflicting or missing DCB information. Probably either no file name is specified or the file name TEXTC count is illegal.
14	07	OPNF	Cannot open file DCB OUT with REL.
14	08	OPEN	Illegal private pack device type.
14	11	OPEN	Code conversion was requested for a tape drive not having that feature.
14	12	OPEN	800 bpi was requested for a tape drive not having the dual density feature.
14	13	OPEN	Code conversion option requested for an ANS tape not at the load point or code conversion requested for Xerox labeled tape.
14	14	OPNF	Access has been granted to an execute-only file because of the execute authorization.
15	00	DELREC or WRITE	An improper sequence of operations has been requested for an update file, or the FPARAM address did not belong to the user. For example, a WRITE or DELREC was issued for a keyed file and there is no key given on the WRITE or DELREC.
15	01	READ or PRECORD	Improper operation sequence on a shared keyed file.
16	00	WRITE	The NEWKEY option was specified, but the key already exists.
17	00	WRITE	The NEWKEY option was not specified for an output or scratch file.
18	00	WRITE	An attempt was made to write a keyed file sequentially with an out-of-order key.

Table B-1. Abnormal Codes – Insufficient or Conflicting Information (cont.)

Abnormal Code	Subcode	Originating Monitor Routine	Meaning of Code
19	00	OPEN/ CLOSE	Illegal operation on M:UC DCB.
1A	00	MOVECAL (RDL)	No error or abnormal address specified in the MOVE CAL FPT.
1A	01	MOVECAL (RDL)	The output DCB is missing.
1A	02	MOVECAL (RDL)	One or both DCBs are not open.
1A	03	MOVECAL (RDL)	The input DCB is not open IN or the output DCB is not open OUT.
1A	04	MOVECAL (RDL)	The MOVECAL is not allowed for device or ANS DCBs.
1A	05	MOVECAL (RDL)	The MOVECAL was aborted by Break, Y^c, or operator abort.
1A	42	MOVECAL (RDL)	KMAX of input DCB is greater than KMAX of output DCB.
1A	4A	MOVECAL (RDL)	The specified buffer does not belong to the user.
20	01	READ	A private pack is locked out.
20	02	READ	An attempt was made to use a private pack that is for exclusive use of another user.
20	03	READ	A private pack was not properly requested.
20	04	OPEN	An on-line user has requested a spindle which is down but which was previously allocated to him and was not in use.
20	05	OPEN	A private pack set contains multiple primary volumes.
21	00	OPEN/ CLOSE	Private pack consistency check failure.
22	00	OPEN	An error occurred on a private pack while trying to open an existing file.
2E	00	OPEN	An attempt was made to open a DCB that is already open.
30	01	LBLT	The user label is bad. All ANS labels must be 80 bytes in length. User header labels must begin with UHL1 and user trailer labels must begin with the characters UTL1. (The byte count is not part of the label because all ANS labels are 80 bytes long; however, it is automatically restored in the first byte of the label buffer when a label is read.)
30	03	LBLT	The file name is greater than 17 characters in length or is equal to zero.
30	04	LBLT	EXPIRE, NEVER was specified.
30	05	LBLT	The format code is illegal.
3F	35	JOBENT	The user tried to enter a job with an illegal account or priority.
3F	36	JOBENT	Job entry has been disallowed by the operator.

Table B-1. Abnormal Codes – Insufficient or Conflicting Information (cont.)

Abnormal Code	Subcode	Originating Monitor Routine	Meaning of Code
3F	37	JOBENT	The user is not allowed to use the service he requested.
3F	38	JOBENT	A function inconsistency exists.
3F	39	JOBENT	The id requested for deletion is not valid.
3F	3A	JOBENT	It is too late to delete job. Either the job is scheduled to run, is running, or has been completed.
3F	3B	JOBENT	No more symbiont space is available or the queue is full.
3F	3C	JOBENT	The user is not allowed to use job entry service.
3F	3D	JOBENT	The system is nonsymbiont, or the LL device is not a symbiont printer or is not defined as a symbiont device.
3F	3E	JOBENT	A DCB has been specified and it is already open.
3F	3F	JOBENT	The specified buffer address is not in the user's program.

Note: In all of the above cases, return is made to the user's program for continuation of execution if no abnormal address is specified in the DCB.

Table B-2. Abnormal Codes – Device Failure or End-of-Data

Abnormal Code	Subcode	Originating Monitor Routine	Meaning of Code
04	00	PRECORD/READ/OPEN	The beginning-of-file has been encountered.
05	00	PRECORD or READ	The end-of-data has been encountered.
06	00	READ	The end-of-file has been encountered (or first read of ! card).
07	00	READ	Data has been lost because the buffer was smaller than the record read, or a parity error was detected.
1C	00	READ, WRITE or PRECORD	The end-of-tape has been encountered.
1C	01	WRITE	The end-of-tape has been encountered on a common journal.
1D	00	READ or PRECORD	The beginning-of-tape has been encountered or a bad command has been sent to the terminal.
1F	00	WRT/IOD/IORT	BIN (or VFC) is not valid for this device.
23	00	COC	On-line terminal read timed out.
24	00	COC	On-line conditional read issued with no type-ahead.

Note: In all of the above cases, return is made to the user's program for continued execution if no abnormal address is specified in the I/O CAL FPT.

Table B-3. Error Codes – Insufficient or Conflicting Information

Error Code	Sub-code	Originating Monitor Routine	Meaning of Code
40	00	READ	A request was made to read an output file.
42	00	READ, WRITE or RANDOM	The key was not valid. The key length was zero or greater than the key maximum for the file or a random file granule number is out of legal range.
42	01	STPNR	Illegal buffer size on assign/merge read or write.
43	00	READ	No record having the specified key was found.
44	00	WRITE	A request was made to write in an input file.
46	xx	READ	The DCB contains insufficient information to open a closed DCB on a Read operation. Subcodes corresponding to the OPEN abnormal codes above describe why the implicit OPEN failed.
46	21	READ or WRITE	A private disk pack logic inconsistency exists.
46	22	READ or WRITE	A private disk pack error occurred trying to open an existing file.
46	48	READ	On-line user is not allowed to access the card reader.
47	xx	WRITE	The DCB contains insufficient information to open a closed DCB on a Write operation. Subcodes corresponding to the OPEN abnormal codes above describe why the implicit OPEN failed.
47	2B	OPEN	Invalid OP Label in DCB.
47	48	WRITE	The symbiont use flag was not set for on-line user.
48	00	OPEN	The symbiont use flag was not set for the given device.
48	01	OPEN	On-line user is not allowed to access the card reader.
49	00	PV	The user's peripheral use flags do not permit the use of tapes.
49	01	OPEN	No tape drives or disk spindles are available (on-line maximum exceeded or all drives or spindles in use). This error only occurs for on-line or ghost jobs.
49	02	OPEN	The user's tape drive or disk spindle limit from LIMIT card is exceeded.
49	03	OPEN	There is insufficient DCB space for the requested serial numbers.
4A	00	READ, WRITE or ENQ	Either the specified buffer or the indirect address in FPT does not belong to user.
4A	01	IOCHECK	The time parameter is too big on M:CHECKECB.
4A	02	IOCHECK	ECB is in the wrong state on M:CHECKECB or M:CHECK W/ECB.
4A	03	IOCHECK	There is an infinite wait condition on M:CHECKECB or M:CHECK W/ECB.
4A	04	IOCHECK	There is no monitor work space to honor M:CHECKECB or M:CHECK W/ECB.
4A	05	IOCHECK	The access code is incorrect for ECB address on M:CHECKECB or M:CHECK W/ECB.
4B	00	READ or WRITE	An attempt was made to open a file that the user already has opened.
4C	00	READ or WRITE	An attempt was made to open a file that another user already has opened.
4D	00	CLOSE	An attempt was made to close and release a file that someone else is reading.
4E	00	ARDL	There is an ANS block count error and no ABCERR was specified.
4E	01	READ or CVOL	A volume sequence number error occurred on an ANS tape.
4E	04	LBLT	BOF encountered on ANS tape with no block count error.
4E	05	READ or CVOL	An ANS block count error exists and end of tape and end of file has been encountered.
4E	07	READ or CVOL	An ANS block count error exists and end of file has been encountered.

