Xerox Control Program-Five (CP-V)

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

Batch Processing Reference Manual

90 17 64H-1(9/78)

CAL1-TO-FUNCTION INDEX

① These procedures are described in detail in the CP-V/SP Reference Manual, 90 31 13.

② This procedure is for on-line use and is described in detail in the CP-V/TS Reference Manual, 90 09 07.

③ The diagnostic routine associated with this procedure determines which of the three procedures (M:DMOD#, M:DPART, or M:DRET) was called.

④ These procedures are described in detail in the CP/SM Reference Manual, 90 16 74.

XEROX

XEROX CONTROL PROGRAM-FIVE (CP-V)

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

Batch Processing

Reference Manual

90 17 64H
90 17 64H-1

September 1978

File No.: 1X13
XL89A, Rev. 0

Printed in U.S.A.

REVISION

This publication documents the F00 version of Control Program Five (CP-V). The publication consists of the H edition of this manual (90 17 64H, dated November 1976) and the revision package numbered 90 17 64H-1 (dated September 1978). Vertical lines in the margins of pages labeled 90 17 64H-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 Remote Batch Terminal/Reference Manual	90 16 02
Xerox Remote Batch Terminal/Operator's Manual	90 16 26
Xerox Control Program-Five (CP-V)/TS Reference Manual	90 09 07
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)/TS User's Guide	90 16 92
Xerox Control Program-Five (CP-V)/RP Reference Manual	90 30 26
Xerox Control Program-Five (CP-V)/TP Reference Manual	90 31 12
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 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/Library Technical Manual	90 15 24
Xerox Extended FORTRAN IV/OPS Reference Manual	90 11 43
Xerox FORTRAN Debug Package (FDP)/Reference Manual	90 16 77
Xerox FLAG/Reference Manual	90 16 54
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 Report Program Generator (RPG)/Reference Manual	90 19 99
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 General Purpose Discrete Simulator (GPDS)/Reference Manual	90 17 58
Xerox Data Management System (DMS)/Reference Manual	90 17 38
Xerox SL-1/Reference Manual	90 16 76
Xerox 1400 Series Simulator/Reference Manual	90 15 02
Xerox Mathematical Routines/Technical Manual	90 09 06
Xerox CIRC-AC/Reference Manual	90 16 98
Xerox CIRC-DC/Reference Manual	90 16 97
Xerox CIRC-TR/Reference Manual	90 17 86

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.

CONTENTS

APPENDIXES

FIGURES

TABLES

PREFACE

Control Program-Five (CP-V) is a general-purpose system that operates on a Xerox 560 or Sigma 5, 6, 7, or 9 computer and a variety of peripheral devices. The system provides for five concurrent modes of operation.

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

This manual is the principal source of reference information for the batch processing features (i.e., job control commands, system procedures, I/O procedures, program loading and execution, debugging aids, and service processors). The purpose of the manual is to define the rules for using the batch processing features. Manuals describing other features of CP-V are outlined below.

- The CP-V Time-Sharing Reference Manual, 90 09 07, is the principal source of information for the time-sharing features. It defines the rules for using the Terminal Executive Language and other terminal processors.
- 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.
- 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 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 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 (local and remote users, system managers, remote site operators, and central site operators) is included in the manual.
- 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, applications programmers, and computer operators.
- 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. {A / id} 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.

1. INTRODUCTION

OPERATING SYSTEM

The CP-V monitor functions as the major control element in an installation's operating system. The operating system consists of the monitor and a number of processing programs: language processors, execution control processors, service processors, application processors, and user processors. In general, the monitor governs the order in which programs are executed and provides common services to all of them (see Figure 1).

The number, types and versions of the programs in an operating system vary, depending upon the exact requirements at a particular installation. Each operating system consists of a selection of monitor routines and processing programs that are closely integrated for a given range of applications.

The operating system required for a particular installation is generated through use of the System Generation programs, which are described in the CP-V/SM Reference Manual, 90 16 74.

As the requirements of an installation increase, the operating system can easily be enlarged, modified, or updated. The ability to adapt conveniently to new requirements is inherent in the system design. Once a system is generated, it can be quickly expanded to include user's programs, data, and system libraries. User's programs and the standard system processors are equivalent in that they are stored, cataloged, and referred to within the system in the same way. They are also written using the same conventions for communicating with the monitor.

The operating system is self-contained and requires operator intervention only under exceptional conditions.

PHILOSOPHY OF OPERATION

The monitor uses sophisticated techniques for efficient machine operation in a production environment. The ability to process a continuous series of jobs with little or no operator intervention is one of the most important features of the system. By reducing the need for operator participation, the operating system ensures faster throughput, and operations are less subject to error. For the most part, the operator should only have to perform routine tasks such as loading and unloading tape reels.

Complete and easy-to-use I/O services are available to user programs, thus relieving the programmer of many coding chores. Device assignment is general and automatic, enabling the user's program to exploit the complete flexibility of peripheral units.

I/O service is comprehensively organized to simplify programming and make machine utilization efficient. I/O transfers are automatically buffered, and I/O peripherals are serviced on a queue basis (by job). Jobs can thus be executed sequentially even though they might normally be I/O-bound and delay use of the CPU or other I/O devices.

The job scheduler permits selective job operation based on job type or administrative priority to maximize throughput efficiency or environmental needs. The computer operator

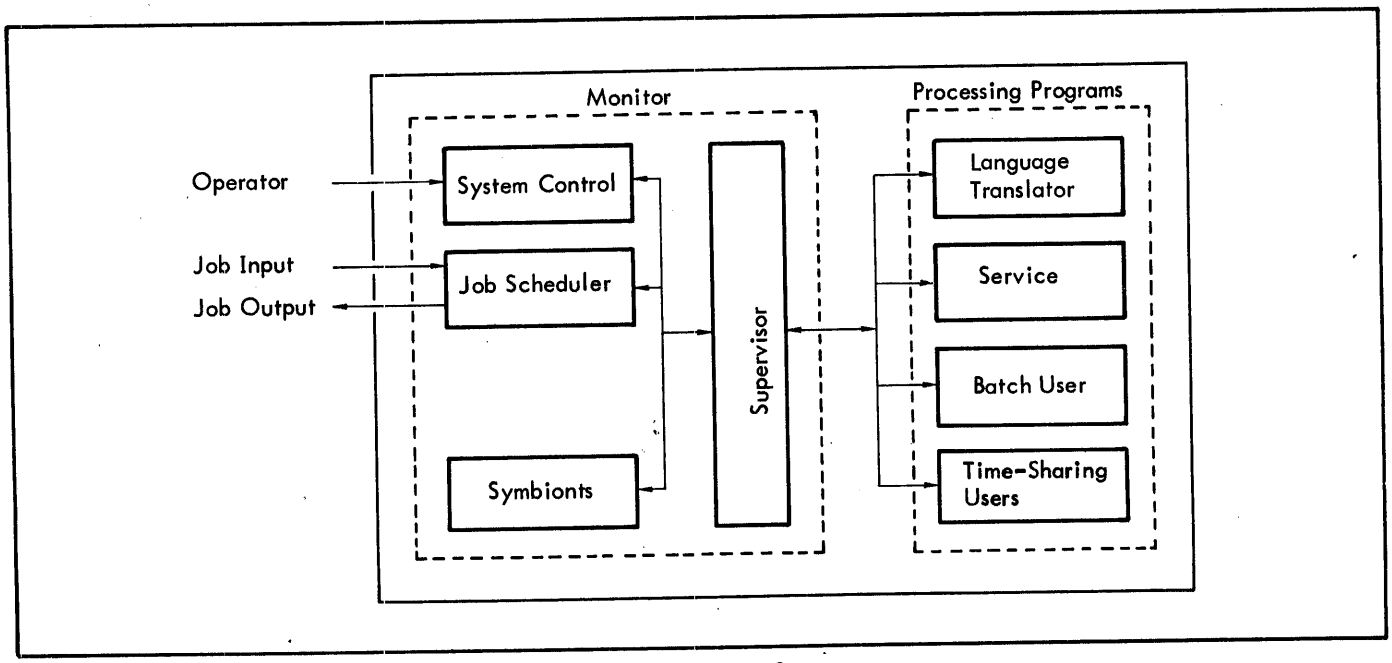


Figure 1. Operating System

GLOSSARY

addend value — a hexadecimal constant to be added to the value of a relocatable address. The constant is expressed as a signed integer appended to the address; e.g., START+12 or HERE-F1.

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.

ANS tape — a tape that has labels written in American National Standard format.

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.

common page — a page of core storage that is available to the user's program and in which stored data is retained until the current job is terminated or until the page is released by the user's program.

concatenation — a process whereby a number of files with the same filename and format e treated as one logical file. Concatenation is only applicable to ANS tapes.

conflicting reference — a refere .ce 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 command interpreter (CCI) — a monitor routine that interprets control commands.

control function — any monitor function initiated by a control command or control key-in.

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 (also see "symbiont").

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

disk pack — a secondary storage system of removable rotating memory.

dummy section — a type of program section that provides a means by which more than one subroutine may reference the same data (via an external definition used as a label for the dummy section).

element file — a user's file consisting of program elements, such as relocatable object modules or library load modules.

error severity level code — a four-bit code indicating the severity of errors noted by the processor. This code is contained in the final byte of an object module.

external definition — a load item that assigns a specific value to the symbolic name associated with a particular external definition name number. An external definition allows the specified symbolic name to be used in external references (see below).

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. When additional output is produced through the DCB, it is added to the previous contents of the file, thereby extending the file.

file management routines — monitor routines that interpret and perform I/O functions.

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 secondary storage file consisting of relocatable object modules formed by a processor.

granule — a block of disk sectors large enough to contain 512 words (a page, or 2048 bytes) of stored information.

job control language (JCL) a language consisting of control commands that provide job specifications to the monitor.

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 that uniquely identifies a record.

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

library load module a load module that may be combined with relocatable object modules, or other library load modules, to form a new executable load module.

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

load information information (i.e., control information, data, and instructions) generated by a processor and contained in one or more modules capable of being linked to form an executable program.

load location counter a counter established and maintained by the monitor to contain the address of the next location into which information is to be loaded.

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 input 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 attributes. 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.

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

option an elective operand in a control command or procedure call, or an elective parameter in a function parameter table.

overlay loader a monitor routine that loads and links elements of overlay programs.

overlay program a segmented program in which the element (i.e., segment) currently being executed may overlay the core storage area occupied by a previously executed element.

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").

postmortem dump a listing of the contents of a specified area of core memory, usually following the abortive execution of a program.

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.

protective mode a mode of tape protection in which only ANS expired tapes may be written on through an ANS DCB; no unexpired ANS tape may be written on through a non-ANS DCB; all ANS tapes must be initialized by the Label processor; no tape serial number specification is allowed at the operator's console; specification of an output serial number in an ANS DCB forces processing to be done only on a tape already having that serial number; tapes mounted as IN may not be written; and tapes mounted as other than IN must have a write ring. (See "semiprotective mode".)

pseudo file name a symbolic name used to identify a logical device in a user's program.

public library — a set of library routines declared during system generation to be public (i.e., to be used in common by all concurrent users).

rapid access data (RAD) storage system — a secondary storage system of rotating memory.

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 the single copy of the program's instructions has a separately mapped 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.

relocating loader — a program capable of loading one or more object modules and linking them to form an executable program.

remote processing — an extension of the symbiont system that provides flexible communication between CP-V and a variety of remote terminals.

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

response time — the time between the completion of terminal input and the first program activation.

scheduler — a monitor routine that controls the initiation and termination of all jobs, job steps, and time slice quanta.

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

segment loader — a monitor routine that loads overlay segments from disk storage at execution time.

semi-protective mode — a mode of tape protection in which a warning is posted to the operator when an ANS DCB attempts output on a non-ANS tape or an unexpired ANS tape, when a non-ANS DCB attempts output on an unexpired ANS tape, or when a tape mounted as INOUT has no write ring. The operator can authorize the overwriting of the tape or the override of INOUT through a key-in (OVER and READ). ANS tapes may be initialized by the Label processor or may be given labels as the result of an operator key-in; tape serial number specification is allowed at the operator's console; and specification of an output serial number in an ANS DCB forces processing to be done only on a tape already having that serial number unless the operator authorizes an overwrite. (See "protective mode".)

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

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

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

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

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

star file — a file created by the system containing temporary user context. The system may create up to five star files for each batch job or on-line session. These files are transparent to the user, are not cataloged in the File Directory, and always cease to exist at the end of the batch job or on-line session.

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

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 reigster 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

rountines during program execution or to allow entry to the program in cases of traps, breaks, etc. The TCB is job-step assiciated.

TEXT format an EBCDIC character string that begins and ends at word boundaries. The character string is left-justified in the field and is paddedwith trailing blanks.

TEXTC format a character string preceded by a byte that contains the number of characters in the character string. (The count byte does not include itself in the count.) For many CP-V functions, the character string need not consist of printable characters.

TSS temp stack a push-down stack established by the monitor for use by an executing program (unless NOTCB was specified for the load module.)

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

user-identification banner an identifying prescript and/or postscript for output through a logical device stream. Line printer and card punch device streams are given user-identification banners; other device streams are not. A punch stream receives a one-card prescript. A printer stream receives a one-page prescript and, provided that it has not been given a FORM name, a one-page post-script. A prescript card and each line of a prescript page contains the user's account number and name.

maintains complete control over the job stack on secondary storage. Jobs can be suspended or initiated on a priority basis.

Rapid access data (RAD) and disk pack (DP) storage devices are used for secondary storage. Secondary storage management is essential to efficient operation of the monitor, since such storage is fully exploited in various ways. It is used for system storage to overlay portions of the monitor, minimizing core memory residency. Service processors (compilers, assemblers, etc.) are contained on secondary storage for immediate access and they, too, capitalize on rapid overlay techniques to minimize core memory requirements at execution time. Scratch storage for service processors and user programs is available on secondary storage. Finally, the secondary storage accommodates permanent and temporary user files.

User files may be stored on public RAD or disk packs or on private disk packs or magnetic tape. Three file structures are available: random (direct), consecutive, and key-indexed (indexed-sequential). Access may be either direct (keyed) or sequential. Programs coded to access the simpler consecutive files may correctly access the more complex keyed files sequentially without program change. Files are protected from unauthorized use by passwords and by explicit lists of users authorized to read or to update them.

User programs can avail themselves of the secondary storage and the overlay service of the monitor. With these facilities, user programs that require more operating core memory storage than is physically available can be easily segmented and controlled so that only part occupies available core memory at any one time. The monitor accepts the overlay structure of the user's program and ensures proper sequencing and transfer ng of program elements. It also detects inconsistencies in the logical overlay structure and log them as a diagnostic message to the user.

The monitor provides for complete accounting of user job activity on the computer. Because of the system's multiusage capability, the accounting information indicates both elapsed time and actual machine facility utilization of each job.

The monitor provides job accounting and validation of each user's job activity:

- Validity or authorization checks are made on the user's name and account number combination. Jobs are aborted where the name and account number are not previously validated by the installation manager.
- A discrete accounting record is written at the termination of each batch job.
- Standard accounting can be supplemented by the user supplying initiation and termination routines for a job.

The monitor's loader function relocates user programs into the currently available core memory space, satisfies all library subroutine references, and links all program elements called for by the user. In addition, run-time debugging calls are recognized and established for the binary programs.

The full multiuse capability of the monitor provides for five concurrent modes of operation:

1. Batch processing.
2. Interactive time-shared processing.
3. Remote processing.
4. Real-time processing.
5. Transaction processing.

BATCH PROCESSING

Batch jobs may be submitted to the batch job stream through the central site card reader, through an on-line terminal (using the Batch processor), or through remote processing. Batch processing facilities are described in this document.

TIME-SHARED PROCESSING

CP-V allows multiple on-line terminal users to concurrently create, debug, and execute programs. 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/TS Reference Manual, 90 09 07.)

REMOTE PROCESSING

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, or 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 users 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. Users 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.)

PROCESSORS

The CP-V system is illustrated in Figure 2 at two levels. The upper level represents the monitor and its various routines. The lower level lists the various processors. These processors are described in the following paragraphs.

COMMAND PROCESSORS

The four processors in this group are: LOGON/LOGOFF, EASY, TEL, and CCI. The first of these processors is available to on-line and batch users, the second and third are available to on-line users only, and the last is available to batch users only.

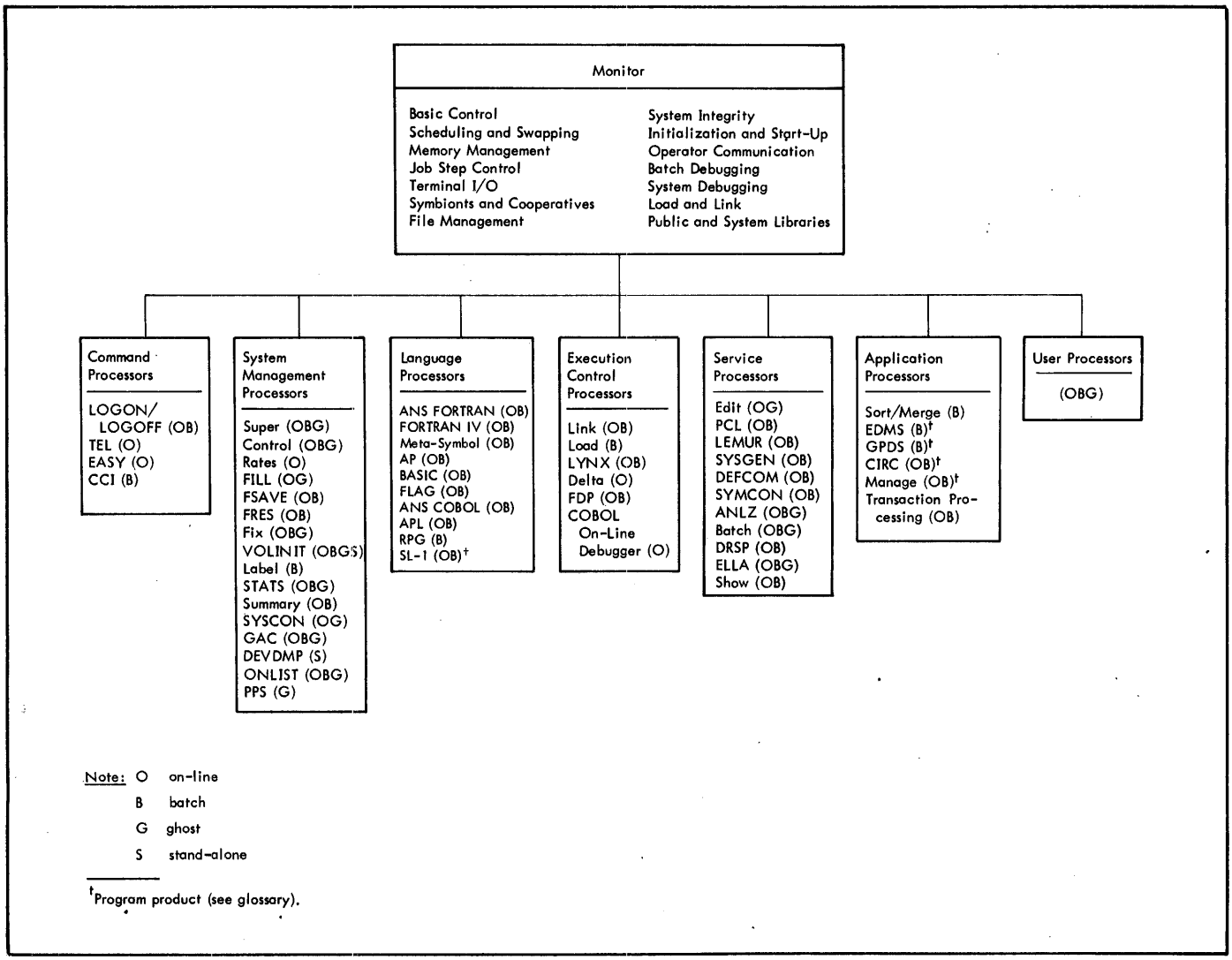


Figure 2. CP-V Operating System

LOGON/LOGOFF

LOGON admits on-line users to the system. LOGOFF disconnects a user (on-line or batch) from the system and does the final cleanup and accounting.

EASY

EASY is a shared processor that enables the user to create, edit, execute, 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 CP-V. (Reference: EASY/LN, OPS Reference Manual, 90 18 73.)

TERMINAL EXECUTIVE LANGUAGE

TEL is the default command processor for time-sharing and serves as the terminal user's interface to the various services of CP-V. TEL is functionally equivalent to the batch mode Control Command Interpreter. Some of the functions performed by TEL are:

1. Calling user programs and system processors.
2. Changing the log-on password.
3. Assigning I/O devices and DCB parameters.
4. Requesting extended memory mode.
5. Determining on-line user status.
6. Changing terminal platen size.
7. Sending messages to the operator.
8. Logging off.

(Reference: CP-V/TS Reference Manual, 90 09 07.)

CONTROL COMMAND INTERPRETER

The Control Command Interpreter is the batch counterpart of TEL. It provides the batch user with control over the processing of batch programs just as TEL provides on-line users with control over the processing of on-line programs.

LANGUAGE PROCESSORS

Language processors translate high-level source code into machine object code. Nine processors of special importance are described below. Eight of these (Extended FORTRAN IV, Meta-Symbol, BASIC, FLAG, ANS COBOL, APL, Manage, and SL-1) can be used in both on-line and batch modes. The other one (RPG) can be used in batch mode only.

ANS FORTRAN

The ANS FORTRAN compiler is compatible with most features of the forthcomong (new) American National Standard Institute 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

(Reference: ANS FORTRAN/LN Reference Manual, 90 32 00, and the ANS FORTRAN/OPS Reference Manual, 90 32 01.)

XEROX EXTENDED FORTRAN IV

The Xerox Extended FORTRAN IV language processor 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 235 subprograms and is available in a reentrant version. Both the compiler and run-time library are reentrant programs that are shared among all concurrent users to reduce the utilization of critical core resources. (Reference: Extended FORTRAN IV/LN Reference Manual, 90 09 56, and Extended FORTRAN IV/OPS Reference Manual, 90 11 43.)

META-SYMBOL

Meta-Symbol is a procedure-oriented macro assembler. It has services that are available only in sophisticated macro assemblers and a number of special features designed to permit the user to exercise dynamic control over the parametric environment of assembly. It provides users with a highly flexible language with which to make full use of the available Xerox 560 and Sigma hardware capabilities. (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 based on 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 applications of a computational nature.

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. (Reference: BASIC/LN, OPS Reference Manual, 90 15 46.)

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.)

ANS COBOL

The Xerox ANS COBOL compiler offers the user a powerful and convenient programming language facility for the implementation of business or commercial applications. The language specifications fully conform to the ANSI standard for the various functional processing modules. The compiler's design takes full advantage of the Xerox 560 and Sigma unique hardware features, resulting in rapid compilation of source code, rapid execution of the resulting object code, and the generation of compact programs. The result is a highly efficient programming system requiring a minimum amount of storage. (Reference: ANS COBOL/LN Reference Manual, 90 15 00.)

RPG

Xerox RPG (Report Program Generator) is a convenient means of preparing reports from information available in computer-readable forms, such as punched cards, magnetic tape, and magnetic disks. In addition, it is a means of establishing and updating files of information, usually in conjunction with preparation of reports.

RPG provides its capabilities through generation (compilation) of object programs, each of which is tailored to produce a different set of reporting results and/or file processing desired by the user. The RPG object programs are capable of accepting input data, retrieving data from existing files, performing calculations, changing formats of data, updating existing files, creating new files, comparing data values to one another and to specified constants to determine appropriate handling, using user-defined processing subroutines, using system library subroutines, and printing reports derived from the input and file data.

Xerox RPG has several advantages over the more traditional method of writing object programs in a symbolic programming language. The RPG language is oriented toward the user's problem, describing reporting requirements, rather than toward the mechanics and manipulations of computer usage. The language and specification techniques are easily learned. A user can become proficient in RPG after writing only a few programs, whereas an equal facility in symbolic programming would require considerable experience. (Reference: RPG/Reference Manual, 90 19 99.)

APL

Xerox APL (A Programming Language) is a processor and programming language based on Kenneth Iverson's APL. It is an interpretive, time-sharing, problem-solving language. As an interpretive language, APL interprets each line of input as it is entered and produces code that is immediately executed. As a problem-solving language, APL requires minimal computer programming knowledge.

Because APL is powerful, concise, easy to learn, and easy to use, it is widely used by universities, engineers, statisticians, and businessmen. One of APL's major strengths is its ability to manipulate vectors and multidimensional arrays as easily as it does scalar values.

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.)

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 560 and Sigma 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.)

EXECUTION CONTROL PROCESSORS

Processors in this group control the execution of object programs. Load can be used in batch mode only. Link and FDP can be used in either batch or on-line mode. Delta and the COBOL On-Line Debugger can only be used in the on-line mode.

LINK

Link is a one-pass linking loader that constructs a single entity called a load module, which is an executable program formed from relocatable object modules (ROMs). Link is designed to make full use of mapping hardware. It is not an overlay loader. If the need for an overlay loader exists, the overlay loader (Load) must be called by entering the job in the batch stream. (Reference: CP-V/TS Reference Manual, 90 09 07, and Chapter 6 of this manual.)

LOAD

Load is a two-pass overlay loader. The first pass processes

1. All relocatable object modules (ROMs).
2. Protection types and sizes for control and dummy sections of the ROMs.
3. Expressions for definitions and references (primary, secondary, and forward references).

The second pass forms the actual core image and its relocation dictionary. (Reference: Chapter 6.)

LYNX

LYNX is a load processor that is available in both the on-line and batch modes. LYNX has most of the capabilities of the Load 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 processor. (Reference: Chapter 6.)

DELTA

Delta is designed to aid in the on-line debugging of programs at the assembly-language or machine-language levels. It operates on object programs and tables of internal and global symbols used by the programs 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. (Reference: CP-V/TS Reference Manual, 90 09 07.)

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, repair program errors.

The debugger can be used in batch and on-line modes. 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 operations, the debugging commands are entered through the terminal keyboard when requested by the debugger. Such requests are made when execution starts, stops, or restarts. The debugger normally has control of such stops. (Reference: FDP/ 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 the 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.)

SERVICE PROCESSORS

The processors in this group perform such functions as editing and transferring data between RAD storage and central site peripheral devices. Four of the processors (SYSGEN, ANLZ, DRSP, and ELLA) are for system management or system programming use only and are not described in the following paragraphs.

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 disk 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. (Reference: CP-V/TS Reference Manual, 90 09 07.)

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 pack, and RAD storage.

PCL is controlled by single-line commands supplied through on-line terminal input 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 9.)

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 6 and CP-V/TS Reference Manual, 90 09 07.)

DEFCOM

DEFCOM makes the DEFs and their associated values in one load module available to another load module. It accomplishes this by using a load module as input and by producing another load module that contains only the DEFs and DEF values from the input module. The resultant load module of DEFs can then be combined with other load modules. DEFCOM is used extensively in constructing the CP-V monitor and the shared run-time libraries. (Reference: Chapter 9.)

SYMCON

The Symbol Control Processor (SYMCON) provides a means of controlling external symbols in a load module and of building a global symbol table. Its primary function is to give the programmer a means of preventing double definitions of external symbols. It may also be used to reduce the number of external symbols. For example, if certain load modules cannot be combined because their control tables are too large, the tables may be reduced in size by deleting all but essential external symbols. (Reference: Chapter 9.)

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. A file 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 on 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 or batch 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. (Reference: Chapter 9.)

APPLICATION PROCESSORS

The processors in this group perform such functions as sorting, simulation, and data management. They all operate in the batch mode only.

SORT/MERGE

The Xerox Sort/Merge processor provides the user with a fast, highly efficient method of sequencing a nonordered file. Sort may be called as a subroutine from within a user's program or as a batch processing job by control cards. It is designed to operate efficiently in a minimum hardware environment. Sorting can take place on from one to sixteen keys and each individual key field may be sorted in ascending or descending sequence. The sorting technique used is that of replacement selection tournament and offers the user the flexibility of changing the blocking and logical record lengths in explicitly structured files to different values in the output file. (Reference: Sort-Merge/Reference Manual, 90 11 99.)

GPDS (PROGRAM PRODUCT)

The General Purpose Discrete Simulator provides engineers and administrators, whose programming experience is minimal, with a system for experimenting with and evaluating system methods, processes, and designs. Providing a means for developing a broad range of simulation models, it allows organizing, modeling, and analyzing the structure of a system, observing the flow of traffic, etc. (Reference: GPDS/ Reference Manual, 90 17 58.)

EDMS (PROGRAM PRODUCT)

EDMS is a generalized data management system that enables the user to create an integrated data base. It is designed to be used with COBOL, FORTRAN, and Meta-Symbol processors. It simplifies programming by performing most of the I/O logic and data base management for the application programmer. (Reference: EDMS/Reference Manual, 90 30 12.)

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.

Manage consists of four subprograms: Dictionary, Fileup, Retrieve, and Report. The Dictionary subprogram is a data file and is the central control element in the Manage system. It consists of definitions and control and formatting parameters that precisely describe the characteristics of a data file. The Fileup subprogram initially creates and then maintains a data file. The Retrieve subprogram extracts data from a data base file according to user-specified criteria. The Report subprogram automatically prepares printed reports from data extracted by the Manage retrieval program. (Reference: Manage/Reference Manual, 90 16 10.)

TRANSACTION PROCESSING

Transaction Processing is designed for applications that require the entry and processing of on-line transactions. It is a collection of general-purpose components and supporting monitor services available under the CP-V operating system. Transaction Processing (TP) enables business to move from cyclic batch processing to remote on-line operations, where transactions are entered directly from their point of origin. The Xerox system consists of

- The CP-V monitor and standard processors such as COBOL, Meta-Symbol, and FORTRAN.
- Terminal Interface Controller.
- Utility processors that create files for external system control.
- Transaction Processing Controller.
- Extended Data Management System (EDMS).

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

CIRC (PROGRAM PRODUCT)

CIRC is a set of three computer programs for electronic circuit analysis on Xerox 560 and Sigma 5-9 computers: 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 in any of three modes of operation: 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 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.)

USER PROCESSORS

Users may write their own processors and add them to CP-V or replace CP-V processors. The rules governing the creation and modification of processors are described in the CP-V/SP Reference Manual, 90 31 13.

MONITOR

The monitor responds to the moment-by-moment requirements of controlling machine operation, switching between programs requiring service, and providing services at the explicit request of the user's program. The monitor processes that perform these functions are listed below:

1. Basic Control.
2. Scheduling and Swapping.
3. Memory Management.
4. File Management.
5. Multibatch Job Scheduling.
6. Job Step Control.
7. Terminal I/O Handling.
8. Symbionts and Cooperatives.
9. System Integrity.
10. Initialization and Start-Up.
11. Operator Communications.
12. Batch Debugging.
13. Load-and-Link.
14. System Debugging.

The basic control system is an I/O interrupt service and handling routine. It includes trap and interrupt handlers, routines that place requests for I/O in a queue, and basic device I/O handling routines.

The scheduling and swapping module makes the decision to swap, selects the users to swap in and out, sets up the I/O command chains for swap transfers, and selects the next user(s) for execution. It also ensures that any associated, but not currently resident, shared processors are brought in with each user. Special algorithms control I/O scheduling and the balance of machine use between on-line and batch.

The memory management module controls the use of core and disk storage. Specifically, it controls the allocation of physical core memory, maintains the map and access images for each user, services the "get" and "free" service calls for memory pages, and manages the swapping disk space.

File management routines control the content and access to physical files of information. These routines perform such functions as indexing, blocking and deblocking, managing of pools of granules on RADs and disk packs, labeling, label checking and positioning of magnetic tape, formatting for printer and card equipment, and controlling access to and simultaneous use of a hierarchy of files.

The multi-batch job scheduling routines select jobs to be run from the waiting input queue depending on priority, position in queue, and resources available within partitions defined by the installation.

Job step control routines are entered between major segments of a job or an on-line session. They perform the monitor functions required between job steps such as

1. Processing error, exit, and abort CALs.
2. Handling monitor aborts.
3. Processing interpretive exits to associate shared processors or to load program modules.
4. Merging DCB assignments for execution.

Terminal I/O handling routines pe orm read-write buffering and external interrupt handling for I/O directed to user terminals. These routin s also translate character codes, insert page headers and VFC control cha acters, simulate tabs, and perform other formatting tasks.

Symbiont routines transfer data from the card reader to disk storage and from disk storage to the card punch or line printer. Input cooperatives intercept card read commands in user programs and transfer data from disk storage where it has been stored by the symbiont routines. Output cooperative routines intercept output directed from a user program to a line printer or card punch and transfer the data to disk storage.

System integrity facilities provide error detection and recovery capabilities. This includes security to user files and automatic high-speed restart in case of system failure. Sufficient information is recorded to isolate errors and failures caused by hardware or software.

Initialization and start-up routines are stored on tape and are booted into core storage. After they are in core, they load the monitor root into core and turn control over to the root. The monitor root then completes the initialization of the monitor by starting and running the program called GHOST1 which completes the patching of the system and the initialization of the swapping disk and hardware.

Operator communication routines provide for communication between the monitor and the operator. They transmit messages to the operator and process key-ins received from the operator.

Batch debugging routines provide batch programs with debugging capability through the use of procedure calls. Any batch program may take a snapshot dump of a specified segment of memory, either on an unconditional or a conditional basis.

System debugging routines provide debugging services to system programmers.

Load-and-link routines give batch programs three types of loading and linking capability. Through the use of procedure calls, a batch program may:

1. Load an overlay segment into core storage.
2. Store the calling program on disk storage, load the called program into core storage, and transfer control to the called program.
3. Load a program into core storage, transfer control to the called program, and release the core area used by the calling program.

CP-V has two FORTRAN libraries. One is a public library and the other is a system library. In the standard release of CP-V, the public library contains two sets of programs. One set (P1) contains a useful set of Extended FORTRAN IV runtime library routines, the other set (P0) contains P1 and the FORTRAN Debug package. These two libraries are so constructed that a single copy is shared among all concurrent users. The system library contains a collection of routines that are less frequently used than the public library routines. They are in library load module form and are loaded only with programs that reference them.

SYSTEM COMMANDS

Control Command Definitions

Control Command	Definition
ASSIGN	Relates an operational label or a pseudo file name to a device. A pseudo file name may be assigned to an operational label.
INCL	Directs the overlay loader to allocate public library routines in a segment.

Control Command	Definition
JOB	Signals the completion of a previous job and the beginning of a new one. All jobs must have a JOB control command.
LDEV	Attaches an information stream to a physical device (identified by a logical device stream name) and defines attributes of the physical device.
LIMIT	Estimates the system job parameters (i.e., number of pages of output, number of cards to be output, time job is to run, etc.) for the job.
LINK	Directs the Link loader to form a relocatable load module and enters it in the user's element file if a load module name is specified.
LOAD	Directs the Load loader to form a relocatable load module and enters it in the user's element file if a load module name is specified.
MESSAGE	Causes the specified message to be typed to the operator at the time that it is encountered by the system.
OLAY	Equivalent to LOAD control command.
OVERLAY	Equivalent to LOAD control command.
POOL	Tells the monitor the number of core pages to be allocated for buffers and tables associated with I/O operations.
processor name	Tells the monitor which processor is to operate and what options the processor is to execute.
PTREE	Tells the monitor that a tree control command is to be read from the user's file.
RUN	Tells the monitor to transfer control to the user's program.
SET	Performs the same function as the ASSIGN control command.
STEP	Provides conditional execution of job steps.
TITLE	Causes the specified title to be output at the beginning of each logical page of output on the LO device.
TREE	Specifies the symbolic representation of the overlay structure.
XEQ	Initiates processing of control commands from a command file.

Debug Control	Definition
AND	Causes a specified test to be made at a specified location. Only if the condition is true and the specified test identifier is set does it remain set; otherwise, it is reset (see SNAPC control command).
COUNT	Specifies the range and the steps within the range where the test identifier is set (see SNAPC control command).
IF	Causes a specified test to be made at a specified location. The specified test identifier is set only if the condition is true; otherwise, the identifier is reset or remains reset (see SNAPC control command).
MODIFY	Allows the user to insert a modification into a user program before execution.
OR	Causes a specified test to be made at a specified location (if a specified test identifier is reset). If the condition is true, the specified test identifier is set; otherwise, it remains unchanged (see SNAPC control command).
PMD	Causes the monitor to dump the selected area of memory, in hexadecimal form, if an error occurs during execution.
PMDE	Causes the monitor to dump (in addition to the information obtainable by PMD) the PSD, registers, etc.
PMDI	Causes the monitor to dump the selected area of memory, in hexadecimal form, regardless of whether errors have been detected.

Debug Control	Definition
SNAP	Causes a snapshot of the specified memory and registers at the location specified to be performed.
SNAPC	Causes a snapshot of the specified memory and registers at the location specified to be performed only when the specified test identifier is set.
SWITCH	Produces the initial settings of the pseudo sense switches.

Input Control	Definition
BCD	Serves as a terminator for a binary input source.
BIN	Informs the monitor that the information to follow is binary.
DATA	Informs the monitor that the information to follow is data.
EOD	Causes an end-of-data abnormal return to the monitor, indicating the end of a series of data records.
FIN	Specifies the end of a stack of jobs.
NCTL	Allows noncontrol input files to be entered from the card reader.
PFIL	Position n files on unlabeled magnetic tape.
REW	Rewinds the specified tape.
WEOF	Writes a physical end-of-file on magnetic tape.

Procedure Definitions

Procedure	Definition
M:AND	Causes a specified test to be made at a specified location. Only if the condition is true and the specified test identifier is set does it remain set; otherwise, it is reset or remains reset (see M:SNAPC procedure).
M:CHECK	Checks type of I/O completion.
M:CHECKECB	Checks for the completion of an event or a set of events.
M:CLOSE	Terminates all I/O associated with a given Data Control Block (DCB).
M:COUNT	Specifies the range and the steps within the range where a specified test identifier is set (see M:SNAPC procedure).
M:CT	Changes terminal type. (This procedure is for on-line use only and is described in the CP-V/TS Reference Manual, 90 09 07.)
M:CVM	Changes Virtual Map.
M:CVOL	Causes the control program to advance to the next volume of a data set before the physical end of the current volume is detected. This call is meaningful only for tapes.
M:DCB	Defines a Data Control Block.
M:DELREC	Specifies that a data record is to be deleted from the file.
M:DEQ	Dequeues resources.
M:DEVICE	Allows the user to set special device procedures.
M:DISPLAY	Reports system load parameters.
M:ENQ	Enqueues resources.
M:ERR	Returns control to the monitor and the monitor honors all PMD and ASSIGN control commands while ignoring all other control commands until it encounters a FIN, JOB, or a processor name control command.
M:EXIT	Returns control to the monitor which then honors all output control commands of the form IPMDI.
M:EXU	Requests that the monitor execute a privileged instruction for the user.

Procedure	Definition
M:FP	Frees page of main storage owned by a given task.
M:FCP	Frees common page.
M:FVP	Frees virtual page.
M:GCP	Gets common pages.
M:GDDL	Gets dynamic data limits.
M:GL	Gets common limits.
M:GP	Allocates pages of main storage to the requesting task.
M:GVP	Gets virtual page.
M:IF	Causes a specified test to be made at a specified location. Only if the specified test condition is true is the test identifier set; otherwise, it is reset or remains reset (see M:SNAPC procedure).
M:INT	Connects a console interrupt.
M:JOB	Inserts a file into or deletes a file from an existing symbiont file.
M:KEYIN	Writes the specified message to the operator on the operator's console and returns the operator's reply to the program issuing the procedure.
M:LDEV	Attaches an information stream to a physical device (identified by a logical device name) and defines attributes of the physical device.
M:LDTRC	Loads the specified load module if a reenterable copy is not available in memory, deletes the calling module, and transfers control to the loaded load module.
M:LINK	Loads the specified load module if a reenterable copy is not available in memory and links to it.
M:MASTER	Allows a special processor to operate in the master (and master protected) mode.
M:MERC	Allows the user to have the monitor process any system abnormal or error code, overriding an ABN or ERR exit.
M:MESSAGE	Writes the specified message on the Operator Console.
M:MOVE	Copies a file, record by record.
M:OPEN	Causes the specified file associated with the specified DCB to be opened for use.
M:OR	Causes a specified test to be made at a specified location (if a specified test identifier is reset). If the condition is true, the specified test identifier is set; otherwise, it remains unchanged (see M:SNAPC procedure).
M:PC	Sets prompt character. (This procedure is for on-line use only and is described in the CP-V/TS Reference Manual, 90 09 07.)
M:PT	Allows the user to generate FPTs in either protected or unprotected storage.
M:PFIL	Causes the specified tape to be positioned past the number of end-of-files specified and in the direction specified.
M:PRECORD	Causes the tape specified by the DCB to be positioned in the direction specified by the specified number of records.
M:PRINT	Writes the specified message on the listing log (LL) output media.
M:RAMR	Reads the assign/merge record.
M:READ	Causes the next data record to be read into the location specified by the user.
M:REW	Rewinds a tape specified by the DCB.

Procedure	Definition
M:SEGLD	Loads a specified overlay segment into memory.
M:SETDCB	Sets error or abnormal addresses in a specified DCB.
M:SLAVE	Allows any master mode program to return to the slave mode.
M:SMPRT	Sets memory protection.
M:SNAP	Causes a snapshot of the registers and memory specified to be performed.
M:SNAPC	Causes a snapshot of the registers and memory specified to be performed if the specified test identifier is set. Whether the test identifier is set or not is dependent on the M:IF, M:AND, M:OR, and M:COUNT procedures.
M:STIMER	Sets the interval timer with a specified interval.
M:STRAP	Simulates a trap.
M:SYS	Allows special processors to use privileged services.
M:TFILE	Causes a specified DCB to be closed, on return to the user's program, and the associated file to be registered as a scratch file.
M:TIME	Communicates the time of day and the current date to the executing program.
M:TRAP	Sets and resets the traps to go to a user routine or the standard system routine. Also sets and resets the maskable traps.
M:TRTN	Restores control to the executing program from a trap or timer routine.
M:TRUNC	Causes the blocking buffer reserved for a specified DCB to be released.
M:TTIMER	Gives the time remaining in the interval that was previously set by M:STIMER procedure and optionally cancels the interval in effect.
M:TYPE	Writes the specified message to the operator on the operator's console.
M:WAIT	Suspends program.
M:WAMR	Writes the assign/merge record.
M:WEOF	Writes an end-of-file mark on an unlabeled tape specified by the DCB.
M:WRITE	Causes the contents of a specified buffer to be transmitted to an output device or file.
M:XCON	Allows a program to regain control after termination.
M:XXX	Causes the monitor to terminate the job and not honor any further commands until it reads another JOB or FIN control command.

2. FILES AND FILE USAGE

INTRODUCTION

A general understanding of files and the way that the monitor deals with them will help the user to obtain the high level of performance available.

A file is an organized collection of information. This collection of information may consist of one or more programs, one or more sets of data, or some combination of programs and data. Under CP-V, a user always accesses files through the monitor — never directly. An option does exist, however, that allows a user to deal with a file (e.g., a non-standard set of data on an unlabeled magnetic tape) as though he were accessing it directly.