Table B-3. Error Codes – Insufficient or Conflicting Information (cont.)

Error Code	Sub-code	Originating Monitor Routine	Meaning of Code
51	00	CLOSE	The file is still open in the input mode through another DCB. The file being closed is deleted.
52	00	OPEN	Insufficient privilege to use this CAL.
54	00	READ	The user has tried to read a control command via the control input (C) device more than once through the same DCB.
55	00	OPEN	Too many files are open simultaneously (the monitor's file-use tables cannot handle that many files).
56	00	CLOSE or CVOL	The system is unable to complete a tape volume switch because the reel number has not been specified or an error occurred opening the new volume.
75	00	CLOSE	The free sector pool contains erroneous information. (This message appears only in ERR-LOG.)
75	01	READ	Data records were lost due to a bad disk address in master index.
75	02	READ	The master index is inaccessible due to bad disk address in preceding master index.
75	03	OPEN	The entire file is inaccessible due to bad disk address in file directory or bad information in file information table.
75	04	OPEN or CLOSE	One or more files are inaccessible due to an error in the file directory.
75	05	OPEN	All files in account were lost due to bad disk address in account directory.
75	06	OPEN	A bad disk address link to next account directory exists. The current account and other accounts are gone.
75	07	OPEN	An error exists in the pyramid. (This message only appears in ERRLOG.)
75	4x		75/40 – 75/47 are the same as 75/00 – 75/07 except that in addition, a hardware error has been detected.
75	7D	OPEN	An error has been detected while trying to perform a fast open. The open will be retried. (This message only appears in ERRLOG.)
75	7E	RDF	Error in main directory granule. The dual granule will be read. (This message only appears in ERRLOG.)
75	7F	RDF	File inconsistency corrected by software. (This message only appears in ERRLOG.)

Note: In all of the above cases, the job is aborted if no error address is specified in the DCB. In batch mode, the monitor skips to the next job; in on-line mode, control is returned to TEL which prints the message and awaits further user commands. For error code 54, the job is aborted in all cases.

Table B-4. Error Codes – Device Failure or End-of-Data

Error Code	Sub-code	Originating Monitor Routine	Meaning of Code
41	00	READ	An irrecoverable read error has occurred.
41	01	COOP	A bad disk address was detected by the input cooperative when reading the input symbiont file.
41	02	READ	Labeled tape read error encountered on block in which requested record was contained. Byte 0 of SR1 contains the number of records in the block.
41	03	READ	Labeled tape read error encountered on block in which requested record was contained. Requested record not transmitted to the user.
41	04	READ	Partial record transmitted following Error 41/03.
45	00	WRITE	An irrecoverable write error has occurred.
45	01	WRITE	An irrecoverable write error has occurred on a common journal.
4F	00	WRITE	There was an unrecoverable error after the reflector on a tape.
57	00	READ or WRITE	Public secondary storage is exhausted, or the user has exceeded his secondary storage authorization.
57	44	RANDOM	There has been a Write request with a specified byte count, and not enough granules remain in a random file to satisfy the Write request, or the beginning relative granule number on a Read request is valid but the specified byte count extends beyond the end-of-file.

Note: In all of the above cases, the job is aborted if no error address is specified in the I/O CAL FPT. In batch mode, the monitor skips to the next job; in the on-line mode, control is returned to TEL which prints the message and awaits further user commands.

Table B-5. Other Monitor Error Codes

Error Code	Sub-code	Originating Monitor Routine	Meaning of Code
7F	1D	INITRCVR	Single user abort due to software check 1D.
7F	21	INITRCVR	Single user abort due to software check 21.
7F	22	INITRCVR	Single user abort due to software check 22.
7F	31	INITRCVR	Single user abort due to software check 31.
7F	32	INITRCVR	Single user abort due to software check 32.
7F	49	INITRCVR	Single user abort due to software check 49.
7F	60	TEL	TEL couldn't get a page.
7F	61	INITRCVR	Single user abort due to software check 61.
7F	6A	INITRCVR	Single user abort due to software check 6A.
7F	79	INITRCVR	Single user abort due to software check 79.
7F	7C	INITRCVR	Single user abort due to software check 7C.
7F	7E	INITRCVR	Single user abort due to software check 7E.
A0	00	ASP	An attempt was made to RUN under an invalid debugger name, or a request for an invalid debugger through TEL.
A1	00	ASP	An attempt was made to associate a debugger with a shared processor.
A1	01	ASP	An attempt was made to debug an execute-only load module.
A1	02	ASP	Conflict between library's overlays and debugger's data.
A2	00	ASP	An attempt was made to access a processor for which the user is not authorized (e.g., an on-line call to CCI).
A2	01	STEP	Access to non-system processor denied.
A2	02	STEP	Access to processor denied by processor restriction list.
A2	xx	STEP	Access to processor denied. (xx is the error code indicating why the system processor restriction file could not be read and is one of the error/abnormal codes given in Tables B-1 through B-5.)
A3	00	TRAP	Trap control cannot be given to the user because his task control block (TCB) does not exist or is full, or his pointer has been destroyed.
A3	01	TRAPC	No environment present for return.
A3	02	TRAPC	That trap should not be simulated.
A4	00		Unspecified trap.
A4	01	TRAP	Trap 40 - Nonexistent instruction.
A4	02	TRAP	Trap 40 - Nonexistent memory reference.
A4	03	TRAP	Trap 40 - Privileged instruction.
A4	04	TRAP	Trap 40 - Memory protect violation.
A4	05	TRAP	Trap 41 - Unimplemented instruction.
A4	06	TRAP	Trap 42 - Stack overflow.
A4	07	TRAP	Trap 43 - Fixed point overflow.
A4	08	TRAP	Trap 44 - Floating point fault.
A4	09	TRAP	Trap 45 - Decimal arithmetic fault.
A4	0A	TRAP	Trap 46 - Watchdog timer.

Table B-5. Other Monitor Error Codes (cont.)