A file has one base name but may have other names synonymous with it. Information is retrieved from a file by specifying the file name (or its synonym), its password and account, and the desired record within the file.

The monitor maintains a directory of accounts having files maintained between jobs. This is called an Account Directory, and contains, with each account number, an address of a directory of files (termed a File Directory) for that account. A File Directory contains, with each file name, an address of a table containing file attributes and disk locations for that file. The table is called a File Information Table. To summarize, the monitor has a single Account Directory, which in turn points to a File Directory for each account. Each File Directory, in turn, points to a File Information Table (FIT) for each file.†

Each file has associated with it (in the FIT) information controlling who may access the file and how it may be accessed. This information can include both a password and a list of which accounts may read or update the file. To access a file, a user must be running under an account which is authorized to access the file and provide the proper password. In addition to access control information, the FIT also contains the file's creation date, date of last modification, date of last access, and expiration date.

A file may be shared among several users if none of them updates the file or attempts to replace the file. A job cannot, however, create a file in an account other than its own.

Three prime concerns of the user in regard to files are

1. File organization — the way in which a file is logically constructed.
2. The methods that a user can apply to find, extract, insert or delete information from a file.
3. The way that a file is stored on specific devices.

†For each batch job and on-line session, the system may create up to five special files containing temporary user context. These files, known as "star files", are transparent to the user, are not cataloged in the File Directory, and always cease to exist at the end of the batch job or on-line session.

FILE ORGANIZATION

Each file is identified by a file name. In addition, access to each file is controlled by the account number of the user who created it and a password, if he chose to include one.

The information contained in a file may be structured in one of three ways. It may be a keyed, consecutive, or random file.

KEYED FILES

Keyed files are those in which each record has an identifying key associated with it. A key consists of a byte string, the first byte of which states the number of bytes in the string. The contents of each byte may be a binary number or a character. A key may consist of up to 31 characters.

As the file is being created, a master index is also created with an entry for each keyed record in the file. The entry contains such information as the key, disk address of the record, size of the record, and position of the record within the blocking buffer.

The records are automatically packed into blocking buffers with the last portion of the last record extending into another buffer as necessary. If the record is large, it is written directly from the user's area instead of being packed into a buffer. Keyed files may be accessed by direct or sequential access.

Keyed files have a multilevel index structure. A multilevel index structure is a collection of hierarchical levels of index blocks, where the entries in a higher level point to index blocks at the next lower level and the entries in the lowest level (called level 0) point to data records. This is best illustrated by an example as shown in Figure 3. The multilevel structure is initially built during a CLOSE if a keyed file has more than three level 0 index blocks.

In the example shown in Figure 3, the keyed file has

- 31,150 records and the keys at level 0 point to these data records. Based on an 11-byte maximum key length, there are 80 keys in each level 0 block and 127 keys in each higher-level block.
- 390 index blocks at level 0, four index blocks at level 1, and one index block at level 2. The next higher-level is built if the last level has more than three index blocks.

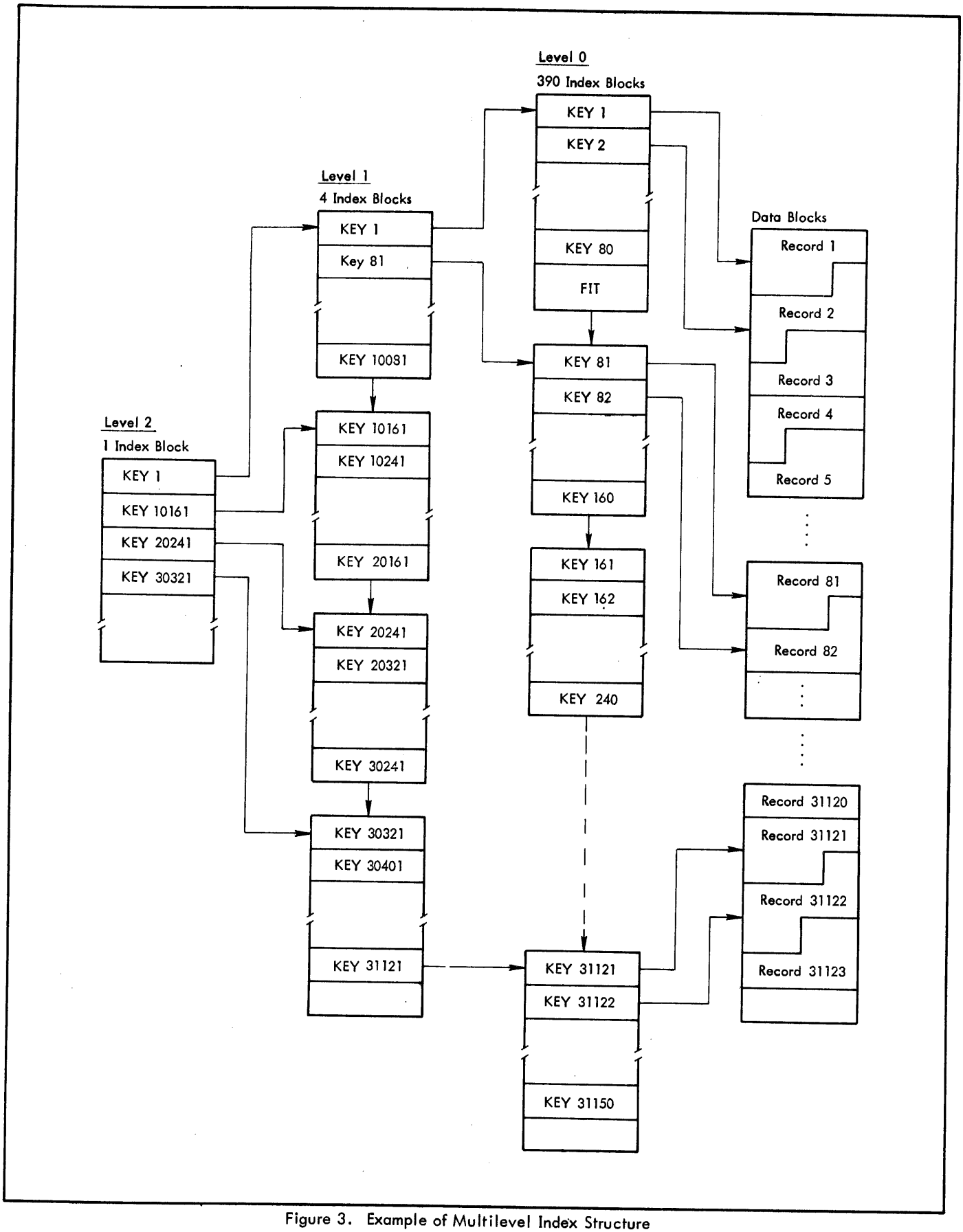


Figure 3. Example of Multilevel Index Structure

Each entry in a higher-level index block contains the disk address of an index block at the next lower level, and the key of the first key in that block.

The multilevel index structure can considerably speed up the direct access of a large keyed file, at only a small cost of secondary storage space. Since the keys are ordered in ascending sequence, at most it would take three index block accesses to locate a data record as shown in the example. Without the higher-level structure, it could take up to 390 index block accesses.

The user has control over the initial creation of the multilevel index structure and he can specify when and if the higher-level structure should be rebuilt. This can be specified by using the NEWX option on the ASSIGN control command or the M:OPEN and M:DCB procedures.

The space required to hold a given file can be estimated by applying the following rules:

Data Blocks

1. Each data block contains 2048 bytes.

2. Each data granule contains one data block.

3. Each data block is compact, except that all records start on word boundaries.

4. Each record or record segment (if a record resides in more than one data block) has a level 0 index entry associated with it.

Level 0 Index Blocks

1. Each index block contains 2048 bytes.

2. Each index block is compact except that 12 bytes are preempted and spare space may be reserved at user request.

3. Each index entry occupies key size (KEYM) plus 14 bytes.

Higher-Level Index Blocks

1. Each higher-level index block contains 2048 bytes.

2. Each higher-level index block is compact except that 12 bytes are reserved.

3. Each higher-level index entry occupies KEYM plus five bytes.

The following formulas can be used to estimate the disk space requirement of keyed and consecutive files. The first formula calculates the number of keys per index block at level 0 (KL). The second formula calculates the total disk storage in granules.

N = number of records in the file.

R = record size in bytes.

SS = spare space expressed as a decimal number (e.g., 20% = .2).

K = key length in bytes.

$$KL = \frac{2048*(1-SS)-12}{14+K}$$

$$\text{Total disk storage (in granules)} = \frac{N*R}{2048} + \left(\frac{N}{KL}\right) + \left(\frac{N}{KL}\right)\left(\frac{5+K}{2036}\right)$$

Note that:

$\frac{N*R}{2048}$ is the number of data granules.

$\left(\frac{N}{KL}\right)$ is the number of Level 0 Index granules.

$\left(\frac{N}{KL}\right)\left(\frac{5+K}{2036}\right)$ is the number of Level 1 Index granules.

The following two examples show the cost to build the multilevel index structure, i.e., disk accesses to build it and disk storage required to contain it, and the saving in time when accessing it.

Example 1

Number of records	N = 40,000
Record size	R = 60 bytes
Key size (KEYM)	K = 3 bytes

Spare space SS = 10% (= .1)

Keys/Level 0 Index block (KL) = $\frac{2048*(1-SS)-12}{14+K}$ =

$\frac{2048*(1-.1)-12}{14+3}$ = 108

Data granules = $\frac{N*R}{2048} = \frac{40{,}000*60}{2048}$ = 1172

Level 0 Index granules = $\left(\frac{N}{KL}\right) = \left(\frac{40{,}000}{108}\right)$ = 371

Level 1 Index granules = $\left(\frac{N}{KL}\right)\left(\frac{5+K}{2036}\right)$ =

$\left(\frac{40{,}000}{108}\right)\left(\frac{5+3}{2036}\right)$ = 2

This file requires a total of 1545 (1172 + 371 + 2) granules of storage of which two are required to store the multilevel index. It would cost 373 disk accesses to build the structure when the file is closed. With the multilevel structure, each random record fetch requires 3-1/2 device accesses, whereas without it each fetch would be 186 accesses.

Example 2

Number of data records = maximum for each device (see below).

Record size R = 1024 bytes

Key size (KEYM) K = 15 bytes

Spare space SS = 0

Keys/Level 0 Index block (KL) = $\frac{2048*(1-SS)-12}{14+K}$ =

$\frac{2048*(1-0)-12}{14+15}$ = 70

Level 0 Index granules = $\left(\frac{N}{KL}\right) = \left(\frac{N}{70}\right)$

Level 1 Index granules = $\left(\frac{N}{KL}\right)\left(\frac{5+K}{2036}\right)$ =

$\left(\frac{N}{70}\right)\left(\frac{5+15}{2036}\right) = \frac{20N}{142{,}520}$

Item	7232 RAD	7242 Disk Packs
Number of data records.	6144	24000
Level 0 granules.	88	343
Level 1 granules.	1	4
Level 2 granules.		1

The cost to build the multilevel structure in the 7242 example is 348 device accesses. Without the multilevel structure a random fetch could take 344 device accesses in the worst case; with it, four accesses are required.

CONSECUTIVE FILES

Consecutive files are files whose records are organized in a consecutive manner; i.e., the user is aware of no identifying keys associated with the records. The records may only be accessed sequentially.

The principal benefit of consecutive files to system operation is a reduction in the number of granules required for the files on disk and RAD and a consequent reduction in the time required to process the files. This is most dramatic for files containing short records. A file of 1000 eighty-character records requires 49 granules for keyed files. For consecutive files, this requirement is reduced to 42, a savings of over 14 percent. For twenty-character records, the requirement of 1000 records drops from 19 to 12 granules, a reduction of over 36 percent. For large records there is still a small reduction. For example, 1000 2048-byte records requires 1002 granules for consecutive files as opposed to 1009 for keyed files. Traversing a file of 1000 2048-byte records requires 1002 disk reads for consecutive files as opposed to 1017 disk reads for keyed files. For 1000 eighty-character records, the reduction is from 57 reads to 42 reads (over 26 percent). For 1000 twenty-character records, the drop is from 27 to 12 reads (a 55 percent reduction).

All position operations for consecutive files is done without I/O. Positioning operations are PRECORD, PFIL, and OPEN with extension. The positioning is only effected when a data transfer operation is about to take place. At that time, there will be three known points in the file that can be used as a starting point (beginning of file, end of file, and the position reflected by the DCB). The starting position chosen will be the one that requires the fewest record skips to be made.

For consecutive files, the FIT is maintained in words 4 through 83 of the first granule of the file.

RANDOM FILES

Random files provide an organization for those users desiring to manage their own files or who do not wish to incur the overhead imposed by system file management.

Random organization differs from keyed and consecutive organization as follows:

1. A random file is simply a collection of contiguous granules on the specified device type. The number of granules is specified at the time the file is created (and may not be expanded after it has been created). If the requested number of granules are not available contiguously, an abnormal code (major code X'01', subcode X'0B') is returned to the user and the file is not opened.

2. The user must specify a relative starting granule number with each read or write and a byte count (the default byte count in the DCB may be used). If the starting granule number does not fall between 0 and the total number of granules allocated at "OPEN"-1, inclusive, an error code of X'42' is then returned to the user. If the byte count exceeds granule size, the operation will continue in the next contiguous granule(s) until all requested bytes have been transferred. The system will return the next available relative granule number to the user (in the KBUF field of the DCB) at the completion of each read/write. If there are not sufficient granules to accommodate the specified byte count, an error code (major code X'57', subcode X'44') is returned to the user and the actual number of bytes transmitted is placed in the RWS and ARS fields of the DCB.

3. Each write/read consumes the entire specified granule. The contents of the granule include no system information. Management of the user's data is the responsibility of that user.

4. Function has the following meaning for random files: when any random file is opened it is first checked for existence.

   - If the file does not exist and function is IN or INOUT, an abnormal code of X'03' is given. If the file does not exist and OUT or OUTIN is specified, a new random file is allocated unless the associated account number differs from the user's account number (in this case, the file will not be opened and an abnormal code of X'14' will be returned).

   - If the file does exist, the user is checked for appropriate access permission (read/write account numbers, password), and an abnormal code X'14' is returned if there is a violation. If there is no violation, the user may proceed to read (unless opened OUT) or write (unless opened IN). If the file is opened OUT or OUTIN, the function is changed to INOUT. Note that the user may write in a granule in which he has already written, and may also read a granule in which he has not written. A random file that is in existence may not be replaced or extended by reopening that file. If the user wishes to replace or extend an existing file, the existing version must be released prior to such action.

5. If a file is opened OUT and a file of the same name already exists, an I/O abnormal code 14-00 will occur if one but not both of the files is random.

6. If a random file is opened OUT, and a random file of the same name already exists, the mode of the open will be changed to INOUT. This change occurs even if the open specifies an RSTORE value that is different from the size of the existing file.

7. If a file is opened OUT on private pack and a file of the same name already exists on the pack, an I/O abnormal 14-00 will occur if both files do not have the same organization.

Thus, the monitor provides allocation of granules, security checks and normal I/O queuing service and clean up. The user is responsible for record management.

FILE FUNCTION AND FILE DISPOSITION

A file may be opened in one of four modes: two of these, IN or input and INOUT or update, access a file that existed prior to this open; the other two, OUT or output and OUTIN or scratch, create a new data aggregate that had not existed prior to this open. There are three possible specifications for the file disposition option: REL or release, SAVE, and JOB. Any one of the three may be specified at open time and either REL or SAVE may be specified at close time. The impact of each of these options and several significant combinations of them is described below.

To create a new file, specify OUT or OUTIN. If the REL disposition option is used or implied (see below) with such an open, it indicates that the file to be created must be released when it is closed; thus, it is an obvious error to combine OUT with REL, and such an open is rejected with an abnormal 14-07. The other combination, OUTIN with REL, results in a true scratch file which is never to be entered into the file directory and thus has no identification other than the device control block with which it is associated. Storage space requirements for such a file are accounted for against the user's temporary granule authorization.

If a file opened OUT or OUTIN is closed with an explicit specification of SAVE, it will be entered into the file directory unless the open process failed to explicitly specify SAVE or JOB, in which case the file is unconditionally

released at close time and the file directory contents are not altered. If an explicit SAVE specification is not made when an OUT or OUTIN file is closed, again the file is released and the file directory contents are not altered. Note that when a job step is completed, all open device control blocks are closed with no explicit disposition specification, and so all open output files are released at that time. The only exception to this is M:DO, which is closed with explicit SAVE in order to ensure that diagnostic output will be received.

Consider the cases of files opened OUT or OUTIN with either SAVE or JOB disposition. All such combinations indicate the intent to create a new file which will probably be entered into the file directory. (See the above paragraph for a discussion of how to overcome this intent.) Unless a job is executing at a high privilege level of X'CO' or greater, it cannot create a new file in an account other than the one under which the job is logged on with one exception. If a file already exists with the same identification as that desired for the new file; if, further, the already existing file permits WRITE access to the user in question; and if, finally, the already existing file is not currently open, then the user may create a new copy of such a file. When an OUT or OUTIN file with SAVE or JOB is closed with explicit SAVE, the name is entered into the file directory; and any previously existing file with the same identification is released. In addition, if JOB has been specified on the open, the file identification is given the same treatment as though it had been mentioned in a M:TFILE procedure call. All files which have had their identification mentioned in such a procedure call are released when the creating user logs off. A JOB file may only be accessed by the creating user or a user with at least X'CO' privilege. Storage space requirements for JOB files are again accounted for against the user's temporary granule authorization. All storage space requirements for files other than true scratch or JOB files are accounted for against the user's permanent granule authorization.

The file disposition option at open time for input and update opens is essentially insignificant and the disposition is completely controlled by the specification on the close. (There is a name substitution option available for locating JOB files which is only operative in the event of explicit JOB disposition specification at open time.) If the specification is explicitly REL, the file is released and the identification is erased from the file directory; otherwise, the file is retained and no change is made to the file directory. When an existing JOB file is reopened in the update mode, the disposition in the device control block is forced to JOB so that granule accounting may be correctly handled.

FILE ACCESS

Records may be accessed within a file by either of two means, direct or sequential access. The interaction of the type of access used for a given operation and the mode in which the file is opened results in some rules, or limitations. These rules are listed below for each type of access and each mode in which a file may be opened.

DIRECT ACCESS

For consecutive files, the only effect of direct access is to inhibit read-ahead. For keyed files, the following rules apply.

OUTPUT FILES (OUT)

When a WRITE is given, a key must be specified. The keys do not need to be given in a sorted order. They will be ordered as they are stored on disk.

Unlike sequential output files, a WRITE never causes forward information to be deleted.

Reading is not allowed.

SCRATCH FILES (OUTIN)

A scratch file is identical to an output file, except that reading is permitted before the file is closed. As for output files, a key must be specified on each WRITE. The keyed record is merged into the file.

A READ may or may not specify a key. If a key is specified, a search is made of the file until the key is found and the record is then read. If the key is not found, an error return is executed. If a key is not specified, the next sequential record is read.

The FWD and REV options apply on read operations not specifying a key. If a key is specified, these options are ignored. PRECORD operations are performed in the same way as for sequential output files. A WRITE does not cause forward information to be deleted. A READ before the first WRITE returns an X'06' abnormal code.

INPUT FILES (IN)

Records may only be read; writing is not allowed. The READ function is the same as that for scratch files. PRECORD operations are allowed.

UPDATE FILES (INOUT)

The READ function is the same as for scratch files. PRECORD operations are allowed.

The WRITE function may or may not have a key specified. If a key is not specified, the WRITE function must have been preceded by a READ. If it is, the record just read is updated; if not, an X'15' abnormal code is signaled.

New records may be added to the file. The NEWKEY or ONEWKEY option must be specified, and a search of the keys will be made to locate the proper place to merge the new key. If the key already exists, and the NEWKEY option only was specified, an abnormal code X'16' is returned.

Records may also be replaced. The NEWKEY option must not be specified in this case. Thus, the NEWKEY option is used when adding new records to the file and notification of an attempted duplication of a key is desired.

The ONEWKEY option is used when adding new records and replacing old records. No notification is given when an existing record is replaced by a new record with the same key.

The absence of either a ONEWKEY or NEWKEY parameter implies that the record to be written already exists and is to be replaced. If the record does not already exist, an abnormal notification will be given and the record will not be written.

The DELETE function may be used. If a key is specified, a search of the directory is made to find the specified key. The record is then deleted. If a key is not specified, the DELETE operation must have been preceded by a READ, and the key just read will then be deleted.

SEQUENTIAL ACCESS

Sequential access may be used when accessing records with keyed or consecutive organization.

OUTPUT FILES (OUT)

When a file is opened in the OUT mode, records may only be written; reading is not allowed. If the file has been declared a keyed file, a key must be given with each write operation and this key must be a new key (i.e., it must not have been used before). If the key has already been used, no information is written and an abnormal X'16' is returned. The keys must be given in a sorted order. For example, if the user writes records with keys A, C, and D, respectively, and then writes a record with key B, the record will not be written and an X'18' error return will be executed.

The PRECORD FWD (position record forward) and PRECORD REV (position record backward) operations are allowed on both keyed and consecutive files. A BOF is given when the beginning-of-file is reached, and an EOF is given when the end-of-file is reached. Otherwise, for keyed files, the pointer to the current entry in the master index is decremented or incremented. For consecutive files, a directional count of records to skip from the current position is established. Positioning will not occur until the next read, write, or delete operation. A WRITE operation following PRECORD causes all forward records to be deleted.

When closing the file, the SAVE option must be specified in both an explicit CLOSE statement and in the OPEN statement if the file is to be saved.

SCRATCH FILES (OUTIN)

The same rules that apply to output files also apply to scratch files, except that reading is allowed, following a write. Reading may be directional; either forward or reverse. A READ with REV implies that the record preceding the current position is to be read. If no direction is specified, FWD is assumed. A READ order issued prior to the first WRITE will result in an X'06' abnormal return.

When reading a keyed file, a key may or may not be specified. If a key is specified, a search is made for the specified key. The FWD and REV options are ignored when a key is specified. If a key is not specified, READ FWD implies that the next record in sequence is to be read. READ with REV implies that the record immediately preceding the current record is to be read. Whenever a keyed file is read, the KBUF field of the DCB contains the address at which the key of the record just read is stored.

Reading a consecutive file is the same as reading a keyed file without specifying a key.

A WRITE deletes all forward information.

INPUT FILES (IN)

This is the same as for direct access input files.

UPDATE FILES (INOUT)

For a keyed file, this is the same as for direct access update files. For a consecutive file, a WRITE deletes all forward information.

SIMULTANEOUS FILE USAGE

REQUIREMENTS FOR MULTIPLE ACCESS TO A SINGLE FILE

Under some conditions, a file may be accessed by more than one DCB at the same time.

TAPE FILES

Single-File Tapes. Only one DCB may be open to the file at a time. If an attempt is made to open a tape file that is already referenced by another DCB, an abnormal return will be executed.

Multi-File Tapes. Only one DCB may be opened to the tape at a time. The user may not reference another file on the tape until the previous file is closed. To do so will cause an abnormal return.

DISK FILES

Random Files (SHARE mode not specified). Any number of DCBs can be opened to the same file in the input mode. Only one DCB may be opened to a file in the update mode. However, one update DCB and one or more input DCBs may be open to the same file at the same time. The order in which DCBs are opened or closed, when sharing the same file, does not make any difference. Only one DCB may be opened in the output or scratch mode and it may only be opened if no file of the same name already exists.

Keyed and Consecutive Files (SHARE mode not specified). Several users may simultaneously access a file. Some kinds of simultaneous uses are allowed and some are not. The rules governing such usage are described below and are summarized in Table 1.

If a file exists, it may be opened once in output mode and any number of times in input mode. Furthermore, the user must open the file for output first. All referencing DCBs in input mode must be closed before the output DCB may be closed. (If the input DCBs are not closed first, the output file will be discarded.)

A scratch file is not considered an output file in the above sense. Since the scratch attribute (OPEN, OUTIN, REL) is declared at the time a file is opened and all file information is local to the using DCB, multiple scratch files of the same name may be open. Remember, however, that scratch files are automatically released at the end of each job step or when the DCB is closed, whichever is sooner.

If a file is successfully opened for output, the effect is that no other users may have the file open in any mode (except scratch) and they will not be allowed access until it is finally closed.

A file opened as OUT or OUTIN with a save disposition does not replace an already existing file of the same name until the former is saved via a CLOSE operation. Thus, tw files of that name exist during the creation of the newer one.

Several DCBs may be simultaneously opened to the same file if the proper protocol is observed, as follows. A file may be opened as OUT, INOUT, or OUTIN only if not already open. If a file is already open as IN, OUT, or OUTIN, it may be opened again as IN only. No simultaneous usage of an INOUT file is allowed. A file opened as OUT or OUTIN may be closed and saved only if it is not also open as IN. If the above protocol is violated, an abnormal code is returned from the OPEN or CLOSE operation. In the latter case (CLOSE), the OUT or OUTIN file is released, also.

Since a file may be opened up to 127 times in the input mode, a user may attempt to release it upon closing if he is unaware of other usage. If this occurs, the request is not honored.

An OUTIN file that has been opened without a specific save disposition is a scratch file. Operations on a scratch file are always local to the DCB and are unaffected by the operations of any job on files with the same name. Any attempt to open a file in IN or INOUT mode with a release disposition is the same as if the M:OPEN has specified a save disposition. Table 1 summarizes the rules for opening and closing files with the same name. It also shows the error codes generated.

To open an OUT file with a release disposition is useless and wasteful.

Random and Keyed Files (SHARE mode specified). Up to 127 updaters and up to 127 readers may simultaneously have access to a keyed or random file. In this case, all DCBs must specify the SHARE mode. The order of opening is not significant in this case and there are no restrictions on the order of closing. When the last DCB which has a shared keyed file open is being closed, the mode is switched to EXCLusive to assure that a file information table is posted before the file is reopened. A close with release is treated as a close with save for DCBs opened in the SHARE mode.

Table 1. Simultaneous File Usage - Keyed or Consecutive

New File Operation			Result		
			Open-File Status (different DCB)		
Mode	Operation	Disposition	In	Outin Out	Inout
IN	OPEN	SAVE	V	V	E(14/01)
		REL[1]			
	CLOSE	SAVE	V	V	I
		REL	does not REL	V	I
OUT OUTIN	OPEN	SAVE	E(14/01)	E(14/01)	E(14/01)
		REL[3]	L	L	L
	CLOSE	SAVE[2]	REL	I	I
		OPEN = SAVE REL	V	I	I
		OPEN = REL	L	L	L
INOUT (UPDATE)	OPEN	SAVE	E(14/01)	E(14/01)	E(14/01)
		REL[1]			
	CLOSE	SAVE	I	I	I
		REL	I	I	I

Notes:

1. When IN and INOUT files are opened, SAVE is forced.
2. SAVE must have been specified when the file was opened. If not, REL is forced.
3. If mode is OUTIN and disposition is REL, the file is a scratch file and is local to the opening DCB. The file is released at the end of the job step and is never shared. If mode is OUT, REL is illegal and I/O error 14-07 results.

File Status Letters

E indicates an error or abnormal operation and is followed by an error code (Appendix B) in parentheses.

I indicates an impossible situation.

V indicates an allowed operation.

L indicates the file is local and the operation is allowed.

COORDINATING MULTIPLE ACCESS TO A SINGLE FILE

The SHARE mode feature extends the use of keyed and random files by permitting simultaneous access to a file by up to 127 updaters and up to 127 readers. Thus several user programs executing concurrently in separate jobs may be generating reports from a data file while other user programs are concurrently modifying data items within the file.

Responsibility for coordinating concurrent update activity is divided into two parts, one controlled and provided by the operating system and the other by the application programs via use of the system's enqueue/dequeue services. The operating system guarantees the physical integrity of the file so that it remains properly connected regardless of the update activity and also assures that readers are provided with the most up-to-date information in response to their requests.

Coordinating and guaranteeing logical integrity of the file (primarily the data content) is the responsibility of the application programs, since for the keyed file organization any connection of the data in one record of a file with that in another record of the same or another file is carried in the application program, not in the file itself. A single example will serve to illustrate this.

Suppose that a file contains records recording a parts inventory – each containing the available number of bolts, washers, nuts, etc., in various sizes. Without any special coordination, the number of any given item can be determined by querying the file even in the face of additions and removals by a concurrent updater. If, however, the application needs to first determine the available number and then remove a quantity from stock, then the record must be locked between the read and the update to preclude the possibility of the stock being taken by another updater.

More elaborate record locking requirements may exist depending on the application. For example if a fastener must be made up of a bolt, a nut and a lock washer, then these three records must be acquired and locked prior to making the needed updates.

Applications use the system's enqueue/dequeue facility to gain exclusive access to the records. Enqueue/dequeue is a generalized service and guarantees exclusive or shared access to named items as required and requested. It is the responsibility of all users of the service to agree on the meaning of the names – for example the names of the records containing inventory count of nuts, bolts, and washers.

PROTOCOL REQUIREMENTS

In a shared update environment, there are four broad classes of operation to be considered (some have interesting variations):

1. Statistical read – the process of reading without concern as to whether the current record, or other records associated with the current record because of application considerations are being updated.
2. Exact read – the process of reading with the assurance that the current record and possibly other application associated records are not in the process of an update which is only partially complete.
3. Update – changing the data content of a record or a group of application associated records.
4. Positioning – execution of M:PRECORD, M:PFIL, or sequential (not specifying a key) M:READ CALs.

To accomplish a statistical read or to execute a M:PFIL CAL, there are no special protocol requirements; however, for the other operations above, it is necessary to obtain some protection from other use of the record(s) in question. The enqueue/dequeue facility has been provided for this purpose.

In order to process an exact read, it is necessary to obtain shared use of the record(s) in question, while, to process an update, exclusive use is a requirement. It is not anticipated that M:PRECORD CALs specifying more than a single record move will be a common occurrence in the shared update mode; but if they are required, then a shared use of the entire file is required. Sequential reads and one position moves can be accomplished without protection if they are for statistical purposes only. If any other use is required though, the key presented in KBUF (in the DCB) after the move should be enqueued appropriately. It is suggested that sequential reads be effected with a zero length buffer and then a reread can be accomplished after the enqueue has been accomplished.

In addition, once an operation has been completed, the enqueued items should be dequeued promptly. It is essential that all users conform to the above described protocol or inefficient operations and data damage may ensue. CP-V does not enforce a correct enqueue/dequeue sequence, but only assures that:

1. The master index structure of a keyed file will not be modified so as to produce a permanent process error (75-02) situation.
2. The processing of an M:WRITE or M:DELREC CAL for a shared keyed file will not proceed until the completion of any other such CAL on the same file, if any, is in progress.
3. If a 75-02 error is about to be reported on a shared keyed file during the processing of an M:READ or M:PRECORD CAL because of the failure to pass a specified link test, a check will be made to see if any modification has been made to the master index linking structure since the inception of processing of the CAL in question. If so, the operation will be terminated with a 15-01 abnormal; and no error log entry will be made.

EXTENSIONS TO M:DCB, M:OPEN, ASSIGN AND SET

The function option of the M:OPEN, M:DCB procedures and of the ASSIGN and SET commands include the following options:

IN[,{SHARE / EXCL}] (for readers)

INOUT[,{SHARE / EXCL}] (for updaters)

where EXCL guarantees exclusive access to the file and SHARE permits sharing. If neither is specified, the option used is that from the DCB – either from the M:DCB process or remaining from a previous operation on the DCB. If no specification is then made by M:DCB, ASSIGN, or M:OPEN, then EXCL results.

The SHARE option is valid only for keyed and random files and permits updaters and readers to have the file open concurrently.

As a final note on processing, when the final user of a shared update keyed file is closing the file, the CFU mode is changed to exclusive during the close process of finding and updating the File Information Table. Thus, until the close is completed, any attempt to reopen the file will result in an 14-01 abnormal. In addition, for shared keyed files, a release specification on a M:CLOSE CAL will be treated as no release. To delete a keyed file, it is necessary to open the file in the exclusive mode and then issue a M:CLOSE CAL with the REL specification.

HASHING QUEUE NAMES

Examples in the Enqueue/Dequeue Resources section of Chapter 4 illustrate a common technique which an application may use to ensure data integrity: Enqueue for the file and subqueue for the record or records of interest by name. The queueing may use the actual file name and account and the actual record keys or some agreed upon abbreviation for them, however, it must be unique.

Since there is the possibility of an extensive monitor data area for enqueue tables if long names are used, it is appropriate to compress the queue and subqueue names by hashing techniques. The EDMS routines use a hash of the file identifier which results in a 24 bit value for any file name/account pair (because ENQ/DEQ carries names in TEXTC and rounds up to full words). The following program displays the hashing algorithm used. Tests on several large file sets indicate an incidence of duplicate hashes of considerably less than one percent.

```
            LI,1        BA(FILENAME)
            LI,3        0
            LB,2        0,1
*TEXTC COUNT
ST1         AI,1        1
            LB,4        0,1
            AW,3        4
            SCS,3       6
            BDR,2       ST1
            LI,1        BA(ACCT)
            LI,2        8
*8 CHARACTERS IN AN ACCT
ST2         LB,4        0,1
            AW,3        4
            SCS,3       6
            AI,1        1
            BDR,2       ST2
*REGISTER 2 CONTAINS ZERO
            DW,2        PRIME
            STW,2       HASH
*REMAINDER IS HASH VALUE
              .
              .
              .
PRIME       DATA        16777213
```

It is suggested that this algorithm can be used effectively and that a similar hashing technique be used on keys when the key max for the file is greater than three.

Whether hashing techniques are used or not, it should be emphasized that the above described protocol must be followed by all shared update users of a keyed file to obtain desired results. Also, if hashing is used by any shared update user for his calls to enqueue/dequeue, the identical hashing algorithm must be used by all users of the file. When using any hashing technique, the user must be prepared for the X'3101' and X'3102' abnormal returns from an enqueue CAL since more than one element may produce identical hash values.

DATA ENCRYPTION

A data encryption facility is provided for keyed and consecutive files. This service is not designed to provide facilities sufficiently secure for highly secret and classified material, but rather is designed to make sensitive information not readily readable (i.e., to put it into ciphered form). For example, buffers of data from encrypted files that appear in dumps taken by system analysts will not be understandable without a non-trivial code cracking exercise. In this manner, files which contain information such as employee salaries can be protected.

To initiate data encryption for keyed and consecutive files, the user must issue an M:SETDCB for the file's DCB after it has been opened to the file. An option of the M:SETDCB procedure allows the user to specify the address of a location which contains a data encryption seed or to specify that data encryption is to be turned off. The seed is used by a pseudo random number generation process for both data encryption or decryption. (Even if the content of the location is zero, encryption/decryption will occur.)

It is very important to note that the seed(s) for data encryption process are not carried in the file, nor anywhere else within the file system. Thus, even users with high privilege who do not know the seed(s) are unable to read anything but gibberish without a significant code cracking process. The other side of this coin is that a user who forgets or cannot reconstruct the encryption seed(s) that were used has essentially lost the encrypted file.

Data encryption is different for keyed and consecutive files, as the keys are used in the encryption process. If an encrypted keyed file is stripped of its keys, the file cannot be decrypted.

FILE STORAGE DEVICES

The three general types of storage media available for user files are (1) disk, (2) labeled magnetic tape, and (3) other physical devices (e.g., cards, unlabeled magnetic tape, etc.).

DISK STORAGE

Both RAD and disk pack devices are used for secondary storage. Any combination of these devices can be defined at SYSGEN time. A disk pack device has dismountable volumes and can be declared either a public or private device at SYSGEN time, while a RAD device, not having dismountable volumes, can only be declared a public device.

A public disk pack device has only one volume that can be recognized by the monitor and that volume must be mounted at all times while the system is active.

A private disk pack device has any number of dismountable volumes that can be recognized by the monitor. The operating system requires that only those volumes needed for execution of the user's job be made available and be mounted.

STORAGE ALLOCATION UNITS

For allocation purposes a disk pack device is partitioned into logical units, either granule or cylinder. RADs are partitioned and allocated in granule units only. A granule unit equals 512 words and is equivalent to two sectors.

FILE ALLOCATION

Keyed and consecutive file space is allocated on a demand basis as the file is being created or updated. Therefore the file does not necessarily exist in contiguous areas on a RAD or disk pack device and can exist on many different physical devices. Random file space is contiguous and is allocated when the file is opened.

A public file resides on a public device (RAD and/or disk pack); a private file resides on private disk pack volumes. A public file can be allocated in granule or cylinder units; a private file is always allocated in cylinder units.

Files on Public RAD and Disk Pack. Allocation of space for files on RAD and/or disk pack follows a set of rules that may be altered and controlled by both the user for individual files and by the system manager on an account or system-wide basis. The scheme provides for best system performance, in absence of specification by the user or system manager, or for good performance of individual jobs by careful selection of disk pack (DP) or RAD (DC) to optimize the program's performance.

Although the rules stated below control the preferred allocation, the system will continue to look for space on other devices on request as long as the user-allowed limit is not exceeded and the space physically exists.

In the absence of other specifications, the monitor uses the following rules to determine the placement of files on RAD or disk pack:

1. All permanent files (opened INOUT or OUTIN and SAVE) prefer disk pack.
2. All temporary files (opened OUT or OUTIN and RELease) prefer RAD.
3. All account directories (AD), file directories (FD), and file information tables (FIT) prefer RAD.
4. All star files (system temporary files for ROMs, LMs, debuggers, etc.) prefer RAD.

Two methods are available to the system manager for control of file space allocation.

1. Using ANLZ and its subcommand DELTA, the system cell RADIST may be set nonzero. In this case, the normal preference is overridden and all space requests (except explicit CYLINDER allocation) prefer RAD.

2. Using SUPER, the system manager may separately limit the amount of space on RAD or on disk pack available to an individual user. For example, by setting the disk pack allowance to zero, all files of that user will be forced to RAD. An error to the user program results if no RAD space is available.

A user program or job may control the allocation of files to RAD or disk pack using either ASSIGN control commands or the M:OPEN program procedure. The required specifications are NOSEP and DEVICE,DP or DEVICE,DC for preferring disk pack or RAD, respectively. If CYLINDER is specified, cylinder-allocated disk packs are preferred.

Public Random Files. A public random file is allocated on a public device by the default rules or by the type specified, either RAD (DC) or disk pack (DP). If disk pack was specified, the monitor attempts to allocate in cylinder units before allocating in granule units.

Private Files. All the index and data blocks of a keyed or consecutive private file are allocated from one or more private disk pack volumes. A keyed, consecutive, or random file can extend beyond volume boundaries.

RECORD BLOCKING

The system will automatically block records for keyed and consecutive files in 512-word blocks to provide more efficient use of disk space. The user has no knowledge of this blocking and, when reading, will receive the appropriate record within the block and not the entire block.

When updating a keyed file, the user may rewrite a record in a size larger or smaller than the original record size. If necessary, the monitor will allocate additional disk space to accommodate the larger size.

A write with a 0 byte count to a keyed file will result in a master index entry for the record with fields in the entry pertaining to disk address, record size, and displacement into the blocking buffer all set to zero. A write with a 0 byte count to a consecutive or random file will be ignored.

LABELED TAPE

CP-V handles two types of labeled tape, Xerox labeled tape and ANS labeled tape. Xerox tape labels and ANS tape labels are described in the UTS File Management Technical Manual, 90 19 89. (Xerox tape labels are currently referred to as UTS tape labels in the technical manual.)

XEROX LABELED TAPE

A Xerox labeled tape is given standard Xerox labels when I/O is first performed on the tape. No tape initialization is required.

For labeled tapes, record blocking is performed similarly to blocking disk records. In BACKSPACE or FORESPACE operations, the correct tape positioning is accomplished by reading each block and determining the number of records within the block (see Figure 4).

ANS LABELED TAPE

An ANS labeled tape is given standard ANS format labels either through the ANS tape initialization processor (Label) or as the result of an operator key-in.

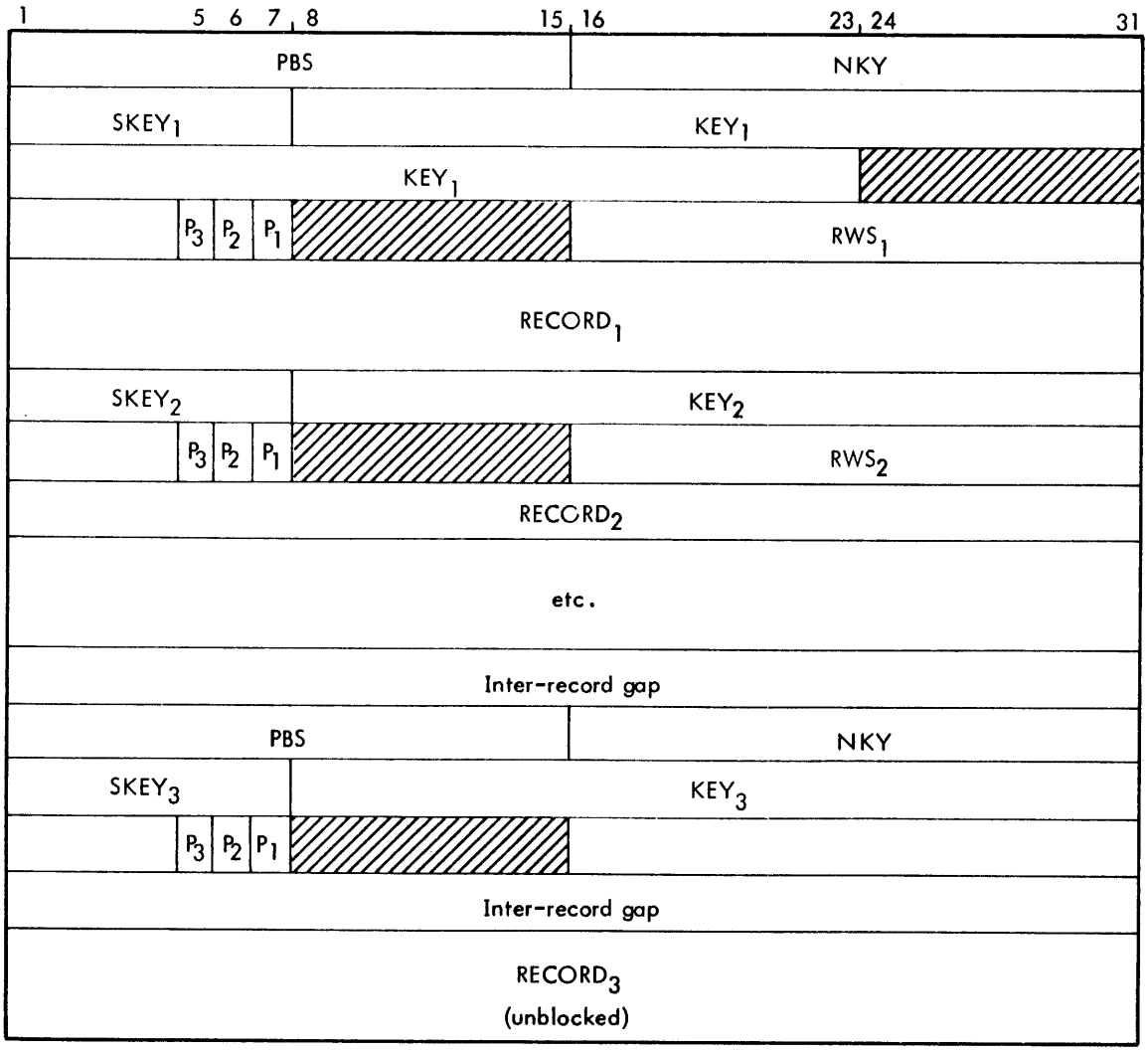
Important features of ANS labeled tapes include

1. When an input operation is performed, files may be processed by filename and volume sequence number, thereby eliminating the requirement of having enough serial number storage space in the DCB for all volumes to be processed.

2. The nondestruction of an unexpired tape can be, to some extent, guaranteed. There are two modes of tape protection that are applicable to ANS tapes:

   The protective mode, in which only ANS expired tapes may be written on through an ANS DCB, no unexpired ANS tape may be written on through a non-ANS DCB, all ANS tapes must be initialized by the Label processor, no tape serial number specification is allowed at the operator's console, specification of an output serial number in an ANS DCB forces processing to be done only on a tape already having that serial number, tapes mounted as IN may not be written, and tapes mounted as other than IN must have a write ring.

   The semi-protective mode, in which a warning is posted to the operator when an ANS DCB attempts output on a non-ANS tape or an unexpired ANS tape when a non-ANS DCB attempts output on an unexpired ANS



PBS contains previous block size.

NKY contains number of entries in block.

SKEY contains size of key (maximum = 31).

KEY contains key.

P_3 = 1 means record is unblocked.

P_3 = 0 means record is blocked.

P_2 = 1 means record continued into next block.

P_2 = 0 means not continued.

P_1 = 1 means first part of record.

P_1 = 0 means not first part.

RWS contains size of record in block.

Figure 4. Labeled Tape Format for Variable-Length Blocked Records

tape, or when a tape mounted as INOUT has no write ring. The operator can authorize the overwriting of the tape or the override of INOUT with the OVER and READ key-ins. ANS tapes may be initialized by the Label processor or may be given labels as the result of an operator key-in. Tape serial number specification is allowed at the operator's console, and specification of an output serial number in an ANS DCB forces processing to be done only on a tape already having that serial number unless the operator authorizes an overwrite.

The mode of ANS tape protection is determined at SYSGEN.

3. A number of files with the same filename and format may be treated as one logical file. This process is known as concatenation of files. Files may be concatenated in either of two ways.

   a. The number of files to be concatenated is specified using the CONCAT keyword and the serial numbers are specified, in order, using the SN keyword.

   b. The number of files to be concatenated is specified using the CONCAT keyword but no serial numbers are specified. In this case, exactly n files will be processed regardless of volume serial numbers. (The value n is specified following the CONCAT keyword.) The files will be concatenated in the order in which they are mounted.

   The concatenation feature is highly useful in situations where several portions of one logical file have been generated asynchronously, but the effect is transparent for input operations.

The user should be aware of the following restrictions for ANS labeled tapes:

1. Tape cataloging is not available in CP-V. Therefore, Generation Data Groups are not applicable.

2. Blocking and deblocking is the responsibility of the user or the run-time subroutines of the processors.

3. Multifile tape sets are processed via serial number only.

EXCLUSIVE USE OF TAPE FILES

Single-File Tapes. Once a user has opened a file, no other user may access the file until the original user closes it.

Multifile Tapes. Once a user has opened a file on a multifile tape, no other user may access the tape until the original user has closed the file. If the REW option is specified, the tape is rewound and a message is typed requesting the operator to dismount the reel. Otherwise, the tape remains at the current position and, if a DCB is opened using tape, one of two actions occurs:

1. On input or update, the tape is scanned forward for the desired file.

2. On output, the tape is positioned to the end of the current file and the new file is written at that position.

PHYSICAL DEVICES

On physical devices (unlabeled magnetic tape, punched cards, and typewriter output) it is frequently desirable for an operating system to intersperse certain control information with user data, to maintain system control, device independence (to user), etc. On the other hand, users occasionally desire to control a specific device entirely as if they were doing the I/O themselves.

These requirements give rise to the need for several formats for external media.

FORMATTED DATA RECORDS

These records are formatted and/or interpreted by the monitor. The mode is specified by the NODRC option of the M:DCB procedure. Exact actions are listed below.

1. Cards – Each binary record is represented on one card. An EBCDIC record can be represented as one or two cards. When the mode is changed (between two records), a mode control card is interjected (!BCD signals that an EBCDIC card follows; !BIN signals that a binary card follows). End-of-data is signaled by an !EOD card.

2. Typewriter – Each record is made up of data of a specified size and terminated by an NL (15_{16}) byte. End-of-data is signaled by an !EOD record.

3. Unlabeled Magnetic Tape – Records do not contain any formatted information. End-of-data is signaled by a physical EOF mark.

The actions resulting from various monitor I/O requests are as follows:

M:READ Read the next record and transfer either the byte count requested or the number of bytes in the record, whichever is smaller, eliminating the format information. Set the mode in the DCB according to the mode of the record. Position to read the following record.

M:WRITE Write the specified record as formatted data.

M:WEOF Output a physical EOF mark, if unlabeled tape; an !EOD, if card punch or typewriter; and a top-of-form, if line printer.

M:CLOSE (output mode) If unlabeled tape, output two physical EOF marks and position the tape between them. If card punch, output an !EOD.

M:CLOSE (input mode) No action.

DIRECT DATA RECORDS

These records are not formatted. Direct is specified by the DRC option of the M:DCB procedure. The user's I/O request is performed exactly as if he had control of the device. The data records are represented exactly as user specified in all cases. End-of-data is signaled by a physical EOF mark on magnetic tape and by !EOD on cards or typewriter. The C device cannot be read with DRC specified.

The actions resulting from various monitor I/O requests are as follows:

M:READ Reads as follows:

1. Unlabeled Magnetic Tape – Read the next record or the specified number of bytes, whichever is smaller. Position to read the following record. The specified number of bytes is limited to 32767.

2. Cards – Read the next card in the mode specified by the DCB (EBCDIC or binary) and transfers either the entire record or the number of bytes requested, whichever is smaller.

3. Typewriter – Read the specified number of bytes.

M:WRITE Output the specified record intact. If punched cards, use mode specified in the DCB.

M:WEOF Output a physical EOF mark, if unlabeled magnetic tape; an !EOD, if punched card or typewriter; and a top-of-form, if line printer.

M:CLOSE No action.

SYNONYMOUS FILES

Synonymous files are null files used to connect several names to one file. They are used in practice almost exclusively by the loader to handle libraries. System and utility processors that copy file to file should be able to handle these files, which exhibit unusual characteristics when they are opened or read. Here is how they work.

OPENNEXT

When an attempt is made to open a synonymous file in an opennext operation, an abnormal return with code X'08' is made. The file parameters are returned if requested and are indeed those of the synonymous file itself. Only the X'01' (name) and X'0B' (parent name) variable length parameter fields are present. If 'TEST FILE' is specified, the return is not abnormal, and if there is an X'11' variable length parameter field in the DCB, its data word has bit 17 set.

To copy the file, the output DCB should be opened with these file parameters and then immediately closed since there are no records to read. It is imperative that the parent file (the file the synonymous file is synonymous to – the name in the X'0B' field) exist where the file is being copied to, and the usual technique is to make two passes of opennext – the first ignoring synonymous files, and the second copying only them.

EXPLICIT OPEN

When a synonymous file is opened by name, no abnormal return is given, but the file that is opened is the parent rather than the synonym. The file parameters returned to the user are those of the parent while the name field in the DCB is that of the synonym. The X'0B' field in the DCB is not filled in with the parent name. If it is necessary to copy the synonym rather than the parent, several steps are required.

1. First it is necessary to detect whether the file that was opened is synonymous or not. The best way to do this is to compare the file name used in the open to the one returned in the file parameters. If they differ, the file is synonymous.

2. In order to copy a synonymous file once it is detected, a special open FPT is necessary. F1 should be set, and there should be at least X'01' (name) and X'0B' (parent name) variable length parameter fields. For example:

```
OPENSYNON  GEN,8,24   X'14',M:EO
           DATA       X'41000001'
           DATA       ABN          ABN
           DATA       4            INOUT
           DATA       X'01000808'
PARENT     RES        8
           DATA       X'0B010808'
SYNONYM    RES        8
```

3. To copy the file, the name field from the file parameters should be moved to PARENT and the name field from the DCB to SYNONYM. The output DCB should be opened with this FPT and then closed since there are no records to read. This comples the copy.

4. Once the synonymous file has been created, the SYNON name must be turned off in the DCB in order to output non-synonymous files through that DCB. This can be accomplished by including X'0B000001' in the variable parameters list of the open or adjust DCB FPT.

Again it should be remembered that the parent must be present in the account to which the synonym is being copied. It may be necessary to copy the file whose name is in the original file parameters before proceeding with the synonymous file copy.

3. MONITOR CONTROL COMMANDS

INTRODUCTION

The operating system is directed by means of a job control language (JCL) consisting of control commands. These commands control the construction and execution of programs and provide communication between a program and its environment. The environment includes the monitor and processors (such as Meta-Symbol, COBOL, and FORTRAN IV), the operator, and the peripheral equipment.

Monitor control commands discussed in this manual may be categorized as follows:

System	Input	Utility
JOB	BIN	PFIL
LIMIT	BCD	REW
STEP	DATA	WEOF
POOL	EOD	SWITCH
MESSAGE	FIN	
TITLE		
ASSIGN		
LDEV		
XEQ		

Program Load and Execution	Debug
LINK	PMD
LOAD	PMDE
LYNX	PMDI
OVERLAY	SNAP
OLAY	SNAPC
INCL	IF
TREE	AND
PTREE	OR
RUN	COUNT
MODIFY	

System, Input, and Utility control commands are described in this chapter. Program Load and Execution control commands are described in Chapter 6, and Debug control commands are described in Chapter 7.

The term "alphanumeric" when used in conjunction with any of the following control commands is defined as any combination of the following characters:

A-Z a-z 0-9 ← $ * % : # @ - +

except where explicitly noted otherwise.

Monitor control commands have the general form

```
!mnemonic specification
```

where

! in column 1, optionally followed by none or one or more spaces, identifies the beginning of a control command or a control key-in function. No spaces, however, are allowed between ! and JOB or any of the input control commands. Note that to avoid problems, any processor control command or continuation to a monitor control command whose first few characters match any of the input control commands or JOB should be used with spaces following ! and preceding the term.

mnemonic is the mnemonic code name of a control function or the name of a processor. If it is the name of a processor, it may consist of up to eight alphanumeric characters with no embedded blanks. If it is the name of a function, it must be spelled exactly as shown in this manual, with no embedded blanks.

specification is a listing of required or optional specification subfields. This may include keyword operands (shown in this manual in uppercase letters), labels, or numeric values appropriate to the specific command. The specification field may begin one or more spaces after the mnemonic field, but spaces (blanks) may not be embedded within options.

The required or optional specifications of a command function are identified in this manual in the following ways.

Commas are used to separate fields and subfields and are required where shown, as in

DEVICE,name

Parentheses are used to indicate the subfield groupings and are required, as in

(SN,value,value)

Brackets are used to indicate selective options. They are not to be used in the control command and the operations shown need not appear in any particular sequence relative to each other in a specific control command. For example:

[(option 1)][,(option 2)]...[,(option n)]

Braces are used to enclose options vertically, thus indicating a choice can be made, as in

```
!{LOAD   }
 {OVERLAY}...
 {OLAY   }
```

Single quotations are used in the specification field as constant delimiters (see Meta-Symbol/LN, OPS Reference Manual, 90 09 52), and are to be used when shown. For example:

'ALL' or 'value'

A period may be used after the specification field (or after the mnemonic field if the command is one with no specifications) as an explicit command terminator. A period is not required if no comment is to follow the specification field. A period may also be used in place of the mnemonic field (i.e., in column 2) to indicate that the "command" contains a comment only.

A semicolon is used as a continuation indicator for the specification field or for comments to be continued from one record to the next. (Processor calls cannot be continued.) For example:

```
!fication

!mnemonic speci;
```

Note the ! in column 1 of the continuation card. Annotational comments detailing the specific purpose of a command may be written following the command terminator. (Generally a period is used as the command terminator. However, if the command consists of a mnemonic alone, one or more blanks may be used as the command terminator.) Comments in a control command record may not contain a semicolon (except as a continuation character).

Communication between the operator and the monitor is accomplished through control commands, key-ins, and messages. Control key-ins are always input through the operator's console. Control commands are usually input to the monitor via punched cards; however, any input device(s) may be designated for these functions (see "ASSIGN", below). All control commands and monitor messages are listed on the output device designated as the listing log (normally a line printer). In this manner, the monitor keeps the operator informed about the progress of each job.

SYSTEM CONTROL COMMANDS

JOB Signals the beginning of a new job.

- Must be the first control command in each job.
- No spaces are allowed between ! and JOB.
- May not be continued from one record to the next.
- Subfields must be separated by a comma.
- Must specify a legal account and name combination and (optionally) a priority authorized for that user.

The form of the JOB control command is

```
!JOB account,name[(ext. actg.)][,priority [,wsn
   [,password]]]
```

where

account specifies an authorized batch processing account number of from one to eight characters.

name identifies the user. The name may consist of from one to more than 12 alphanumeric characters, but only the first 12 will be used.

ext. actg. identifies the user's accounting information as a subset of the user's name. It may consist of from one to more than 24 characters, but only the first 24 will be used. Legal values are any alphanumeric characters except commas (,) and parentheses (()).

priority specifies the priority of the job. Legal values are

0	(hold in job queue until priority is changed by a PRIORITY key-in).
1-F_{16}	(lowest to highest priority).

The default value is 1.

wsn specifies a workstation name and is only applicable to remote processing (see the CP-V/RP Reference Manual, 90 30 26). If the workstation name is present and valid, the job's output will be assigned to the specified workstation. If it is not present, the job's output is returned to the workstation at which the job originated.

password specifies a user password which may consist of one to eight characters and is only applicable to remote batch or locally submitted batch jobs. The printing of the password is suppressed when the JOB control command is listed.

Example:

```
!JOB C6400314,SMITH-1234(3211-XYZ), F.3211,3291
```

This example specifies that the account number of the job is C6400314, the user is SMITH-1234 with extended accounting 3211-XYZ (employee 1234 named Smith with extended accounting information specifying 3211-XYZ), and the job has priority F (the highest possible). The period following the specification fields indicates that the remainder of the record consists of comments.

LIMIT Specifies (in decimal integers) maximum values for various system resources required by the job.

- LIMIT control commands are optional, and, if included, must follow the JOB control command.
- Job aborts when limit for any system resource is exceeded.
- LIMIT commands cannot be continued. However, multiple LIMIT commands are allowed in a job.

The form of the LIMIT control command is

```
!LIMIT (option)[,(option)|. . .
```

where the parameter options are

ACCOUNT specifies that no other batch job with this account is to be run concurrently. The default is to allow the execution of concurrent batch jobs under the same account. Specification of ACCOUNT has no effect on on-line jobs.

CORE,value specifies, in K units where K = 1024 words, the maximum amount of core required for the user's data, DCBs, and procedure. The core space for shared processor procedure called by the user and context items required by the monitor (such as JITs and buffers) is not included in this limit.

DO,value specifies the maximum number of printed pages that may be output for diagnostics in the current job. (Output is via the M:DO DCB.) The maximum value that may be specified is 32,767. Note that !PMD output is not subject to this limitation.

LO,value specifies the maximum number of printed pages (excluding diagnostic output) that may be listed by shared processors for the current job. The maximum value that may be specified is 32,767.

MOUNT,$\left(\left\{\begin{matrix}X[ij]\\S[ij]\end{matrix}\right\},sn,\ldots\right)\left[,\left(\left\{\begin{matrix}X[ij]\\S[ij]\end{matrix}\right\},sn,\ldots\right)\ldots\right]$

specifies which packs are required and whether they are to be shared or are to have exclusive use, where

X indicates that the referenced disk packs are not to be shared (i.e., they are to have exclusive use).

S indicates that the referenced disk packs are to be shared.

sn specifies the serial number of the disk pack.

ij specifies the type of disk pack, such as SP. If ij is omitted, the type is assumed to be SP.

NORDER specifies that this job is not dependent upon the outcome of any previously submitted jobs.

ORDER specifies this job is to be run only after all previously entered jobs with the same account number have been run.

PDISK,value specifies, in decimal, the maximum number of public (disk pack) storage granules that are to be allocated for permanent files by the current job.

PO,value specifies the maximum number of punched cards that may be produced in the current job. The maximum value that may be specified is 32,767.

PSTORE,value specifies, in decimal, the maximum number of public (RAD) storage granules that are to be allocated for permanent files by the current job.

RERUN requests that in the event of a system failure while the job is running, the job be rerun after the recovery. The request will be honored unless the job is suspected (by the system) of causing the failure. Note that RERUN is not appropriate for all jobs. For example, it could be disastrous to rerun an interrupted job that updates a data base.

resource name,value specifies the maximum number of resources, where resource name is a system management defined label such as

9T in which case the value specifies the maximum number of 9-track tape drives.

7T in which case the value specifies the maximum number of 7-track tape drives.

SP in which case the value specifies the maximum number of spindles required exclusively for disk pack use. This value determines which partitions are available for the current job. Shared spindles are not counted for partition fit and are not included in the SP count.

For example, (9T,5) declares that a maximum of five 9-track tapes are required for the current job.

TDISK,value specifies, in decimal, the maximum number of public (disk pack) storage granules that are to be allocated for temporary files by the current job.

TIME,value specifies, in minutes, the maximum execution time for the current job.

TSTORE,value specifies, in decimal, the maximum number of public (RAD) storage granules that are to be allocated for temporary files by the current job.

UO,value specifies the maximum number of printed pages that may be output by an executing user program (nonshared processor) in the current job. The maximum value that may be specified is 32,766.

Example:

```
!LIMIT(TIME,10),(LO,100),(PO,2500),(DO,50),(UO,75)
```

The above example specifies that the current job may require no more than 10 minutes of execution time, 100 pages of object listings, 2500 object cards, 50 pages of diagnostics output, and 75 pages of output produced by the execution program.

STEP Provides conditional execution of job steps. It operates on and tests the value of the step condition code (SCC), a monitor item that is located in the JIT. At the beginning of a job, the SCC is set to zero. During the job, the SCC may be modified in one of two ways:

1. The SCC may be set at the end of a job step to reflect the manner of completion. The following values are used:

   2 – the step was skipped

   4 – the step was errored

   6 – the step was aborted

   The SCC will only be set if the job step did not execute successfully and if the new value is greater than the current value in the SCC.

2. The SCC may be set as the result of a STEP control command to any hexadecimal value in the range 0-F. The value is specified in the STEP control command and is only used to set the SCC under certain conditions (described below).

The STEP control command has the form

```
!STEP op,v1[,v2]
```

where

op specifies the type of comparison to be made. Possible specifications are

GT – greater than

LT – less than

EQ – equal to

GE – greater than or equal to

LE – less than or equal to

NE – not equal

v_1 specifies the hexadecimal value to be compared with the SCC.

v_2 specifies the hexadecimal value (in the range 0-F) used to reset the SCC.

If the logical expression (SCC op v_1) is true, the next job step is executed and the SCC is set to the value specified by v_2. (If v_2 is not specified, the SCC remains unchanged.) If the logical expression is false and the current value of the SCC is less than 2, control commands are skipped up to the next STEP control command (or to the end of job if there is not a subsequent STEP control command) and the SCC is set to indicate that the step was skipped.

The STEP control command may be placed anywhere in a job except in the middle of debug control commands. The STEP control command will not be honored and the entire job will be aborted if any of the following conditions is encountered:

1. Invalid JOB or LIMIT control command (e.g., invalid syntax in command).

2. Operator abort key-in.

3. Violation of values specified in LIMIT control command (e.g., specified execution time exceeded).

Example:

```
!STEP LE,2,0
```

The above example tests the results of previous job steps as recorded in the SCC. If the SCC is less than or equal to 2, the SCC is set to 0 and the next job step is executed. Otherwise, the SCC remains unchanged and all control commands are skipped up to the next STEP control command or until the end of job.

POOL specifies the number of buffers to be allocated to the monitor for file indexes and file data. A POOL control command may appear anywhere except between the JOB and LIMIT commands or within a series of debug commands.

If POOL is not specified, system limits are assumed. The maximum number of buffers allocated will never exceed available storage.

The form of the POOL control command is

```
!POOL (FPOOL,value)
```

where value specifies the number of 512-word buffers to be allocated for file management. The value specification must be in the range 4 to 22. For optimum performance, each DCB open to a labeled tape or disk file should have the following number of buffers:

labeled tape	1
consecutive disk file	1
keyed disk file	2
random disk file	0

If fewer than the optimum number of buffers are available, the buffers will be shared.

MESSAGE sends a message to the operator console (OC device) and listing log (LL device) at the time that it is encountered by the monitor.

- May not continue from one record to the next.
- More than one MESSAGE control command in succession is permissible.

The form of the MESSAGE control command is

```
!MESSAGE message string
```

where message string specifies the message to be typed.

Legal values are all characters, including blanks.

Example:

```
!MESSAGE SEND ALL SAVE TAPES TO BEN NEVIS
```

The above example causes the following message to be output on the LL and OC devices.

```
*id: MESSAGE SEND ALL SAVE TAPES TO BEN NEVIS
```

where id specifies the user's job identification.

TITLE inserts a heading at the beginning of each logical page listed on the LO device.

- May not continue from one record to the next.
- Has no effect if a header has been specified for LO output (see M:DEVICE procedure under "Specify Output Header"), or if LO output is not assigned to a listing type device.
- Within a job, the most recent TITLE control command is in effect, and page numbering begins at 1 when each TITLE control command occurs.

The form of the TITLE control command is

```
!TITLE title string
```

where title string specifies the title that is to appear on each page.

Legal values are all characters, including blanks.

Example:

```
!TITLE*STRESS-ENERGY TENSOR ANALYSIS*
```

The above example causes the title string to be output at the top of each logical page listed on the LO device by the executing program.

ASSIGN An ASSIGN control command can be used to assign a user's logical I/O device to a system logical or physical device. A logical device in a user's program is controlled by a data control block and is referred to symbolically by a name beginning with the characters "F:" or "M:".

If a DCB contains all necessary information when assembled into the load module of the user's program, then no ASSIGN command is needed. However, if the DCB is incomplete or if the user wishes to use an ASSIGN command to alter one or more of the parameters, this may be done at any time in the job prior to execution of the program containing the DCB. Any parameters altered in this way will remain altered throughout execution of the user's program unless explicitly changed by a call to a system function (see "M:OPEN" and "M:CLOSE" procedures). If a series of ASSIGN commands is given specifying the same DCB name, each successive command cancels all effects of the previous one. The DCB parameters then reflect the explicit options of the most recent

ASSIGN command for that DCB; parameters not included in the most recent command revert to the values of the DCB established when the user's program was assembled.

The total number of words (the assign/merge information) required to express ASSIGNs (or SETs) for a job, whether on-line or batch, may not exceed 512 words (an error message results if it does). Each assignment requires a minimum of four words, plus the number of words in the DCB name, plus the number of words in the open FPT for the requested assignments. Assignments may be replaced or deleted during the course of the job.

The parameters required in a DCB depend on the types of I/O operations to be performed and the types of devices and/or files to which the DCB may be assigned. In general, a DCB must contain at least the following parameters at the time that I/O is to be done:

1. Device or file name defining the assignment of the DCB.
2. File function (IN, OUT, etc).
3. Buffer address (if data is to be read or written).
4. Number of bytes to be transferred (if data is to be read or written).

The above parameters may be assembled into the DCB via the M:DCB procedure or may be specified in an M:OPEN procedure call. The first two may also be specified by an ASSIGN command, and the last two may be specified in an M:READ or M:WRITE procedure call.

Output to a labeled tape or disk file through a monitor DCB such as M:BO, M:LO, etc., will exist as a single file provided that the DCB is not reassigned between job steps via an ASSIGN control command or an M:OPEN procedure call.

Note: The following restrictions exist. The M:C DCB, i.e., the C device, is normally assigned to the card reader. It cannot be reassigned with the ASSIGN control command. In addition, the following system DCBs cannot be reassigned with the ASSIGN control command: M:=, M:*, F:CF, and M:OC.

When the M:GO DCB is assigned to a file, that file is automatically deleted at job termination.

Disk pack devices are declared either public or private at SYSGEN time. The volume on a public disk pack device becomes part of the system's secondary storage and must be in place at all times while the system is active. A file residing on a public device (or devices) is called a public file. The dismountable volumes on a private disk pack device are used in much the same way that tape reels are used.

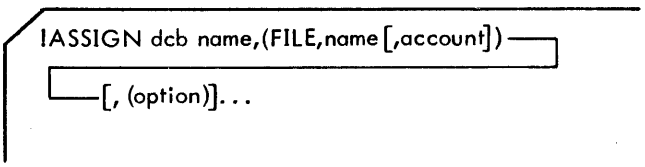
A file residing on a private volume is called a private file. A private volume-set is defined as a collection of removable volumes that the user has grouped together containing any number of files with any type of organization (consecutive, keyed, or random).

A private volume-set is identified by the volume serial numbers specified in the SN option of the ASSIGN command when the first file is written on the set. Volumes may be added to the set by the addition of a new volume serial number in the SN list, but a volume may not be removed. The system builds both an Account Directory and a File Directory (containing information about all files in the account) on each private volume-set so that a set of volumes is a self-contained entity and can be transported from one computer site to another.

There are four general types of assignments that can be made via the ASSIGN control command: disk files, Xerox labeled tape files, ANS labeled tape files, and files on other devices. Described below are four types of ASSIGN control commands, one for each type of assignment. In each case, only the options that are normally appropriate to the specific type of assignment are listed. Deleting assignments is discussed after the description of the four types of ASSIGN commands.

DISK FILE ASSIGN COMMAND

The form of the ASSIGN control command for disk files is



where

dcb name specifies the name (not exceeding 31 characters in length) of the DCB to be referenced. This must be the first subfield following ASSIGN, and must be followed by the FILE keyword. The first two characters of a user's DCB name must be "F:" (e.g., F:PRINT or F:BI). The first two characters of a monitor DCB name are "M:" (e.g., M:LO).

FILE,name[,account] specifies the name of the public or private disk file that is to be assigned to the DCB. The name may consist of up to 31 alphanumeric characters. The named file will be maintained on RAD or DP storage. If

the file belongs to a different account than that of the current job, the file's account number must be given. Otherwise, the file's account number is defaulted to the user's account. If the file is private, the SN option must be used to specify the serial number(s) of the private volume-set.

The options are as follows:

Physical Device

DEVICE,name specifies the type of physical device to be used for file storage (e.g., DC, DP). Name may be any disk device which was declared at SYSGEN. Use of the DP device type causes the system to request the default disk type that was defined at SYSGEN. The DEVICE option on an ASSIGN command is normally used only when ASSIGNing a file on a private volume set.

If device type is not specified for public files, space for the file will be allocated on any available RAD and/or public disk pack devices. Otherwise, space for the file will only be allocated on the type of device specified.

Organization (one of the three types given below)

CONSEC specifies that the records in the file are consecutively organized and each record is to be processed in order. This is the default if no organization is specified.

If a private file has consecutive organization, the system only requires that one volume in the private volume-set be mounted at any time. As another volume is required, the system will request that it be mounted.

KEYED specifies that the location of each record in the file is determined by an explicit identifier (key). If a private file has keyed organization, all volumes in the set must be mounted when the file is opened and remain mounted until the file is closed.

RANDOM specifies that the data in the file is to be written in contiguous areas of a random access device and accessed by specifying the starting block number. If device type is not specified, the file will be allocated on RAD or disk pack, whichever is available.

If a private file has random organization, all volumes in the set must be mounted when the file is opened and remain mounted until the file is closed.

Access (one of the two access means given below)

SEQUEN specifies that records in the file are to be accessed in the order in which they appear within the file. This is the default if neither SEQUEN nor DIRECT (see below) is specified.

DIRECT specifies that the next record to be accessed is to be determined by a key.

Function (one of the four modes given below)

IN[,SHARE / ,EXCL] specifies the input mode. This is the default if function is not specified. SHARE specifies share mode for the DCB which allows more than one IN and/or INOUT user to access the file concurrently. EXCL specifies exclusive mode for the DCB which means that the user must have exclusive use of the file. The default is EXCL.

OUT specifies the output mode.

INOUT[,SHARE / ,EXCL] specifies the input and output mode (i.e., the update mode). SHARE specifies share mode for the DCB which allows more than one IN and/or INOUT user to access the file concurrently. EXCL specifies exclusive mode for the DCB which means that the user must have exclusive use of the file. The default is EXCL.

OUTIN specifies the output and input mode (i.e., the scratch mode).

File Disposition (one of the two specifications given below; meaningful only for OUT or OUTIN files).

REL specifies that the secondary storage allocated to this file is to be released when the file is closed. See FILES, in the discussion of M:CLOSE.

SAVE specifies that the secondary storage allocated to this file is not to be released when the file is closed, unless specified otherwise by an M:CLOSE procedure call. If SAVE is not also specified in the M:CLOSE, the secondary storage allocated to this file will be released.

JOB specifies that the file is temporary. It is to be saved across job steps but is to be released when the job ends.

Other Options

CYLINDER specifies that the public file is to be allocated from public devices having cylinder allocation units. If CYLINDER is not specified, the public file

is allocated from public devices having granule allocation units. In either case, the file will only be allocated on the type of device specified with the DEVICE option. If the DEVICE option is not specified, the system looks for space on public disk packs first and RADs last. If space is not available in the units requested, the file will be allocated in the available units from public devices of the type requested. CYLINDER only has meaning for public files.

EXECUTE[,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. This option is valid only for OUT and OUTIN files. If no value is specified, all execute account entries in the DCB are reset.

EXPIRE,{mm,dd,yy / ddd / NEVER} specifies either an explicit expiration date (mm,dd,yy), the number of days to retain the file (ddd), or that the file is never to expire (NEVER). If not specified, the default value as established in the authorization record for the user will determine the expiration date. Files will be automatically purged from the public file system if they have expired whenever secondary storage space passes below a SYSGEN-established threshold.

The value specified may not exceed the maximum expiration period authorized for the user. If the maximum expiration period is exceeded or unspecified, the default expiration period authorized for that user will be used. If this option is omitted from the M:DCB procedure call it will not appear in the DCB and, consequently, may not be used in an ASSIGN control command or M:OPEN procedure call referencing the DCB. If EXPIRE is specified but no value given in the M:DCB call, two words are reserved for the value (to be inserted via an ASSIGN control command or M:OPEN procedure call).

KEYM,value 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.

NEWX,slides[,consecutive slides] allows the user to specify "when" and "if" a keyed file's higher-level index structure should be built (or rebuilt). Unless otherwise specified, the higher-level index structure is built for the first time when a keyed file that has more than three level 0 index blocks is closed.

slides 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.

consecutive slides 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.

NOSEP specifies that the index blocks of a public file are to be allocated in the same way that the data blocks are allocated. (Note that only keyed files have index blocks.) If NOSEP is not specified, the index blocks of a public file are allocated from public devices having granule allocation units. In either case, the file will only be allocated on the type of device specified with the DEVICE option. If the DEVICE option is not specified, the system looks for available granules on public disk packs first and RADs last. If space is not available in granule units, the system looks for space on public disk packs with cylinder allocation units. NOSEP only has meaning for public files with keyed or consecutive organization.

PASS,name specifies the password that will allow access to a password protected file (after any other security checks have been made). The password may be from one to eight characters in length and will be omitted from the listing of the ASSIGN commands.

READ[,value]. . . specifies, for OUT or OUTIN files only, the account number of those accounts that may read but not write the file. The value ALL may be used to specify that any account may read the file (e.g., READ,ALL). The value NONE may be used to specify that no other account may read the file. If no value is specified, or if READ (and WRITE, see below) is omitted, ALL or NONE, as specified in the authorization record for that user, is assumed by default. The total number of accounts explicitly specified in the READ and WRITE options must not exceed 16. READ is applicable to OUT or OUTIN files only. Also see WRITE and EXECUTE.

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 an ASSIGN 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.

RSTORE,limit specifies, in decimal, the number of granules to be allocated to a RANDOM file. RSTORE is only honored when the file is first created.

SN[,serial number]. . . specifies the serial numbers of the private disk pack volumes that are to be used for file input or output. The serial number may be from one to four alphanumeric characters and a maximum of three serial numbers may be specified for system DCBs. If SN is not specified (by ASSIGN, M:DCB or M:OPEN), the file is assumed to be on public devices. If no serial number is specified, all serial number entries in the DCB are reset.

For a file on a private volume-set:

1. When the first file on a private volume-set is created, all serial numbers in the set must be specified and the first volume in the set will become the primary volume

2. If the private volume-set has been established, only the serial number of the primary volume need be specified. The primary volume contains a list of all serial numbers in the set.

3. If one or more volumes are to be added to the set, the serial numbers of the new volume(s) must be specified following the primary volume.

The SN option must be specified in the M:DCB procedure call for it to appear in the DCB so that it may be used by the ASSIGN control command or the M:OPEN procedure call. When SN is specified in the M:DCB procedure call but no serial numbers are given, three words are reserved for the serial numbers which can be inserted through ASSIGN or M:OPEN.

The INSN and OUTSN options used in the previous version of the monitor were replaced with the SN option. For compatibility, the INSN and OUTSN options are acceptable in lieu of SN.

SPARE,n 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. The value specified may not exceed 255 bytes; if it does, it is treated modulo 256. If SPARE is not specified or is zero, it is set to 102 bytes by default. This spare space is used so that additional keys can be inserted in a minimum time when updating the file with direct access (as in EDIT). If the file will never be updated with direct access, a spare value of one should be specified.

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.

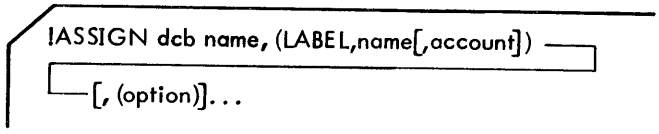
WRITE[,value]. . . specifies, for OUT or OUTIN files only, the account number of those accounts that may have both read and write access to the file. The values ALL and NONE may be used, as with the READ option; and, if a conflict exists between READ and WRITE specifications, those of the WRITE option take precedence. NONE is assumed by default.

The READ and WRITE option must be specified in the M:DCB procedure call for it to appear in the DCB so that it may be used by the ASSIGN control command or the M:OPEN procedure call. When READ or WRITE are specified in the M:DCB procedure call but no account numbers are given, 16 words are reserved for either READ or WRITE and can subsequently be filled by ASSIGN or M:OPEN.

UNDER[,name] 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. If no name is specified, the processor entries in the DCB are reset.

XEROX LABELED TAPE ASSIGN COMMAND

The form of the ASSIGN control command for Xerox labeled magnetic tape files is



where

dcb name specifies the name (not exceeding 31 characters in length) of the DCB to be referenced. This must be the first subfield following ASSIGN, and must be followed by the LABEL keyword. The first two characters of a user's DCB name must be "F:" (e.g., F:PRINT or F:BI). The first two characters of a monitor DCB name are "M:" (e.g., M:LO).

LABEL,name[,account] specifies the name of the magnetic tape file that is to be assigned to the DCB. The name may consist of up to 31 alphanumeric characters. If the file belongs to a different account than that of the current job, the file's account number must be given. Otherwise, the file's account number is defaulted to the user's account. If the file is to be input, the SN option must be used to specify the tapes containing the file.

The options are as follows:

Physical Device

DEVICE,name specifies the type of tape drive to be used for file input or output. Name may be any tape device which was declared at SYSGEN.

The standard tape devices are:

7T = 7-track tape drive.

9T = 800 bpi 9-track tape drive.

BT = 1600 bpi 9-track tape drive.

MT = installation dependent (see below).

Use of the MT device type causes the system to request the default tape type that was defined at SYSGEN.

Organization (one of the two types given below)

CONSEC specifies that the records in the file are consecutively organized and each record is to be processed in order. This is the default if no organization is specified.

KEYED specifies that the location of each record in the file is determined by an explicit identifier (key).

Access (one of the two access means given below)

SEQUEN specifies that records in the file are to be accessed in the order in which they appear within the file. This is the default if neither SEQUEN nor DIRECT (see below) is specified.

DIRECT specifies that the next record to be accessed is to be determined by a key.

Function (one of the four modes given below)

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).

Other Options

KEYM,value specifies the maximum length, in bytes of the keys associated with records within the file. A key may consist of up to 31 alphanumeric characters. The default value is 11.

PASS,name specifies the password that will allow access to a password protected file (after any other security checks have been made). The password may be from one to eight characters in length and will be omitted from the listing of the ASSIGN command.

READ[,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 the file (e.g., READ,ALL). The value NONE may be used to specify that no other account may read the file. If no value is specified, or if READ (and WRITE, see below) is omitted, ALL or NONE, as specified in the authorization record for that user, is assumed by default. The total number of accounts explicitly specified in the READ and WRITE options must not exceed 16. READ is applicable to OUT or OUTIN files only.

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 an ASSIGN 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.

SN[,serial number]... specifies the serial numbers of the tape reels that are to be used for file input or output. The serial number may be from one to four alphanumeric characters and a maximum of three serial numbers may be specified for system DCBs.

For a file on labeled tape:

1. Serial numbers must be ordered in the proper sequence for a file to be opened in the IN or INOUT mode. If SN is not specified (by ASSIGN, M:DCB or M:OPEN), the DCB is not opened and an abnormal code of X'14' is returned.
2. The file will be written in the order in which the serial numbers are specified for a file to be opened in the OUT or OUTIN mode. If SN is not specified (by ASSIGN, M:DCB or M:OPEN), available scratch tape(s) of the type specified in the DEVICE option (or by default, any type available) will be used.

The INSN and OUTSN options used in the previous versions of the monitor were replaced with the SN option. For compatibility, the INSN and OUTSN options are acceptable in lieu of SN. If no serial number is specified, serial number entries in the DCB 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.

VOL,value specifies which tape reel in the SN list is to be used initially. A value of 1 designates the first (in the list), a value of 2 designates the second, etc. If VOL is omitted, a value of 1 is assumed by default.

WRITE[,value]... specifies the account number of those accounts that may have both read and write access to the file. The values ALL and NONE may be used, as with the READ option; and, if a conflict exists between READ and WRITE specifications, those of the WRITE option take precedence. NONE is assumed by default. WRITE is applicable to OUT or OUTIN files only.

The READ or WRITE option must be specified in the M:DCB procedure call for it to appear in the DCB so that it may be used by the ASSIGN control command or the M:OPEN procedure call. When READ or WRITE is specified in the M:DCB procedure call but no account numbers are given, 16 words are reserved for either READ or WRITE and can subsequently be filled by ASSIGN or M:OPEN.

DEN,value specifies the density at which a magnetic tape is to be read or written. Only values of 800 or 1600 are acceptable.

ASCII specifies that conversion of code between EBCDIC in core and ASCII on tape is to be performed. ASCII is legal only for tape drives having this feature.

EBCDIC specifies that no conversion of code is to take place (see ASCII above) and that the tape is to be read and written in EBCDIC.

ANS LABELED TAPE ASSIGN COMMAND

The form of the ASSIGN control command for ANS labeled magnetic tape files is

```
!ASSIGN dcb name,(ANSLBL,name)[,(option)]...
```

where

dcb name specifies the name of the DCB to be referenced. This must be the first subfield following ASSIGN, and must be followed by the ANSLBL keyword. The first two characters of a user's DCB name must be "F:" (e.g., F:PRINT or F:BI). The first two characters of a monitor DCB name are "M:" (e.g., M:LO).

ANSLBL,name specifies the name of the magnetic tape file that is to be assigned to the DCB. The name may consist of up to 17 characters. If the file name contains special characters, it must be enclosed by single quotation marks. When a single quotation mark is to be used as part of the file name, it must be coded as two successive quotation marks. There should be no blanks between the last character and the terminating quotation mark.

The options are as follows:

Physical Device

DEVICE,name specifies the type of tape drive to be used for file input or output. Name may be any tape device which was declared at SYSGEN.

The standard tape devices are:

7T = 7-track tape drive.

9T = 800 bpi 9-track tape drive.

BT = 1600 bpi 9-track tape drive.

MT = installation dependent (see below).

Use of the MT device type causes the system to request the default tape type that was defined at SYSGEN.

Function (one of the four modes given below)

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).

Other Options

ABCERR specifies that block count errors (inconsistencies between the tape-specified and the system-accumulated block counts) are not to result in program abortion.

BLKL,value specifies block size in bytes. The value may be in the range 1 to 32,767. If a value less than 18 bytes is specified, 18 bytes are written.

CONCAT,value specifies the number of identically named files that are to be read as one logical file (concatenated). The value may be in the range 2 through 255.

EXPIRE, {mm,dd,yy / ddd} specifies either an explicit expiration date (mm,dd,yy) or the number of days to retain the file (ddd). If not specified, the default value as established in the authorization record for the user will determine the expiration date. Files will be automatically purged from the public file system if they have expired whenever secondary storage passes below a SYSGEN-established threshold.

The value specified may not exceed the maximum expiration period authorized for the user. If the maximum expiration period is exceeded or unspecified, the default expiration period authorized for that user will be used. If this option is omitted from the M:DCB procedure call it will not appear in the DCB and, consequently, may not be used in an ASSIGN control command or M:OPEN procedure call referencing the DCB. If EXPIRE is specified but no value given in the M:DCB call, two words are reserved for the value (to be inserted via an ASSIGN control command or M:OPEN procedure call).

FORMAT,character specifies the record formats. The character may be

F – fixed length.

D – variable specified in decimal.

V – variable specified in binary.

U – undefined.

LRECL,value specifies the logical record size in bytes. The value may be in the range 1 to 32,767.

SN[,serial number]... specifies the serial numbers of the tape reels that are to be used for file input or output. The serial number must consist of six alphanumeric characters (blanks are permissible) and a maximum of three serial numbers may be specified by system DCBs. If blanks are used, the serial numbers must be enclosed within quote marks. ANS serial numbers are stored in encoded format so that they will fit within 32 bits.

The INSN and OUTSN options used in the previous versions of the monitor were replaced with the SN option. For compatibility, the INSN and OUTSN options are acceptable in lieu of SN. If no serial number is specified, serial number entries in the DCB 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.

VOL,value specifies which tape reel in the SN list is to be used initially. A value of 1 designates the first (in the list), the value 2 designates the second, etc. If VOL is omitted, a value of 1 is assumed by default.

DEN,value specifies the density at which a magnetic tape is to be read or written. Only values of 800 or 1600 are acceptable.

ASCII specifies that conversion of code between EBCDIC in core and ASCII on tape is to be performed. ASCII is legal only for tape drives having this feature.

EBCDIC specifies that no conversion of code is to take place (see ASCII above) and that the tape is to be read and written in EBCDIC.