Error Code	Sub-code	Originating Monitor Routine	Meaning of Code
A4	0B	TRAP	Trap 47 - Programmed trap.
A4	0D	CSEHAND	Trap 4D - Instruction exception trap.
A5	00	STEP	User's load module exceeds user limit or available core.
A5	02	STEP	Virtual core is not available for special shared processor.
A5	04	STEP	While in the extended memory mode, the current job step was aborted so that TEL could be accessed.
A5	06	STEP	Current special shared processor was aborted so that TEL could be accessed.
A5	07	STEP	Procedure overlaps currently allocated common pages.
A5	08	STEP	Physical core is not available for special shared processor.
A5	09	STEP	Either virtual core or physical core was not available to obtain a buffer for a cooperative file.
A5	51	STEP	Bad data bias for core library. The load module is pre-B00.
A6	03	STEP	Specified load module does not exist.
A6	14	STEP	Load module access denied.
A6	30	STEP	Bad DCBs or DCB table.
A6	31	STEP	Bad head record.
A6	32	STEP	Load module bias not on page boundary.
A6	33	STEP	Pure procedure not on page boundary.
A6	34	STEP	DCBs not on page boundary.
A6	35	STEP	Head record is incomplete.
A6	36	STEP	Tree record is incomplete.
A6	37	STEP	No debugs allowed with link-built LMNs.
A6	38	STEP	Program too big for user area.
A6	39	STEP	File not keyed, not a LMN.
A6	3A	STEP	DCB links bad or circular.
A6	3B	STEP	TCB address is not within the data area.
A6	42	STEP	The module exists but it is not a load module.
A6	43	STEP	The module exists but it is not a load module.
A6	44	STEP	The requested program can't be found.
A6	50	STEP	The DCBs are biased below the user area. The load module is pre-B00.
A6	51	STEP	PMD/SNAP/MODIFY not allowed with an execute only load module.
A6	xx	STEP	The xx subcode specifies the reason the DCB could not be opened and will be the abnormal/error codes given in Tables B1-B5.
A8	00	STEP	An error or abort CAL was issued. (RNST bits are also set.)

Table B-5. Other Monitor Error Codes (cont.)

Error Code	Sub-code	Originating Monitor Routine	Meaning of Code
A9	00	UCAL	An error on a read or write of the assign/merge record occurred.
AA	00	STEP	A request was made for core library that does not exist.
AC			An attempt was made to read the card reader by an on-line user.
AD	00	STEP	Extending processing limits were exceeded.
AE	00	CALPROC ALTCP	The user issued a CAL with unknown codes.
AF	00	CALPROC	A CAL1 instruction referenced a non-DCB.
B0	00	DUMP	The program specified snapshot dumps but did not have an M:DO DCB.
B0	01	DUMP	The program attempted snapshot dump of inaccessible or nonexistent memory.
B0	02	DUMP	Inaccessible flag address given on conditional debug command.
B0	03	DUMP	Illegal parameter in DEBUG CAL.
B1	00	SEGLOAD	Monitor cannot find the segment named in the user M:SEGLD DCB.
B1	01	SEGLOAD	Bad tree table.
B1	02	SEGLOAD	Circular tree table encountered.
B1	03	SEGLOAD	Data size specified in tree is too large.
B1	04	SEGLOAD	Procedure size specified in tree is too large.
B1	05	SEGLOAD	Overlay limits as defined in TREE lie outside of limits defined in HEAD record.
B1	06	SEGLOAD	Unable to get a page for segloading. (System error.)
B1	07	SEGLOAD	Page obtained by M:CVM procedure encountered.
B1	08	SEGLOAD	The paged load module is greater than 255 segments.
B2	00	ENTRY	The user issued a CAL2, CAL3, or CAL4.
B3	00	WRTD	Limit exceeded.
B3	01	WRTD	Punch limit. (PO)
B3	02	WRTD	Printer page limit for processor. (LO)
B3	03	WRTD	Printer page limit for user. (UO)
B3	04	WRTD	Printer page limit for debugging. (DO)
B3	08	WRTD	Execution time limit.
B4	00	STEP	Exit.
B4	01	STEP	User issued M:ERR.
B4	02	STEP	User issued M:XXX.
B4	03	STEP	Operator E (error) key-in.
B4	04	STEP	Operator X (abort) key-in or user abort.
B5	xx	LDLNK	See STEP (error code A5 and A5) subcodes and I/O error codes.
B5	62	LDLNK	M:LINK and M:LDTRC are not permitted when a shared processor is associated with the user program.

Table B-5. Other Monitor Error Codes (cont.)

Error Code	Sub-code	Originating Monitor Routine	Meaning of Code
B5	63	LDLNK	The program must not be loaded with Link.
B5	64	LDLNK	The user must own all memory from data through dynamic data.
B5	65	LNKTRC	Page acquired by CVM encountered.
B5	66	LNKTRC	Out of pages. (System error.)
B5	67	LDLNK	A logically impossible exit to Load and Link has occurred.
B5	68	LDLNK	Illegal information supplied in transfer file.
B5	69	LDLNK	A Load and Link cleanup occurred without a previous Load and Link operation.
B5	6A	LNKTRC	Load and Link to command processor not allowed.
B5	6B	STEP	A load and link to a linked program is not allowed.
B5	6C	STEP	A load and link to a special shared processor is not allowed.
B5	6D	LNKTRC	Insufficient physical core exists for core library following LNKTRC.
B5	6E	LNKTRC	M:LINK/LDTRC illegal for programs with transaction processing CALs outstanding.
B5	6F	LNKTRC	M:LDTRC attempt to execute a previously executed load module.
B5	70	LNKTRC	M:LINK/M:LDTRC illegal for programs with real-time ICBs associated.
B6	00	STEP	M:LINK: Not SEGLOAD DCB.
B6	01	STEP	The DCB name chain must be in the DCB record.
B6	02	STEP	The DCB name chain may not be linked.
B6	03	STEP	The DCB name chain is irregular.
B6	04	STEP	The DCB has no name.
B6	05	STEP	A user cannot have more than 509 DCBs.
B6	06	STEP	The DCB is outside of the buffer.
B6	07	STEP	A DCB may not cross a page boundary.
B6	08	STEP	A DCB must be at least 22 words long.
B6	09	STEP	KBUF must lie within the DCB.
B6	0A	STEP	FLP must lie within the DCB.
B6	0B	STEP	The FLPs overlap into KBUF.
B6	0C	STEP	M:SEGLD DCB needs 10 words for variable length parameters.
B7	00	OPNLD	Unrecognized stream-id.
B7	01	OPNLD	Unrecognized DEV specification.
B7	02	OPNLD	The function specified (IN or OUT) is not legal for this device.
B7	03	OPNLD	A nonzero workstation name is specified for an unauthorized user (i.e., the processor is not a shared processor and the privilege level of the user is less than X'C0').
B7	04	OPNLD	The peripheral use flag is not set for this DCB.
B7	05	OPNLD	Multiple copies are not allowed in concurrent output mode.
B7	06	OPNLD	Concurrent output mode is illegal for an IRBT.
B7	07	OPNLD	User is not authorized for concurrent output mode.
B8	01	RTROOT	M:QFI was attempted when no ICBs were associated with the user.
B8	02	RTROOT	M:INTRTN was attempted and there were no active interrupts associated with the user.

Table B-5. Other Monitor Error Codes (cont.)

Error Code	Sub-code	Originating Monitor Routine	Meaning of Code
B8	03	RTROOT	A real-time user has issued a restricted CAL after having locked himself in core (with M:HOLD).
B8	04	RTNR	A real-time user provided an illegal interrupt address or an unknown interrupt label.
B8	05	RTNR	A real-time user provided an FPT that is illegal because it is missing a required parameter.
B8	06	RTNR	The user did not specify a time value on an M:CLOCK request.
B8	07	T:JOBENT/ GRAN	A real-time user has requested a service from a system ghost job after having blocked the ghost job by locking himself in core (with M:HOLD).
B9	01	ALTCP/ RTROOT	User has insufficient privilege to issue this CAL1,5.
B9	02	RTROOT	The device specified via M:IOEX doesn't exist or is not preempted, or the specified DCB is not opened properly.
B9	04	ALTCP	The effective address of an M:EXU CAL is in protected memory.
B9	05	ALTCP	The instruction to be executed via M:EXU has an invalid op code.

XEROX LABELED TAPE ERROR HANDLING

After a block is read from labeled tape and an error (after normal retries) is encountered, the tape remains positioned after the last record read. The monitor then performs a consistency check on the record control information in the block. If the record control information is judged valid, the record is transferred to the user's buffer, as requested, and an error code 41/02 is returned. Byte 0 of SR1 will contain the number of records in the block. These records, although of questionable quality, are available to the user if he requests them. If the record control information is invalid, the user will receive an error return 41/03 and no information from the block is transmitted.

If after error condition 41/03 the following read causes a partial record (continuation of a record whose first part was contained in the block error) to be transmitted, an error return of 41/04 is given.

ENQUEUE/DEQUEUE ABNORMAL AND ERROR CODES

When an abnormal condition is encountered, return is made to the instruction following the CAL if no ABN address was supplied. If an ABN address was supplied, return is made to the ABN address and the user's register 10 is set to the appropriate abnormal code (see Table B-6). In either case, when an ECB address is supplied, the ECB is set to reflect the queue state.

When an error condition is encountered, the program is aborted if no ERR address was supplied. If an ERR address was supplied, return is to the ERR address and the user's register 10 is set to the appropriate error code (see Table B-7). In the latter case, when an ECB address is supplied, the ECB is set to reflect the queue state.

If an M:ENQ or M:DEQ procedure call is issued in a system that was generated without these services, the user is aborted with the error code as defined in Table B-7.

Table B-6. Enqueue/Dequeue Abnormal Codes

Abnormal Code	Subcode	Originating Monitor Routine	Meaning of Code
31	00	ENQ	A dequeue was attempted on a resource/element for which the user was not queued.
31	01	ENQ	An enqueue was attempted on resource/element for which the user was already queued. If an ECB address was given, the ECBP bit is reset to 0 if the user is still waiting for the resource/element or is set to 1 if the user has control of the resource/element.
31	02	ENQ	An enqueue SHARE was attempted on a resource/element for which the user was already queued as EXCLusive. The SHARE request is ignored and the EXCLusive request remains in the queue. If an ECB address was given, the ECBP bit is reset to 0 if the user is still waiting for the resource/element or is set to 1 if the user had control of the resource/element.
31	03	ENQ	The requested resource/element is not presently available on an enqueue TEST or enqueue NOWAIT request. If it is an enqueue NOWAIT request, the user is queued for the resource/element. The ECB is reset to 0.
31	04	ENQ	The enqueue request was aborted by a BREAK or CONTROL Y. The request is not queued.

Table B-7. Enqueue/Dequeue Error Codes

Error Code	Subcode	Originating Monitor Routine	Meaning of Code
4A	00	ENQ	An address in the FPT is not in the user's area or some other inconsistency was detected in the FPT.
58	00	ENQ	The request would result in a deadlock. Not only is the request rejected, but the user should dequeue all elements to allow other users to complete their operations, thus freeing the elements.
58	01	ENQ	There are no more empty entries in the monitor's enqueue tables. Not only is the request rejected, but the user should dequeue all resource/elements to permit other users to proceed.
58	02	ENQ	The enqueue request is for ALL and the user has sub-queues other than NULL, thus creating a deadlock.
58	03	ENQ	The user is not authorized to use the enqueue service.
AE	00	CALPROC	An M:ENQ or M:DEQ procedure call was issued in a system that does not include the enqueue/dequeue optional feature. The job step is aborted.

APPENDIX C. CONVERTING FROM BTM TO CP-V

INTRODUCTION

Converting an installation from BTM to CP-V is a relatively simple matter because of the compatibility of the file system and the processors. This appendix lists the procedures that must be used to accomplish the conversion. It then compares the time-sharing services of CP-V with those of BTM. The common processors of the two systems are compared and the differences of use, including added facilities offered by CP-V, are outlined where they exist for the assembler, FORTRAN, loaders, Edit, Delta, BASIC, and FERRET/PCL. Finally, a list of miscellaneous differences of detail is given.

CONVERSION PROCEDURES

Because the file systems are compatible and most processors are identical, changeover from BTM to CP-V is relatively simple. Three main steps are required.

1. Files recorded on FAST SAVE tapes in BTM must be entered into CP-V. This is done simply by a standard FRES restore.

2. Load modules of BTM are not compatible with CP-V since they are biased differently. Each must be reloaded from ROMs. Similarly, SAVE files must be recreated.

3. Any installation produced on-line processors using CAL3 calls must be modified to use the equivalent CP-V services. This is usually a simple matter using Delta. Batch programs and on-line programs using standard BPM CALs will run without change.

COMPARISON OF CP-V AND BTM SERVICES

The following is a detailed comparison of the time-sharing services of CP-V with those of BTM from the terminal user's point of view. It is assumed that the reader is familiar with BTM on-line capabilities described in the BTM/TS Reference Manual, 90 15 77.

TELETYPE OPERATIONS

The CP-V terminal is activated with the same procedure used for BTM. Once the terminal is operational under CP-V, the system responds by typing

```
XEROX CP-V AT YOUR SERVICE - site id
time date USER# sysid LINE# line
optional operator message
LOGON PLEASE:
```

The user inputs his account, name, and an optional password (in that order) as he would for BTM. There will be a short delay before CP-V responds; the LOGON data must be printed on the operator's console first.

To terminate an on-line session, the user types the OFF command. This serves the same function as the BTM BYE command. CP-V responds by typing the following statistics:

```
CPU = m.mmm    CON = hh:mm:ss    INT = nn
    CHG = xxxx
```

where

CPU is the CPU time, in minutes.

CON is the terminal connect time, in hours, minutes, and seconds.

INT is the number of terminal interactions during the session.

CHG is the number of charge units for the session.

Unlike BTM, the number of RAD and disk granules used during the on-line session is not printed. The number of granules remaining at any time can be determined with the DISPLAY command.

If the user wants to log on again while the line is still connected to CP-V, he does not have to hit the BREAK key as he would for BTM. All he need do is wait for a few seconds and CP-V will type the LOGON request again.

Several special Teletype characters for CP-V have different (or new) meanings from their BTM counterparts. These characters are listed in Table C-1. In addition, CP-V supports 2741-compatible terminals with all combinations of Selectric and EBCD code sets with APL and standard keyboards.

Table C-1. Additional or Different Teletype Characters for CP-V

CP-V Character	BTM Character	Meaning
RUBOUT or ESC RUBOUT	ESC RUBOUT	Erase last character.
ESC ESC, 4 BREAKS, or Y^c	ESC ESC	Return to executive.
ESC T	(none)	Toggle tab simulation.
L^c or ESC L	(none)	End of page; go to top of next
ESC F	!EOD	End of file
ESC U	(none)	Toggle upper/lowercase
ESC C	(none)	Toggle tab relative mode
ESC S	(none)	Toggle space insertion mode
ESC (	(none)	Uppercase shift
ESC)	(none)	Lowercase shift
Q^c (X-ON)	(none)	Turn on paper tape reader
S^c (X-OFF)	(none)	Turn off paper tape reader

Notes:

1. In all cases, CP-V responds to an ESC character pair with the character followed by a backslash (\).
2. The superscript c indicates that the CONTROL key is to be depressed.
3. The meaning of the excape sequences X, R, P, I, E, CR, LF, and Q are identical in CP-V and BTM.