JOURNAL ASSIGN COMMAND

The form of the ASSIGN control command for a common journal file is

```
!ASSIGN dcb name,(JRNL,name[,account])
```

where

dcb name specifies the name (not exceeding 31 characters in length) of the DCB to be referenced. This must be the first subfield following ASSIGN, and must be followed by the JRNL keyword. The first two characters of a user's DCB name must be "F:" (e.g., F:PRINT or F:BI). The first two characters of a monitor DCB name are "M:" (e.g., M:LO).

JRNL,name[,account] specifies the name of the common journal that is to be assigned to the DCB. The name may consist of up to 31 alphanumeric characters. If the file belongs to a different account than that of the current job, the file's account number must be given.

DEVICE ASSIGN COMMAND

The form of the ASSIGN control command for devices other than disk or labeled magnetic tape file is

```
!ASSIGN dcb name,(DEVICE,name)[,(option)]...
```

where

dcb name specifies the name (not exceeding 31 characters in length) of the DCB to be referenced. This must be the first subfield following ASSIGN, and must be followed by the DEVICE keyword. The first two characters of a user's DCB name must be "F:" (e.g., F:PRINT or F:BI). The first two characters of a monitor DCB name are "M:" (e.g., M:LO).

DEVICE,name specifies the system physical device name, device type, operational label, or logical device stream that is to be assigned to the DCB. These labels are defined by the installation manager at SYSGEN. Tables 2 and 3 list the standard assignments.

Table 2. Standard Operational Labels, Device Types, and Physical Device Name

Type	Code	Description
Operational Label	BI, BO, C, CI, CO, DO, EI, EO, LL, LO, OC, PO, SI, SL, SO, UC (see Table 3). 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 the batch mode. No assignment, i.e., no default, is to be applied. Read operations through this DCB will return an end-of-file. Write operations will be ignored.
Standard Device Types	CP CR LP TY 9T 7T BT MT DP	Card punch Card reader Line printer Typewriter 9-track tape 7-track tape 1600 bpi 9-track tape Default magnetic tape type (defined at SYSGEN) Default disk type (defined at SYSGEN)
Physical Device Name	yyndd (the ndd portion is ignored but is allowed for compatibility with previous versions of the system)	yy specifies the device type as indicated above. n specifies the IOP letter (A-H corresponding to unit address 0-7). dd specifies the device number in hexadecimal, where: $00 \leq dd \leq 7F$ Refers to a device number. $80 \leq dd \leq FF$ Refers to a device controller number (8-F) followed by a device number.
Logical Device Stream	L1 C1 P1 (and others defined at SYSGEN)	Line printer Card reader Card punch

Table 3. Operational Label Conventions

Label	Reference	Comments	Typical Batch Device Assignment†
BI	Binary input	Binary coded input will be received from the device to which this label is assigned.	CR
BO	Binary output	Binary coded output will be transmitted to the device to which this label is assigned.	CP
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.	CR

Table 3. Operational Label Conventions (cont.)

Label	Reference	Comments	Typical Batch Device Assignment†
CI	Compressed input	Compressed symbolic input will be received from the device to which this label is assigned.	CR
CO	Compressed output	Compressed symbolic output will be transmitted to the device to which this label is assigned.	CP
DO	Diagnostic output	Diagnostic program dumps will be output on the device to which this label is assigned.	LP
EI	Element input	Element file input will be received from the device to which this label is assigned.	CR
EO	Element output	Element file output will be transmitted to the device to which this label is assigned.	CP
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.	LP
LO	Listing output	Source and object listings for assemblies and compilations will be output on the device to which this label is assigned.	LP
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.	TY
PO	Punch output	FCD or binary coded output will be transmitted to the device to which this label is assigned (normally a card punch).	CP
SI	Source input	Symbolic (source language) input will be received from the device to which this label is assigned.	CR
SL	Source listing	A listing of symbolic (source language) input will be transmitted to the device to which this label is assigned.	LP
SO	Source output	Symbolic (source language) output will be transmitted to the device to which this label is assigned.	CP
UC	User's console	This is for on-line use (see the CP-V/TS Reference Manual, 90 09 07). The batch mode defaults to OC (operator's console).	TY

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

The options are as follows:

Function (one of the four modes given below)

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).

Format Control (one of the following two options)

VFC specifies that the first character of each record is a format-control character for printing (see Table 4).

NOVFC specifies that the records do not contain format-control characters.

Mode (any of the following seven options for a device I/O mode)

BCD specifies that EBCDIC mode is to be used.

BIN specifies that the binary device mode is to be used.

Table 4. Line Printer Format Control Codes

Code (hexadecimal)	Action
C0, 40	Space no additional lines.
60, E0	Inhibit space after printing.
C1	Space 1 additional line before printing.
C2	Space 2 additional lines before printing.
C3 . . . CF	Space 3 additional lines before printing. . . . Space 15 additional lines before printing.
F0	Skip to Channel 0 (bottom of page) before printing.
F1	Skip to Channel 1 (top of page) before printing.
F2 . . . FF	Skip to Channel 2 before printing. . . . Skip to Channel 15 before printing.

FBCD specifies that FORTRAN BCD conversion is to be used. Note that this does not preclude use of the binary mode.

NOFBCD specifies that FORTRAN BCD conversion is not to be used.

PACK specifies that the packed binary mode (7-track tape) is to be used. PACK is not valid unless BIN is specified.

UNPACK specifies that the unpacked binary mode (7-track tape) is to be used. UNPACK is not valid unless BIN is specified.

L specifies that a listing-type device is to be used (FORTRAN programs).

Notes: BIN/BCD controls the mode of writing to CP or 7T, and reading from 7T. It also controls the mode of reading from CR if DRC has been specified.

FBCD causes conversion from the FORTRAN BCD set to EBCDIC on reading from CR or 7T and the opposite conversion when writing to CP or 7T.

PACK/UNPACK specifies packed or unpacked binary on 7T if BIN is also specified.

If no mode is specified, the current mode established for the Data Control Block (DCB) is used.

Other Options

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 type of device used) will apply. The value specified in an ASSIGN command will override that assembled into the DCB but will not override the RECL specifications of an M:OPEN call or the SIZE specification of an M:READ or M:WRITE procedure call.

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.

The following options are device-dependent and will be ignored by the monitor if not applicable to the device type used.

COUNT,tab specifies that a page count is to appear at the top of each page, beginning in the column specified by "tab". If COUNT is specified for the LO device and a TITLE control command is also specified, the page count will be superimposed on the title line.

Example:

```
COUNT,60
```

The above example specifies that the most significant digit of the page count is to appear in column 60 at the top of each page.

DATA,col specifies that output is to begin on each page (or card, if EBCDIC) in the column specified by "col".

LINES,value specifies the number of printable lines per logical page. The greatest value that may be specified is 32,767. If LINES is not specified, the value established at SYSGEN time will apply.

SEQ[,id] specifies that the punched output is to have decimal sequencing in columns 77-80. If id is specified, it will appear in columns 73-76 of the punched output. Sequencing begins with 0000.

SN,serial number[,serial number]... specifies the serial numbers of tape reels that are to be used for file input or output. The serial number may be from one to four alphanumeric characters and a maximum of three serial numbers may be specified for system DCBs.

The SN option must be specified in the M:DCB procedure call for it to appear in the DCB so that it may be used by the ASSIGN control command or the M:OPEN procedure call. When SN is specified in the M:DCB procedure call but no serial numbers are given, three words are reserved for the serial numbers which can be inserted through ASSIGN or M:OPEN.

The INSN and OUTSN options used in the previous versions of the monitor were replaced with the SN option. For compatibility, the INSN and OUTSN options are acceptable in lieu of SN.

SPACE,value specifies the spacing between lines. A value of one indicates that lines are to be single-spaced. The greatest value that may be specified is 15.

TAB,value[,value]... specifies the values of up to eight tab settings for an output device. The values must be in ascending order.

DEN,value specifies the density at which a magnetic tape is to be read or written. Only values of 800 or 1600 are acceptable.

ASCII specifies that conversion of code between EBCDIC in core and ASCII on tape is to be performed by the system. ^SC'I is legal only for drives having this feature.

EBCDIC specifies that no conversion of code is to take place (see ASCII above) and that the tape is to be read and written in EBCDIC.

Example:

```
!ASSIGN F:OUT,(DEVICE,LO),(SEQ,OUT)
```

This example specifies that the user's DCB name F:OUT is to be assigned to the output device to which the operational label LO is assigned (normally a line printer). Sequence numbers are to be printed in columns 77-80 and the identification "OUT" is to be printed in columns 73-76 of each record.

DELETING AN ASSIGNMENT

The form of an ASSIGN command used to delete an assignment is

```
ASSIGN dcb name [0]
```

where dcb name specifies the name (not exceeding 31 characters in length) of the DCB for which the assignment is to be deleted.

SET DCB assignments may also be made through the SET command in much the same manner as the ASSIGN command. The batch mode SET command has the same format as the on-line mode SET command.

ASSIGNING I/O DEVICES AND DCB PARAMETERS

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.

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.

The several formats of the SET command are:

```
SET dcb [0]

SET dcb [ oplabel
          device
          stream-id
          tapecode [tapeid] ] [;dopt]...

SET dcb [ [tapecode [tapeid] [-rt]] /fid ] [;fopt ...
        [ filecode [-rt]            ]

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:C, M:UC, M:OC, and M:XX are not allowed.)

oplabel specifies an operational label (BI, C, CI, etc.). (See Table 5.)

device specifies a device code (CP, PL, LP, etc.). (See Table 5.)

stream-id specifies the name of a logical device stream (C1, L1, P1, etc.). (See Table 5.)

tapecode specifies a magnetic tape code (LT, AT, or FT). (See Table 5.)

filecode specifies a secondary storage code (DP). (See Table 5.)

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

name [.account / .account.password / ..password]

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. ↵
!
```

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:IN 1 LT#4003/M ↵
!
```

Table 5. 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 3) (and others defined at SYSGEN)	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	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.

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
!
```

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
!
```

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
```

or the equivalent

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

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 5.

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 ASSIGN command are also valid for the SET command. Table 7 contains the list of file options.

Alternatively, PCL compatible keywords (as shown in Table 6) may also be used. As an example, FORMAT is a valid ANS labeled tape option in the ASSIGN command and the 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.

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. EBDIC 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.
COUNT = value	Turns on page counting and specifies the column number at which the page number is to be printed.
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.

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

Format	Description
FBCD, NOFBCD	Controls the automatic conversion between external Hollerith code and internal EBCDIC code (FORTRAN BCD conversion). NOFBCD is assumed by default.
IN OUT INOUT OUTIN	Specifies the input mode. Specifies the output mode. Specifies the input and output mode (i.e., the update mode). 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.

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

Type	Format	Disk	Xerox Tape	ANS Tape	Description
Access	SEQUEN DIRECT	X X	X X		Records will be accessed sequentially. Records will be accessed by key.
Function	IN [,{SHARE / EXCL}]	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.
	INOUT [,{SHARE / EXCL}]	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.
	{LRECL / REC} = 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 persisting across job steps.

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] = { mm,dd,yy | ddd | NEVER }	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][= { acct[,acct]... | ALL | NONE }]	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 option with no arguments resets all previous execute account entries in the DCB.

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

Type	Format	Disk	Xerox Tape	ANS Tape	Description
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 serial 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.
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.

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

Type	Format	Disk	Xerox Tape	ANS Tape	Description
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.

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 default physical device, and attribtues 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 and/or attributes to satisfy the requirements of his job. He makes any necessary changes through use of the LDEV command or the M:LDEV procedure. 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 LDEV control command can almost be viewed as another level of the ASSIGN control command. If the DCB has been ASSIGNed to a logical device stream, the LDEV command can be used to attach the logical device stream to a physical device (if the default physical device for the stream is inappropriate). ASSIGN stores information in a DCB, while LDEV stores information in a cooperative context block.

The LDEV control command has the form

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

where

stream-id specifies the two-character name of the stream to be referenced. This must be the name of one of the logical device streams defined during SYSGEN (for example, C1, L1, P1).

options specify the device streams attributes, such as device type, stream direction, form, format control, workstation name, etc. The options are as described below; they may appear in any order.

Options

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 been set as the result of previous LDEV commands) are not to be changed. ASAVE cannot be used for the LABEL option. DEV and WSN are subject to the restrictions noted in the Remote Processing Reference Manual 90 30 26.

COPIES,value specifies the number of times the file is to be processed to produce multiple copies. The value can be any integer from 1 to 255 inclusive. 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 device. (Note that if COUNT is specified for the LO device 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 menmonics are those defined for symbiont devices during SYSGEN (for example, CR, LP). Remote mnemonics are those specified when defining a workstation with Super (for example, OC, CR).

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 the 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.)

FPC,name specifies the one- to four-character name of an installation-determined form overlay and is used in output scheduling for the Xerox 1200 or a similar device. The default is to have no special scheduling (i.e., the operator will determine which overlay to use if any).

IN and OUT specifies the direction of the stream. The default is OUT.

JDE,value specifies the job descriptor entry to be used in output scheduling for the device. The value must be in the range 0-89 and specifies an installation defined procedure describing printer setup attributes (e.g., VFC tape).

LABEL,text specifies a text string to be appended to the stream's user-identification banner lines (see "user-identification banner" in glossary). The text may not include period, semicolon, or right parenthesis characters. Up to 255 characters of text may be specified. However, the length of the text that will be used is limited by the size of a line on the device.

LINES, value specifies the number of printable lines per logical page. The greatest value that may be specified is 255 lines per page. If this option is not specified, the value established at SYSGEN time will apply.

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.

NAME,name specifies the one- to four-character name of a noncontrol input file (see below, "Noncontrol Input Files"). If used on output, 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. (NAME and FORM may be used interchangeably.)

NOVFC see VFC below.

OUT see IN above.

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. A value of 0 or 1 results in single spacing. The greatest value that may be specified is 15. The default is single spacing.

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 he wishes to receive it as the first byte of records if this is an input stream. This option is used only for remote processing.

VFC and NOVFC specifies whether or not vertical format control characters are to be used. (These two options are legal only 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.

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, then 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.

WSN, {name / $} specifies the workstation name of the remote device that is to receive the stream, where name can be from one to eight alphanumeric characters. The default is local output. If a dollar sign ($) 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)
```

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)
```

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

```
!!NCTL [name]
```

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

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 jobs that are to be run at a later time.

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 specifies a name or requests the operator to do so, the user can access the file using the NAME,xxxx or FORM,xxxx option. (The operator gives the file a name using the key-in Syyndd,F'xxxx' where xxxx must be identical to xxxx 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.

XEQ The XEQ command initiates the execution of control 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 (FILE,name[,account[,password]])[, (REC,value)]
```

where

FILE,name [,account [,password]] specifies the command file. The name cannot exceed 11 characters in length.

REC,value specifies an optional starting record number. 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 a record number is not specified, command execution begins with the first record of the command file.

As each command in the file is executed, it is output on the line printer preceded by its record number.

Example:

```
!XEQ (FILE,ABJOB)
**** ABJOB EXECUTED AT RECORD 0001 ****
0001 - ASSIGN F:IN,(FILE,PAYROLL)
  .
  .
  .
0009 - ASSIGN F:11,(FILE,EMPS)
****ABJOB TERMINATED AT RECORD 0009 ****
```

Upon termination of command file processing, the system resumes reading control commands from the card reader.

Command files executed by XEQ must consist of valid control commands with a ! character in the first position of the record. The ! character is not printed on the line printer, however. Also, the user must provide assignments for all DCBs through which the program or processor will read data. Note that M:C cannot be assigned in the batch mode.

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

1. An end-of-file occurs for the command file.
2. A EOD or FIN command is encountered.
3. A syntax error occurs while processing a command in the command file.
4. An illegal command is encountered. There are three illegal commands: JOB, BIN, and BCD.
5. Another XEQ command is encountered within the command file. The second XEQ terminates the current command file and processes the newly specified command file.

INPUT CONTROL COMMANDS

Input control commands must not have any spaces between the exclamation character and the mnemonic. Input control commands are not listed on the LL device.

BIN The BIN control command informs the monitor that the information to follow will be in binary. The termination of the binary information is specified by a BCD control command (see "BCD", below).

The form of the BIN control command is

```
!BIN
```

Binary decks punched by monitor processors are identified as binary by means of a code in the first column of each card. Therefore, such decks need not be preceded by a BIN command. However, all user-formatted binary decks must be identified as binary by means of the BIN control command; otherwise, read-check errors may cause the job to be aborted.

BCD The BCD control command is used as a terminator for binary input, i.e., it causes the monitor to revert to its normal EBCDIC input mode.

The form of the BCD control command is

```
!BCD
```

DATA The DATA control command is used to inform the monitor that no more debug control commands are to follow. It is required immediately after the last debug command following a RUN control command (or after the RUN command itself if there are no debug commands) unless no data deck is to be read from the C device (in which case the monitor assumes that no debug commands are to follow).

The DATA control command has the form

```
!DATA
```

EOD The EOD (i.e., end of data) control command may be used to define data blocks in a data deck. This is accomplished by inserting EOD control commands at the end of each block of data. When an EOD command is encountered, the monitor returns an abnormal code of 05 in SR3 (if the user has specified an abnormal address in the M:READ procedure).

The EOD control command has the form

```
!EOD
```

Any number of EOD control commands may be used in a job.

FIN The FIN control command is used to inform the monitor that there are no more jobs to be processed.

The FIN control command has the form

```
!FIN
```

On encountering a FIN control command, the monitor automatically suspends symbiont activity for that input device.

UTILITY CONTROL COMMANDS

The utility control commands described below allow the user to manipulate magnetic tape files. These commands are only valid when used with library DCBs (of the form M:xxx).

PFIL The PFIL (i.e., position file) control command may be used to cause a designated number of physical files on unlabeled tape to be moved (i.e., skipped) in a specified direction. For unlabeled tape, the tape will be positioned after the end-of-file in the direction skipped.

The form of the PFIL control command is

```
!PFIL dcb name[,(BACK)][,(files)]
```

where

dcb name specifies the name of the DCB associated with the files to be skipped.

BACK specifies that skipping will take place in the reverse direction (the default option is skipping in the forward direction).

files specifies the (decimal) number of files to be skipped. If the files option is not specified, 1 is assumed.

REW The REW control command may be used to cause the tape associated with a specified DCB to be rewound.

The form of the REW control command is

```
!REW dcb name
```

where dcb name specifies the name of the DCB associated with the tape to be rewound.

WEOF The WEOF control command may be used to cause a physical end-of-file to be written on unlabeled tape (see M:WEOF procedure).

The form of the WEOF control command is

```
!WEOF dcb name
```

where dcb name specifies the name of the DCB associated with the tape on which the end-of-file is to be written.

SWITCH Any of six pseudo sense switches may be set or reset by means of the SWITCH control command.

The form of the SWITCH control command is

```
!SWITCH [(SET,value[,value]...)]
        [,(RESET,value[,value]...)]
```

where

SET,value specifies which pseudo sense switches are to be set. The "value" may be from 1 to 6, and more than one value may be specified. If ALL is specified, all pseudo sense switches will be set.

RESET,value specifies which pseudo sense switches are to be reset. The "value" may be from 1 to 6, and more than one value may be specified. If ALL is specified, all pseudo sense switches will be reset.

If a conflict exists between the SET and RESET options, the last setting specified in the command will apply, since the monitor processes the options in sequence (i.e., left to right).

Three library routines (L:TSS, L:SSS and L:RSS) are also provided to allow processors and user programs to set, reset, and test specified pseudo sense switches. The entire sense switch simulation is based on the use of a pseudo sense switch register contained in a Task Control Block (TCB) established and maintained by the monitor. The first two words of the TCB comprise a Stack Pointer Doubleword (SPD), and the subsequent words contain additional information used by the monitor to control the current task. When the monitor transfers control to a user's program (or a processor), it places the word address of the TCB in general register 0.

When a user's program calls any of the sense switch library routines, general register 0 must contain the word address of the TCB. General register 6 is used for passing the number of the specified sense switch. The link register is general register 11, and general registers 6-10 are volatile (not preserved by the library routine).

The linkage

```
BAL,11    L:TSS
```

causes the sense switch specified in general register 6 to be tested. If the switch is set, the condition codes are all set (to 1); otherwise, the condition codes are reset (to 0).

The linkage

```
BAL,11    L:SSS
```

causes the sense switch specified in general register 6 to be set. If the number of the specified switch is not within the range 1-6, the routine will ignore the request.

The linkage

```
BAL,11    L:RSS
```

causes the sense switch specified in general register 6 to be reset. If the number of the specified switch is not within the range 1-6, the routine will ignore the request.

4. SYSTEM PROCEDURES

INTRODUCTION

Monitor procedures enable the user's symbolic Meta-Symbol program to request a variety of monitor functions. When a procedure call is encountered during the processing of a program, the processor responds by retrieving a symbolic calling sequence from the procedure library, modifying it according to the parameters specified in the procedure call, and inserting the modified symbolic code into the user's source program (to be translated into object code during a subsequent processing phase).

At execution time, the calling sequence calls an appropriate monitor routine that, in turn, performs the desired function. In this manual, the monitor routine called at execution time, as the end result of a procedure call having a command mnemonic of the form M:XYZ, is referred to as monitor routine XYZ.

When using Meta-Symbol, the monitor procedure library is invoked via the directive

```
SYSTEM BPM
```

This directive defines all of the monitor procedures. The Sigma computer instruction set is invoked by the directive

```
SYSTEM SIG7[F][D][P]
```

where F specifies the floating point option, D specifies the decimal option, and P specifies privileged instructions.

The Xerox 560 and Sigma 9 computer instruction sets are invoked by the directive

```
SYSTEM SIG9[P]
```

where P specifies the privileged instruction set.

Thus, both the SYSTEM BPM and the SYSTEM SIG directives should be used. When the SYSTEM BPM directive is processed, two control sections are declared for use in generating function parameter lists for monitor procedures subsequently used. Normally, FPTs are generated in unprotected storage (protection type 0). The M:PT procedure allows the user to generate FPTs in either unprotected storage (protection type 0) or protected storage (protection type 1).

BPM procedures are of three forms: standard, list, and execute. The standard form results in both executable code and FPT generation during procedure expansion with FPTs being placed in an internal CSECT exclusive to System BPM.

Example:

```
SYSTEM BPM
:
:
M:OPEN F:ANS,IN      Generate OPEN CAL and
:                    OPEN FPT.
:
```

The execute form results in only executable code generated during procedure expansion with the argument field interpreted as the FPT address.

Example:

```
M:OPEN,E  OPENFPT    Generate OPEN CAL using
                     OPENFPT.
```

The list form results in only FPT generation during procedure expansion.

Example:

```
OPENFPT  M:OPEN,L  F:ANS,IN  Generate
                             OPENFPT.
```

Often it is desirable to be able to symbolically reference the parameter list associated with a particular procedure. The second element of the label field list is used for this purpose.

For example, assume the user wanted to use the label RD to identify the address of the CAL generated for the read function on the C device, and also wanted to use the label RDFPT to identify the address of the Function Parameter Table (FPT) for the same function. He could do so by means of the following Meta-Symbol procedure reference in his program:

Label	Command	Argument
RD, RDFPT	M:READ	M:C, (BUF, ALPHA)

Examples:

WR	M:WRITE	arglst
, OPNFPT	M:OPEN	arglst

In the first example, above, the label WR identifies the address of the CAL generated for the write function specified by the argument list (represented here by "arglst"). In the second example, the label OPNFPT identifies the address of the first word of the FPT generated for the OPEN function specified by the argument list (the associated CAL, in this example, is not given a label; hence, the comma preceding OPNFPT).

Because code generated for system procedures is placed in a different control section than the user's program, the $ symbol should not be used in procedure calls to represent the "present location".

The format conventions used by the procedure calls is of the same form as previously defined for control commands (see Chapter 3). In addition, an asterisk (*) is used in procedure calls to denote indirect addressing (* address) or to show that the indirect addressing technique can be used where a variable (* value or * limit) is the address of a location containing the variable. The terms SR1, SR2, and SR3 are used to refer to system registers 1, 2, and 3 (which are more commonly known as general registers 8, 9, and 10).

Each procedure call described in this chapter shows the Function Parameter Table (FPT) it generates. In the formats given, an asterisk in bit one of a word indicates that indirect addressing may be used. (Indirect addressing will be used if the bit is set to one.)

GENERAL-PURPOSE PROCEDURES

SET FPT PROTECTION TYPE

M:PT Normally, FPTs are generated in unprotected storage (protection type 0). The M:PT procedure allows the user to generate FPTs in either unprotected storage (protection type 0) or protected storage (protection type 1). The procedure has the form

[label1][,[label2][,label3]] M:PT type

where

label is set to the current value of the location counter ($).

label2 is set to the address of the beginning of the protection type 0 (unprotected) control section used for generating SYSTEM BPM FPTs.

label3 is set to the address of the beginning of the protection type 1 (protected) control section used for generating SYSTEM BPM FPTs.

type sets the new protection type (0 or 1) to be used when generating SYSTEM BPM FPTs in all subsequent procedure reference lines until the next M:PT is encountered.

Example

```
          :
          :
          SYSTEM    BPM
,PT0,PT1  M:PT      1
          DEF       PT0,PT1
          M:OPEN    M:EI
          M:READ    M:EI,(BUF,X)
          M:PT      0
          M:CLOSE   M:EI
          :
          :
```

In this example PT0 is the address of the beginning of the unprotected control section and PT1 is the address of the beginning of the protected control section used for generating SYSTEM BPM FPTs. The FPTs generated by M:OPEN and M:READ are protected (type 1) and the FPT generated by M:CLOSE, along with all FPTs generated until the next M:PT, is unprotected (type 0).

The M:PT procedure does not generate an FPT or any executable code.

Caution: Through the use of the Meta-Symbol USECT directive together with the label field of the M:PT procedure, the user can direct generation of code into the control section intended for the exclusive use of the BPM system procedures. This can lead to unpredictable results and is therefore discouraged.

LOAD OVERLAY SEGMENT

M:SEGLD The monitor routine SEGLD causes a specified overlay segment to be loaded into core storage. If an I/O error occurs in executing the SEGLD routine, or if the specified segment is not found, the job is aborted. The routine brings in both the data and procedure segments, unless they are already resident in core.

The M:SEGLD procedure call has the form

M:SEGLD {[*]address / 'segment name'}

where

address specifies the address of the first word of a byte string containing the name (in TEXTC format) of the overlay segment to be loaded.

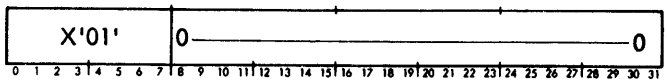
'segment name' specifies the name of the overlay segment to be loaded.

Calls generated by the M:SEGLD procedure have the form

CAL1,8 fpt

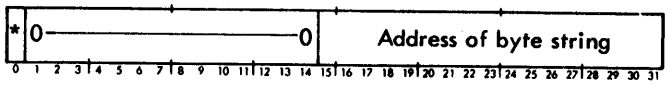
where fpt points to word 0 of the FPT shown below.

word 0



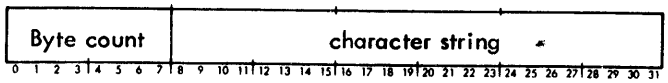
Caution: If bit 8 is not set, control returns to the word following the FPT, rather than the word following the CAL.

word 1

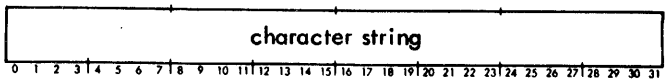


If 'segment name' is specified in the procedure call, the following words are also generated as part of the FPT. They contain the segment name in TEXTC format. The address in word 1 points to word 2 in this case.

word 2



word n



LINK TO A LOAD MODULE

M:LINK The monitor LINK routine causes the calling load module's core information (i.e., program and data, except common dynamic data) to be saved in disk storage. The calling module's core area is made available to the called module. The called module is then loaded into core storage (overlaying the calling program) and control is transferred to it. If there is no transfer address associated with the called module, the job is aborted. The user's temporary and permanent load module libraries are searched for the specified load module. If it is not found or an I/O error occurs in executing the LINK routine, the job is aborted. A return to the calling module may be effected in one of two ways:

1. The called module may return explicitly by making use of the monitor's M:LDTRC procedure.

2. The EXIT or ERROR options may be used in M:LINK by the calling program, causing the called module to return implicitly when it exits, errors, or aborts.

Any communication between the calling and called load modules must be accomplished through the general registers or common dynamic storage.

The monitor-assigned file name of the calling program is contained in SR1, allowing the called module to return to the calling location + 1 (via M:LDTRC, returning control to the overlaid program). If a program is entered via a RUN command, however, SR1 is set to zero. (Remember that SR1 is system register 1, or general register 8.)

Programs to be linked to or transferred to must have been loaded by Load or LYNX, but not by Link. M:LINK cannot be used to link to a command processor.

Pages obtained by M:GVP must have been freed before an M:LINK or M:LDTRC is issued or an error will result. Pages released from the user's assembled data area must have been restored before an M:LINK or M:LDTRC is issued or an error will result. That is, the virtual memory area that includes assembled data, program, and dynamic data must be continuous. Note that when DELTA has not been associated by "START...UNDER", the pages required for DELTA's symbols are acquired via M:GVP and must be released by ;k under DELTA.

If exit control has been established in the calling program, the issuing of an M:LINK procedure will cause control to be passed to the effective exit control routine (see M:XCON). Specifying EXIT or ERROR on the M:LINK procedure has no effect on either the exit control that may have been established in the calling program or any exit control that may be established by the called program.

The M:LINK procedure call has the form

```
M:LINK  'name'[,['account'][,'password']]
        [,(CMD,[ [*]addr])][,(ERROR)]
                           (EXIT )
```

where

'name' specifies the name of the load module to which control is to be transferred. The name is limited to 11 characters not including the quotes.

'account' specifies the account from which the load module is to be obtained.

'password' specifies the password associated with the load module.

CMD,[[*]addr] is the address of a TEXTC string to be passed (via J:CCBUF, the control command buffer contained in the JIT) to the called program. When in the on-line mode, the text string begins in byte zero of J:CCBUF and has a carriage return character (X'0D') appended to it. The character count of the resulting string is passed in JB:CCARS. In the batch mode, byte zero of J:CCBUF is a blank (X'40'), the string begins in byte one, and the remainder of J:CCBUF is blank filled. Text strings are limited to 79 characters. If longer strings are given, they are truncated to 79 characters. If CMD is specified but addr is not specified, then 'name' is placed in J:CCBUF following the above conventions. If CMD is not specified, J:CCBUF is not altered.

EXIT specifies that return should be made to the calling program following the M:LINK CAL when the linked-to program exits normally.

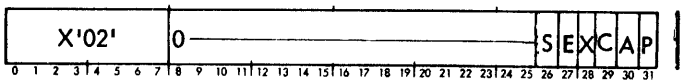
ERROR specifies that return should be made if the linked-to program errors or aborts (via M:ERR or M:XXX). ERROR also implies EXIT.

Calls generated by the M:LINK procedure have the form

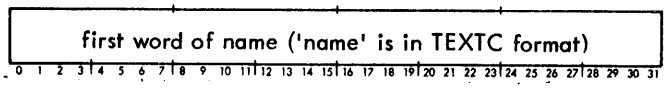
```
CAL1,8    fpt
```

where fpt points to word 0 of the FPT shown below.

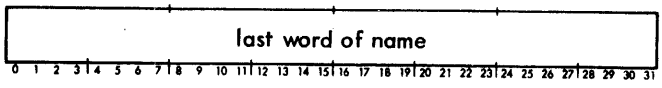
word 0



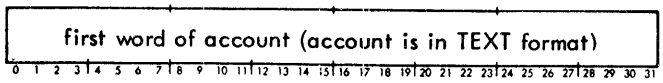
word 1



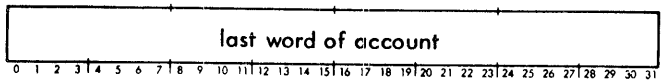
word n



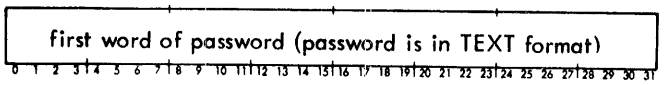
word n+1



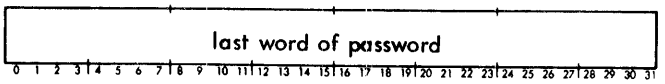
word n+2



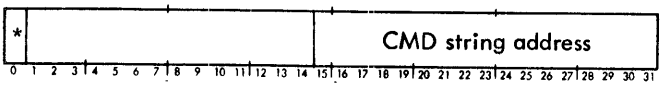
word n+3



word n+4



word n+5



where

S is set if CMD was specified and the command string address was NOT specified.

E is set if ERROR was specified.

X is set if EXIT was specified.

C is set if CMD was specified.

A is set if the account was specified.

P is set if the password was specified.

LOAD AND TRANSFER CONTROL

M:LDTRC The monitor routine LDTRC loads a specified load module (either one that had been partially executed and then saved as the result of an M:LINK procedure call, or else a "new" one), releases the core area used by the calling module, and transfers control to the starting address of the called module. If the called module was a previously saved program, it will be entered at a point immediately following the original M:LINK call, provided that the monitor-assigned file name of the previously saved program (communicated to the user via SR1) was stored into word 1 of the FPT associated with the M:LDTRC call prior to executing the M:LDTRC call.

The user's temporary and permanent load module libraries are searched for the specified load module. If it is not found or an I/O error occurs in executing the LDTRC routine, the job is aborted.

Any communication from the calling module to the called module must be accomplished through the general registers or common dynamic storage.

Programs to be linked to, or transferred to, must have been loaded by Load or LYNX, but not by Link.

Pages obtained by M:GVP must have been freed before an M:LINK or M:LDTRC is issued or an error will result. Pages released from the user's assembled data area must have been restored before an M:LINK or M:LDTRC is issued or an error will result. That is, the virtual memory area that includes assembled data, program, and dynamic data must be continuous. Note that when DELTA has not been associated by "START... UNDER", the pages required for DELTA's symbols are acquired via M:GVP and must be released by ;k under DELTA.

If exit control has been established in the calling program, the issuing of an M:LDTRC procedure will cause control to be passed to the effective exit control routine (see M:XCON).

No special shared processors other than public libraries may be associated with the program at the time an M:LDTRC is issued.

Shared processors may use M:LDTRC; however, special shared processors may not use M:LDTRC. M:LDTRC may also be used to initiate the processing of a TEL or CCI command (!XEQ) file. No special options need be specified on the M:LDTRC procedure call. If the following conditions are met, the system will terminate the current program and simulate an !XEQ command:

1. The file specified in the M:LDTRC procedure call must be either unkeyed (consecutive) or Edit keyed (KEYM = 3).
2. The program issuing the M:LDTRC must not have been loaded by an M:LINK, either directly or indirectly.
3. Command file processing must not be in effect at the time of the M:LDTRC.

This feature may be disallowed by the installation manager.

The M:LDTRC procedure call has the form

```
M:LDTRC    'name'[,['account'],'password']
```

where

'name' specifies the name of the load module or command file to which control is to be transferred. The name is limited to 11 characters not including quotes.

'account' specifies the account from which the load module is to be obtained.

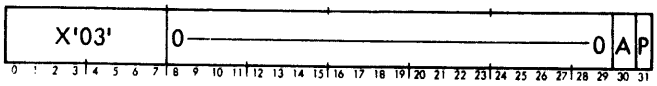
'password' specifies the password associated with the load module.

Calls generated by the M:LDTRC procedure have the form

```
CAL1,8    fpt
```

where fpt points to word 0 of the FPT shown below.

word 0



where A, P, and the subsequent words of the FPT are of the same form as shown previously for M:LINK.

GIVE TIME AND DATE

M:TIME The monitor TIME routine gives the time of day and the current date.

The M:TIME procedure call has the form

M:TIME [*]{address[,TMS]

where

address specifies the address of a four-word block where the time and date are to be stored. The four-word block where the time and the date are to be stored cannot lie within registers. (I/O buffers cannot lie within general registers and space for the time and the date is considered an I/O buffer.) The (EBCDIC) byte format of this block is shown below.

word 0

h	h	:	m

word 1

m	b	m	o

word 2

n	b	d	d

word 3

,	'	y	y

where

hh is the hour ($00 \leq hh \leq 23$).

mm is the minute ($00 \leq mm \leq 59$).

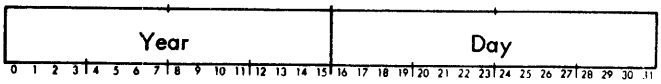
mon is the month (standard 3-letter abbr.).

dd is the day ($01 \leq dd \leq 31$).

yy is the year ($00 \leq yy \leq 99$).

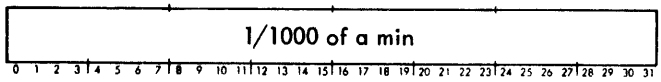
TMS indicates that the date and time (including resolution down to basic timer units) are to be returned in binary. If TMS is specified, SR1 will contain the year and Julian days, SR2 will contain the hour of day, minute of hour, second of minute, and number of 2-millisecond units since last thousandth of a minute, and SR3 will contain thousandths of a minute since last minute, as:

SR1



SR2

SR3



where

Year is a binary value, for example, 1970 is represented as X'46'.

Day is the Julian day of year represented in binary; for example, September 14 is represented as X'101'.

Hour is the hour of day (0-23).

Min is the minute of hour (0-59).

Sec is the second of minute (0-59).

TMS is the number of two millisecond units since the last 1/1000 of a minute (0-29).

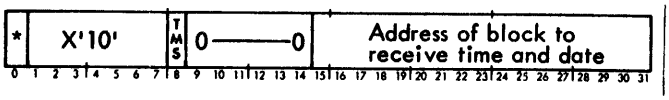
1/1000 min is the thousandths of minute since last minute. This is supplied because the monitor records time in 1/1000 minutes as the smallest increment in job accounting.

Calls generated by the M:TIME procedure have the form

CAL1,8 fpt

where fpt points to word 0 of the FPT shown below.

word 0



If TMS is specified, bit 8 is set to 1

TYPE A MESSAGE

{M:TYPE / M:MESSAGE} The monitor TYPE and MESSAGE routines output a specified message to the operator, on the operator's console typewriter. In batch operations, the two routines are equivalent.

The M:TYPE and M:MESSAGE procedure calls have the form

{M:TYPE / M:MESSAGE} (MESS, [*]address)

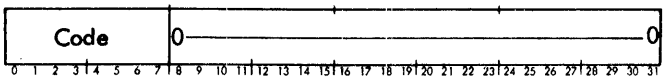
where MESS, [*] address specifies the word address of the beginning of the message to be typed. The first byte of the message must specify the number of characters in the message. The message may consist of not more than 136 alphanumeric characters. The address can be indirect to a register; however, the message cannot be in registers.

Calls generated by the M:TYPE and M:MESSAGE procedures have the form

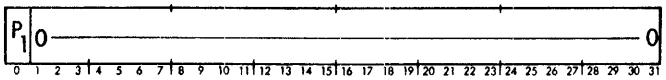
CAL1,2 fpt

where fpt points to word 0 of the FPT shown below.

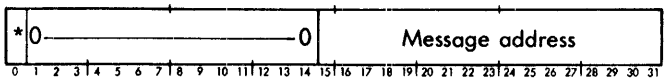
word 0



word 1 (parameter-presence-indicator word)



word 2



Code equals X'02' for M:TYPE FPTs, and zero for M:MESSAGE.

P_1 must be equal to 1, indicating that the following parameter word (word 2) is present.

The M:TYPE procedure supplies REFs for M:LL and M:OC.

REQUEST A KEY-IN

M:KEYIN The monitor KEYIN routine types a specified message to the operator and enables the operator's reply to be returned to the user's program.

The M:KEYIN procedure call has the form

M:KEYIN (MESS,[*]address)[,(REPLY,[*]address)] [,(SIZE,value)][,(ECB,[*]address)] [,(OC)]

where

MESS,[*]address specifies the word address of the beginning of the message to be output to the operator. The first byte of the word must specify the number of characters in the message. The address can be indirect to a register, however the message cannot be in registers. The message may consist of not more than 136 alphanumeric characters.

REPLY,[*]address specifies the word address of the location at which the beginning of the operator's reply is to be stored. The first byte of the word will (automatically) contain the number of characters in the reply. Indirect addressing can be made to a register; however, the message may not be in registers. If REPLY is not specified, the operator's reply is stored at the address specified by the MESS parameter.

SIZE,value specifies the maximum number of alphanumeric characters to be accepted from the operator's key-in and stored. If SIZE is not specified, a size of 60 characters is assumed.

ECB,[*]address specifies the word address of the Event Control Block (ECB) to be posted when a reply has been received. Bit 0 of the ECB is set to 1 until the reply has been received, then it is set to 0. Indirect addressing can be made to a register· however, the word may not be in a register.

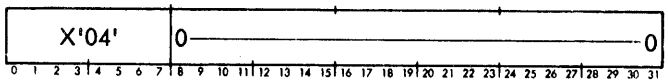
OC specifies that the message is to go to the operator's console and that the reply is to be received from the operator's console. (In the on-line mode, the message goes to the user's console and the reply is received from the user's console if this option is not specified.)

Calls generated by the M:KEYIN procedure have the form

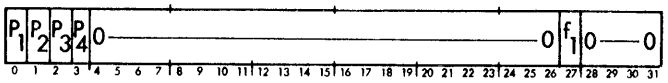
CAL1,2 fpt

where fpt points to word 0 of the fpt shown below.

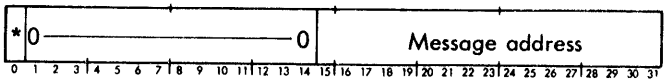
word 0



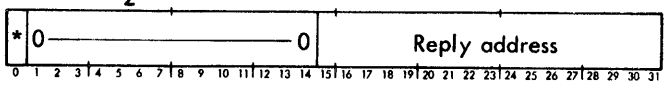
word 1



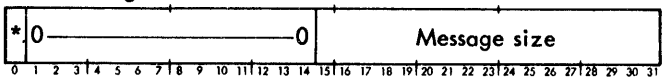
word 2 (P_1)



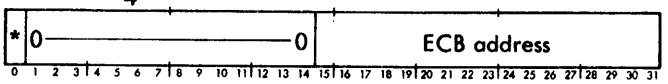
word 3 (P_2)



word 4 (P_3)



word 5 (P_4)



P_1 must be set to 1, indicating that word 2 is present. If f_1 is set, the OC option was specified.

The M:KEYIN procedure supplies REFs for M:LL and M:OC.

WRITE TO LISTING LOG

M:PRINT The monitor PRINT routine outputs a specified message on the listing log (LL) device.

The M:PRINT procedure call has the form

```
M:PRINT     (MESS,[*]address)
```

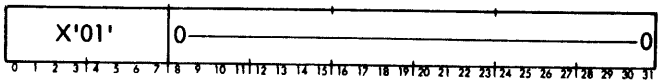
where MESS, [*]address specifies the word address of the location containing the beginning of the message to be output. The first byte must specify the number of characters in the message. The message may consist of not more than 136 alphanumeric characters (132 if LL is assigned to a line printer). Indirect addressing can be made to a register, however, the message may not be in registers.

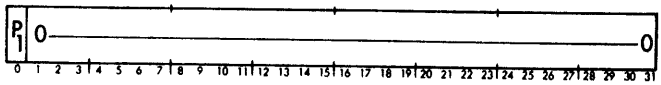
Calls generated by the M:PRINT procedure have the form