PROGRAMMING CONVENTIONS

The special CAL3s for terminal I/O in BTM are not implemented in CP-V. Most of the services these calls provide, however, are available to the CP-V on-line user in other forms, as follows:

CAL3,0	M:READ, M:KEYIN
CAL3,1	M:TYPE, M:WRITE, M:PRINT
CAL3,4	M:LINK, M:LDTRC, M:SEGLD
CAL3,5	M:ASP, M:DSP
CAL3,6	M:EXIT (to TEL)
CAL3,10	(Error messages available in ERRFILE)
CAL3,14	M:GL
CAL3,15	M:DATE, M:TIME
CAL3,7,8,9,11,13	Are not required in CP-V since the services are provided automatically.

CP-V presents a uniform programming interface to the user program regardless of whether the program is run in on-line or batch mode; the same CALs execute in the same way. The difference between batch and on-line environments is that the operational labels for on-line are directed to the user's terminal rather than the central site card reader and line printer.

TERMINAL EXECUTIVE LANGUAGE (TEL) VERSUS BTM EXEC

Some of the new or different features provided by TEL as compared to BTM are

1. All TEL commands are terminated by a RETURN or LINE FEED. There is no system activation on two characters of a name or on punctuation as in BTM.

2. Many functions that had to be accomplished via a subsystem parameter in BTM can be accomplished by a single TEL command under CP-V. Either such a function is carried out directly by TEL (e.g., the SET command) or an implicit call is made to the proper processor (e.g., the TEL command BUILD results in an implicit call to Edit).

3. In contrast to the BTM processors FORTRAN, Symbol, and Loader, the on-line user does not have to preassign his files to source input, binary output, and listing output for the corresponding CP-V processors (FORT4, META, and LINK). In fact, all of the control commands needed to perform assembly or load (assignment of DCBs, processor call, processor options) are combined into one TEL command.

4. CP-V allows a properly-authorized user on-line access to peripheral devices (printer, punch, paper tape, card reader) and magnetic tape. Such capabilities do not exist for the BTM user.

5. Any load module under CP-V may be called for execution by an on-line user via TEL. This includes load modules under any account (not just :SYS).

6. Whereas BTM recognizes the word HERE to mean the user's terminal, CP-V recognizes the word ME.

7. In CP-V, a dollar sign may be used to refer to a program just assembled, compiled, or loaded during the current on-line session.

8. Under BTM, the only device-type assignment permitted is the assignment of a DCB to the user's terminal. With the CP-V SET command, it is possible to set most of the DCB parameters that are set by the batch ASSIGN command and many of the parameters that are set by BPM OPEN and DEVICE procedures. The LDEV command allows the user to assign an I/O stream to a particular device and to define attributes of the device.

9. TAB characters are handled somewhat differently in CP-V. The effect of a tab character on input and output is dependent upon the DCB tab settings, the space insertion mode (ESC S), the tab simulation mode (ESC T), and the tab relative mode (ESC C).

   ESC T controls whether or not tabs are to be simulated at the terminal. Normally, the ESC T switch is ON, i.e., tabs are simulated. When a tab character, I^c or ESC I, is encountered either on input or for output, spaces are sent to the terminal to bring the carrier to the proper position as defined by tab stop settings and the nominal beginning of the line (see below). When the mode is off, the terminal is assumed to have a real physical tab mechanism and the tab character itself is sent.

   Tab stops are set in the DCB for locating the simulated tab stops via either the TABS TEL command for the M:UC DCB or an M:DEVICE CAL for some other input or output DCB.

   ESC C, the Tab Relative Mode, controls whether counting for simulated tab stops begins at the left margin (as would physical tab stops) or is offset by any message just output (the physical position when the M:READ was issued). This latter is useful when providing input to a processor that reads with a prompt character or with a prompt sequence, such as the line number prompt used by Edit.

   ESC S, the Space Insertion Mode, controls whether a series of spaces (normal) or the tab character itself is sent from the terminal to the reading program. This is useful for processors that do not normally expect the tab character. The number of spaces sent is controlled by the tab stop settings and the ESC C mode.

   In all cases where tab simulation is required and no tab stops are specified, a single space is sent.

A summary of TEL commands and the comparable BTM commands appears in Table C-2. The first column contains the TEL command format; the middle column contains the corresponding BTM command(s) required to achieve the same function. The command function is described in the third column. File identification is designated by "fid" and has the format

	CP-V		BTM
name	. account	name	(account)
	. account. password		(account,password)
	. . password		(,password)

The prompt character (!) has been left off the TEL and BTM EXEC commands. Prompt characters for processors, however, are indicated.

Table C-2. TEL Command Summary and Equivalent BTM Command(s)

TEL Command	BTM Command(s)	Description
BACKUP fid	(none)	Saves the specified file on a system tape.
BATCH fid[,fid]...(Simplified format)	ASSIGN M:SI, (FILE, file) BPM INSERT JOB?Y	Enters the specified file in the batch job stream.
B[UILD] fid	EDIT *BUILD fid	Allows a new file to be created from the terminal using the Edit processor.
BYE or OFF	BYE	Disconnects terminal from systems and provides accounting directory.
CANCEL jid[,jid]...	BPM DELETE JOB Y ID=jid	Cancels a previously submitted batch job.

Table C-2. TEL Command Summary and Equivalent BTM Command(s) (cont.)

TEL Command	BTM Command(s)	Description
COBOL [sp] [ON OVER [rom] [, list]] where sp = fid or ME. rom = fid, stream-id, CP, or No. list = fid, stream-id, LP, ME, or No.	(none)	Compiles the specified COBOL program.
COMMENT {ON OVER} list where list is fid, LP, ME, or stream-id.	ASSIGN M:D0,(list) where list = FILE,name or HERE	Directs error commentary (from an on-line assembler or compiler) to the specified device.
CONTINUE, GO, or PROCEED	PROCEED	Continues processing from the point of termination.
C[OPY] sf {TO OVER INTO} df where sf = source specification df = destination specification	FERRET ≥C[OPY] fid$_1$, fid$_2$ or ≥E[XAMINE] fid	Copies a file to the specified device.
COUPLE	(none)	Allows other terminals to couple to this terminal.
COUPLE line	(none)	Establishes a link between the user's terminal and another terminal.
DECOUPLE	(none)	Releases the connection between coupled terminals.
D[ELETE] {[DC/] DP[#serial no.][-rt]/} fid [,fid]...	FERRET ≥D[ELETE] fid	Deletes the specified file.
DI[SPLAY]	FERRET ≥S[TATISTICS]	Lists the current values of various system parameters.
DONT COMMENT	(none)	Stops error commentary output.
DONT COUPLE	(none)	Causes attempts to couple to this terminal to be rejected.
DONT ECHO	(none)	Suppresses printing of commands being executed from a command file via XEQ.
DONT LIST	(none)	Stops listing output.
DONT OUTPUT	(none)	Stops object output.
DONT SEND	MESSAGE OFF	Inhibits operator messages.
DONT ERROR	(none)	Prevents the user from using CONTROL Y to exit from conditions requiring operator intervention when dealing with tapes.

Table C-2. TEL Command Summary and Equivalent BTM Command(s) (cont.)

TEL Command	BTM Command(s)	Description
ECHO	(none)	Causes printing of commands being executed from a command file via XEQ.
E[DIT] [fid]	EDIT *EDIT fid	Calls Edit to modify a file.
ERASE	(none)	Deletes unwanted printer output.
ERROR	(none)	Allows the user to enter CONTROL Y to exit from conditions requiring operator intervention when dealing with tapes.
EXTEND	(none)	Requests the extended memory mode.
FORT4[sp] [ON OVER][rom][, list] where sp = fid or ME rom = fid, stream-id, CP, or No. list = fid, LP, ME, stream-id, or No.	ASSIGN M:SI,(FILE,file) ASSIGN M:BO,(FILE,file) ASSIGN M:LO,(FILE,file) FORTRAN	Compiles the specified FORTRAN program.
GET fid or RESTORE fid	RESTORE fid	Restores the previously saved core image.
GO, CONTINUE, or PROCEED	PROCEED	Continues processing from the point of termination.
JOB jid[,jid]...	BPM STATUS CHECK ?Y ID = jid	Requests the status of a previously entered batch job.
L [LT#reel-id[-rt][(s)] [DC][, acct] (s)] LT#serial no.[-rt][(s)]/fid[(s)][, fid[(s)]]... fid[(s)][, fid[(s)]]... DP#reel-id[-rt][(s)] DP#serial no.[-rt]/fid[(s)][, fid[(s)]]... FT#serial no.[-rt]]	(none)	Lists file names and, optionally, attributes from the account dictionary, tape, or disk pack.
LDEV stream-id[,(option)]...	(none)	Attaches a cooperative stream to a physical device and/or defines attributes of the physical device.
LINK[options]rom[,rom][...,rom] [ON OVER lmn][;lid[, lid]... [, lid]] where rom = fid or $ lid = library fid codes include (L), (NL), (D), (ND), (C), (NC), (M), (NM)	LOAD ELEMENT FILES:[fid]... [fid] . . . SAVE lmn	Forms a load module as specified.

Table C-2. TEL Command Summary and Equivalent BTM Command(s) (cont.)

TEL Command	BTM Command(s)	Description
LIST $\begin{Bmatrix}\text{ON}\\ \text{OVER}\end{Bmatrix}$list where list = fid, LP, ME, or stream-id.	ASSIGN M:LO, (list) where list = FILE, name or HERE	Directs the listing output to the specified device.
lmn[sp]$\left[\begin{matrix}\text{ON}\\ \text{OVER}\end{matrix}\text{[rom][,list]}\right]$	(none)	Initiates execution of a load module where sp is assigned to M:SI; rom is assigned to M:GO; list is assigned to M:LO.
M[ESSAGE] text	FERRET >M[ESSAGE]text	Sends the specified message to the operator.

META sp $\left[\begin{matrix}\text{ON}\\ \text{OVER}\end{matrix}\text{[rom] [,list]}\right]$
where sp = fid or ME
rom = fid, stream-id, CP, or NO.
list = fid, LP, ME, stream-id, or NO.
	ASSIGN M:SI, (FILE,name)
	ASSIGN M:BO, (FILE,name)
	ASSIGN M:LO, (FILE,name)
	SYMBOL
		Assembles the specified source program.

OFF or BYE	BYE	Disconnects terminal from system and provides accounting directory.
OUTPUT $\begin{Bmatrix}\text{ON}\\ \text{OVER}\end{Bmatrix}$rom	ASSIGN M:BO, (FILE,name)	Directs rom output to a specified file.
PASSWORD, old password, new password	(none)	Assigns a new log-on password for the user.
PLATEN[w][,l]	(none)	Sets the value of the terminal platen width and page length (not including header) or displays the current platen size.
PRINT	(none)	Sends output to the line printer and card punch without waiting for the user to log off.
PROCEED, CONTINUE, or GO	PROCEED	Continues processing from the point of termination.
Processor Calls ANSF APL BASIC COBOL DELTA E[DIT] FDP FORT4 LINK META PCL lmn (user's program)	 (none) (none) BASIC (none) DELTA EDIT (none) FORTRAN LOAD SYMBOL FERRET (none) BPM RUN	Turn over control of the terminal executive to the processor.
PROMPT [character]	(none)	Sets the default prompt character.
ESC ESC, Y^c, or 4 BREAKs followed by QUIT	Escape Command (ESC ESC)	Terminates the current job step.

Table C-2. TEL Command Summary and Equivalent BTM Command(s) (cont.)

TEL Command	BTM Command(s)	Description
R[ESET] or SET dcb[0]	ASSIGN dcb	Clears DCB of previous parameters.
RESTORE fid or GET fid	RESTORE fid	Restores the saved core image.
RUN [options][ef[,ef]...] [ON OVER lmn] [;[libname][.[libacct][password]]...	LOAD ELEMENT FILES:[fid]... [fid] . . . XEQ?Y	Loads the specified load module and starts execution.
SAVE {ON OVER} fid	SAVE fid	Saves the current core image on the designated file.
SEND	MESSAGE ON	Allows the operator to send messages to the user terminal at any time.
SET dcb [0] or R[ESET]	ASSIGN dcb	Clears DCB of previous parameters.
SET dcb [oplabel device stream-id tapecode [tapeid]] [;dopt]... where dopt = device options	ASSIGN dcb (HERE)	Assigns device to a DCB or sets a DCB parameter.
SET dcb[[tapecode[tapeid][-rt] filecode[-rt]]/fid] [;fopt]... where fopt = file options	ASSIGN dcb (FILE,fid) [,(option)...]	Assigns file to a DCB or sets a DCB parameter.
SHOW [option [,option]...]	(none)	Lists the user's system information.
S[TART][lmn $][U[NDER DELTA]]	RUN LOAD MODULE FID:lmn (executed under a subset of BTM DELTA)	Begins execution of a load module, either with or without an associated debugger.
ST[ATUS]	FERRET >S[TATISTICS]	Displays the current accounting values.
SWITCH[S[ET]=n,...[R[SET]=m...]]	(none)	Sets and resets pseudo sense switches.
TABS	(none)	Displays the simulated tab stop settings.
TABS s[,s]... (maximum = 16)	TABS[s]...[,s] (maximum = 8)	Sets the simulated tab stops at the terminal.
T[ERMINAL]type [,algorithm] where type =33, 35, 37, 7015, APL, C360 DATA[POINT], EAPL, ESTD, EXEC[UPORT], MEMO[REX], SAPL, SSTD, TI, or installation-supplied translation table name.	(none)	Sets the terminal type for proper I/O translations.

Table C-2. TEL Command Summary and Equivalent BTM Command(s) (cont.)

TEL Command	BTM Command(s)	Description
T[ERMINAL][STATUS]	(none)	Lists current values used in controlling the terminal operation.
TP	(none)	Makes a terminal available as a slave transaction processing terminal.
U	(none)	Causes the words UNDER DELTA to be inferred in the next command.
WHERE account,name	(none)	Returns the line number of the user with the specified account and name (if the user is logged on).
XEQ fid[,record number]	(none)	Initiates processing of TEL commands from a command file.