```
CAL1,2      fpt
```

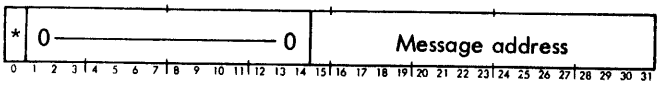
where fpt points to word 0 of the FPT shown below.

word 0





word 2



P_1 must be equal to 1.

The M:PRINT procedure supplies REFs for M:LL and M:OC.

SUSPEND PROGRAM

M:WAIT The WAIT routine causes the specified number of units of real-time to elapse before the next instruction in sequence is executed. It is used by programs that are to be executed periodically or must wait for files in use by another program. When execution of a program has been suspended by this CAL, it is resumed at a priority just higher than the current compute bound users.

The procedure call is of the form

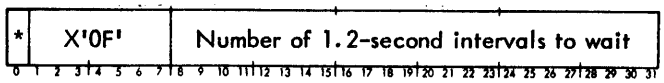
```
M:WAIT      units
```

where units specifies the number of 1.2-second intervals to elapse before the next instruction is executed and is treated modulo 24 hours.

Calls generated by the M:WAIT procedure have the form

```
CAL1,8      fpt
```

where fpt points to the FPT shown below.



EXCEPTIONAL CONDITION CONTROL PROCEDURES

The procedures described in this section allow the user program to take program control to a specific location for processing whenever one of the exceptional conditions occurs. Also described in this section are procedures that allow the program to return after processing the exceptional conditions, to exit completely from the program, and to simulate and test certain of the exceptional conditions.

Exceptional conditions may be divided into two categories: I/O related and the rest. They are distinguished both by the method of requesting control and by the difference in the environment available to the routine taking control.

I/O related conditions are requested by establishing error and abnormal addresses in the DCB, usually using the M:OPEN procedure, or by establishing them with each M:READ or M:WRITE. The resulting four types of conditions are described in Appendix B, "Monitor Error Messages". This appendix also describes the contents of the general registers when the error or abnormal routine takes control. Certain registers are made to contain the error codes and the DCB address and no environment is saved, as is the case with the rest of the exceptional conditions described below in this section. Two procedures in this section relate to I/O errors: M:MERC which allows an error or abnormal processing routine to return control of the error to the monitor, and M:XCON which takes control of all exceptional conditions including I/O errors.

Four of the pracedures are used to request exceptional condition control:

M:TRAP	to take control of machine traps.
M:STIMER	to take control after a time interval.
M:INT	to take control of the console interrupt or terminal break key.
M:XCON	to take control of all exceptional conditions.

These routines simply establish the address of a program routine to be entered when the condition occurs. The first three take precedence over the last — that is, if both TRAP and XCON are requested and a trap occurs, control will go to the address specified by M:TRAP.

M:XCON allows control to be taken in several cases not otherwise available: exceeding LIMIT card specifications,

operator aborts, line hangups, and program exits (M:EXIT, M:ERR, M:XXX).

When the exceptional condition processing routine specified by any of the above four procedures is entered, a common environment is established as described below. This consists of two parts:

1. The contents of the general registers on entry to the routine.

2. The machine environment at the time the exceptional condition occurred, which is stored in the pushdown stack in the user's task control block (TCB). (See Chapter 6 for other details of the TCB.)

Exceptional Condition Registers

On entry to exceptional condition processing routines the general registers have the contents shown below. For M:XCON processing only, additional registers are set as specified later in this chapter in Table 9. Unspecified registers have arbitrary contents.

Register 0 contains the address of the stack pointer doubleword in the TCB.

Register 1 contains the address of the PSD within the TCB stack.

When running on the Xerox 560, register 7 contains the contents of the internal control register Q30 decremented by one, which is effectively the address of the last successful branching type instruction executed by the machine immediately prior to the user's trap. If the address is below A000, the user has not executed a successful branch type instruction yet and the address reported is meaningless. (Q30 is described in detail in the Xerox 560 Computer/ Reference Manual, 90 30 76.)

Exceptional Condition TCB Stack Contents

On entry to an exceptional condition processing routine, the program environment at the time of occurrence of the condition may be found in the user's TCB pushdown stack. The environment consists of 20 or 21 words as shown in Figure 5.

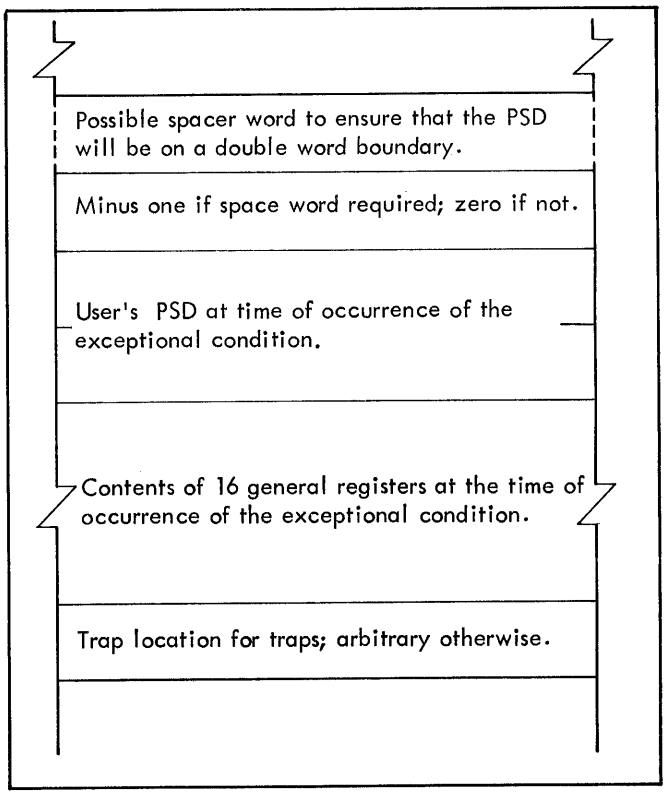


Figure 5. TCB Stack Contents on Exceptional Condition

SET TRAPS

M:TRAP The monitor TRAP routine sets and resets the trap conditions. Any trap condition that occurs while in the "trap" state causes control to go to a user's routine; any trap condition that occurs while in the "abort" state causes the user's program to be aborted. Maskable traps (i.e., fixed-point and decimal arithmetic) may be masked off so they do not occur, by placing them in the "ignore" state.

Each time the monitor TRAP routine is entered, the previous contents of the Program Trap Conditions (PTC) become available. The PTC always contains the current trap settings. Prior to the first M:TRAP, the mask bits are all zero. The first word of the PTC (returned in SR1) indicates which traps are in the "trap" or "abort" state and which maskable traps are in the "ignore" state.

The second word of the PTC (returned in SR2) contains the transfer address of the previous trap condition. Using the RESTORE option (see below) and some previous PTC, the trap settings can be restored to a previous setting.

The M:TRAP procedure call has the form

M:TRAP[transfer address][,(ABORT,traps)]

[,(TRAP,traps)][,(IGNORE,mask traps)]

[,(PERMIT,mask traps)]

or

M:TRAP (RESTORE,ptc address)

where

transfer address specifies the address of a user's routine to handle any traps caused by the TRAP option (see below).

ABORT,traps specifies the traps to be set to the "abort" state. Any combination of the following (separated by commas) may be specified:

Trap	Designated Trap(s)
ALL	All traps listed below.
CAL	Bad CAL.
DEC	Decimal arithmetic.
FP	Floating-point arithmetic.
FX	Fixed-point arithmetic.
NAO	Nonallowed operation.
PS	Push-down stack limit.
UI	Unimplemented instruction.

TRAP,traps specifies the traps to be set to the "trap" state. Any combination of the above may be specified.

IGNORE,mask traps specifies which maskable traps are to be set to the "ignore" state. Any of the following may be specified.

Mask T aps	Designated Trap(s)
FX	Fixed-point arithmetic.
DEC	Decimal arithmetic.
BOTH	Both of the above.

PERMIT,mask traps specifies which maskable traps are to be set to the "permit" state. Any of the traps shown may be specified.

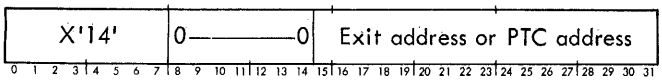
RESTORE,ptc address specifies the address of a previous PTC.

Calls generated by the M:TRAP procedure have the form

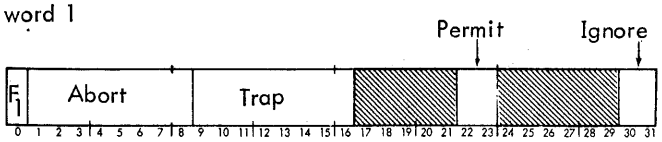
```
CAL1,8     fpt
```

where fpt points to word 0 of the FPT shown below.

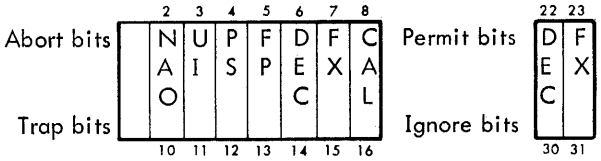
word 0



word 1



where control bits are shown below and F_1 specifies whether to restore (F_1 = 1) or to set new trap conditions (F_1 = 0).



When a user's trap routine is to be entered, due to the occurrence of a trap condition for a set-trap, the information stored in the user's stack and the contents of registers are as described at the beginning of this section. The top word of the stack contains the trap location (X'40' - X'4B'). The condition codes are those set by the hardware trap, so the user may determine which type of nonallowed operation caused the trap.

The trap return function (see "M:TRTN") can be used to return to the trapped program. The monitor does not increment the PSD when a trap occurs. If the PSD for the trapped program is to be changed, the user must change the PSD (in the user's stack) before control is returned to the trapped program.

Trap conditions are accumulated from one M:TRAP to succeeding ones, and the most recent transfer address is used regardless of which trap occurs.

SIMULATE A TRAP

M:STRAP The monitor STRAP routine simulates the occurrence of a trap condition. The trap condition and environment are specified by a block of temporary storage at the top of the user's TCB temp stack as described above under "Exceptional Condition TCB Stack Contents". The traps that may be simulated are locations X'40' through X'45' and X'48' through X'4B'. The monitor pulls the environment from the user's stack and simulates the occurrence of the specified trap with that environment.

The M:STRAP procedure call has the form

```
M:STRAP
```

Calls generated by the M:STRAP procedure have the form

```
CAL1,9     4
```

No FPT is required by M:STRAP.

SET INTERVAL TIMER

The interval timer is manipulated by the two monitor procedures M:STIMER and M:TTIMER. M:STIMER sets the interval timer and M:TTIMER tests the interval timer.

The basic timing unit used by CP-V is 2 milliseconds and it refers to user execution time. Thus, M:STIMER is used to give the user program control at a specified address after the specified amount of execution time has elapsed. I/O time does not count, nor does time spent in the monitor or executing other user programs.

The time calculation is made each time a time-slice is required (each QUAN if compute bound and running alone). Thus, the actual time of the STIMER transfer of control will vary from the time specified to later than that by an amount no greater than QUAN. The user may determine the exact elapsed interval by examining JIT cells that contain the exact time.

In summary, all M:STIMER and M:TTIMER calls are based on the closest number of interval timer units (2 milliseconds) with accuracy of -0 to +QUAN.

M:STIMER The monitor STIMER routine sets the interval timer with the specified value and specifies what action is to be taken. The interval is to be decremented only when the job issuing the M:STIMER procedure is operating, and only one such timing function may be in progress at any one time. An M:STIMER must be issued for each interval to be timed.

When the time expires, the PSD and registers are stored in the user's TCB stack as described in Figure 5. The user's program is entered at "exit address" with registers 0 and 1 set as described above. The interrupted program may be reinstated by use of the M:TRTN procedure.

The M:STIMER procedure call has the form

M:STIMER {(MIN, value) / (SEC, value) / (TUN, value)}, [*]exit address

where

MIN,value specifies (in minutes) the interval to which the timer is to be set.

SEC,value specifies (in seconds) the interval to which the timer is to be set.

TUN,value specifies (in interval timer units) the interval to which the timer is to be set.

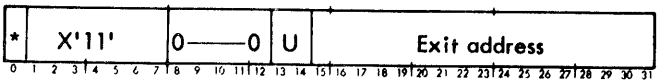
[*]exit address specifies the address of a routine to be entered when the specified interval ends. If omitted, zero is assumed.

Calls generated by the M:STIMER procedure have the form

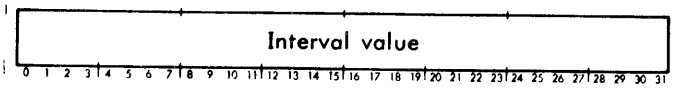
CAL1,8 fpt

where fpt points to word 0 of the FPT shown below.

word 0



word 1



where U specifies the type of units represented by the interval (0 means seconds, 1 means minutes, 2 means interval timer units).

TEST INTERVAL TIMER

M:TTIMER The monitor TTIMER routine causes an indication of the time remaining in the time interval (previously set by the STIMER routine) to be returned to SR1, and optionally allows the interval currently in effect to be canceled.

The M:TTIMER procedure call has the form

M:TTIMER [unit][, CANCEL]

where

unit specifies the units in which the time indication is to be returned to SR1. Unit may be either SEC, MIN, or TUN (see M:STIMER procedure); if omitted, TUN is assumed.

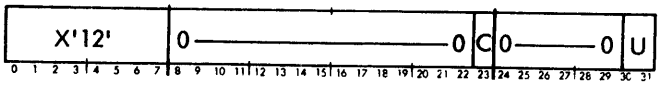
CANCEL specifies that the interval currently in effect is to be canceled. The exit address (see M:STIMER procedure) is ignored.

Calls generated by the M:TTIMER procedure have the form

CAL1,8 fpt

where fpt points to word 0 of the FPT shown below.

word 0



where

C specifies whether the interval in effect is (C = 1) or is not (C = 0) to be concluded.

U specifies the units in which the time indication is to be returned to SR1 (0 means seconds, 1 means minutes, 2 means interval timer units).

CONNECT CONSOLE INTERRUPT

M:INT The monitor INT routine may be called to connect a console interrupt (via the INT key-in) to a user's

program, allowing execution of the program to be controlled from the operator's console. When control is given to the INT routine, the information stored in the user's TCB stack is as described in Figure 5.

When a user's interrupt routine is entered, the condition codes are those loaded by the execution of the interrupt. The TRTN routine (see M:TRTN) may be used to restore control from a console interrupt.

The M:INT procedure call has the form

M:INT address

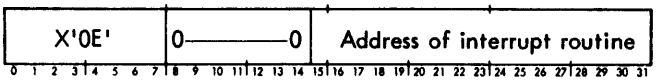
where address specifies the location of the entry to the user's console interrupt routine.

Calls generated by the M:INT procedure have the form

CAL1,8 fpt

where fpt points to word 0 of the FPT shown below.

word 0



EXIT CONTROL

This facility allows a user program to gain CPU control following an exit, abort, or line disconnect. This is usually for the purpose of cleanup or postmortems in the event of an unexpected problem.

Control is established via the M:XCON procedure. When an exit or abort occurs, control is passed to the location which was specified in M:XCON. In addition, the reason for the exit or abort is passed to the specified routine. Limits on output and time are reestablished to control the exit routine. Exit control entries may be nested by the user.

M:XCON Exit control is established with the M:XCON procedure, which has the form

M:XCON address [,LAST]

where

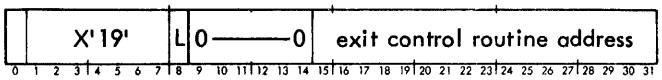
address specifies the address of a routine or a location to be entered upon exit of the current program whether normal or abnormal. No indirect address is allowed.

LAST specifies that this exit control request is intended to be the last one and subsequently no other exit control request will be honored unless issued while processing an exit. Thus, the program specifying LAST will be the first to receive control on exit.

Calls generated by the M:XCON procedure have the form

CAL1,8 fpt

where fpt points to word 0 of the FPT shown below:



where bit 8 is set if the LAST option is specified.

The exit control request may be unsuccessful if some previous M:XCON has specified the LAST option. In this case, exit control is not established and CC1 is set on return from the CAL. If a preceding part of the program has established exit control, the exit control routine address is returned in SR1 as is done for the M:TRAP CAL. This address may be saved so that the exit control routine may reestablish exit control to its previous entry point after performing the desired cleanup. The exit control may be nested.

The absence of the LAST option will imply that the exit control is no longer intended to be the last request even if it has been previously designated to be such. If the exit routine address is zero, exit control is reset. Thus, there is an inherent difference between the procedures M:XCON 0 and M:XCON 0,LAST, although the application of the latter should be directed with special purpose in mind.

Returns from the exit control routine may be performed via the M:TRTN service with the XCON option specified. Although exit control is similar to trap control, the two are kept separate to provide the user the convenience of different addresses for each function and to eliminate need to pass through one exit control routine to get to the other (e.g., through the trap control to obtain the exit control address).

Entry to the Exit Control Routine. Conditions that cause control to be passed to the established exit control routine are described in Table 8.

I/O errors are treated first by established error and abnormal routines associated with the I/O. The exit control routine is entered only if the program would have exited – that is, the error is not taken care of by a program error routine.

If a program processes only some of the errors and returns control via M:MERC, then control will pass through an established exit control routine.

Trap control is similar. A program establishing control via M:TRAP obtains control when a trap occurs for all requested traps. Those not requested will result in an error exit and be passed to exit control, if established.

The exit control facility imposes some restrictions on the linking process to a load module from a calling program. If an exit control has been established in the calling program, the issuing of a M:LINK procedure will cause control to be passed to the effective exit control routine. In the

Table 8. Exits to the Monitor

Condition	Monitor Action
Class I Normal exit from user program (M:EXIT). Abnormal exit from user program (M:ERR or M:XXX). Transfer to another load module (M:LINK, M:LDTRC). I/O error not handled by the user. Operator errored the user. Monitor detected error. User program trap.	Users current limits are not modified. No time limit is imposed.
Class II A resource limit has been exceeded (RAD, time, etc.).	Users current limits are incremented by a fixed amount (SYSGEN specified). Time limit is imposed for batch jobs, none for on-line jobs.
Class III Line disconnect or operator abort.	Users current limits are incremented by a fixed amount (SYSGEN specified). Time limit is imposed for both batch and on-line jobs.

exit control routine, desired actions can be taken upon determining the fact that entry to the routine is caused by a linking call. If the linking process is still deemed necessary, the M:LINK procedure can be reissued (by employing the EXU instruction) in the exit control routine. The Monitor will detect such a situation and will cause the linking process to occur. Exit control is then automatically reset before the transfer to the called program. This new level of control in the usage of linking to a load module can be regarded as a potentially powerful feature for a library routine where the necessary supervisory tasks can be performed upon a user program. Once the linking process is underway, the currently applicable restrictions are in effect. That is to say, if the called process is not successfully completed (for example, a limit is exceeded while executing the called routine), control will not be passed to the exit control routine in the called program. If the called program correctly executes a M:LDTRC procedure to return control to the caller, control will return to the exit control routine (which previously intercepted and then reexecuted the original M:LINK). The exit control routine must then reestablish exit control via M:XCON, if so desired. User programming conventions can be adopted to maximize security and reliability through the use of exit control. For instance, programs which are linked to should take exit control and always return to the caller in abort or abnormal situations.

Once exit control is in effect, previously established timer and break controls (i.e., by M:STIMER and M:INT) are reset. They can be reestablished in the exit control routine if so desired.

Limits: Standard system exit control limit increments are established by SYSGEN. When an unconditional abort event occurs (Classes II and III – limit exceeded, line disconnect, or operator abort), the users current limits are incremented by the exit control values. An exception is the time limit, which is not treated as an increment. When maximum time must be set (see Table 8), the exit control default is stored as the new maximum run-time limit. Because the limits are established only for the exit control entry first encountered, the system is protected from looping exit control routines by the SYSGEN established limits for exit routines since some limit will eventually be exceeded.

If an exit condition should again occur while the user is processing the exit control routine, different actions,

depending on the current and existing conditions, are dictated by the monitor as follows:

1. A Class I type exit, while the user is processing a Class I, Class II, or Class III exit condition, will cause control to be passed to the currently effective exit control routine without establishing the processing limits.

2. A Class II type exit, while the user is processing a Class I exit condition, will cause control to be passed to the currently effective exit control routine.

   A Class II type exit, while the user is processing a Class II exit condition, will cause the batch user to be unconditionally logged off. In the case of an on-line user, control will be passed to the current command processor. This is the case where the user has exhausted the extended processing capabilities granted to him by the system. A Class II type exit, while the user is processing a Class III exit condition, will cause the user, whether batch or on-line, to be unconditionally logged off.

3. A Class III type exit, while the user is processing a Class I or Class II exit condition will cause control to be passed to the currently effective exit control routine. In the case of processing a Class I exit condition, the extended processing limits will be appropriately set up.

   A Class III type exit, while the user is processing a Class III exit condition will cause the user to be unconditionally logged off.

Information Provided the Exit Control Routine. On entry to the exit control routine, registers are set to indicate the cause of the exit. In addition, several values are returned, each right-justified in an otherwise zero register. Register contents are described in Table 9.

Bits that are set correspond to the bits currently established in JIT. Other bits may be assigned meaning from time-to-time so no code should be written that depends explicitly on zero values.

Before control is passed to the exit control routine, the run status, error code, and error subcode are cleared in the JIT.

Three cases will cause the exit control routine to be entered with zero run status – M:EXIT, M:LINK, and M:LDTRC. The specific cause may be identified by examining the instruction pointed to by the exit control environment.

When the exit control routine is entered, the PSD and general registers of the user program are placed in the TCB temp stack as described in Figure 5. The exit control routine is entered in the slave mode with decimal and fixed-point traps inhibited.

The environment may be returned to for additional execution via the M:TRTN procedure. For example, if a print limit is exceeded, it may be desirable to return to the program to finish a current page of output before taking a final exit. Returning to the users environment has questionable meaning if the user program has issued M:EXIT, M:ERR, or M:XXX CAL.

To resume execution of a program after receiving exit control, the M:TRTN service with the XCON option specified should be used. This will serve to signal that exit control is no longer in process and will ensure the proper use of the LAST option on the M:XCON CAL.

If there is no TCB stack or if it is full, the exit control routine is entered without placement of the user's environment, with bit zero of register 12 set to 1, and with the PSD from the environment placed in registers 2 and 3.

If an exit control routine wishes to exit unconditionally from the current job step, it can issue an exit, error, or abort CAL. If communication to other steps in the same job is desirable, the step condition code may be set via an option on the M:EXIT, M:ERR, or M:XXX procedure.

EXITS TO THE MONITOR

To enable the monitor to provide continuous system operation, control of the system must be returned to the monitor by each user's program when it has terminated execution of its operations for any reason. The monitor provides three exit returns by which a user's program may relinquish control after termination. The monitor performs an implicit "Close" for any DCBs that are open when a program terminates via one of the three exit returns.

M:EXIT An EXIT return should be used when the user's program has completed its operations in a normal manner. When control is returned via the EXIT routine, the monitor either returns to the exit control routine in the user's program or performs any PMDI dumps that have been specified for the program and proceeds to the next control command. Return to the user's program occurs only if exit control has been established in the user's program.

The M:EXIT procedure call has the form

M:EXIT [scc / *address]

where

scc specifies a new value for the step condition code (see STEP Control Command).

*address specifies the address of a location in which the step condition code and the exit type are contained.

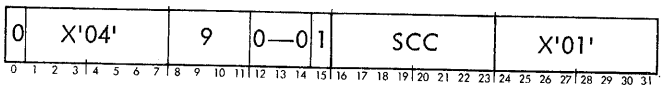
Calls generated by the M:EXIT procedure with no options specified have the form

```
CAL1,9     1
```

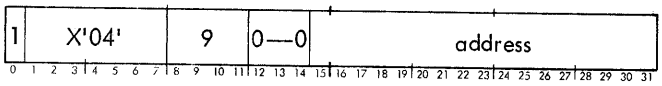
Table 9. Register Contents for Exit Control

Register	Name	Contents
8	Run Status	If exit was normal, following bits reset; otherwise, appropriate bit is set as follows: 24 An M:ERR was issued by the program or processor. 25 An M:XXX was issued by the program or processor; or an on-line user program with exit control entered TEL via Control Y, then implicitly aborted (e.g., by calling PCL). 26 The operator errored the program (ERROR or E key-in). 27 The operator aborted the program (ABORT or X key-in). 28 Terminal has hung up or line has disconnected. 29 Some limit was exceeded (see Register 9). 30 I/O Error (see Registers 10 and 11). 31 Trap (see Registers 10 and 11).
9	Limits	If bit 29 of Register 8 is set, then the limit exceeded is identified by 23 Disk granule allocation (net permanent). 24 CPU time. 25 Scratch tapes. 26 Temporary disk granules acquired. 27 Permanent disk granules acquired. 28 Diagnostic print pages output. 29 User-generated print pages output. 30 Processor-generated print pages output. 31 Punch cards output.
10	Error Code	See Appendix B.
11	Error Subcode	See Appendix B.
12	Stack Flag	Bit 0 reset if environment is placed in the stack; set if not. Bit 1 set if the exit is from M:LINK or M:LDTRC.
0	TCB Address	User's TCB address.
1	PSD Address	PSD address (in user's TCB).
2, 3	PSD	PSD at exit point if environment is not placed in TCB.

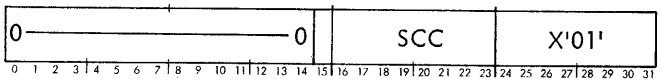
The CAL code generated by the M:EXIT procedure with scc specified is of the form of a CAL1,9, as follows:



The CAL code generated by the M:EXIT procedure with *address specified is of the form of a CAL1,9, as follows:



with the location pointed to by the specified address containing the following:



Bit 15 in the address field, if set, indicates that the step condition code value is to be established.

No FPT is required by M:EXIT.

M:ERR An ERR return is used when an error has occurred during program execution and the user wants the monitor to discontinue execution of the current program. When control is returned via the ERR routine, the monitor outputs the message

JOB ERRORED BY USER AT xxxxx

where xxxxx is the address of the last instruction executed in the program.

This message plus the contents of the current register block and Program Status Doubleword (PSD) are listed on the LL device. The PSD contains the address of the last instruction executed in the errored program. The step condition code is set to 4.

The monitor then either returns to the exit control routine in the user's program or performs any specified postmortem dumps and proceeds to the next job step. Return to the user's program occurs only if exit control has been established in the user's program.

The M:ERR procedure call has the form

M:ERR [scc / *address]

where

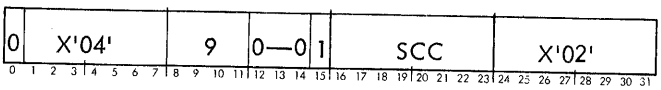
scc specifies a new value for the step condition code (see STEP Control Command).

*address specifies the address of a location in which the step condition code and the exit type are contained.

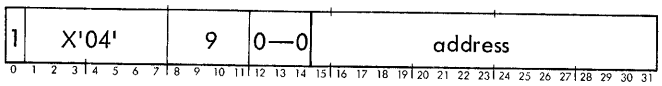
Calls generated by the M:ERR procedure with no options specified have the form

CAL1,9 2

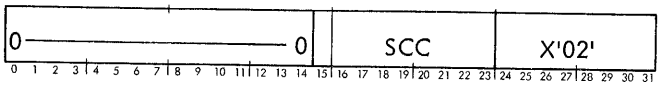
The CAL code generated by the M:ERR procedure with scc specified is of the form of a CAL1,9, as follows:



The CAL code generated by the M:ERR procedure with *address specified is of the form of a CAL1,9, as follows:



with the location pointed to by the specified address containing the following:



Bit 15 in the address field, if set, indicates that the step condition code value is to be established.

No FPT is required by M:ERR.

M:XXX The XXX (abort) return is used when an irrecoverable error has occurred and the current job step is to be aborted. When a job step is aborted, the monitor lists the message

JOB ABORTED BY USER AT xxxxx

where xxxxx is the address of the last instruction executed in the program.

This message plus the contents of the current register block and Program Status Doubleword (PSD) are listed on the LL device. The PSD contains the address of the last instruction executed in the aborted program. The step condition code is set to 6.

The monitor then either returns to the exit control routine in the user's program (if exit control has been established in the user's program) or performs any specified postmortem dumps and proceeds to the next job step.

The M:XXX procedure call has the form

M:XXX [scc / *address]

where

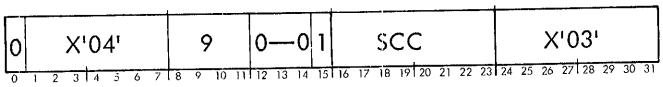
scc specifies a new value for the step condition code (see STEP Control Command).

*address specifies the address of a location in which the step condition code and the exit type are contained.

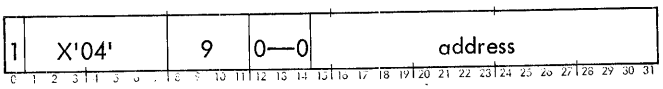
Calls generated by the M:XXX procedure with no options specified have the form

```
CAL1,9    3
```

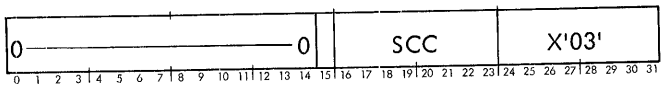
The CAL code generated by the M:XXX procedure with scc specified is of the form of a CAL1,9, as follows:



The CAL code generated by the M:XXX procedure with *address specified is of the form of a CAL1,9, as follows:



with the location pointed to by the specified address containing the following:



Bit 15 in the address field, if set, indicates that the step condition code value is to be established.

No FPT is required by M:XXX.

EXIT FROM TRAP, INTERRUPT, TIMER, OR EXIT CONTROL ROUTINE

M:TRTN The monitor TRTN routine restores control to a trapped program.

The M:TRTN procedure call has the form

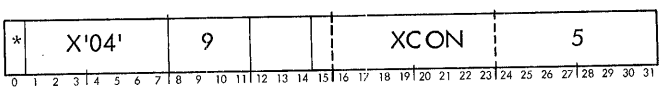
```
M:TRTN [*][XCON]
```

where XCON identifies an exit control return.

Calls generated by the M:TRTN procedure with no options specified have the form

```
CAL1,9    5
```

The CAL code generated by M:TRTN, with the XCON option specified, has the form of a CAL1,9, as follows:



where XCON specifies a normal M:TRTN (XCON = 0), or an exit control return (XCON = 1).

M:TRTN pulls the last standard environment from the users TCB, as described under "Exceptional Condition TCB Stack Contents". The user should note that he may effect M:TRTN himself (so long as the XCON option is not required) by loading the 16 registers from the TCB, adjusting the stack pointer appropriately, loading condition codes from the PSD, and branching to the location specified in the PSD. This can be a lot faster than using M:TRTN. If a trap return, M:TRTN, is attempted with no environment in the stack, then the usual error code, A301, results and exit control takes effect.

If M:TRTN XCON is used when not in an exit control routine, return is to the program at CAL+1 with CC1 set.

MONITOR ERROR CONTROL

M:MERC The monitor MERC routine enables the user's program to specify an error or abnormal code and subcode (see Appendix B) in SR3 and have the monitor handle it, overriding any user's abnormal or error routines that might otherwise apply.

The call can also be used to return control to the monitor from a user's error or abnormal routine if that routine can handle only certain codes. If a user's error or abnormal routine is called to handle an abnormal or error condition beyond its capability, it must leave the contents of communication registers SR1 and SR3 intact and call the MERC routine to handle the condition.

The M:MERC procedure call is of the form

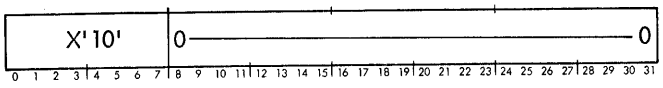
```
M:MERC
```

Although no parameters are specified in this call, communication register SR3 must contain the error or abnormal code in bit positions 0-7 and subcode in bit positions 8-14 when MERC is entered. For I/O error or abnormal conditions, the address of the associated DCB must also be contained in SR3, in bit positions 15-31. This information is placed in SR3 by the monitor, when an error or abnormal condition is detected. When bit positions 0-7 of SR3 contain X'40'-X'FF', the current job is aborted and the error code is used to obtain the appropriate error message from the system error message file. It should be noted that SR1 will contain the address + 1 of the offending CAL when the error or abnormal address is entered. It must also be preserved so that MERC can return properly.

Calls generated by the M:MERC procedure have the form

```
CAL1,2    fpt
```

where fpt points to word 0 of the FPT shown below.



DATA MEMORY MANAGEMENT

CP-V permits either relative or specific allocation of core memory in the data area of a user program. When this type of allocation is used, pages may be allocated sequentially from either end of unallocated virtual memory (Figure 6). Pages allocated from the lowest toward the highest unallocated page address are called dynamic pages. Pages allocated from the highest toward the lowest unallocated page address are called common dynamic pages or simply common pages. Dynamic and common pages may not overlap. If there is an attempt to overlap these pages, an error indication will result.

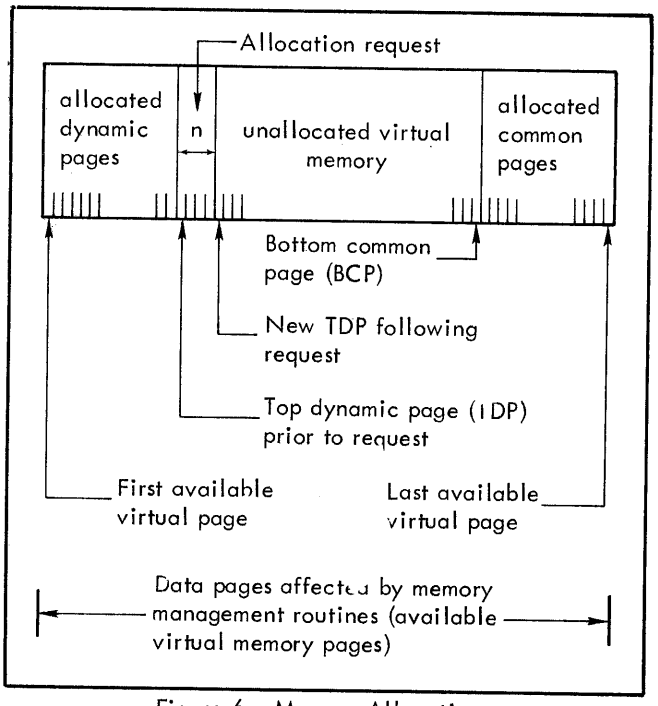


Figure 6. Memory Allocation

Specific allocation allows a user program to get or release any page in the data area by reference to the address of the first word on the page. If an attempt is made to get pages that have been allocated or to release pages that have been released, an error indication will result.

Relative-allocation CALs may not be used to allocate a page already allocated by a specific allocation CAL. Specific-allocation CALs may not be used to release pages allocated by a relative allocation CAL.

The number of physical pages that may be allocated for all purposes is limited. This limit is initially set by SYSGEN and may be modified by the performance control program (see CP-V/SM Reference Manual, 90 16 74).

No program may acquire more than 185 pages of physical memory in addition to JIT. This limit includes space for the program, assembled data, dynamically obtained pages, blocking, index, cooperative buffers, and DCBs. Not included in this limit are associated libraries or shared processors (e.g., compilers, debuggers).

No program may acquire more pages than the number of pages on the machine minus the size of the monitor (including any overlay required by the program). This limit includes space required for shared processors and libraries.

In addition, users are constrained by the authorized user file and the LIMIT control command.

GET COMMON LIMITS

M:GL The GL routine returns the lowest and highest word addresses of the common data area presently allocated. The lowest address is returned in SR1 and the highest address is returned in SR2. If no pages have been requested before, or if all pages requested have been returned, SR1 and SR2 are equal.

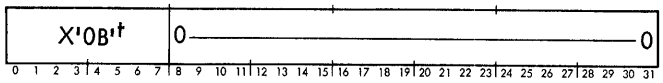
The M:GL procedure call is of the form

```
M:GL
```

Calls generated by the M:GL procedure have the form

```
CAL1,8    fpt
```

where fpt points to the FPT (function parameter table) shown below.



GET DYNAMIC DATA LIMITS

M:GDDL The GDDL routine returns several limiting values of the dynamic data area. The dynamic data area is the region of memory from which dynamic, common dynamic, and, sometimes, virtual pages are allocated. The address of the first word of this area is returned in SR1 and the address of the last word in SR2. Returned in SR3 is the maximum number of pages that a user could obtain through the M:GP, M:GCP, and M:GVP CALs. This value does not include those pages already allocated (for example, if the user already obtained all but one page, a value of 1 would be returned).

---

t Hexadecimal numbers in this manual are expressed in the form X'n', where n is the hexadecimal number.

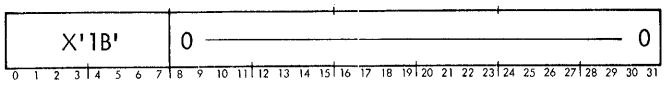
The M:GDDL procedure is of the form

M:GDDL

Calls generated by the M:GDDL procedure have the form

CAL1,8 fpt

where the fpt points to the FPT shown below:



GET COMMON PAGES

M:GCP The GCP routine allocates a specified number of pages at successively lower addresses of common storage starting with the next lower page (BCP). It also decrements that page number.

Pages are obtained and allocated at successively lower addresses beginning with that of BCP until

1. The required number of pages are obtained.
2. The installation-set or user-set limit on the number of physical core pages is reached or a page already allocated via M:GP or M:GVP is encountered.

This information returned for each of the two cases is

Case	CC1	SR1	SR2
1	0	Number of pages allocated	address of lowest word allocated
2	1	Number of pages allocated	

In each case, BCP is the page number of the next lower page yet to be allocated.

Access codes for the allocated pages are set to 00 (read, write, and execute).

The M:GCP procedure call is of the form

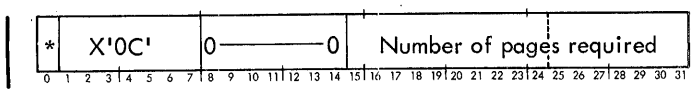
M:GCP [*] pages

where pages specifies the number of memory pages by which common storage is to be extended.

Calls generated by the M:GCP procedure have the form

CAL1,8 fpt

where fpt points to the FPT shown below



FREE COMMON PAGES

M:FCP The FCP routine releases a specified number of pages at successively higher addresses of common storage beginning with the current lowest page (BCP+1). It also increments that page number.

Pages are released beginning at BCP+1 toward successively higher addresses until

1. The requested number of pages have been released.
2. The last available virtual page is released.

In the first case, CC1 is set to zero. In the second case, CC1 is set to one. The number of pages released is returned in SR1.

Pages released by FCP have access codes set to 11 (no access). Any subsequent reference to these pages will result in a trap.

The M:FCP procedure call is of the form

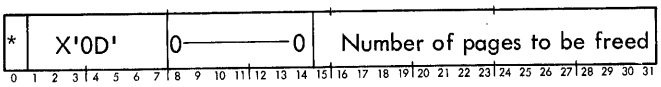
M:FCP [*] pages

where pages specifies the number of pages to be freed.

Calls generated by the M:FCP procedure have the form

CAL1,8 fpt

where fpt points to the FPT shown below.



GET DYNAMIC PAGES

M:GP The GP routine allocates a specified number of pages beginning with the next higher page (TDP) of dynamic storage. It also increments that page number until

1. The required number of pages are allocated.
2. The installation-set or user-set limit on the number of physical core pages is reached or a page already allocated via M:GCP or M:GVP is encountered.

The information returned for the two cases is

Case	CC1	SR1	SR2
1	0	Number of pages allocated	address of lowest word allocated
2	1	Number of pages allocated	

In each case, TDP is the page number of the next higher page yet to be allocated.

Access codes for all allocated pages are set to 00 (read, write, and execute).

The M:GP procedure call is of the form

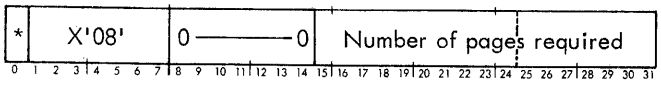
M:GP [*] pages

where pages specifies the number of additional pages requested.

Calls generated by the M:GP procedure have the form

CAL1,8 fpt

where fpt points to the FPT shown below.



FREE DYNAMIC PAGES

M:FP The FP routine releases a specified number of pages at successively lower addresses of dynamic storage beginning with the current highest page (TDP-1). It also decrements that page number until

1. The requested number of pages have been released.
2. The first available virtual pag is released.

In the first case, CC1 is set to zero. In the second case, CC1 is set to one. The number of pages released is returned in SR1.

Pages released have their access codes set to 11 (no access) and any subsequent reference to these pages will result in a trap.

The M:FP procedure call is of the form

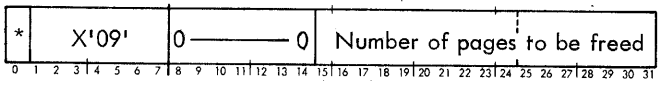
M:FP [*] pages

where pages specifies the number of pages to be freed from use by the user program.

Calls generated by the M:FP procedure have the form

CAL1,8 fpt

where fpt points to the FPT shown below.



GET VIRTUAL PAGE

M:GVP The GVP routine allocates a specific page of virtual memory to the user program. If the request is allowed, access for the page is set to 00 (read, write, or execute) and CC1 is set to zero. The result is disallowed if

1. The installation-set or user-set limit on number of pages allowed would be exceeded.
2. The page has already been allocated.
3. The page requested is outside the limits of unallocated virtual memory.

In all three cases, CC1 is set to one and no page is allocated.

The M:GVP procedure call is of the form

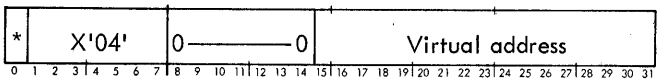
M:GVP [*] virtual address

where virtual address specifies the address of the first word in the virtual page desired.

Calls generated by the M:GVP procedure have the form

CAL1,8 fpt

where fpt points to the FPT shown below.



FREE VIRTUAL PAGE

M:FVP The FVP routine is called to release a specific page of virtual memory. The indicated page is released and CC1 is set to zero except when the request is for a page that does not belong to the user, in which case, CC1 is set to one and no page is released. The number of pages released (0 or 1) is returned in SR1. Pages of data that are loaded with the program or processor (those assembled with the program) may be released using this mechanism. These are the pages below the first available virtual page.

The M:FVP procedure is of the form

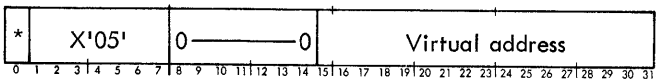
M:FVP [*] virtual address

where virtual address specifies the address of the first word in the virtual page to be released.

Calls generated by the M:FVP procedure have the form

CAL1,8 fpt

where fpt points to the FPT shown below.



SET MEMORY PROTECT

M:SMPRT The SMPRT routine sets the access codes on pages owned by the user. It does not affect write locks.

The M:SMPRT procedure call is of the form