PROCESSOR COMPARISONS

In several cases, the BTM on-line subsystems are toned-down versions of more powerful processors available to batch users. In contrast, the CP-V system allows both batch and on-line users access to the same processors. Differences between the CP-V and BTM processors are described below.

CP-V META AND BTM SYMBOL ASSEMBLERS

The on-line Meta-Symbol assembler for CP-V (META) has several advantages over the BTM Symbol assembler.

1. The limitations imposed by the Symbol language are lifted for the CP-V on-line user. He can form as sophisticated assembly language programs as the CP-V (and BTM) batch user.

2. The META subsystem recognizes more assembly options than the BTM Symbol subsystem.

AC	(ac., . . . ,ac_n)
BO[†]	Binary ROM output
CI[†]	Compressed input
CN	Concordance commands
CO	Compressed output
DC	Default concordance
GO[†]	Binary output on GO file
LO	Listing output
LU	List update
NS	No symbol

3. The parameters CI, SI, LO, BO, and GO do not need to be preassigned before calling META.

4. An on-line user can update a CI file with a source file built under Edit simply by specifying both files as output.

```
!SET   M:CI   DC/comp.
!META update ON bin, LP
WITH > CI
```

CP-V FORT4 AND BTM FORTRAN

The CP-V FORT4 processor is an Extended FORTRAN IV compiler. The BTM FORTRAN processor is an Extended FORTRAN IV-H compiler. Since FORTRAN IV-H is a subset of FORTRAN IV, the CP-V user can compile FORTRAN programs on-line which in BTM would have to be compiled in batch mode.

CP-V does not create SOTEMP files of source statements when input is directly to FORTRAN and SYMBOL from the terminal.

When entering FORTRAN programs a line at a time, syntax checking is performed after each line is received. If a statement is to be continued, each continued line must end with a colon (:) and each continuation line must use column 6. This is the exact opposite of BTM, where the colon indicates no continuation.

CP-V LINK LOADER AND BTM LOADER

Both CP-V LINK and BTM Loader subsystems form nonoverlayed, ready-to-run program images in core memory from ROMs and libraries. In addition, LINK forms a load module for later execution. Several options are also available under LOAD.

Take, for instance, internal symbol tables (ISTs). Considerable flexibility exists with regard to the construction of ISTs by LINK for use under DELTA. The user can specify whether or not he wants an IST to be built for each input file. Also, the ISTs for several input files can be merged. These capabilities contrast to the BTM Loader subsystem, whereby the D (debug) option allows only all-or-none IST construction.

In addition to the load map option (M), (available in both on-line loaders), LINK recognizes two other display options. The (D) option produces a list of all unsatisfied external and internal symbols at the completion of the linking process. The (C) option results in a display of all conflicting internal and external symbols. These displays may be inhibited by the (ND) and (NC) options. (BTM always outputs an undefined external symbol map. It outputs the undefined, internal symbol map if debug is specified.)

Both on-line loaders search libraries to resolve unsatisfied external references. (Such a library is a file containing ROMs "linked" together.) LINK, however, does not restrict its search to the :BLIB file of any account, as does the BTM Loader subsystem. Instead, it searches any file specified in the library (lid) portion of the LINK command. In this way, the CP-V user is relieved of maintaining all of his library-type ROMs in one unique file.

LINK places code in the 00 and 01 protection-type sections according to the dictates of the input ROMs. It does not force the entire load module into protection-type 00, as does the BTM Loader.

CP-V EDIT AND BTM EDIT

Features of CP-V Edit that are different from those of BTM are listed below.

1. The following edit commands may be given via TEL:

   BUILD fid
   EDIT [fid]
   DELETE fid

2. The file identification (fid) must follow the CP-V file identification structure.

[†] Implicitly specified in the META command.

3. CP-V terminates the entire command if the BREAK key is depressed.

4. A new command is available, TA. It sets the tab positions and has the following format:

   TA $\begin{Bmatrix} \text{F} \\ \text{M} \\ \text{S} \end{Bmatrix}$

   where

F	implies FORTRAN; tab set at column 7, 16, and 34.
M	implies Meta-Symbol; tabs set at columns 10, 19, and 37.
S	implies Meta-Symbol, short form; tabs set at columns 8, 16, and 30.

   The actual tab simulation is carried out exactly like the TABS command.

CP-V DELTA AND BTM DELTA

Features of CP-V Delta that are different from those of BTM Delta are listed below:

1. Delta may be called by the following means:

   a. To execute a load module under Delta, give the TEL command

      START lmn UNDER DELTA

   b. To call Delta after a program has already started executing, strike the CONTROL and Y keys simultaneously to return to TEL. Then give the TEL command DELTA.

   c. To call Delta to write and check a short program, give the TEL command DELTA.

   d. To call Delta after a program has aborted, give the TEL command DELTA.

2. Symbol tables can be manipulated at load time (see Chapter 8).

3. The following new commands are available:

   a. Symbol table control

;U	Display undefined symbols.
;KI	Remove current internal symbol tables.
;KG	Remove global symbol table and any symbols defined at terminal.

   b. Execution control

)	Execute current instruction and display next one.

   c. Memory searching and modification

e1,e2;W	Store e2 through mask in locations that match e1 through the mask.

   d. Breakpoint control (data and transfer)

e,r,val,m;DR	Data breakpoint whenever contents of e, masked by m, are in relation r to val (r options are LS, EQ, GR, GQ, NQ, LQ).	
e,r,val,m;DTr	Same as above in trace mode.	
n;D	Remove nth data breakpoint.	
0;D	Remove all data breakpoints.	
;D	Display list of active data breakpoints.	
;Y	Set transfer breakpoint mode.	
;YT	Same as above in trace mode.	
0;Y	Turn off transfer breakpoint mode.	
loc;Y	Start transfer breakpoint execution mode at loc.	
loc;YT	Same as above in trace mode.	
l,m,n,o;YS	Set entries in SAT.	
l,m,n,o;YR	Release entries in SAT.	
;YR	Release all entries in SAT.	
;YD	Display SAT.	
loc,opt1,opt2;Y	Set transfer breakpoints as follows:	
	opt1=0	all branches except those in SAT.
	opt1=1	only SAT branches.
	opt2=0	no trace on BIR/BDR.
	opt2=1	trace BIR/BDR.

Note: For each entry in this index, the number of the most significant page is listed first. Any pages thereafter are listed in numerical sequence.

Note: For each entry in this index, the number of the most significant page is listed first. Any pages thereafter are listed in numerical sequence.

Note: For each entry in this index, the number of the most significant page is listed first. Any pages thereafter are listed in numerical sequence.

Note: For each entry in this index, the number of the most significant page is listed first. Any pages thereafter are listed in numerical sequence.

Note: For each entry in this index, the number of the most significant page is listed first. Any pages thereafter are listed in numerical sequence.

Note: For each entry in this index, the number of the most significant page is listed first. Any pages thereafter are listed in numerical sequence.

Note: For each entry in this index, the number of the most significant page is listed first. Any pages thereafter are listed in numerical sequence.

e. Printer output

;J	Divert DELTA output to line printer.
a,b;0	Print hex dump from a to b on line printer. A header for the dump may be appended to 0.

f. Miscellaneous

e1,e2,v;Z	Store v in memory from e1 through e2.
;RK	Display addresses as CSECT type symbol plus any hex offset.

CP-V BASIC AND BPM/BTM BASIC

The following differences are completely described in Chapter 5 of the BASIC/Reference Manual, 90 15 46 – Revision B or later.

1. Language extensions

   a. String capability

      String variables (scalars, matrices, substrings).
      String expressions.
      Character strings.
      Length and value assignments (LEN, VAL).
      Numeric-to-string conversion (STR).
      Assignment and concatenation.
      Comparison.
      I/O.
      String-to-alphanumeric constant conversion.

   b. New intrinsic functions – CSC, SEC, COT, ASN, ACS, HSN, HCS, HTN, LTW, DEG, RAD.

   c. CHAIN LINK statement

2. New edit mode commands

   a. CLEAR

      ARRAYS
      STRINGS

   b. NULL

      ARRAYS
      STRINGS
      SIMVARS

   c. FILE PACK

   d. SET

   e. EXECUTE

3. Increased edit mode-execute mode commands

   a. Changes in BREAK-PROCEED logic

   b. Changes in direct statement capability

      (1) Smaller number of nondirect statements

      (2) Direct capability in edit mode

CP-V COUNTERPARTS TO FERRET COMMANDS

Most of the functions of the BTM FERRET subsystem can be accomplished with CP-V TEL and PCL commands. The FERRET commands and their CP-V counterparts are listed in Table C-3.

MISCELLANEOUS INFORMATION

Miscellaneous differences between BTM and CP-V are listed below:

1. In CP-V, read operations, through a DCB assigned to the NO operational label return an end-of-file code.

2. BTM and CP-V load modules have different formats. Therefore, load modules currently running under BTM must be reformed under CP-V before they will execute correctly. ROMs are compatible between BTM and CP-V.

3. Under BPM/BTM, X'15' corresponds to a carriage return and X'25' corresponds to a line feed. Under CP-V, X'15' corresponds to a line feed and X'0D' corresponds to a carriage return; X'25' is unassigned.

4. CP-V does not set ASN in the DCB to 5 if the DCB is assigned to a Teletype as in BTM. CP-V sets ASN to 3, DEVF to 1, and TYPE to 10. TYPE is not set until the DCB is opened. Prior to the opening of the DCB, it may contain an OPLB code or the EBCDIC representation of that OPLB.

5. In CP-V, all input/output through COC routines (M:UC) is restricted to 140 characters per M:WRITE or M:READ.

6. CP-V includes an elaborate checking algorithm for validating DCBs. Either the M:DCB procedure or the DCBs in the library should be used to assure correct DCBs.

7. On batch ASSIGN cards, device assignments are interpreted as oplabel assignments, i.e.,

   ```
   !ASSIGN    M:SI,(DEVICE,CRA03)
   ```

   is assumed to mean

   ```
   !ASSIGN    M:SI,(DEVICE,CR)
   ```

8. Number of on-line DCBs is increased from the BTM limit of four to a limit constrained only by the size of combined DCB and buffer POOL area which must total less than 10K.

Table C-3. FERRET Commands and Corresponding CP-V Commands

FERRET Command	CP-V Command	Description
>L[IST][acct]	!L [.acct]	Lists the specified account directory.
>T[EST] file	!L fid	Tests file accessibility.
>A[CTIVITY] file	(none)	Checks file activity.
>S[TATISTICS]	!ST[ATUS]	Displays accounting statistics for this on-line session.
>LOG		
>RAD		
>RADS		
>CPU		
>IO		
>SERV		
>S[TATISTICS]	!DI[SPLAY]	Displays system load parameters.
>N		
>D[ELETE] file	!D[ELETE]file	Deletes specified file.
>C[OPY] file$_1$,file$_2$	!C[OPY]fid$_1$ to fid$_2$	Copies file$_1$ to file$_2$.
>K[OPY] file$_1$,file$_2$		Copies file$_1$ to file$_2$ retaining keys.
>E[XAMINE] fid	!C[OPY] fid[TO ME]	Examines a file.
>I[NSPECT] fid		Examines a file and displays keys.
>M[ESSAGE] text	!M[ESSAGE] text	Sends message to operator.
>P[UNCH] fid	(none)	Punches file to paper tape.
>G[RANULES][(acct)]	(none)	Displays number of granules.
>R[EVIEW]	!PCL <REVIEW fid$_1$ [,fid$_2$]	Lists and selectively keeps or deletes all files in account.

INDEX

Note: For each entry in this index, the number of the most significant page is listed first. Any pages thereafter are listed in numerical sequence.

XEROX **Publication Revision Sheet**

SEPTEMBER 1978

CORRECTIONS TO XEROX CONTROL PROGRAM-FIVE (CP-V)/TIME-SHARING REFERENCE MANUAL
(Xerox 560 and Sigma 5/6/7/9 Computers)

PUBLICATION NO. 90 09 07H-1(9/78)

The attached pages contain changes which reflect the F00 version of Control Program-Five (CP-V). Pages in the H edition (11/76) of the manual that are to be replaced are: title page/ii, iii through viii, 7 through 10, 21 through 34, 37 through 50, 53/54, 57/58, 65 through 104, 105 through 118, 121 through 124, 125/126, 129 through 142, 147 through 156, 159 through 162, 163/164, 167/168, 171/172, 185 through 198, 201 through 206, and 209 through 216. (Pages 101.1, 101.2, 111.1, and 111.2 are new pages.)

Pages that are to be inserted are: 104.1/104.2, 124.1/124.2, and 162.1/162.2.

Revision bars in the margins of replacement pages identify changes. Pages without the publication number 90 09 07H-1(9/78) at the bottom are included only as backup pages; revision bars appearing on such pages identify changes made in a previous revision. A revision bar adjacent to a page number indicates that the material on the page has been reorganized without the content being changed.

21798
1978
Printed in U.S.A.

XEROX

ꞏeader Comment Form

We would appreciate your comments and suggestions for improving this publication

Publication No.	Rev. Letter	Title	Current Date

How did you use this publication?

□ Learning □ Installing □ Sales

□ Reference □ Maintaining □ Operating

Is the material presented effectively?

□ Fully Covered □ Well Illustrated □ Well organized □ Clear

What is your overall rating of this publication?

□ Very Good □ Fair □ Very Poor

□ Good □ Poor

What is your occupation?

Your other comments may be entered here. Please be specific and give page, column, and line number references where applicable. To report errors, please use the Xerox Software Improvement or Difficulty Report (1188) instead of this form.

Your name & Return Address

Thank You For Your Interest. (fold & fasten as shown on back, no postage needed if mailed in U.S.A.)

CUT ALONG LINE

PLEASE FOLD AND TAPE–
NOTE: U. S. Postal Service will not deliver stapled forms

FOLD ALONG LINE

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 59153 LOS ANGELES,CA 90045

POSTAGE WILL BE PAID BY ADDRESSEE

HONEYWELL INFORMATION SYSTEMS
5250 W. CENTURY BOULEVARD
LOS ANGELES, CA 90045

ATTN: PROGRAMMING PUBLICATIONS

FOLD ALONG LINE

Honeywell

Honeywell Information Systems
In the U.S.A.: 200 Smith Street, MS 486, Waltham, Massachusetts 02154
In Canada: 155 Gordon Baker Road, Willowdale, Ontario M2H 3N7
In the U.K.: Great West Road, Brentford, Middlesex TW8 9DH
In Australia: 124 Walker Street, North Sydney, N.S.W. 2060
In Mexico: Avenida Nuevo Leon 250, Mexico 11, D.F.

20425, 1478, Printed in U.S.A.