```
M:SMPRT    value, [*] from[,[*]to]
```

where

value specifies the value of the requested memory protection setting and may be any one of the following:

Value	Access
0	Read, Write, Execute
1	Read, Execute
2	Read
3	No access

from specifies the address of the first page to which the specified setting is to apply. If no "to" is specified, only this page will be affected.

to specifies the address of the last page to which the specified setting is to apply.

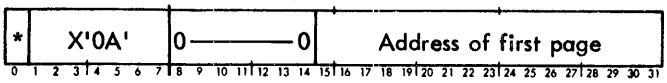
This procedure call may not be used to reduce the amount of protection on a given page from its initial value. Thus, data areas may be given any protection value and program areas may be given any protection value except 0. JITs and DCBs may not be set to a value lower than two.

Calls generated by the M:SMPRT procedure have the form

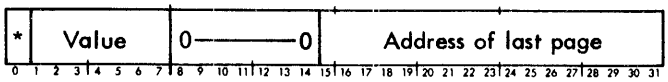
```
CAL1,8     fpt
```

where fpt points to word 0 of the FPT shown below.

word 0



word 1



CHANGE VIRTUAL MAP

M:CVM The CVM routine allows certain processors and privileged programs, such as those of the system programmer, to examine, display, or change data portions of real physical core.

Restrictions

1. The user issuing this CAL must have the proper privilege level: X'80' for read access and X'B0' for store access (see CP-V/SM Reference Manual, 90 16 74).
2. The virtual address (VA) must not already be assigned to the calling program.
3. The real-core page must exist.

If any of these restrictions are violated, CC1 is set and control is returned to the user without action.

The number of the real-core page requested is placed in the specified virtual-page entry in the memory map of the calling program. The user specifies both in terms of addresses, however.

Access for the page is set to read or data depending on the user privilege, unless PROT is specified. When PROT is specified, access is set to read.

The M:CVM procedure call is of the form

```
M:CVM [*] rpa,[*]va [,PROT]
```

where

rpa is the address of the real-core page to be placed in the user's map.

va is the virtual-page address in the calling program onto which rpa is to be mapped.

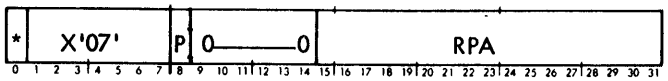
PROT specifies that the virtual-page is to have read and execute access, regardless of the user's privilege level.

Calls, generated by the M:CVM procedure to change the map, must have the form

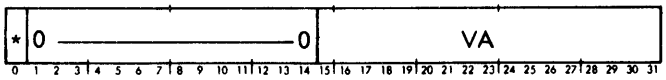
```
CAL1,8     fpt
```

where fpt points to word 0 of the FPT shown below.

word 0



word 1



If P is set, the access code for the virtual page is to be set to 01 (read and access instructions). If P is not set, the access code is to be set to 00 (read, access instructions, and write).

ENQUEUE/DEQUEUE RESOURCES

The enqueue/dequeue feature permits users to coordinate the use of a resource among themselves. This includes, but is not limited to, coordinating shared use of random files for simultaneous update by several jobs.

The use of enqueue/dequeue presumes the existence of a resource that may (or may not) contain elements and that both the resource and its elements have names that are known to all programs using that resource. For example, the resource might be a random file called DATAFILE and the elements of the resource might be the granules in the file, the granules being referred to as 001, 002, 003, etc.

The use of enqueue/dequeue does not allocate any real, physical resource. The resource and element names have no meaning to the monitor other than to identify the pseudo-resources and elements that are being queued upon. However, properly used, the enqueue/dequeue feature permits programs to coordinate the use of real, physical resources.

When a user enqueues on a particular resource/element, he is effectively put in a queue to wait for availability of the resource element. He remains in the queue until he specifically dequeues on the particular resource/element or until the monitor automatically dequeues him (either at the end of the job step or at the end of the job, dependent upon options he has specified).

An option is provided in the enqueue service that allows the user to enqueue simply to obtain a position in the queue even though he is not ready to use the resource/element. This generally ensures him of priority in the queue over subsequent users who want the entire resource.

A resource/element may be requested either for EXCLusive use or SHAREd use. If it is requested for EXCLusive use, no other user may access the resource/element while the EXCLusive user has it. If it is SHAREd, all users requesting SHAREd use of the resource/element may have simultaneous access.

When a user enqueues on a particular resource/element, either the user will be put to sleep until the resource/element is available to him or an Event Control Block (ECB) is flagged to mark when the resource/element becomes available to that user. In the latter case, the user can use the M:CHECKECB service to determine when the resource/element has become available. Once the resource/element becomes available to the user, it remains available to him until he dequeues or is dequeued by the monitor (as described above).

The following examples use a data management application with a random file data base as the resource and the granules of that file as the elements. Assume the data base is in file DATAFILE in account THATACCT. Further, assume that all data management users of DATAFILE.THATACCT refer to that file for enqueue purposes, as QFILE, and the granules of DATAFILE.THATACCT are referred to as 001, 002, 003, etc. Before reading any data that is contained on granules 5 and 11, then, the program would enqueue on QFILE, 005 and QFILE, 011 with SHARE specified. If at the same time another user wanted data from granules 8 and 11, his copy of the program would enqueue on QFILE, 008 and QFILE, 011 with SHARE specified. Both readers would then proceed and when finished reading they would dequeue those resource/elements.

In the above sequence, no problems exist and the enqueue/dequeue feature was not needed. However, if, during the same period another user tried to update data, some of the data retrieved would not match other data retrieved (a sequence such as read 5, write 5, write 11, read 11 could occur, making the two reads be of different data). The updating program, therefore, would enqueue on QFILE, 005 and QFILE, 011 with EXCLusive specified. If the first user requested SHARE and the second requested EXCLusive, the second user would not be given access until the first user dequeued. But, if both the first and the second user were SHARE and the third user was EXCLusive, the first and the second would have simultaneous access and the third user would have to wait until both the others had dequeued.

On the other hand, assume a reader wanted granules 20 and 23 of DATAFILE.THATACCT and an updater wanted to change granules 12 and 18. The reader would enqueue on QFILE, 020 and QFILE, 023 with SHARE specified while the updater would enqueue on QFILE, 012 and QFILE, 018 with EXCLusive specified. Since these sub-queues do not conflict, both reader and updater could operate simultaneously.

If it were necessary to rebuild the file, the rebuilder would enqueue on the entire file with EXCLusive specified. The rebuilder would then have to wait until all current users dequeued and any new users would be blocked until after the rebuilder dequeued.

Two types of deadlock are possible with enqueue/dequeue: multi-queue and single-queue. A multi-queue deadlock occurs when two or more users have at least one resource/element and are waiting for at least one resource/element such that none of the users involved can ever get the resource/element they are waiting for. For example, user 1 has element A and is waiting for element B, user 2 has element B and is waiting for C, and user 3 has C and is waiting for A. Assuming that user 3 was the last to request an element, CP-V would detect this multi-queue deadlock when user 3 enqueued on A. At that point, user 3 would be given an error return (or aborted if no ERR address were given), and the enqueue request would be ignored. Normally, user 3 would then dequeue on element C, permitting user 2 to finish; user 2 would then dequeue on element B and C, permitting user 1 to proceed, etc. A possible alternative is that user 3 might be able to use element D, an alternate, complete his work and dequeue on C and D. In any event, user 3 is given the error indication and, therefore, user 3 is responsible for unblocking the deadlock.

The single-queue deadlock occurs when two or more users have been granted SHARE access to an element and two of the users are attempting to upgrade their access to EXCLusive. Since they cannot get EXCLusive access as long as there are SHARE users of the element, and since they have not relinquished their existing SHARE positions in the queue, deadlock is created. In this situation, the error is attributed to the second user requesting the upgrade to EXCLusive and that user is responsible for unblocking the deadlock. As with the multi-queue deadlock, the user receiving the error has his enqueue request ignored.

A third type of deadlock is possible when there are no remaining empty entries in the monitor's enqueue table. However, strictly speaking, this is not a deadlock because it is possible for the queues to unwind without special handling. This condition results in an error return, but with a different sub-code than a true deadlock.

Additional information on simultaneous file usage is given in Appendix F.

M:ENQ The M:ENQ procedure call allows a user to enqueue on a particular resource/element or test a particular resource/element for availability. It has the following format:

```
M:ENQ ([*]'qname',[*]'sname'[,STEP][,EXCL ])
                            [,JOB ][,SHARE]

  [ /WAIT                   \ ]
  [( NOWAIT,[*]ecb address   )][,(ERR,[*]address)]
  [,\TEST,[*]ecb address    / ]

  [,(ABN,[*]address)]
```

where

'qname' specifies the name of the queue (resource).

'sname' specifies the name of the sub-queue (element) of the queue or specifies one of the following (not in quotes):

ALL specifies that all sub-queues (elements) of the queue (resource) are to be enqueued.

NULL specifies that the user wants to queue on the resource but not on a particular element of the resource. The NULL specification essentially just reserves a place in the queue for the user. At a later time, the user may queue on an element or elements of the resource. Meanwhile, he has established priority over subsequent users that request ALL.

STEP specifies that use of this resource/element (qname/sname) applies only during this job step or during execution of this job. If STEP is specified, the resource element will automatically be dequeued by the monitor at the end of the job step or program execution unless it has already been dequeued by the user. If neither STEP nor JOB is specified, STEP is assumed.

JOB specifies that use of this resource/element (qname/sname) may continue throughout the job or on-line session. The resource/element will not be automatically dequeued at the end of the program's execution.

EXCL specifies exclusive use. No other user may use this resource/element until it is dequeued by this user. If neither EXCL nor SHARE is specified, EXCL is assumed.

SHARE specifies that the resource/element may be shared with other users that do not require exclusive use.

WAIT specifies that the program is not to resume execution until the resource/element has been made available to this user. WAIT, NOWAIT, and TEST are mutually exclusive. If neither WAIT, NOWAIT, nor TEST is specified, WAIT is assumed.

NOWAIT,[*]ecb address specifies that program execution is to continue regardless of whether or not the resource/element is available. It also specifies the address of the ECB to be associated with this M:ENQ procedure call. When the resource/element becomes available, the ECBP flag of the ECB will be set to one. (ECBs are described in the discussion of M:CHECKECB.)

TEST,[*]ecb address specifies that the resource/element is not to be queued, but rather is to be tested for availability. It also specifies the address of the ECB to be associated with this M:ENQ procedure call. (ECBs are described in the discussion of M:CHECKECB.) The return from an M:ENQ procedure that has TEST specified reflects whether or not the resource/element is available and whether or not the user has already enqueued on the resource/element. The ECBP bit of the ECB reflects whether or not the resource/element is available. If the resource/element is not available or if the user has already enqueued on the resource/element, the return is an abnormal return.

ERR,[*]address specifies the address at which execution resumes if an error condition is detected. Error codes for M:ENQ are listed in Table B-8, Appendix B.

ABN,[*]address specifies the address at which execution resumes if an abnormal condition is detected. Abnormal codes for M:ENQ are listed in Table B-7, Appendix B.

Calls generated by the M:ENQ procedure have the form

CAL1,2 fpt

where fpt points to word 0 of the FPT shown below.

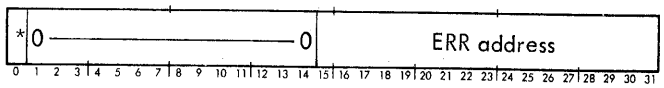
word 0

Bits 0–7	Bits 8–31
X'08'	0 ——— 0

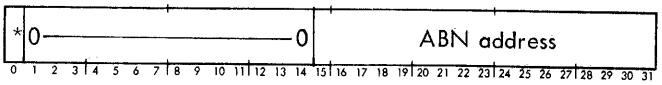
word 1

Bit 0	Bit 1	Bit 2	Bits 3–29	Bit 30	Bit 31
P_1	P_2	P_3	0 ——— 0	F_2	F_1

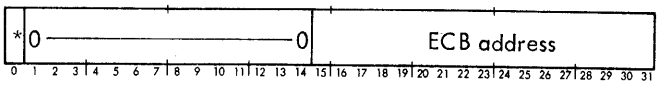
word 2 (P1)



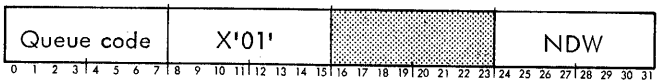
word 3 (P2)



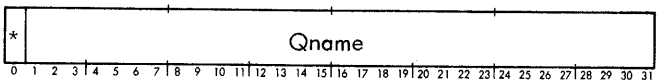
word 4 (P3)



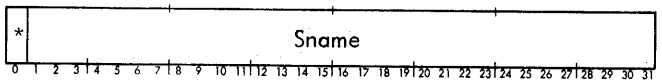
word 5



word 6



word n



where

f_1 is set to one if the NOWAIT option was specified.

f_2 is set to one if the TEST option was specified.

Queue code has the following meanings:

X'01' for an EXCL, STEP request

X'03' for a SHARE, STEP request

X'05' for an EXCL, JOB request

X'07' for a SHARE, JOB request

NDW contains the number of words reserved to contain the qname and sname names (in words 6 and following).

Qname specifies the qname in TEXTC format.

Sname specifies the sname in TEXTC format. The sname starts in the first word following qname (or the indirect address specification of qname). If the count byte of this field is set to X'7F', the ALL option was specified. If the count byte is set to X'40', the NULL option was specified.

M:DEQ The M:DEQ procedure call allows a user to dequeue a particular resource/element or a group of resource elements. It has the following format:

```
M:DEQ ([*]'qname',[*]'sname' [,STEP]
                             [,JOB] )
      [,(ERR,[*]address)][,(ABN,[*]address)]
```

where

'qname' specifies the name of a queue (resource). A qname of ALL (without quotes) specifies that all resource/elements currently enqueued for this job are to be dequeued, the only exceptions being those resource/elements that are queued for the entire job if JOB is not specified in this M:DEQ. If ALL is specified, the sname field should also contain ALL.

'sname' specifies the name of the sub-queue (element) or specifies one of the following (not in quotes):

ALL specifies that all elements of the resource that are enqueued are to be dequeued, the only exceptions being those elements that are queued for the entire job if JOB is not specified in this M:DEQ.

RES specifies that all elements of the resource that are enqueued are to be dequeued with the exception of those elements that are queued for the entire job if JOB is not specified in this M:DEQ. RES does not dequeue (i.e., cancel) any previous M:ENQ that had NULL specified. If ALL was specified in the qname field, RES can be used but it is meaningless.

NULL specifies that a previous NULL enqueue for the resource is to be dequeued.

STEP has meaning in conjunction with a qname or sname of ALL and specifies that just those resource/elements enqueued for this step only are to be dequeued. If neither STEP nor JOB is specified, STEP is assumed.

JOB has meaning in conjunction with a qname or sname of ALL and specifies that all enqueued resource/elements are to be dequeued, both those flagged for this step only and those flagged for the entire job.

ERR,[*]address specifies the address at which execution resumes if an error condition is detected. Error codes for M:DEQ are listed in Table B-8, Appendix B.

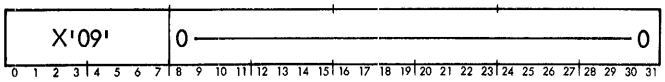
ABN,[*]address specifies the address at which execution resumes if an abnormal condition is detected. Abnormal codes for M:DEQ are listed in Table B-7, Appendix B.

Calls generated by the M:DEQ procedure have the form

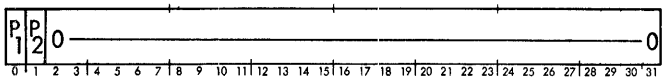
CAL1,2 fpt

where fpt points to word 0 of the FPT shown below.

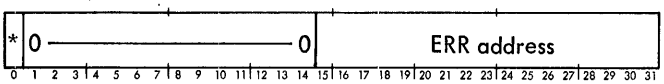
word 0



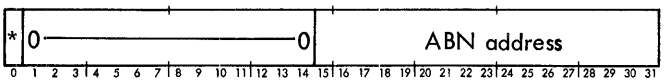
word 1



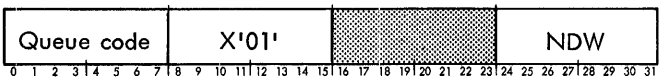
word 2 (P1)



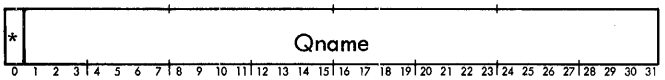
word 3 (P2)



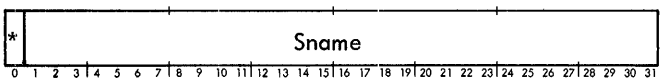
word 4



word 5



word n



where

Queue code has the following meanings:

X'01' for STEP request

X'05' for JOB request

NDW contains the number of words reserved to contain the qname and sname names (in words 6 and following).

Qname specifies the qname in TEXTC format. If the count byte of this field contains X'7F', ALL was specified.

Sname specifies the sname in TEXTC format. The sname starts in the first word following qname (or the indirect address specification of qname). If the count byte is set to X'7F', the ALL option was specified. If the count byte is set to X'7E', the RES option was specified. If the first byte is set to X'40', the NULL option was specified.

OTHER CP-V SERVICE CALLS

There are several additional service calls that are available primarily for use by special system processors, but are also available for use by user programs. These service calls are described in the following paragraphs.

ADJUST DCB CAL

The Adjust DCB CAL merges information from an FPT into a DCB but does not actually open the DCB. The information merged is a combination of that from M:OPEN and M:DEVICE CALs. If the DCB is already open, no adjustments are made.

The format of this call is

CAL1,1 fpt

where fpt points to word 0 of the FPT described in detail below.

OVERALL STRUCTURE OF FPT

There are three parts to the FPT:

1. Basic FPT (required)
2. Variable length parameter list (optional)
3. Device-oriented FPT (optional)

The three parts are contiguous and occur in the order given above. Figure 7 describes the basic FPT and Figure 8 the device-oriented FPT. Table 10 lists the variable length parameter list entries.

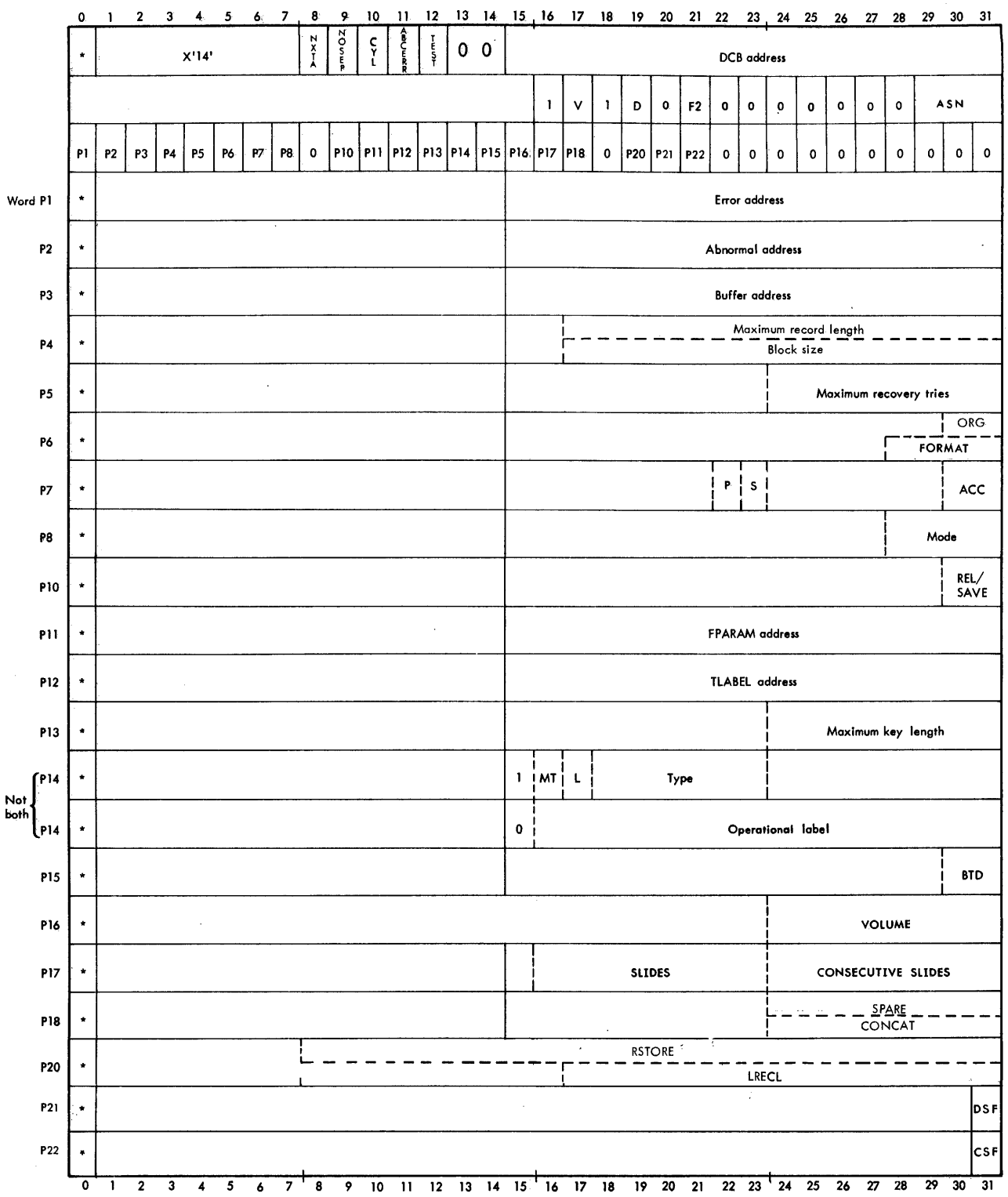


Figure 7. Basic FPT

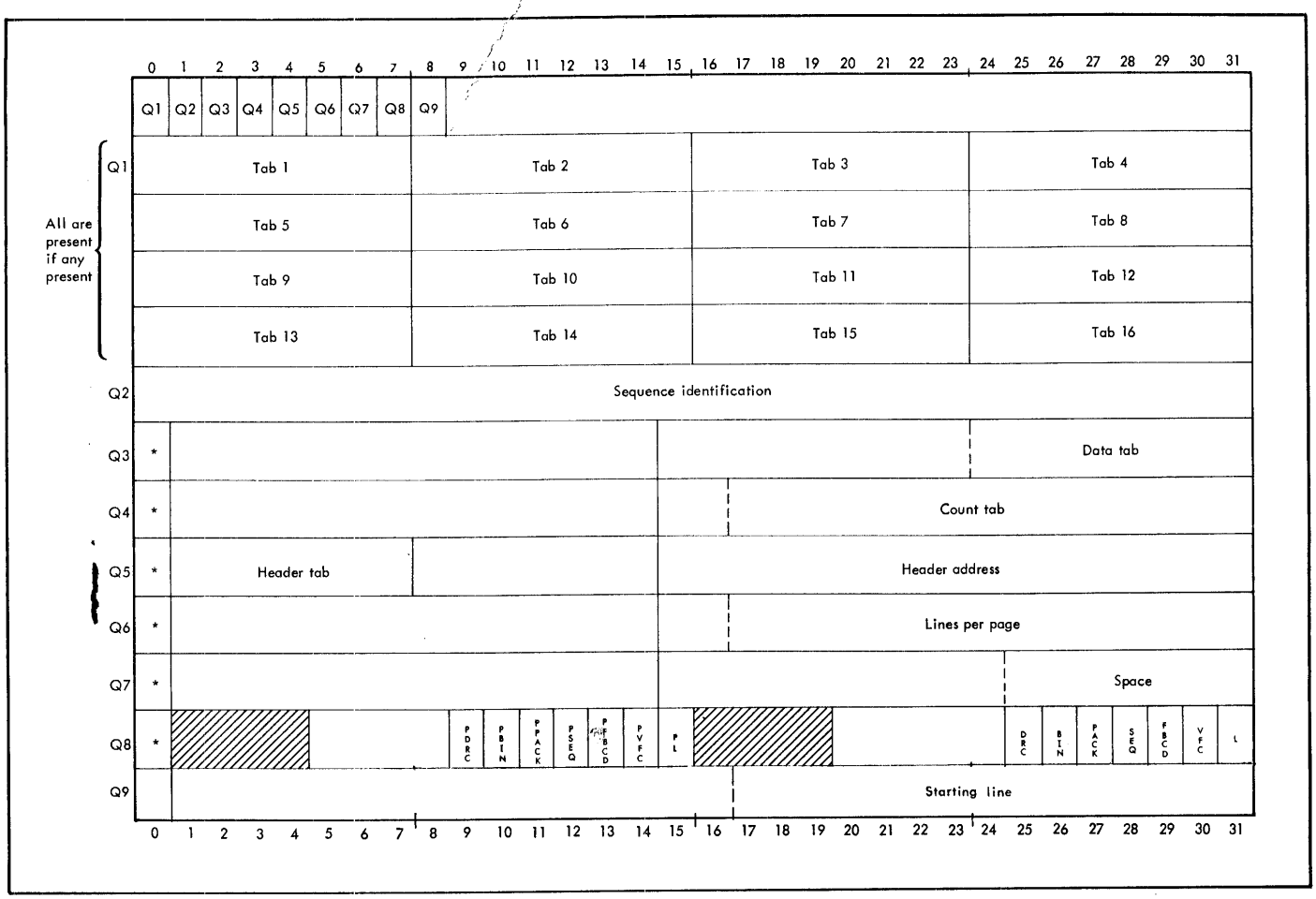


Figure 8. Device-Oriented FPT

Table 10. Variable Length Parameter List

Entry No.	Description	Maximum No. of Subentries[t]	Length of Subentries
1	File Name	1	1-8 words (TEXTC)
2	Account Name	1	2 words
3	Password	1	2 words
4	Expiration Date	1	2 words
5	Read account numbers	19	2 words
6	Write account numbers	19	2 words
7	{SN / INSN} reel numbers	255	1 word

[t]The maximum number of subentries is limited to the space reserved in the DCB.

Table 10. Variable Length Parameter List (cont.)

Entry No.	Description	Maximum No. of Subentries[t]	Length of Subentries
8	OUTSN reel numbers	255	1 word
9, C, D	FPARAM output only	Not valid in parameter list	
A	Modification date	1	3 words
B	SYNON name	1	1-8 words (TEXTC)
E	Creation date	1	2 words
F	Access date	1	2 words
10	Backup date	1	2 words
11	File descriptors	1	1 word
14	Execute account numbers	19	2 words
15	"Execute UNDER" processor or module name	10	3 words (TEXTC)

[t]The maximum number of subentries is limited to the space reserved in the DCB.

If a word Pi is present in an FPT, the corresponding presence bit is set in word 2 (the third word) No gaps may be left for words that are not effectively present (e.g., if P2 and P5 are set and P3 and P4 are reset, then words P2 and P5 must be contiguous). Similarly for Qi in Figure 8. Also in Figure 8, word Q8, bits 9-15 are the presence bits for items in positions 25-31.

Several values in FPT word 1 are:

V = 1 indicates that a variable length parameter list follows the FPT.

D = 1 indicates that following this FPT (or the variable length parameter list of this FPT) is a device-oriented FPT.

The field ASN defines the type of DCB assignment:

000 do not change DCB assignment.

001 assigned to FILE.

010 assigned to XEROX labeled tape.

011 assigned to a device, operational label, or logical device stream.

101 assigned to ANS labeled tape.

VARIABLE LENGTH PARAMETER LIST

Entries have the general form:

Entry No.	1 if last entry else 0	Number of Significant Words	Number of Actual Words
Subentries ⋮			

The entries must be contiguous. Note that an FPT entry of this type may have space reserved (number of actual words > 0) but none used in the current call (number of significant words = 0). Such an entry will cause the number of significant words in the corresponding DCB entry to be set to zero. An entry number of zero can be used to cause an entry to be ignored.

THE EFFECT OF ASSIGN OPTIONS ON THE ADJUST DCB FPT

Because of the similarity of this CAL to M:OPEN and M:DEVICE, and because the user must generate the FPT without a system PROC, the following three lists show all

options allowed, and the effect of each option on the Adjust DCB FPT. Options are grouped by the part of the FPT they affect, and are given in the form used in M:OPEN and M:DEVICE CAL. Names of the FPT items set are those given in Figures 7 and 8.

1. Basic FPT Options

Option	FPT Items Set
ABCERR	Word 0, Bit 11 = 1
RECL,value	P4=1; MAX RECORD LENGTH = value
BLKL,value	P4=1; BLOCK SIZE = value
TRIES,value	P5=1; MAX RECOV TRIES = value
CONSEC	P6=1; ORG = 1
KEYED	P6=1; ORG = 2
RANDOM	P6=1; ORG = 3
FORMAT, character	P6=1; FORMAT = value
SEQUEN	P7=1; ACC = 1
DIRECT	P7=1; ACC = 2
IN[,SHARE / ,EXCL]	P8=1; MODE = 1 [;P=1; S = 1 / ;P=1; S = 0]
OUT	P8=1; MODE = 2
INOUT[,SHARE / ,EXCL]	P8=1; MODE = 4 [;P=1; S = 1 / ;P=1; S = 0]
OUTIN	P8=1; MODE = 8
REL	P10=1; REL/SAVE = 1
SAVE	P10=1; REL/SAVE = 2
FPARAM	P11=1; address of buffer for file parameters
TLABEL	P12=1; address of tape label buffer
KEYM, value	P13=1; MAX KEY LENGTH =value (1-31)
DEVICE, X	P14=1; OP-LABEL=X; F11=1; F12=1
[,L]	see Figure 8 for L
BTD, value	P15=1; BTD=value
VOL, value	P16=1; VOLUME=value
NEWX, slides	P17=1; SLIDES=value
[,consecutive slides]	consecutive slides=value
SPARE, value	P18=1; percent of spare space=value
CONCAT, value	P18=1; number of concatenated files=value
RESTORE, value	P20=1; RESTORE=value
LRECL, value	P20=1; LRECL=value
DEN, {1600 / 800}	P21=1; DSF={0 / 1}
EBCDIC	P22=1; CSF=0
ASCII	P22=1; CSF=1
NXTF	F2=1

2. Variable Parameter Options

Option	FPT Items Set
FILE,	V=1; ASN=1
name	create entry 01; subentry=name
[,account]	create entry 02; subentry=account
LABEL	V=1; ASN=2
name	create entry 01; subentry=name
[,account]	create entry 02; subentry=account
PASS,password	V=1; create entry 03; subentry=password
READ,	V=1;
$account_1,\ldots$	create entry 05 for each $account_n$; $subentry_n=account_n$
WRITE	V=1;
$account_1,\ldots$	create entry 06 for each $account_n$; $subentry_n=account_n$
INSN,	V=1;
$serial\ no_1,\ldots$	create entry 07 for each $serial\ no_n$; $subentry_n=serial_n$
OUTSN,	V=1;
$serial\ no_1,\ldots$	create entry 08 for each $serial\ no_n$; $subentry_n=serial\ no_n$
EXECUTE,	V=1;
$account_1,\ldots$	create entry 14 for each $account_n$; $subentry_n=account_n$
UNDER,	V=1;
account	create entry 15; subentry=processor or load module name

3. Device Oriented FPT Options

Option	FPT Items Set
TAB,value$_1$,...	D=1; Q1=1; for each value$_n$, TAB$_n$ =value$_n$ (16 max.)
SEQ	see below
[,id]	D=1; Q2=1; SEQUENCE ID=id
DATA,value	D=1; Q3=1; DATA TAB=value
COUNT,value	D=1; Q4=1; COUNT=value
HEADER,	D=1, Q5=1
value,	HEADER TAB=value
[*]address	HEADER ADDRESS =[*]address
LINES,	D=1; Q6=1;
value	LINES PER PAGE=value
SPACE,	D=1; Q7=1;
value	SPACE=value
DRC	D=1; Q8=1; PDRC=1; DRC=1
NODRC	D=1; Q8=1; PDRC=1; DRC=0
BIN	D=1; Q8=1; PBIN=1; BIN=1
BCD	D=1; Q8=1; PBIN=1; BIN=0
PACK	D=1; Q8=1; PPACK=1; PACK=1
UNPACK	D=1; Q8=1; PPACK=1; PACK=0
SEQ	D=1; Q8=1; PSEQ=1; SEQ=1
[,id]	see above
NOSEQ	D=1; Q8=1; PSEQ=1; SEQ=0
FBCD	D=1; Q8=1; PFBCD=1; FBCD=1
NOFBCD	D=1; Q8=1; PFBCD=1; FBCD=0
VFC	D=1; Q8=1; PVFC=1; VFC=1
NOVFC	D=1; Q8=1; PVFC=1; VFC=0
L	D=1; Q8=1; PL=1; L=1
NOL	D=1; Q8=1; PL=1; L=0
DEVICE,X	see Figure 7 for OP-LABEL
[,L]	D=1; Q8=1; PL=1; L=1

Also see Table 11, this chapter, for storage of some of the Adjust DCB options.

SPECIFY LOGICAL DEVICE I/O STREAMS

M:LDEV The monitor LDEV routine attaches a logical device stream to a physical device and defines attributes of the logical device stream. LDEV stores the information 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.

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 and/or attributes to satisfy the requirements of his job. He makes any necessary changes through use of the LDEV command or the M:LDEV procedure.

The M:LDEV procedure call has the form

M:LDEV 'stream-id'[,(option)]. . .

where

stream-id specifies the two-character name of the stream to be referenced. This must be the name of one of the logical device streams defined during SYSGEN (for example, C1, L1, P1).

options specify the attributes of the device, such as device type, stream direction, form, format control, etc. The options are as described below; they may appear in any order.

Options

AINIT specifies that the attributes for the stream are to be initialized with the attributes specified in this M:LDEV procedure and that system defaults are to be supplied wherever an attribute is not specified. Any attributes specified for the stream in a previous M:LDEV procedure 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 M:LDEV procedures) is to be released and that the attributes are not to be reinitialized. Any other options specified (except DELETE) in this procedure will be ignored.

ASAVE specifies that the attributes for the stream are to be set only by options explicitly specified in this M:LDEV procedure. Other M:LDEV-specifiable attributes (which may have been set as the result of previous M:LDEV procedures) are not to be changed. ASAVE cannot be used for the LABEL option. DEV and WSN are subject to restrictions noted in the Remote Processing Reference Manual, 90 30 26.

COPIES,value specifies the number of times the file is to be processed to produce multiple copies. The value specified can be an integer from 1 to 255 inclusive. 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 device. (Note that if COUNT is specified for the LO device 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 type mnemonics of the central site (that is, mnemonics defined for symbiont devices during SYSGEN — for example, CR, 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 the designated 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. (See "Noncontrol Input Files" below.)

FPC,'name' specifies the one- to four-character name of an installation-determined form overlay and is used in output scheduling for the Xerox 1200 or a similar device. The default is to have no special scheduling (i.e., the operator will determine which overlay to use if any).

IN and OUT specifies the direction of the stream. The default is OUT.

JDE,value specifies the job descriptor entry to be used in output scheduling for the device. The value must be in the range 0-89 and specifies an installation defined procedure describing printer setup attributes (e.g., VFC tape).

LABEL,[*]address specifies the address of a TEXTC string to be appended to the stream's user-identification banner lines (see "user-identification banner" in glossary).

LINES,value specifies the number of printable lines per logical page. The greatest value that may be specified is 255 lines per page. If this option is not specified the value established at SYSGEN time will apply.

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.

OUT see IN above.

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 printered (top). A value of 0 or 1 results in single spacing. The greatest value that may be specified is 15. The default is single spacing.

VFC and NOVFC specifies whether or not vertical format control characters are to be used. (These two options are legal only 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.

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, then 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.

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

```
!!NCTL [name]
```

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

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 jobs that are to be run at a later time.

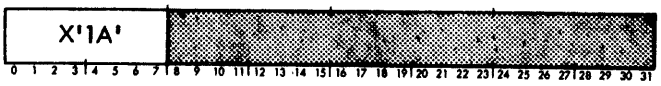
A file created in this manner must be accessed via the LDEV command or M:LDEV procedure using any logical device stream except C1. If the user gives the file a name or requests the operator to do so, the user can access the file using the FORM,xxxx option. (The operator gives the file a name using the key-in Syyndd,F'xxxx' where xxxx must be identical to xxxx on the FORM 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.

Calls generated by the M:LDEV procedure have the form

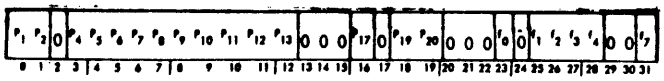
```
CAL1,8      fpt
```

where fpt points to word 0 of the FPT shown below.

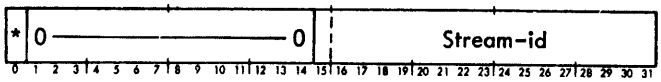
word 0



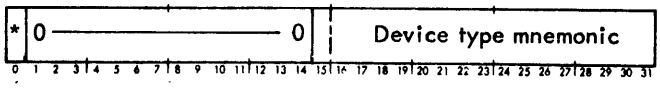
word 1



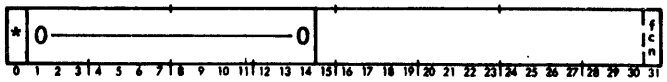
stream-id (P1)



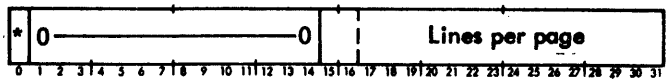
option DEV (P2)



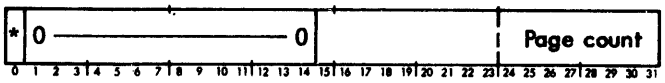
option IN/OUT (P4)



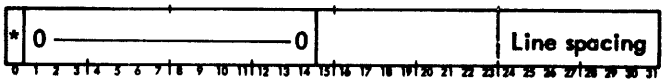
option LINES (P5)



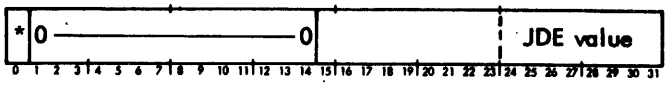
option COUNT (P6)



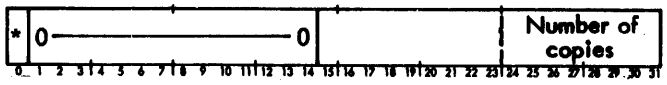
option SPACE (P7)



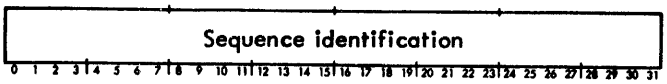
option JDE (P8)



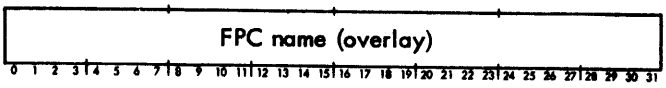
option COPIES (P9)



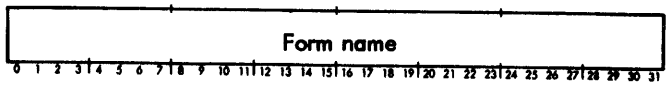
option SEQ (P10)



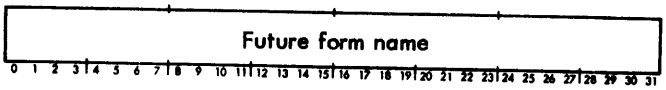
option FPC (P11)



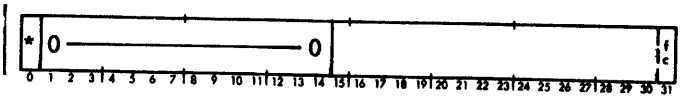
option FORM (P12)



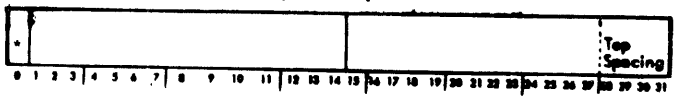
option FFORM (P13)



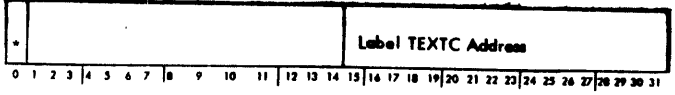
option { VFC / NOVFC } (P17)



option SPACE (P19) suboption top



option LABEL (P20)



where

P_1 through P_{20} (in word 1) specify which of the succeeding words are present; that is, which of certain options have been specified (1 means the word is present, 0 means the word is not present). For example, a 1 in bit 11 of word 1 indicates that the word for the FORM option (see word P12) is present.

f_0 through f_7 (in word 1) specify whether the DELETE, AREL, ASAVE, and DRC options are present (0 means the option is not present, 1 means it is):

- f_0 is for the NOBANNER option.
- f_1 is for the DELETE option.
- f_2 is for the AREL option.
- f_3 is for the ASAVE option.
- f_4 is for the CONCURR option.
- f_7 is for the DRC option.

For example, a 1 in bit 31 of word 1 specifies the DRC option.

fcn (in word P4) specifies the direction of the stream (0 means IN, 1 means OUT).

fc (in word P17) specifies vertical format control override (0 means VFC, 1 means NOVFC).

If an inconsistency is detected in the FPT parameter, return is made to CAL+1 with CC1 set and the error code in SR3 (see Appendix B, Table B-5). Otherwise CC1 is reset.

Table 11 lists most of the service functions (i.e., options) that apply to logical device streams and indicates, for each requesting method, the place where the function information is stored — Data Control Block (DCB) or Stream Context Block (SCB). Null table entries indicate that the particular service option is not allowed for that request method. If the table entry specifies DCB, then presence or absence of the request is stored in the DCB and only operations through that DCB receive the service. If the table entry specifies SCB, then the presence or absence of the request is stored in the SCB and will affect all operations on that stream through any DCB. The importance of Table 11 lies in the fact that when I/O is performed for a logical device stream and the DCB conflicts with the SCB, the DCB takes precedence. Also, when conflicting options are specified for the same DCB or for the same SCB, the last one encountered during execution is used.

READ AND WRITE ASSIGN/MERGE RECORD

Throughout a job or on-line session, I/O service and file assignment information is retained and merged into user or processor DCBs at each job step. This information is maintained in an assign/merge record on disk storage, with one record location assigned to each user by the log-on procedure that places the disk storage address of the assign/merge record in JIT.

Special procedures are used to read (M:RAMR) and write (M:WAMR) this record. Use of M:RAMR and M:WAMR is governed by the following rules:

1. Any program may read an assign/merge record.
2. Only command processors and processors with JIT access may write an assign/merge record.
3. The buffer size should be 2048 bytes.
4. The DCB must be closed and at least 8 words long.
5. Error codes are set to report to the address set in the DCB, as described in Appendix B. Errors in the call may be as follows:

Code	Meaning
X'06'	No record exists (read operation).
X'57'	No granule can be obtained (write operation).
X'2E'	The DCB is open.

Code	Meaning
X'4A'	The buffer address is outside the user's data area or the size is greater than 2048 bytes.
X'14'	A write has been attempted by a processor that is not a command processor or doesn't have special JIT access.

6. If a read or write error was encountered when accessing the assign/merge record, the user is aborted and logged off the system with the following error code.

Code	Meaning
X'A9'	Error on read or write of assign/merge record.

The M:RAMR and M:WAMR procedures are described in the following paragraphs.

M:RAMR The RAMR routine reads the assign/merge record and has the following format:

M:RAMR [*] DCB name [,(option)]...

The options are as follows:

BUF, [*] address specifies the address of the user's buffer into which the record is to be read. An asterisk may be used to indicate that the address is the address of a location containing the buffer address.

SIZE, [*] value specifies the size in bytes of the user's buffer. The buffer should be 2048 bytes. An asterisk may be used to indicate that the value is the address of a location containing the buffer size.

Calls generated by the M:RAMR procedure have the form

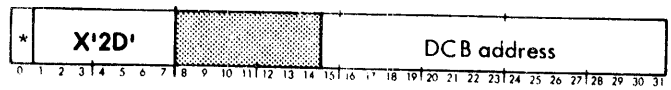
CAL1,1 fpt

where fpt points to word 0 of the FPT shown below.

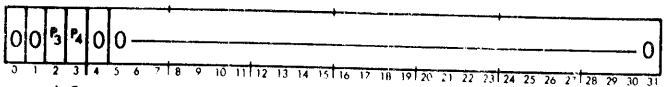
Table 11. Storage of Service Functions

Service Function	Request Method					
	M:DEVICE	Adjust DCB	M:DCB	ASSIGN	M:LDEV	LDEV
BCD	DCB	DCB	DCB	DCB	-	-
BIN	DCB	DCB	DCB	DCB	-	-
COPIES	-	-	-	-	SCB	SCB
COUNT	DCB	DCB	DCB	DCB	SCB	SCB
DATA	DCB	DCB	DCB	DCB	-	-
DEV or DEVICE	-	Both	Both	Both	Both	Both
DRC	DCB	DCB	DCB	-	SCB	SCB
FBCD	DCB	DCB	DCB	DCB	-	-
FFORM	-	-	-	-	SCB	SCB
FORM	SCB	-	-	-	SCB	SCB
FPC	-	-	-	-	SCB	SCB
HEADER	SCB	SCB	SCB	-	-	-
IN	-	Both	Both	Both	Both	Both
JDE	-	-	-	-	SCB	SCB
LINES	SCB	SCB	SCB	SCB	SCB	SCB
NLINES	SCB	-	-	-	-	-
NODRC	DCB	DCB	DCB	-	SCB	SCB
NOFBCD	DCB	DCB	DCB	DCB	-	-
NOVFC	DCB	DCB	DCB	DCB	SCB	SCB
OUT	-	Both	Both	Both	Both	Both
PAGE	DCB	-	-	-	-	-
SEQ	DCB	DCB	DCB	DCB	SCB	SCB
SPACE	DCB	DCB	DCB	DCB	SCB	SCB
SRCB	-	-	-	-	-	SCB
TAB	DCB	DCB	DCB	-	-	-
VFC	DCB	DCB	DCB	DCB	SCB	SCB
WSN	-	-	-	-	-	SCB

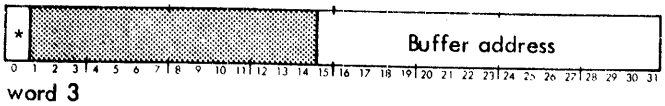
word 0



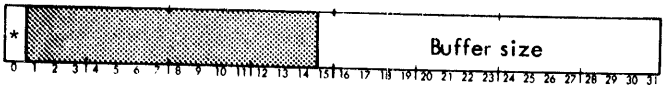
word 1



word 2



word 3



M:WAMR The WAMR routine writes the assign/merge record and has the following format:

M:WAMR [*] DCB name [,(option)]...

The options are as follows:

BUF, [*] address specifies the address of the user's buffer into which the record is to be read. An asterisk may be used to indicate that the address is the address of a location containing the buffer address.

SIZE, [*] value specifies the size in bytes of the user's buffer. The buffer should be 2048 bytes. An asterisk may be used to indicate that the value is the address of a location containing the buffer size.

Calls generated by the M:WAMR procedure have the form

CAL1,1 fpt

where fpt points to word 0 of the FPT shown for M:RAMR with the exception that the code in the first byte of word 0 is X'2E' instead of X'2D'.

REPORT SYSTEM LOAD PARAMETERS

M:DISPLAY The DISPLAY routine returns the current values of three system load parameters. The three system load parameters are

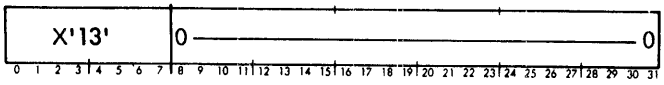
1. The execution time multiplication factor (ETMF).
2. The median value of terminal response time in seconds.
3. The current number of active users.

Integer values for these parameters are returned in registers 5, 6, and 7, respectively. ETMF and response time values apply to all operations during the last full minute of system usage.

The procedure M:DISPLAY has no parameters and generates a CAL of the form

CAL1,8 fpt

where fpt points to the FPT shown below

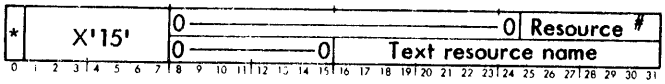


RELEASE RESOURCE CAL

There is one CAL that is used to release job resources back to the operating system under program control. It may be used, for example, to release tape drives used in the first job step of a stream, but not required for the rest of the job. Its format is

CAL1,8 fpt

where fpt points to the FPT shown below



where

resource # is the resource type index and may be obtained from the type field in the DCB (right seven bits of byte 2 of word 1). (See Appendix A.)

text resource name is the name of the resource being released. The name corresponds to the name specified on the LIMIT card (for example, 9T, DP).

In addition, the user must specify in SR1 the number of resources to be released.

The following error conditions are possible:

CC1 = 1 If resources released exceed amount originally allocated or remaining allocated (none are released).

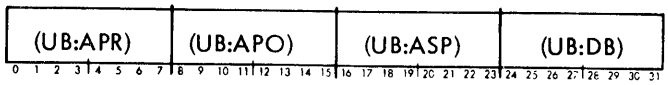
CC2 = 1 If an odd number of core pages is specified for release, and no pages are released.

CC3 = 1 If an index is outside the range of the Resource Allocation Table.

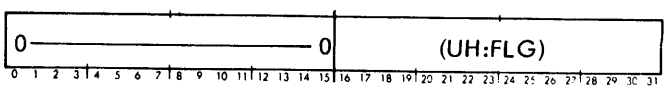
SAVE CAL

The SAVE CAL stores in the user's JIT the current values of the user's tables for the following: associated processor root, associated processor overlay, associated special processor, associated debugger, and user flags. This information is stored as follows:

J:CPROCS



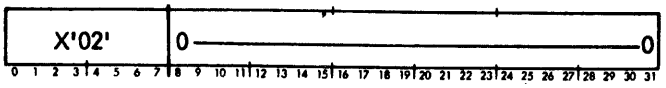
J:CFLGS



The format of the SAVE CAL is

CAL1,4 fpt

where fpt points to the FPT shown below.



There are no restrictions on the use of the SAVE CAL. It always returns with CC1 = 0.

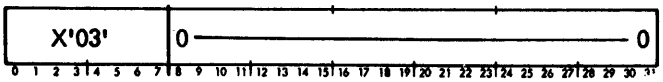
GET CAL

The GET CAL can be invoked only by TEL or CCI. If TIC (TEL-in-control flag) is not set, the return is to CAL + 1 with CC1 set to one. Otherwise, the debugger-associated and debugger-in-control bits from the saved flags (J:CFLGS) are stored in the user's current flags and the saved processor values are transferred from J:CPROCS to the appropriate user tables. The return in this case is to CAL + 1 with CC1 = 0.

The format of the GET CAL is

```
CAL1,4     fpt
```

where fpt points to the FPT shown below



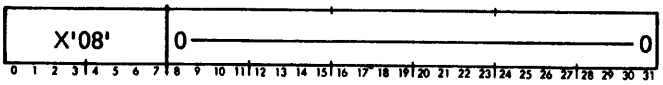
ENTER MASTER MODE

M:SYS The M:SYS procedure allows processors that have a sufficient privilege level (C0 or higher) to operate in master mode with a write key of 0. The calling program is also given the addresses of the monitor's I/O routines, in system-communication registers SR1-SR3. M:SYS does not set the Sigma 9 or Xerox 560 master-protected bit.

The M:SYS procedure has no parameters, and generates a CAL of the form

```
CAL1,6     fpt
```

where fpt points to the FPT shown below.



On return from this procedure, the calling program is operating in master mode with a write key of 0. Register SR1 will contain the address of QUEUE, the monitor routine for I/O through a DCB with no end-action; SR2 the address of QUEUE1, for I/O through a DCB with end-action; and SR3 the address of NEWQ, for I/O with no DCB. (User programs generally may not specify end action. End action routines must be in the resident monitor.) If the caller's privilege level is not sufficient, return is to CAL+1 with CC1 set.

M:CAL The M:CAL procedure allows user control over the CAL3 trap, thus allowing the user to specify a target Program Status Double word (PSD) for the CAL3 trap. Ability to modify the bits in the PSD is controlled by privilege level. The user with less than C0 privilege can alter the instruction address (which will be entered slave mapped), as well as the arithmetic and floating mask bits. The user with C0 or greater privilege can alter any portion of the PSD with the exception of the register block, the write key, or the map bit.

The M:CAL procedure has the form

```
M:CAL     (IA, addr) ,pb ...
```

where

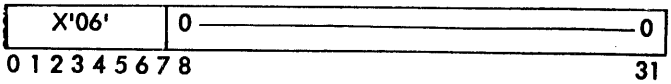
addr specifies either 0 or the user handler address, above JBUPVPA.

pb specifies a two character identifier for various PSD bits. The options are listed below.

CI	Counter interrupt group inhibit
TI	Input/output interrupt group inhibit
EI	External interrupt inhibit
FS	Significance trap mask
FZ	Zero trap mask
FN	Normalize trap mask
DM	Decimal arithmetic fault trap mask
AM	Fixed-point arithmetic overflow trap mask
MM	Set master mode, write key = 0
MP	Set master protect mode, write key = 1 (Sigma 9 and 560 only)

The CAL1,5 that is generated points to word 0 of the following FPT:

word 0



words 1 and 2

Upon issuing the CAL1, the following CC bits are returned to the user:

CC1 SET - bad address given, no action taken.
RESET - connect request satisfied.

CC2 SET - user not privileged to specify MM, MP, or inhibit bits.

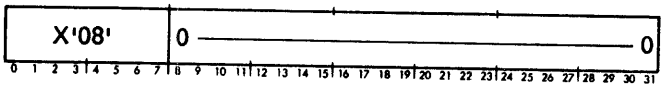
M:MASTER The M:MASTER procedure allows a user with sufficient privilege level (C0 or higher) to operate in the master mode (master-protected mode if running on a Sigma 9 or Xerox 560) with a write key of 1. The format of the procedure call is

```
M:MASTER
```

Calls generated by the M:MASTER procedure have the form

```
CAL1,5     fpt
```

where fpt points to the FPT shown below.



If the caller's privilege level is not sufficient, return is to CAL+1 with CC1 set.

ENTER SLAVE MODE

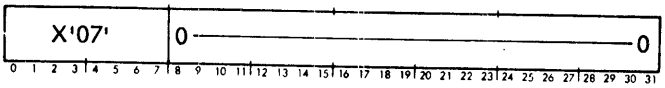
M:SLAVE The M:SLAVE procedure allows any master (and master-protected) mode program to return to the slave mode. The format of the procedure call is

```
M:SLAVE
```

Calls generated by the M:SLAVE procedure have the form

```
CAL1,5    fpt
```

where fpt points to the FPT shown below.



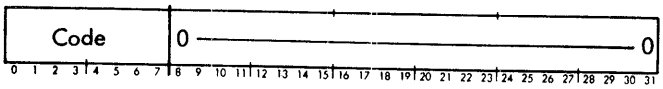
ASSOCIATE OR DISASSOCIATE PUBLIC LIBRARY

Two CALs allow the user to control the association of shared public libraries with his program. Both CALs are of the form

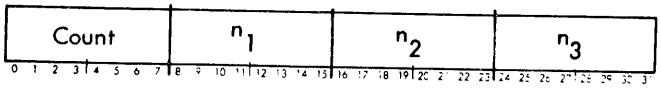
```
CAL1,4    fpt
```

where fpt points to word 0 of the fpt shown below.

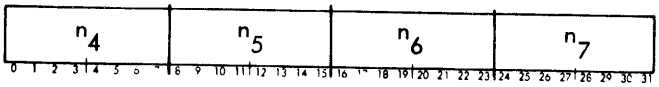
word 0



word 1



word 2



where

code is X'04' for associate and X'05' for disassociate.

count is the number of characters in the name.

n_i are the characters in the name of the public library. Names are limited to 7 characters and must have trailing blanks.

If no library with the specified name can be found, CC1 is set and no further action is taken. An attempt to disassociate when there is no association will cause CC2 to be set. If association of one library requires disassociation of a current library, both disassociation and association will take place and CC3 will be set. If either the virtual core or the physical core is not available, CC4 is set and no further action is taken.

(This page intentionally left blank.)

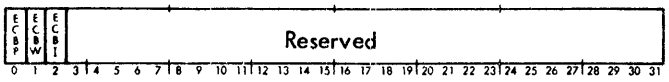
CHECK EVENT CONTROL BLOCK(S) FOR COMPLETION

M:CHECKECB The M:CHECKECB procedure allows a user program to check for the completion of an event or of a set of events and, if necessary, to enter the wait state to await the completion of the event(s).

There are two event-driven services in CP-V, enqueue (M:ENQ) and dequeue (M:DEQ). When one of these services is requested, the user program may allocate a two-word block to be used as an event control block (ECB). The access protection for the ECB must be 00 — all access. When the address of the event control block is specified in the service call, it is saved by the monitor. Upon entry to the service procedure, the system initiates the required action, sets the event control block to an 'in-use' status, and returns control to the user program. More than one event-driven service may be in action at the same time. The user program continues to process until it requires that the action(s) requested be completed. At this time the user program may issue the M:CHECKECB procedure call which, if necessary, will place the user program in a wait state until the action(s) specified within the M:CHECKECB procedure call have completed.

An event control block consists of two words:

word 0



word 1

where

ECBP is set to one when the event has completed.

ECBW is set to one when the event has been referenced in an M:CHECKECB request.

ECBI is set when the ECB has been assigned to an action to be completed by an enqueue or dequeue service. ECBI indicates that a read or write has been issued and is waiting for completion of non-enqueue/dequeue operations.

The settings of the bits ECBP, EDBW, and ECBI are mutually exclusive. That is, only one of these bits may be set to one at any given time. Since the ECB exists within the user program's virtual storage, the state of the ECB may be examined at any time by the user program.

The format of the M:CHECKECB procedure call is

M:CHECKECB (option)[,(option)]...

where the options are:

ECB,[*]address[,[*]value] specifies the address of a set of contiguous event control blocks. Value specifies the number of contiguous event control blocks. The default for value is 1. A value of zero specifies that the set of ECBs is null.

ECBL,[*]address ,[*]value specifies the address of the first word of a list of words, each of which contains a pointer to an event control block. Value specifies the number of words containing event control block pointers. The default for value is 1. A value of zero specifies that the set of pointers to ECBs is null.

EVENTS,[*]value specifies the number of event control blocks that must be posted complete before control is returned to the user program. The default is 1.

TIME,[*]units specifies the number of 1.2-second time units that may elapse before control is returned to the user event if the value specified for EVENTS has not been satisfied. The maximum number of units that may be specified is 65,535. If this option is not specified, then time is not a factor in the completion of the procedure call.

At least one occurrence of ECB or ECBL must be present in the specification of M:CHECKECB. ECB and ECBL may be stated a multiple number of times to combine noncontiguous ECB areas for one M:CHECKECB request.

The M:CHECKECB procedure allows a user to wait for the completion of a specified number of events. To determine which event or events were actually completed, the user should examine the ECBP bit of each ECB that was specified in the list(s). Since there exist certain actions which unconditionally end the M:CHECKECB wait state, the user should always determine the setting of the ECBP bit after the wait state ends. This is true even if the ECB or ECBL parameters specified only one ECB to wait upon. The actions which unconditionally terminate the M:CHECKECB wait state (changing the user's environment to the M:CHECKECB CAL1+1 location) are:

BREAK

Y^C

INT key-in (operator key-in)

E key-in (operator key-in)

X key-in (operator key-in)

ZAP key-in (operator key-in)

TIME value has elapsed

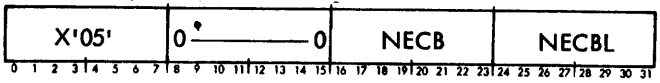
The ECBP bit may not be set correctly if access protection for the ECB is not 00. The monitor makes a security check at the time that an ECB is selected to be posted complete. If access to the ECB is not 00, the portion of the posting operation which would have transferred information to the ECB is omitted.

Calls generated by the M:CHECKECB procedure have the form

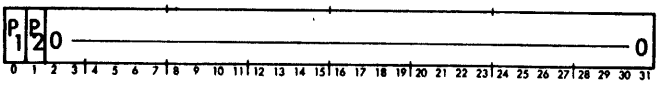
```
CAL1,7     fpt
```

where fpt points to word 0 of the FPT shown below.

word 0

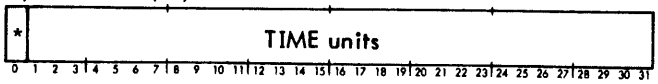


word 1

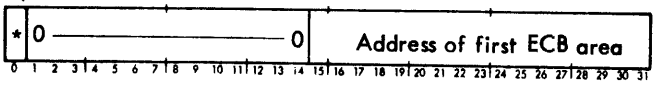


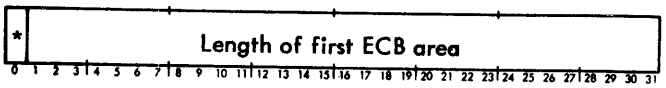
option EVENTS (P1)

option TIME (P2)



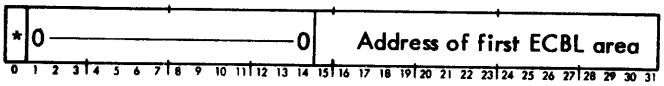
option ECB

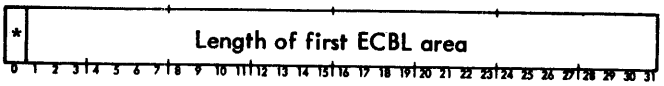




Addresses and lengths of additional ECB areas (one pair for each ECB specified)

option ECBL





Addresses and lengths of additional ECB areas (one pair for each ECB specified)

where

NECB specifies the number of ECB options in the procedure call.

NECBL specifies the number of ECBL options in the procedure call.

Note that for each ECB and each ECBL option, two words are generated.

Condition code settings resulting from an M:CHECKECB CAL are:

0	0	0	0	The M:CHECKECB procedure call was completed with no errors.
0	0	0	1	The TIME specification is greater than 65,535.
0	0	1	0	The ECB is not in the proper state. (Either ECBW is set or ECBI is not reset.)
0	0	1	1	An infinite wait condition has occurred. The number of ECBs specified is less than the EVENTS specification.
0	1	0	0	There was not enough monitor workspace to process the M:CHECKECB procedure call at this time.
0	1	0	1	The ECB does not have an access protection of 00.

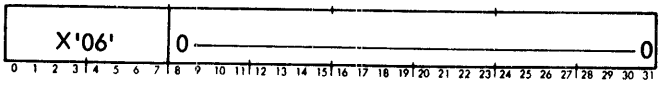
INITIATE GHOST JOB

The following CAL can be used to initiate a ghost job.

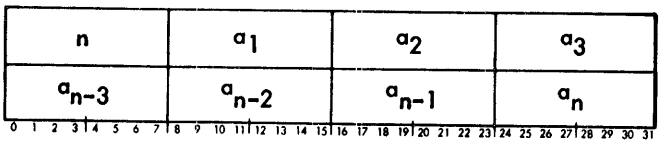
```
CAL1,6      fpt
```

where fpt points to an FPT having the following format:

word 0



words 1 and 2 (Name of job to be initiated)



(Name of job must be in TEXTC format.)

If the program to be initiated is already in execution at the time of the request and is not in a waiting state (WAIT CAL with unexpired time), the normal return is made (CCI = 0). If the program is in a waiting state, it will be activated immediately at the WAIT CAL plus 1 and a normal return is made to the initiating program. A privilege level of C0 or higher is required to utilize this CAL.

EXECUTE PRIVILEGED INSTRUCTIONS

M:EXU The M:EXU procedure allows a user with sufficient privilege level (C0 or higher) to request that the monitor execute a privileged instruction for the user so that the program does not have to run in the master mode. The procedure has the following format:

```
M:EXU     [*]address
```

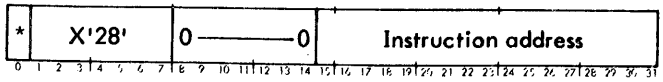
where address specifies the address of the privileged instruction to be executed. The op code of the instruction must be one of the following:

X'4C'	SIO
X'4D'	TIO
X'4E'	TDV
X'4F'	HIO
X'6C'	RD
X'6D'	WD

Calls generated by the M:EXU procedure have the form

```
CAL1,5      fpt
```

where fpt points to the FPT shown below.



The instruction may invoke indirect addressing. Since the condition codes returned are those of the executed instruction, the following abnormal conditions are reported via a program abort (which may be intercepted by the user's trap control routine by specifying the CAL keyword in the M:TRAP procedure).

Code	Subcode	Meaning
B9	01	Insufficient privilege.
B9	04	Illegal op code in referenced instruction.
B9	05	Referenced instruction is in protected memory.

ON-LINE AND BATCH DIFFERENCES

The monitor responds differently to certain CALs depending on whether an on-line or a batch program issued the call. These differences are outlined below.

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 message is listed on the UC device. 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 CAL
```

This message is listed on the UC device. The monitor then returns control to the on-line executive (TEL), 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 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.

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.

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 the INTERRUPT key-in is invoked at the operator's console. When control is given to the INT routine as a result of an interrupt, the monitor pushes the PSD and general registers into the user's temp stack. 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 the user's temp stack. The TRTN routine may be used to restore control to the user program.

The procedure call is of the form

```
M:INT     address
```

where address specifies the location of the entry to the program's BREAK response routine. A zero address resets break control. If the address specified is in the range of virtual addresses assigned to the monitor, then zero is substituted (break control is reset).

5. I/O PROCEDURES

INTRODUCTION

All I/O operations are performed by the monitor for the user (i.e., the user's program never directly accesses an I/O device, but rather requests that the monitor do so). Each request for I/O service from the monitor is made by inclusion of an I/O call in the user's program. This call generates a Function Parameter Table (FPT), which in turn refers to a Data Control Block (DCB). The combination of the I/O call, the FPT and the DCB provides the information that the monitor needs to perform the requested operation.

Generally, the DCB contains the kind of information that is specific to a device (e.g., for output to a line printer, number of lines per page is one value in the DCB). The FPT contains a far smaller set of information that is specific to the operation to be performed (e.g., the location and size of the buffer that is to be output to the printer in this specific operation). Separation of information into the DCB and the FPT allows the user to create one DCB for a type of I/O and reference that DCB throughout his program, whenever he requires that type of I/O. Each time that he references that DCB, he generates an FPT with the specific information required for that particular I/O operation.

In addition to serving as a source of information for the monitor to use in an I/O operation, the DCB also provides a place for the monitor to store information while it is performing an I/O operation. Some of the information stored in the DCB by the monitor may be of some use to the user, and some is of meaning only to the monitor.

The user is responsible for providing the address of a properly initialized DCB with every call to the monitor requesting an I/O operation.

The user may obtain a DCB by

1. Including a copy of one or more preconstructed standard monitor DCBs that are available for most common I/O operations (via an external reference to the appropriate DCB name in his program).

2. Explicitly creating his own DCB at assembly time.

Monitor calls are provided to permit the user to initialize or alter DCBs.

I/O operations involving symbiont devices do not require different procedures. Aside from faster completion, the user is not aware that a device is a symbiont device.

Each DCB is assigned to a physical device either directly, by entry of a device code in the DCB, or indirectly, by entry of an operational label or a resource name in the DCB. Physical devices may be identified by a code of the form yyndd, where yy = device type, n = IOP designation, and dd = device designation. (The ndd portion is ignored but is allowed for compatability with previous versions of the system.) Values for each are listed in Tables 12 through 14. Table 12 lists only the I/O device type codes that are standard in CP-V. Other device types may be defined at SYSGEN. The operational label is the name of a logical system device. The assignment of DCBs to devices through operational labels gives users the capability of changing device assignments for a particular input/output class by changing the normal assignment of the operational labels.

The standard monitor DCBs have names containing the letters of the operational labels to which they are normally assigned. For example, the standard monitor DCB assigned to the SI operational label by default has the name M:SI.

Table 12. Standard I/O Device Type Codes

Device (yy)	Physical Device Name
MT	Default magnetic tape type (defined at SYSGEN)
7T	7-track magnetic tape
9T	800 bpi 9-track magnetic tape
BT	1600 bpi 9-track magnetic tape
CP	Card punch
CR	Card reader
TY	Typewriter
LP	Line printer
DC	Magnetic disk
DP	Default disk pack type (defined at SYSGEN)
NO	No device

Table 13. IOP Designation Codes

Specified Channel Letter (n)	Corresponding Decimal Digit of Unit Address
A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7

Table 14. Device Designation Codes

Hexadecimal Code (dd)	Device Designation
00 ≤ dd ≤ 7F	Refers to a device number (00 through 7F).
80 ≤ dd ≤ FF	Refers to a device controller number (8 through F) followed by a device number (0 through F).

General registers may not be used as I/O buffers.

I/O procedures are provided for the following I/O functions:

1. File Maintenance

   Create a Data Control Block

   Open a File

   Close a File

   Set Error or Abnormal Address

   Check I/O Completion

   Declare Temporary File

2. Data Record Manipulation

   Read a Data Record

   Write a Data Record

   Delete a Data Record

   Truncate a Blocking Buffer

3. File Manipulation

   Position n Records

   Position File

   Close Volume

   Rewind

   Write End-of-File

   Insert or delete a Symbiont File

4. Special Device

   Set Listing Tabs

   Skip to Top of Form

   Set Number of Printable Lines

   Set Line Spacing

   Specify Direct Formatting

   Specify Vertical Format Control

   Specify Page Count

   Change Output Form

   Change Device Mode or Record Size

   Specify Output Header

   Specify Card Punch Sequencing

   Determine Number of Lines Remaining

   Check Correspondence of DCB Assignments

FILE MAINTENANCE PROCEDURES

All procedure calls except M:DCB described in this chapter generate a Function Parameter Table (FPT) of the same general form, consisting of a function identifier, parameter and file option flags, and data.

CREATE A DATA CONTROL BLOCK

If the user's program is written in ANS COBOL or Extended FORTRAN IV, the processor will automatically include in the object modules generated for the program all necessary I/O calls and references to DCBs that the loader will satisfy. However, if the user's program is written in Meta-Symbol, he must provide all necessary I/O procedure calls in his symbolic program.

The user may use copies of monitor DCBs by declaring them as external references in his Meta-Symbol program; otherwise, he must create his own DCBs by means of explicit symbolic code or via M:DCB procedure calls.

When a load module is loaded for execution, any ASSIGN parameters for DCBs contained in that load module are merged. Thus, an option contained in a DCB created explicitly or via M:DCB or contained in a system DCB may be overridden by an ASSIGN control command.

DCB formats are described in the appendix titled "Data Control Block Formats".

M:DCB The M:DCB procedure generates nonexecutable code (i.e., it creates only a data area in the user's program) which must have a label. The label is the name by which the DCB is to be referenced.

The M:DCB procedure call is of the form

```
dcb name M:DCB [(option)] [, (option)] . . .
```

where dcb name specifies the name of the user's DCB. The name may consist of from 3 to 31 alphanumeric characters, the first two of which must be "F:" or "M:". The "dcb name" must previously have been declared a dummy section, via a statement of the form

dcb name DSECT 1

The options are as follows:

name (one or two of the four keyword operands given below).

DEVICE,'name' specifies a device type, a system operational label, or a logical device stream name. Acceptable forms of the name specifications are (1) for a device type – 'CR', 'LP', '7T', '9T', etc.; (2) for an operational label – 'LO', 'EO', 'LL', 'C', etc.; (3) for a logical stream – 'P1', 'C1', 'L1'. DEVICE may be used in conjunction with FILE, LABEL, or ANSLBL.

FILE[,'name'[,'account']][,n] specifies the name of the public or private file that is to be assigned to the DCB. The name may consist of up to 31 alphanumeric characters. The named file will be maintained on RAD or DP storage. If the file is private, the SN option must be used to specify the serial number(s) of the private volume set.

If the named file belongs to a different account than that of the current job, the file's account number must be given (either in the M:DCB call or in an ASSIGN control command or M:OPEN call). If the name and account number are both omitted and n is not specified, eight words are reserved for the name (to be inserted via an ASSIGN control command or M:OPEN call) and two words for the account number. If n is specified, n words will be reserved for the file name. If neither FILE nor LABEL (see below) is specified in the M:DCB call, the DCB may only be assigned to files defined by a system operational label (for example, GO), or to a device.

The following examples illustrate use of the file option:

(FILE) reserves eight words for the file name.
(FILE,n) reserves n words for the file name.
(FILE,'name',n) reserves n words and puts 'name' as the filename.
(FILE,'name','account',n) reserves n words and puts 'name' as the filename and 'account' as the account.

All correct forms reserve two words for the account.

Also see the description of the FILE option for the M:OPEN procedure.

LABEL[,'name'[,'account']][,n] specifies the name of a file on Xerox labeled magnetic tape. The tape may consist of up to 31 alphanumeric characters. If LABEL is specified, the SN option must be used to specify the reel(s) containing the file. If the named file belongs to a different account than that of the current job, the file's account number must be given (either in the M:DCB call or in an ASSIGN control command or M:OPEN call). If the name and account number are both omitted and n is not specified, eight words are reserved for the name (to be inserted via an ASSIGN control command or M:OPEN call) and two words for the account number. If n is specified, n words will be reserved for the file name.

ANSLBL,'name' specifies the name of a file on ANS labeled magnetic tape that is to be assigned to the DCB. The name may consist of up to 17 alphanumeric characters. If the file name contains a special character, it must be enclosed by single quotation marks. When a single quotation mark is to be used as part of the file name, it must be coded as two successive quotation marks. There must be no blanks between the last character and the terminating quotation mark.

ASN,{FILE / LABEL / DEVICE / JRNL / ANSLBL} specifies the value for the ASN field of the FPT. This value will override the default value created by other keywords such as FILE. For example, this option is useful in creating a device DCB which has space reserved for a file name:

....(FILE,8),(DEVICE,'LO'),(ASN,DEVICE)...

The ASN values are

1 - FILE
2 - LABEL
3 - DEVICE
4 - JRNL
X'A' - ANSLBL

org (one of the four file organization types given below. Not applicable to ANS labeled tapes.)

CONSEC specifies that the records in the file are consecutively organized and each record is to be processed in order.

If a private file has consecutive organization, only one volume in the private volume set need be mounted at any time. As another volume is required, the system will request that it be mounted.

KEYED specifies that the location of each record in the file is determined by an explicit identifier (key) that may be used to access the record. A key may consist of up to 31 characters.

If a private file has keyed organization, all volumes in the set must be mounted when the file is opened and remain mounted until the file is closed.

RANDOM specifies that the records in the file are a collection of contiguous granules on the specified device type that are devoid of any system information, and whose internal structure is the responsibility of the user. If device type is not specified, the file is allocated on RAD or disk pack, whichever is available.

If a private file has random organization, all volumes in the set must be mounted when the file is opened and remain mounted until the file is closed.

UNDEF specifies that Xerox labeled tape records are all unblocked and without headers, i.e., equivalent to device format. BLOCK access is forced for this organization. (Applicable only to Xerox labeled tape.)

access (one of the three record access means given below.) Not applicable to ANS labeled tapes unless otherwise noted.

SEQUEN specifies that records in the file are to be accessed in the order in which they appear within the file.

DIRECT specifies that the next record to be accessed is determined by an explicit identifier (key). If specified for consecutive or keyed disk files, read ahead will be disabled.

BLOCK specifies that data blocks are transferred directly between tape and the user's buffer. (Applicable only to Xerox and ANS labeled tape.) This access is forced for UNDEF organization and for ANS tape.

function (one of the four modes given below).

IN [,SHARE / ,EXCL] specifies the input mode. SHARE specifies share mode for the DCB which allows more than one IN and/or INOUT user to access the file concurrently. EXCL specifies exclusive mode for the DCB which means that the user must have exclusive use of the file. The default is EXCL.

OUT specifies the output mode.

INOUT [,SHARE / ,EXCL] specifies the input and output mode (i.e., the update mode). SHARE specifies share mode for the DCB which allows more than one IN and/or INOUT user to access the file concurrently. EXCL specifies exclusive mode for the DCB which means that the user must have exclusive use of the file. The default is EXCL.

OUTIN specifies the output and input mode (i.e., the scratch mode).

file disposition (one of the two specifications given below).

REL specifies that the secondary storage allocated to this file is to be released when the file is closed. REL is significant only for OUT and OUTIN files and is assumed if file disposition is unspecified. See FILES, in the discussion of M:CLOSE.

SAVE specifies that the secondary storage allocated to this file is to be saved when the file is closed, unless otherwise specified by an M:CLOSE procedure call. If SAVE is not also specified in the M:CLOSE, the secondary storage allocated to this file will be released.

JOB specifies that the file is temporary and is to be kept across job steps but is to be released at the end of the job. (See M:TFILE and M:CLOSE.) This option is not available for private packs.

Other options

ABCERR specifies that block count errors are not to force an unconditional abort; i.e., that in the case of inconsistency between the tape-specified and the system-accumulated block counts, return is to the ERR address in the DCB – an abort is to occur only if there is no ERR address in the DCB. ABCERR is applicable for ANS labeled tapes only.

ABN,address specifies the symbolic address of a user's routine that is to be used to analyze any abnormal conditions associated with the makeup of the DCB. Allows the user to handle errors (such as end-of-file) himself rather than having the monitor handle them.

BLKL,value specifies block size in bytes. The value may be in the range 1 to 32,767. If a value less than 18 bytes is specified, 18 bytes are written. BLKL is applicable for ANS labeled tapes only.

BTD,value specifies the byte displacement (0-3) in the user's buffer from which I/O is to take place (i.e., at which byte in the buffer the data begins).

BUF,address specifies the symbolic address of a buffer that is to be used in the transfer of data or trailer labels.

CONCAT,value specifies the number of identically named files that are to be read as one logical file (concatenated). The value may be in the range 2 through 128. The default value is 0. CONCAT is applicable for ANS labeled tapes only.

CYLINDER specifies that the data blocks of a public file are to be allocated from public devices having cylinder allocation units. If CYLINDER is not specified, the data blocks of a public file are allocated from public devices having granule allocation units. In either case, the file will only be allocated on the type of device specified with the DEVICE option. If the DEVICE option is not specified, the system looks for space on public disk packs first and RADs last. If space is not available in the units requested, the file's data blocks will be allocated in the available units from public devices of the type requested. CYLINDER only has meaning for public files with keyed or consecutive organization.

ERR,address specifies the symbolic location of a user's routine that is to be used to analyze any error conditions associated with the makeup of the DCB (see the appendix titled "Monitor Error Messages"). Allows the user to handle errors himself rather than having the monitor handle them.

EXECUTE[,'value'[,'value']...][,n] 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. This option is not checked for any user who would have reserved access to the file by an explicit READ or WRITE specification. If n is specified, n words will be reserved in the variable length parameters for EXECUTE accounts. If EXECUTE is not specified, the default is no execute.

If this option is omitted from the M:DCB procedure call, it will not appear in the DCB and, consequently, will be ignored in the ASSIGN control command or M:OPEN procedure call. If EXECUTE is specified but no values are given, 16 words are reserved for EXECUTE account numbers (to be inserted via an ASSIGN control command or M:OPEN call). EXECUTE is applicable only to files.

EXPIRE, {mm,dd,yy / ddd / NEVER} specifies either an explicit expiration date (mm,dd,yy), the number of days to retain the file (ddd), or that the file is never to expire (NEVER). NEVER is not applicable for ANS labeled tapes. If not specified, the default value, as established in the authorization record for the user, will determine the expiration date. Files will be automatically purged from the public file system if they have expired whenever secondary storage space passes below a SYSGEN established threshold.

The value specified may not exceed the maximum expiration period authorized for the user. If the maximum expiration period is exceeded or unspecified, the default expiration period authorized for that user will be used. If this option is omitted from the M:DCB procedure call it will not appear in the DCB and, consequently, may not be used in an ASSIGN control command or M:OPEN procedure call referencing the DCB. If EXPIRE is specified but no value given in the M:DCB call, two words are reserved for the value (to be inserted via an ASSIGN control command or M:OPEN procedure call).

FORMAT,character specifies the record formats. The character may be

F – fixed length.

D – variable specified in decimal.

V – variable specified in binary.

U – undefined.

The default character is F. FORMAT is applicable for ANS labeled tapes only.

FPARAM,address specifies that the monitor is to pass the file parameters, in the same format as the variable-length parameters, to the user's program, beginning at the specified "address". The area in the user's program that is to receive the file parameters must be 90 words in length. Only the variable-length parameters are passed to the user's program. The account number is not returned, but other permanent file parameters are returned. FPARAM is not applicable for ANS labeled tapes.

KEYM,value specifies the maximum length, in bytes, of the keys associated with records within the file. If KEYM is not specified, the value 11 is assumed. A key may consist of up to 31 characters. KEYM is not applicable for ANS labeled tapes.

LRECL, value specifies the logical record size in bytes. The value may be in the range 1 to 32,767. LRECL is applicable for ANS labeled tapes only. The default value is the BLKL value.

NEWX,slides[,consecutive slides] allows the user to specify "when" and "if" a keyed file's higher-level index structure should be rebuilt. The higher-level index structure is built for the first time when a keyed filed that has more than three level 0 index blocks is closed.

slides 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 never be rebuilt. If NEWX is not specified, the value 254 is used in default.

consecutive slides 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.

NEWX is not applicable for ANS labeled tapes.

NOSEP specifies that the index blocks of a public keyed file are to be allocated in the same way that the data blocks are allocated. If NOSEP is not specified, the index blocks of a public file are allocated from public devices having granule allocation units. In either case, the file will only be allocated on the type of device specified with the DEVICE option. If the DEVICE option is not specified, the system looks for available granules on public disk packs first and RADs last. If space is not available in granule units, the system looks for space on public disk packs with cylinder allocation units. NOSEP only has meaning for public files with keyed organization.

PASS[,'value'] specifies the password that is to allow access to a classified data file. The value may be from 1 through 8 alphanumeric characters. If this option is omitted from the M:DCB procedure call it

will not appear in the DCB and, consequently, may not be used in an ASSIGN control command or M:OPEN procedure call referencing the DCB. If PASS is specified but no value given in the M:DCB call, two words are reserved for the value (to be inserted via an ASSIGN control command or M:OPEN call). PASS is not applicable for ANS labeled tapes.

READ[,'value']...[,n] 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 the file (e.g., READ, 'ALL') provided the user has X'40' privilege or greater. The value 'PUBL' may be used to specify that any user may read the file. Files cataloged under :SYS are accessible to any user as described without regard to privilege. The value 'NONE' may be used to specify that no other account may read the file. If no value is specified, or if READ 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 the READ or WRITE options must not exceed 16. If n is specified, n words will be reserved in the variable length parameters for read accounts. 'ALL' need not be specified unless it is desired to specifically override a default 'NONE' from the user's authorization.

If this option is omitted from the M:DCB procedure call it will not appear in the DCB and, consequently, may not be used in an ASSIGN control command or M:OPEN procedure call. If READ is specified but no values given, 16 words are reserved for READ account numbers (to be inserted via an ASSIGN control command or M:OPEN call). READ is not applicable for ANS labeled tapes.

RECL,value specifies the default record length, in bytes. The greatest value that may be specified is 32,767 if the count starts at 0, or 32,768 if the count starts at 1. If RECL is not specified, a standard value (appropriate to the type of device used) will apply by default. RECL is not applicable for ANS labeled tapes.

RSTORE,limit specifies, in decimal, the number of granules to be allocated to a RANDOM file. RSTORE is only honored when the file is first created. If no RSTORE value is given for a RANDOM file, M:DCB procedure generates one as a value. Unless changed by the time the DCB is opened, a file of one granule is created. RSTORE is not applicable for ANS labeled tapes. RSTORE must be in the range of 1 to $2^{24}-1$.

SN $\left[,\left\{\begin{matrix}\text{'serial number'},\ldots[,n] \\ n\end{matrix}\right\}\right]$ specifies the numbers of words to be reserved for serial numbers or the serial numbers to be used for file input or output. Space may be reserved for more serial numbers than are explicitly specified by serial number.

SN,'serial number',...[,n] specifies the serial numbers of the volumes (tape reels or disk packs) that are to be used for file input or output. The serial number may be from one to four alphanumeric characters for disk packs and Xerox labeled tapes. The serial numbers must consist of six alphanumeric characters for ANS labeled tapes. A maximum of three serial numbers may be specified for system DCBs. If n is specified, n words will be reserved in the variable length parameters (if n is greater than the number of listed serial numbers).

SN specifies that three words will be reserved in the variable length parameters for serial numbers that can be inserted through an ASSIGN control command or M:OPEN.

SN,n specifies that n words will be reserved in variable length parameters for serial numbers which can be inserted through the ASSIGN control command or M:OPEN. This will cause the third byte (byte 2) of the VLP control word for SN to be X'00' to allow the DCB to be used interchangeably as a labeled tape or file DCB. n must not exceed 50.

The SN option must be specified in the M:DCB procedure call for it to appear in the DCB so that it may be used by the ASSIGN control command or the M:OPEN procedure call.

For a file on a labeled tape:

1. Serial numbers must be ordered in the proper sequence. If SN is not specified (by ASSIGN, M:DCB or M:OPEN) for a file to be opened in the IN or INOUT mode, the DCB is not opened and an abnormal code of X'14' is returned.

2. The file will be written in the order in which the serial numbers are specified for a file to be opened in the OUT or OUTIN mode. If SN is not specified (by ASSIGN, M:DCB or M:OPEN), available scratch volume(s) of the type specified in the DEVICE option (or by default, any type available) will be used.

For a file on a private volume set:

1. When the first file on a private volume set is created, all serial numbers in the set must be specified and the first volume in the set will become the primary volume.

2. If the private volume set has been established, only the serial number of the primary volume need be specified. The primary volume contains a list of all serial numbers in the set.

3. If one or more volumes are to be added to the set, the serial numbers of the new volume(s) must be specified following the primary volume.

4. If SN is not specified (by ASSIGN, M:DCB or M:OPEN) for a file on RAD or DP, the file is assumed to be on public devices.

The INSN and OUTSN options used in the previous versions of the monitor were replaced with the SN option. For compatibility, the INSN and OUTSN options are acceptable in lieu of SN.

SPARE,n 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. The value specified may not exceed 255 bytes; if it does, it is treated modulo 256. If SPARE is not specified or is zero, it is set to 1 byte by default. This spare space is used so that additional keys can be inserted in a minimum time when updating the file with direct access (as in EDIT). If the file will never be updated with direct access, a spare value of 1 should be specified. SPARE is not applicable for tapes.

SYNON,'filename' specifies that the "name" given in the FILE option (see above) is to be considered synonymous with the designated filename. The filename must exist in the file directory of the account specified for "name". This option is used to create a synonym for a file name. It forces the DCB to be opened in the update mode. If SYNON is not specified in the M:DCB procedure call, it will not appear in the DCB and, therefore, may not be used in an M:OPEN procedure call referencing the DCB. If SYNON is specified but no value given, eight words are reserved for the file name (to be inserted via an M:OPEN call). SYNON is not applicable for ANS labeled tapes.

TLABEL,address specifies the symbolic address of a user's buffer into which a label is to be read, or from which a label is to be written upon opening a tape file. The first byte of the label information must contain the length (i.e., number of bytes) of the buffer. For ANS labeled tapes, the count must be 80 and the next four bytes of the buffer must contain UHL1.

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.

UNDER[,'name']...[,n] specifies the name(s) of the processor(s) that may access this file if the user does not own the file. The name(s) may be from one to ten characters enclosed within single quotes ('). The processor(s) 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. If n is specified, n words will be reserved in the variable length parameters for UNDER names. Fetch must not be used explicitly.

VOL,value specifies which volume in the SN list is to be used initially. A value of 1 designates the first reel (in the list), the value 2 designates the second reel, etc. If VOL is omitted, a value of 1 is assumed by default. The VOL option only has meaning for tapes and private disk packs.

WRITE[,'value']...[,n] specifies the account numbers of those accounts that may have both read and write access to the file. The values 'PUBL', 'ALL' and 'NONE' may be used, as with the READ option (see above); and if a conflict exists between READ and WRITE specifications, those of the WRITE option take precedence. If no WRITE accounts are specified, NONE is assumed. If n is specified, n words will be reserved in the variable length parameters for WRITE accounts.

If the WRITE is omitted from the M:DCB procedure call it will not appear in the DCB and, consequently, may not be used in an ASSIGN control command or M:OPEN procedure call. If WRITE is specified but no values given, 16 words are reserved for WRITE account numbers (to be inserted via an ASSIGN control command or M:OPEN call). WRITE is not applicable for ANS labeled tapes.

The following options are device-dependent, and will be ignored by the monitor in all cases where they are not applicable to the device used.

ASCII specifies that the data is to be converted between EBCDIC characters in core and ASCII characters on tape. Applies only to ANS labeled tape and unlabeled tape. Causes error code 1413 if used for Xerox labeled tape. Causes error code 1411 if used for drives not having the code conversion feature.

COUNT,tab specifies that a page count is to appear at the top of each page, beginning in the column specified by "tab".

Example:

```
COUNT,60
```

The above example specifies that the most singificant digit of the page count is to appear in column 60 at the top of each page.

DATA,tab specifies that output is to begin on each page (or card, if EBCDIC) in the column specified by "tab".

DENS,value specifies the density for writing a tape on a dual density tape drive. The value must be either 800 or 1600. The default is 1600. Specification of 800 for a drive not having the dual density feature causes abnormal code 1412.

EBCDIC specifies that EBCDIC is to be used when reading and writing a tape (i.e., conversion to ASCII is not to occur).

HEADER,tab,address specifies that the I/O handler is to output a header (heading) on each page. Tab specifies the column at which the header is to begin. Address specifies the symbolic location of the header; the first byte of the header must contain the number of bytes.

LINES,value specifies the number of printable lines per page. The greatest value that may be specified is 32,767. If LINES is not specified, the value established at system generation time will apply.

SEQ[,'id'] specifies that the punched output is to have sequencing in columns 77-80. If 'id' is specified, it will appear in columns 73-76 of the punched output. Sequencing begins with 0000.

SPACE,value[,top] specifies the spacing between lines (value) and the number of the first printed line on the page (top). A value of 1 indicates that lines are to be single spaced. The greatest value that may be specified is 15.

TAB,value[,value]... specifies the values of tab stop settings (for an output device). The values must be in ascending order.

format (any of the following specifications).

VFC specifies that the first character of each record is a format-control character for printing (see Table 4).

NOVFC specifies that the records do not contain format-control characters.

DRC specifies that the monitor is not to do special formatting of records on read or write operations.

NODRC specifies that the monitor is to do record formatting on read or write operations. If neither DRC nor NODRC is specified, NODRC is assumed by default.

mode (any of the following specifications for a device I/O mode).

BCD specifies that the EBCDIC device mode is to be used.

BIN specifies that the binary device mode is to be used.

FBCD specifies that FORTRAN BCD conversion is to be used.

L specifies that a listing type of device is to be used.

NOFBCD specifies that FORTRAN BCD conversion is not to be used.

PACK specifies that the packed binary mode (7-track tape) is to be used. PACK is not valid unless BIN is specified.

UNPACK specifies that the unpacked binary mode (7-track tape) is to be used. UNPACK is not valid unless BIN is specified.

If no mode is specified, BCD is assumed.

The formats of the file, Xerox labeled tape, ANS labeled tape, and device DCBs are shown in the appendix titled "Data Control Block Formats".

SPECIAL NOTE

After generating the DCB, Meta-Symbol will resume assembly in whatever control dummy section was in effect when the M:DCB procedure reference line was encountered. In order to prevent the statements following the M:DCB procedure reference line from being assembled in the same control/dummy section as the DCB, one of the following is recommended:

1. The control section directive preceding an M:DCB procedure reference line should be a CSECT, and the DSECT associated with an M:DCB should precede the CSECT.
2. The statement immediately following an M:DCB procedure reference line should be either a CSECT or a USECT referencing a prior CSECT.

OPEN A FILE (Initialize a DCB)

M:OPEN The monitor OPEN routine initializes specified parameters of a designated DCB.

Files (on RAD, DP, or labeled tape) are normally positioned to the beginning of file, except when file extension is required. File extension will occur when a system output DCB (e.g., M:BO) is opened more than once during the same job, without an intervening ASSIGN control command referencing the DCB. File extension will occur because the second, or subsequent, OPEN will cause the file to be positioned at the end of the last data record to permit additional output to be appended at the end.

Files that are assigned via user DCBs or system input DCBs cannot be extended.

If a READ or WRITE I/O routine is called (see M:READ and M:WRITE procedures) when the DCB has not been opened, the monitor stores the call temporarily and calls the OPEN routine automatically. If the DCB does not get opened, the requested read or write operation is not executed. The DCB will not be opened if the information in the DCB is insufficient, inaccurate, or contradictory, and the resulting abnormal or error code will be returned in byte 0 of SR3. If the OPEN is made with no parameters, the existing parameters in the DCB are used.

The READ or WRITE option must be specified in the M:DCB procedure call for it to appear in the DCB so that it may be used by the ASSIGN control command or the M:OPEN procedure call. When READ or WRITE are specified in the M:DCB procedure call but no account numbers are given, 16 words are reserved for either READ or WRITE and can subsequently be filled by ASSIGN or M:OPEN.

If the specified DCB is already open when the OPEN routine is explicitly called, an abnormal condition is signaled (see the appendix titled "Monitor Error Messages"). If the DCB is not open when the OPEN routine is called, the DCB is reinitialized according to the parameters specified in the M:OPEN procedure call.

In a multiprogramming environment like CP-V, files may not be available on request because of current use by another program. An error code reports this condition (code 14, subcode 01) and programs may need special routines to handle it. A common solution is to use the WAIT CAL for a minute or so and then repeat the request.

In addition, the number of CFUs (current file usage) limits the number of files that may be opened by all active users and batch jobs at a given time since a CFU data recording

block is required for each open file. CFUs take up 19 words. The space required for CFUs is core resident in the monitor at all times.

The M:OPEN procedure call is of the form

M:OPEN [*] dcb name, [(option)][, (option)]. . .

where dcb name specifies the name of the DCB that is to be opened. The options specified in the OPEN call override those previously specified. If no options are specified, the existing ones are used. The optional asterisk (*) can be used for indirect addressing.

The third element of the label field list in the OPEN call is used to set a symbol to the address of the VLP list that is generated by the M:OPEN.

Example:

OPN, OPNFPT, VLPLOC M:OPEN M:EI, (FILE, 'A')

The M:OPEN options are as follows:

name (one or two of the four keyword operands given below).

DEVICE,'name' specifies a device type, a system operational label, or a logical device stream. Acceptable forms of the name specifications are (1) for a device type – 'CR', 'LP', '7T', '9T', etc.; (2) for an operational label – 'LO', 'EO', 'LL', 'C', etc.; (3) for a logical device stream – 'P1', 'C1', 'L1'. DEVICE may be used in conjunction with FILE, LABEL or ANSLBL.

FILE,'name'$_L$,'account'] specifies the name of the public or private file that is to be assigned to the DCB. The name may consist of up to 31 alphanumeric characters. The named file will be maintained on RAD or DP storage. If the file is private, the SN option must be used to specify the serial number(s) of the private volume set. If the named file belongs to a different account than that of the current job, the file's account number (not exceeding 8 characters), must be given. If the file is public and is (or is to be) cataloged under the log-on account, and if the second and third characters of the name are both colons, then for DCBs with the JOB file disposition (see below), the colons are replaced by the system identification number of the current user. This feature allows a processor to be written such that it may be executed concurrently by more than one user logged on under the same account without file conflict.

LABEL,'name'[,'account'] specifies the name (not exceeding 31 characters) of a file on Xerox labeled tape. The SN option must be used to specify the particular tape reel(s) containing an input file. If the named file belongs to a different account that that of the current job, the file's account number (not exceeding 8 characters) must be given.

ANSLBL,'name' specifies the name of a file on ANS labeled magnetic tape that is to be assigned to the DCB. The name may consist of up to 17 alphanumeric characters. If the file name contains a special character, it must be enclosed by single quotation marks. When a single quotation mark is to be used as part of the file name, it must be coded as two successive quotation marks. There should be no blanks between the last character and the terminating quotation marks.

ASN, {FILE / LABEL / DEVICE / ANSLBL} specifies the value for the ASN field of the FPT. This value will override the default value created by other keywords such as FILE. The ASN values are:

1 - FILE
2 - LABEL
3 - DEVICE
5 - ANSLBL

org (one of the four file organization types given below. Not applicable to ANS labeled tapes).

CONSEC specifies that the records in the file are consecutively organized and each record is to be processed in order.

If a private file has consecutive organization, only one volume in the private volume set need be mounted at any time. As another volume is required, the system will request that it be mounted.

KEYED specifies that the location of each record in the file is determined by an explicit identifier (key) that may be used to address the device containing the file. A key may consist of up to 31 characters.

If a private file has keyed organization, all volumes in the set must be mounted when the file is opened and remain mounted until the file is closed.

RANDOM specifies that the contents of the file comprise a collection of contiguous granules on the specified device type. Data is accessed by relative granule number and byte count. If device type is not specified, the file will be allocated on RAD or disk pack, whichever is available.

If a private file has random organization, all volumes in the set must be mounted when the file is opened and remain mounted until the file is closed.

If an organization type is omitted and there is no value in the DCB, CONSEC is assumed.

UNDEF specifies that Xerox labeled tape records are all unblocked and without headers, i.e., equivalent to device format. BLOCK access is forced for this organization. (Applicable only to Xerox labeled tape.)

access (one of the three record access means given below. Not applicable to ANS labeled tapes unless otherwise noted.)

SEQUEN specifies that records in the file are to be accessed in the order in which they appear in the file.

DIRECT specifies that the next record to be accessed is to be determined by an explicit identifier (key). If specified for consecutive or keyed disk files, read ahead will be disabled.

If an access option is omitted and there is no value in the DCB, SEQUEN is assumed. The common defaults for access and organization are shown in Table 15.

BLOCK specifies that data blocks are transferred directly between tape and the user's buffer. (Applicable only to Xerox and ANS labeled tape.) This access is forced for UNDEF organization and for ANS tape.

function (one of the four modes given below.)

IN [,SHARE / ,EXCL] specifies the input mode. SHARE specifies share mode for the DCB which allows more than one IN and/or INOUT user to access the file concurrently. EXCL specifies exclusive mode for the DCB which means that any number of IN but no INOUT users may access the file concurrently. The default is EXCL.

OUT specifies the output mode.

INOUT [,SHARE / ,EXCL] specifies the input and output mode (i.e., the update mode). SHARE specifies share mode for the DCB which allows more than one IN and/or INOUT user to access the file concurrently. EXCL specifies exclusive mode for the DCB which means that the user must have exclusive use of the file. The default is EXCL.

OUTIN specifies the output and input mode (i.e., the scratch mode).

file disposition (one of the two specifications given below.)

REL specifies that the secondary storage allocated to this file is to be released when the file is closed. REL is significant only for OUT and OUTIN files and is assumed if file disposition is unspecified. See FILES, in the discussion of M:CLOSE.

SAVE specifies that the secondary storage allocated to this file is not to be released when the file is closed, unless otherwise specified by an M:CLOSE procedure call. If REL is specified in the M:CLOSE, the secondary storage allocated to this file will be released. SAVE is assumed for IN or INOUT if file disposition is unspecified. (See M:CLOSE procedure, below.)

JOB specifies that the file is temporary and is to be kept across JOB steps but is to be released at the end of the job. (See M:TFILE and M:CLOSE.) Job files may only be opened by the creating user or a user with C0 or greater privilege. This option is not available for private packs.

Table 15. File Defaults

DCB		M:OPEN		Outcome (in DCB)	
org	access	org	access	org	access
Null	Null	Null	Null	CONSEC	SEQUEN
Null	Null	Null	SEQUEN	CONSEC	SEQUEN
Null	Null	CONSEC	Null	CONSEC	SEQUEN
Null	Null	Null	DIRECT	KEYED	DIRECT
Null	Null	KEYED	Null	KEYED	DIRECT
CONSEC	SEQUEN	Null	Null	CONSEC	SEQUEN
CONSEC	SEQUEN	Null	DIRECT	CONSEC	DIRECT
CONSEC	SEQUEN	KEYED	Null	KEYED	SEQUEN
KEYED	DIRECT	Null	Null	KEYED	DIRECT
KEYED	DIRECT	Null	SEQUEN	KEYED	SEQUEN
KEYED	DIRECT	CONSEC	Null	CONSEC	SEQUEN

Other Options

ABCERR specifies that block count errors are not to force an unconditional abort; i.e., that in the case of inconsistency between the tape-specified and the system-accumulated block counts, return is to the ERR address in the DCB – an abort occurs only if there is no ERR address in the DCB. ABCERR is only applicable for ANS labeled tapes.

ABN,[*]address specifies the symbolic location of a user's routine that is to be used to analyze any abnormal conditions associated with the makeup of the DCB (see the appendix titled "Monitor Error Messages"). The address specified must lie within the user's program.

ASCII specifies that the data is to be converted between EBCDIC characters in core and ASCII characters on tape. Applies only to ANS labeled tape and unlabeled tape. Causes error code 1413 if used for Xerox labeled tape. Causes error code 1411 if used for drives not having the code conversion feature.

BLKL,value specifies block size in bytes. The value may be in the range 1 to 32,767. If a value less than 18 bytes is specified, 18 bytes are written. BLKL is only applicable for ANS labeled tapes.

BTD,value specifies the byte displacement (0-3) in the user's buffer from which I/O is to take place (i.e., at which byte in the buffer the data begins).

BUF,[*]address specifies the symbolic address of a buffer that is to be used in the transfer of data (the buffer may not include a general register).

CONCAT,value specifies the number of identically named files that are to be read as one logical file (concatenated). The value may be in the range 2 through 128. CONCAT is only applicable for ANS labeled tapes.

CYLINDER specifies that the public file is to be allocated on public devices having cylinder allocation units. If CYLINDER is not specified, the public file is to be allocated from public devices having granule allocation units. In either case, the file will only be allocated on the type of device specified with the DEVICE option. If the DEVICE option is not specified, the system looks for space on public disk packs first and RADs last. If space is not available in the units requested, the file will be allocated in the available units from public devices of the type requested. CYLINDER only has meaning for public files.

DENS,value specifies the density for writing a tape on a dual density tape drive. The value must be either 800 or 1600. Specification of 800 for a drive not having the dual density feature causes error code 1412.

EBCDIC specifies that EBCDIC is to be used when reading and writing a tape (i.e., conversion to ASCII is not to occur).

ERR,[*]address specifies the symbolic location of a user's routine that is to be used to analyze any error conditions associated with the makeup of the DCB (see the appendix titled "Monitor Error Messages"). The address specified must lie within the user's program.

EXPIRE,{mm,dd,yy / ddd / NEVER} specifies either an explicit expiration date (mm,dd,yy), the number of days to retain the file (ddd), or that the file is never to expire (NEVER). NEVER is not applicable for ANS labeled tapes. If not specified, the default value as established in the authorization record for the user will determine the expiration date. Files will be automatically purged from the public file system if they have expired whenever secondary storage space passes below a SYSGEN established threshold.

The value specified may not exceed the maximum expiration period authorized for the user. If the maximum expiration period is exceeded or unspecified, the default expiration period authorized for that user will be used. If this option is omitted from the M:DCB procedure call it will not appear in the DCB and, consequently, may not be used in an ASSIGN control command or M:OPEN procedure call referencing the DCB.

If EXPIRE is specified but no value given in the M:DCB call, two words are reserved for the value (to be inserted via an ASSIGN control command or M:OPEN procedure call).

FORMAT,character specifies the record formats for ANS labeled tapes. The character may be

F - fixed length.

D - variable specified in decimal.

V - variable specified in binary.

U - undefined.

FPARAM,address specifies that the monitor is to pass the file parameters from the FIT to the memory location beginning at the specified address. The area of the user's program that is to receive the file parameters must be 90 words in length. The format of the file parameters is given in Appendix A and internally is similar to the format of variable length parameters of both M:OPEN and a DCB. FPARAM is not applicable to ANS labeled tapes. Once specified, FPARAM remains in effect until changed. An address of zero must be specified to halt the passing of file parameters.

KEYM,value specifies the maximum length, in bytes, of the keys associated with records within the file. If KEYM is omitted, the value 11 is assumed. KEYM applies to OUT or OUTIN files only and is not applicable

to ANS labeled tapes. A key may consist of up to 31 characters.

LRECL,value specifies the logical record size in bytes. The value may be in the range 1 to 32,767. LRECL is only applicable for ANS labeled tapes.

NEWX,slides[,consecutive slides] allows the user to specify "when" and "if" a keyed file's higher-level index-structure should be rebuilt. The higher-level index structure is built for the first time when a keyed file that has more than three level 0 index blocks is closed.

slides 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 never be rebuilt. If a NEWX is not specified, the value 254 is used in default.

consecutive slides 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.

NEWX is not applicable for tapes.

NOSEP specifies that the index blocks of a public keyed file are to be allocated in the same way that the data blocks are allocated. If NOSEP is not specified, the index blocks of a public keyed file are allocated from public devices having granule allocation units. In either case, the file will only be allocated on the type of device specified with the DEVICE option. If the DEVICE option is not specified, the system looks for available granules on public disk packs first and RADs last. If space is not available in granule units, the system looks for space on public disk packs with cylinder allocation units. NOSEP only has meaning for public files with keyed organization.

NXTA specifies that when the DCB is opened, the next account in the public file's system account directory following the account specified in the DCB is to be accessed. If there is no account specified in the DCB, the first account in the account directory is put in the DCB. If the NXTF option is also specified, the first file in the account is assigned to the DCB and if the file security checks are satisfied, the DCB is opened. If the NXTF option is not specified along with the NXTA option, the requested account number is put in the DCB and returned to the user with the DCB not opened. If there are no more accounts available, an abnormal code of X'02' with a subcode of X'01' is returned. NXTA only has meaning for DCBs assigned to public files. A privilege level of at least '80' is required to use this option.

NXTF specifies that when the DCB is opened for RAD or DP files or Xerox labeled tape, the monitor is to access the next file in sequence (following the one most recently accessed via the DCB). If no file name is specified (currently) in the DCB, the first file on the tape or in the DCB's account file directory is accessed. If the next file is a synonymous file, an abnormal code of X'08' is returned and the DCB is not opened. If there are no more files available, an abnormal code of X'02' is returned and the DCB is not opened. If the file name variable length parameter in the DCB is not 8 words long, abnormal code 1406 is returned.

PASS,'value' specifies the password that allows access to a classified data file. The 'value' may be from 1 through 8 alphanumeric characters in length. (PASS is not applicable to ANS labeled tapes.) The referenced DCB must have room to hold the password.

READ[,'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 the file (e.g., READ,'ALL') provided the user has X '40' privilege or greater. The value 'PUBL' may be used to specify that any user may read the file. Files cataloged under :SYS are available to any user as described without regard to privilege. The value 'NONE' may be used to specify that no other account may read the file. If no value is specified, or if READ 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 the READ and WRITE options together must not exceed 16 (a maximum of 8 READ and 8 WRITE). READ applies to OUT or OUTIN files only and is not applicable to ANS labeled tapes. The referenced DCB must have room to hold the READ list. (Also see WRITE, below.)

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 by default. RECL is not applicable to ANS labeled tapes.

RSTORE,{limit / *address of limit} specifies in decimal the number of granules to be allocated to a random file. RSTORE is meaningful only when a file is first created. The RSTORE value is required to be in the range 1 to $2^{24}-1$. RSTORE is not applicable to ANS labeled tapes.

SN[,'serial number',...] specifies the serial numbers of the volumes (tape reels or disk packs) that are to be used for file input or output. The serial number may be from one to four alphanumeric characters for Xerox labeled tapes and disk packs. The serial number must be six alphanumeric characters for ANS labeled tapes. A maximum of three serial numbers may be specified for system DCBs.

The SN option must be specified in the M:DCB procedure call for it to appear in the DCB so that it may be used by the ASSIGN control command or the M:OPEN procedure call. When SN is specified in the M:DCB procedure call but no serial numbers are given, three words are reserved for the serial numbers which can be inserted through ASSIGN or M:OPEN.

For a file on labeled tape:

1. Serial numbers must be ordered in the proper sequence for a file to be opened in the IN or INOUT mode. If SN is not specified (by ASSIGN, M:DCB or M:OPEN) the DCB is not opened and an abnormal code of X'14' is returned.
2. The file is written in the order in which the serial numbers are specified for a file to be opened in the OUT or OUTIN mode. If SN is not specified (by ASSIGN, M:DCB or M:OPEN), available scratch volume(s) of the type specified in the DEVICE option (or by default, any type available) will be used.

For a file on a private volume set:

1. If the first file on a private volume set is being output, all serial numbers in the set must be specified and the first volume in the set will become the primary volume.
2. If the private volume set has been established, only the serial number of the primary volume need be specified. The primary volume contains a list of all serial numbers in the set.
3. If one or more volumes are to be added to the set, the serial numbers of the new volume(s) must be specified following the primary volume.
4. If SN is not specified (by ASSIGN, M:DCB or M:OPEN) for a file on RAD or DP, the file is assumed to be on public devices.

The INSN and OUTSN options used in previous versions of the monitor were replaced with the SN option. For compatibility, the INSN and OUTSN options are acceptable in lieu of SN.

SPARE,n 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. The value specified may not exceed 255 bytes; if it does, it is treated modulo 256. If SPARE is not specified or is zero, it is set to 1 byte by default. This spare space is used so that additional keys can be inserted in a minimum time when updating the file with direct access (as in EDIT). If the file will never be updated with direct access, a spare value of 1 should be specified. SPARE is not applicable for tapes.

EXECUTE[,'value'] . . . specifies the account numbers of accounts that may execute the file. The value 'ALL' allows any account to execute the file. The value 'NONE' prohibits execution of the file by any account except the creator. The value 'PUBL' is meaningless here. The EXECUTE option is not checked for users that would have received READ or WRITE access.

SYNON,'filename' specifies that the "name" given in the FILE option name attribute is to be considered synonymous with the designated "filename". The filename must currently apply to a file that exists on RAD or DP. This option is used to create a synonym for a RAD or DP filename. It forces the function mode in the DCB to INOUT.

TEST specifies that this is a test file operation. The DCB is not opened and subsequently does not require a CLOSE. TEST is normally used in conjunction with the NXTA and NXTF options to return information regarding files via the DCB variable length parameters and FPARAM. Use of TEST will force the function to IN regardless of what is specified in the DCB or FPT. TEST is not applicable for tapes. If FPARAM is specified, security checks will be made and error 14-00 returned if the user does not have access to the file. Under no circumstances will error 14-01 (file busy) be returned when TEST is specified.

TLABEL,[*]address specifies the symbolic address of the user's buffer into which a label is to be written. The first byte of the label information must contain the length (i.e., number of bytes) of the buffer. For ANS labeled tapes, the count must be 80 and the first four bytes of the label must contain UHL1.

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.

UNDER,['name']... specifies the name(s) of the only processor(s) that may access this file if the user does not own the file. The name(s) may be from one to four characters enclosed within single quotes ('). The processor(s) 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. FETCH must not be stated explicitly.

VOL,value specifies which tape reel in the SN list to be used initially. A value of 1 designates the first reel (in this list), the value 2 designates the second reel, etc. If VOL is omitted, a value of 1 is assumed by default.

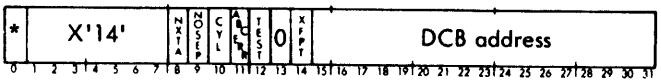
WRITE[,'value']... specifies the account numbers of those accounts that may have both read and write access to the file, The values 'PUBL', 'ALL' and 'NONE' may be used, as with the READ option; and, if a conflict exists between READ and WRITE specifications, those of the WRITE option take precedence. If no WRITE accounts are specified, 'NONE' is assumed. WRITE applies to OUT or OUTIN files only and is not applicable to ANS labeled tapes. The referenced DCB must have room to hold the WRITE list.

Calls generated by the M:OPEN procedure have the form

CAL, 1 fpt

where fpt points to word 0 of the FPT shown below.

word 0

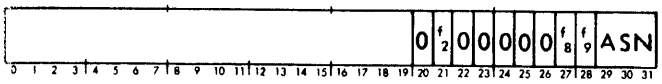


If XFPT= 0 (see word 0, bit 14).

word 1

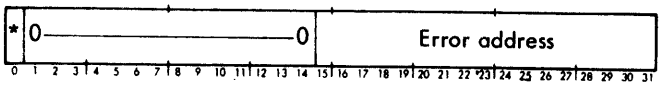
If XFPT = 1 (see word 0, bit 14).

word 1

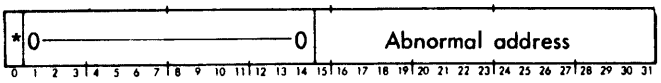


word 2

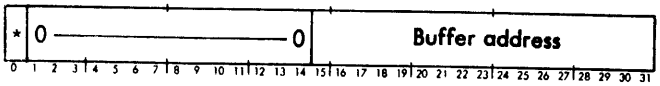
option ERR (P1)



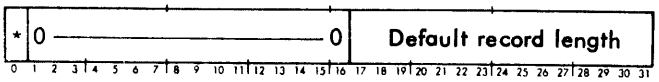
option ABN (P2)



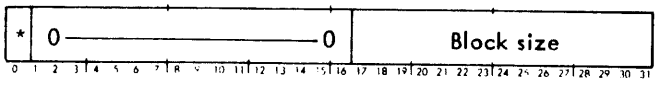
option BUF (P3)



option RECL (P4)

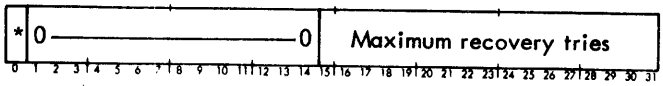


option BLKL (P4) altemate form

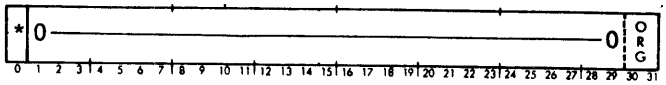


Block size is only applicable for ANS labeled tapes.

option TRIES (P5)

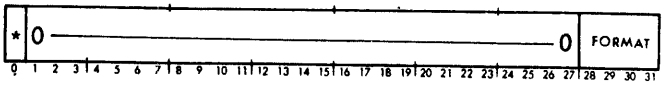


option { CONSEC | KEYED | RANDOM | UNDEF | ORG, { value | *location } } (P6)



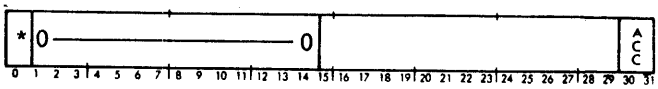
where ORG specifies the file organization type (1 means consecutive, 2 means keyed, 3 means random, 4 means undefined). If this option is omitted, consecutive is assumed by default.

option FORMAT (P6) alternate form



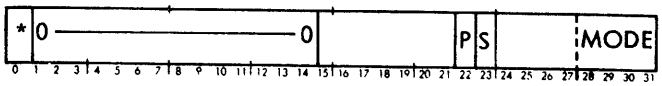
where FORMAT specifies the record format for ANS labeled tapes: 1 = F (fixed length), 2 = D (variable, expressed in decimal), 3 = V (variable, expressed in binary), 4 = U (undefined).

option { SEQUEN | DIRECT | BLOCK | ACC, { value | *location } } (P7)



where ACC specifies the record access method (1 means sequential, 2 means direct, 3 means block).

option { IN | OUT | INOUT | OUTIN | MODE, { value | *location } } [, SHARE | , EXCL] (P8)



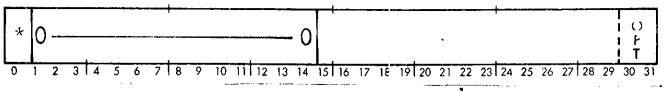
where

MODE specifies the file function mode (1 means IN, 2 – OUT, 4 – INOUT, 8 – OUTIN).

P is a presence bit to indicate whether or not the S field is significant. If P is set to one, S is significant. If P is set to zero, S is not significant.

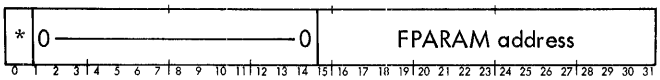
S specifies SHARE (S = 1) or EXCLUSIVE (S = 0). For RANDOM and KEYED files only, more than one user may open the file in the INOUT mode, if, and only if, all such open requests specify SHARE. If a RANDOM or KEYED file is already open OUTIN, all requests to open the file in the IN mode must specify SHARE or EXCLUSIVE exactly as the OUTIN open request did. If a RANDOM or KEYED file is already open in the IN mode, all requests to open the file in the OUTIN mode must specify SHARE or EXCLUSIVE exactly as the IN open request did.

option { RELEASE | SAVE | JOB | DISP, { value | *location } } (P10)

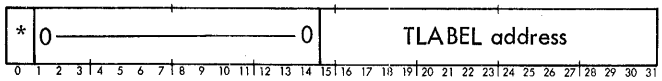


where OPT specifies REL/SAVE/JOB option (1 means RELEASE, 2 means SAVE), 3 means JOB).

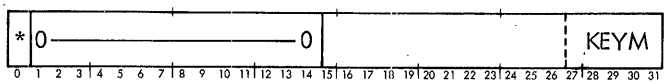
option FPARAM (P11)



option TLABEL (P12)

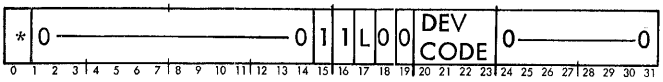


option KEYM (P13)



where KEYM specifies the maximum key length.

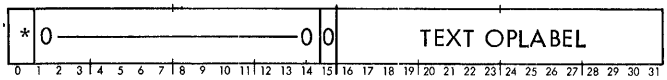
option device code (P14)



where DEV CODE is the device type code set in the DCB.

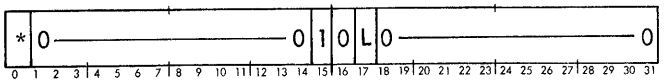
If the assignment mode currently in effect is file and neither DC nor DP is specified, the monitor will allocate files on either device. DP or DC are only meaningful when the function mode is OUT or OUTIN.

option device code (P14) alternate form



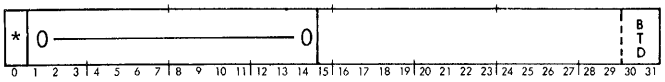
where TEXT OPLABEL is the device name in TEXT format.

option device code (P14) alternate form



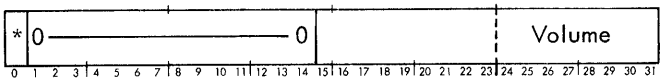
causes the DCB to be opened to any listing type device, i.e., any LP device. The L bit will be set in the DCB as it is set here.

option BTD (P15)

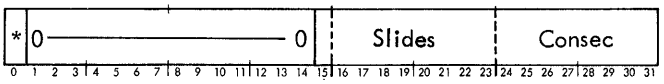


Byte displacement

option VOL (P16)

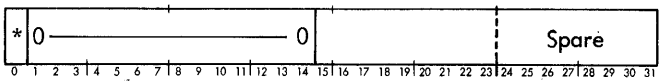


option NEWX (P17)

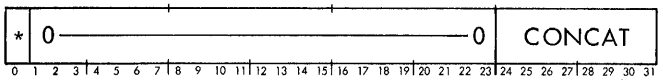


where Slides and Consec are as described in NEWX.

option SPARE (P18)

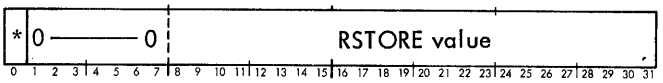


option CONCAT (P18) alternate form

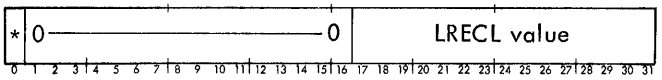


where CONCAT specifies the number of identically named files that are to be read as one logical file (concatenated). CONCAT is only applicable for ANS labeled tapes.

option RSTORE (P20)



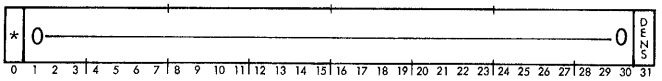
option LRECL (P20) alternate form



LRECL (logical record length) is only applicable for ANS labeled tapes.

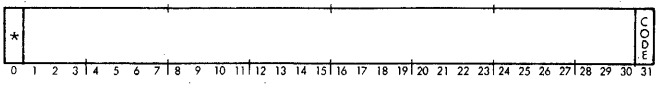
Parameters 21 and 22 are available only if the extended format FPT (XFPT = 1) is used.

option DENS (P21)



where DENS has the following meanings: 0 = 1600 bpi; 1 = 800 bpi.

option $\begin{Bmatrix}\text{EBCDIC}\\\text{ASCII}\end{Bmatrix}$ (P22)



where CODE is 0 for EBCDIC, 1 for ASCII.

Entries for any variable-length parameters follow those for the fixed-length parameters previously discussed. The variable-length entries are identical in format to those of a DCB (see Appendix A).

Flags f_1 through f_9 in word 1 of the FPT have the significance indicated below (when $f_i = 1$).

Flag	Significance (when set to 1)
f_2	The next file of the account is to be opened. If there is no name specified in the DCB currently open, the first file of the account is to be opened.
f_9	VLPs are present.
ASN	A three-bit field indicating the assignment type: 000 = null 001 = file 010 = CP-V labeled tape 011 = device 101 = ANS labeled tape

CLOSE A FILE (Terminate I/O Through a DCB)

M:CLOSE The monitor CLOSE routine terminates and inhibits I/O through a specified DCB, until the DCB is again opened.

In addition, unique positioning and updating operations occur for the following devices and files.

CARD PUNCH DEVICES

If the direct device format option DRC was not specified in the DCB, an !EOD record is output to the device indicating an end-of-file.

TAPE UPDATING

For unlabeled tape, if the last operation performed was a write and the direct device format option DRC was not specified in the DCB, two end-of-file marks are written and the tape is positioned between them.

For labeled tape, if the last operation performed was a write, an end-of-file and an end-of-reel sentinel are written. If the LABEL option in the M:CLOSE is specified, a trailer label is also written as a part of the end-of-file sentinel. If the function mode specified in the DCB is OUT or OUTIN and the SAVE option of M:CLOSE is not specified, the tape will be unconditionally rewound. The SAVE option must be specified if the tape is to have more than one file. The REM option should not be specified until the last file on the tape is closed.

TAPE POSITIONING

Input tapes closed by the monitor CLOSE routine are always saved. The REM (remove) option, if specified, is honored on both labeled and unlabeled tapes; the PTL (position to label) option is honored on labeled tapes only. Output, update, and scratch tapes closed by the monitor CLOSE routine are handled as indicated in Table 16.

FILES

If the file's function mode in the DCB is OUT or OUTIN, and the SAVE option of M:CLOSE is not specified, REL is assumed. The monitor will release all secondary storage allocated to the file and no record of the file will be maintained.

If the file's function mode is OUT or OUTIN and the SAVE option is specified in both the DCB and M:CLOSE, an entry for the file will be created in the account's file directory. If there is already a file in the user's account with the same file name, the new file will replace the old, and the secondary storage allocated to the old file will be released. In this case, if the JOB option was specified in the DCB, the file name will be processed through the M:TFILE procedure.

If the file's function is IN or INOUT SHAREd and the REL option of M:CLOSE is specified, the file is treated as though SAVE had been specified.

If the file's function mode is IN or INOUT EXCLusive and the REL option of M:CLOSE is specified, the monitor will release all secondary storage allocated to the file and delete the file's name from the account's file directory. All files synonymous to the file being released will also be deleted from the file directory.

If the file's function mode is IN or INOUT and the REL option of M:CLOSE is not specified, SAVE is assumed and no further action is required (there are already entries in the file directory).

If the file's function is INOUT EXCLusive, SAVE is specified (either implicitly or explicitly), and the file is not synonymous, file attributes may be altered by specifying any of the following options on M:CLOSE: FILE, PASS, READ, WRITE, EXECUTE, UNDER.

The M:CLOSE procedure call is of the form

M:CLOSE [*]dcb name[, (option)]...

Table 16. Tape Positioning for Output, Update, and Scratch Tapes

SAVE option is specified	Unlabeled tapes	Assigned to OPLABEL	The tape is rewound and SAVE message is output.
		Assigned to device	If REM is specified, the tape is rewound and the drive is taken off-line; otherwise, no action is taken. SAVE message is output.
SAVE option is specified	Labeled tapes	REM is specified	The tape is rewound and the drive is taken off-line. SAVE message is output.
		PTL is specified	The file is positioned to the label at the beginning-of-file; (positioned in front of tape mark preceding :EOF sentinel); if the label is on another tape, SAVE message is output (PTL is ignored for ANS tapes.)
		PTV is specified	(Applicable for ANS tapes only.) The tape is positioned as if an AVR sequence has been specified.
		No options; ANS; no SN specified	If the file is contained on one volume, PTV is performed. If the file is multivolume, REM is performed.
		No options	No action is taken. Tape head is somewhere between tape marks surrounding records, depending on last record read or written, and direction.
SAVE option is not specified	Unlabeled tapes	SN is not specified	The tape is rewound and remains a scratch tape.
		SN is specified	If REM is specified, the tape is rewound; otherwise, no action is taken.
SAVE option is not specified	Labeled tapes	SN is not specified	The tape is rewound and remains a scratch tape. Any other scratch tapes saved (due to CVOL) for the files are released for other use.
		SN is specified	The tape is rewound. If REM is specified, DISMOUNT message is output.

where dcb name specifies the name of the DCB to be closed.

The options are as follows:

REL applies to files, as described above.

SAVE applies to files or labeled tape, as described above.

REM applies to labeled and unlabeled tape, as described above.

PTL applies to Xerox labeled tape, as described above. PTL is ignored for ANS tapes.

PTV applies to ANS tapes only and will cause an ANS M:REW.

LABEL,[*](address) specifies the address of a trailer label (in TEXTC format) to be added as a record following an :EOF sentinel. See "Tape Updating" above. An ANS trailer label is 80 bytes in length and the first four bytes must contain UTL1.

ABN,[*]address specifies the location of the user routine to be entered if an abnormal condition occurs.

ERR,[*]address specifies the location of the user routine to be entered if an error condition occurs.

The following options are meaningful only for disk files but may not be applied to star files (see glossary). Further, they are not effective for shared opens or files opened via a synonymous name.

FILE[,'name'] specifies that the file is to be renamed. The name may consist of up to 31 alphanumeric characters and must not be the same as the name of an already existing file. In order to use this option, there must be a minimum of five file buffers associated with the job. (See the POOL control command.)

PASS[,'value'] specifies a new password that is to be associated with the file. The value may be from 1 to 8 alphanumeric characters. If PASS is specified but no value given, the current password for the file is deleted.

READ[,'value']... specifies a new set of account numbers that may read but not write the file. The value 'ALL' may be used to specify that any account may read the file (e.g., READ,'ALL'). The value 'NONE' may be used to specify that no other account may read the file. If no value is specified, any existing read accounts will be removed and 'ALL' will be assumed by default.

WRITE[,'value']... specifies a new set of account numbers that may have both read and write access to the file. The values 'ALL' and 'NONE' may be used,

as with the READ option (see above); and if a conflict exists between READ and WRITE specifications, those of the WRITE option take precedence. If no value is specified, any existing write accounts will be removed and 'NONE' will be assumed by default.

EXECUTE[,'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. If READ, 'NONE' is not specified, the EXECUTE option will be ignored. If value is not specified, any existing execute accounts will be removed and 'ALL' will be assumed by default.

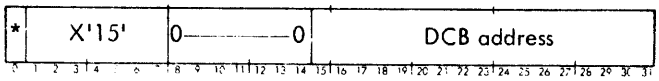
UNDER[,'name']... specifies the name(s) of the only processor(s) that may access this file if the user does not own the file. The name(s) may be from one to ten characters enclosed within single quotes. The processor(s) may be any shared processor or any load module in the :SYS account. If UNDER is specified, without a name, the current UNDER name(s) are deleted from the file attributes.

Calls generated by the M:CLOSE procedure have the form

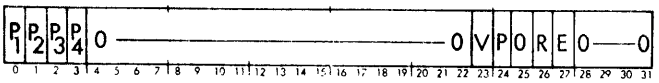
CAL1,1 fpt

where fpt points to word 0 of the FPT shown below.

word 0



word 1, options PTL and REM



where

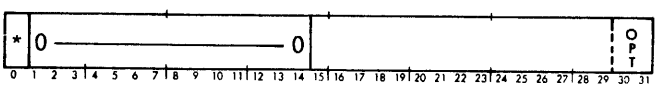
P specifies that the PTV option (see above) has (P=1) or has not (P=0) been requested.

R specifies that the REM option (see above) has (R = 1) or has not (R = 0) been requested.

E specifies that the PTL option (see above) has (E = 1) or has not (E = 0) been requested.

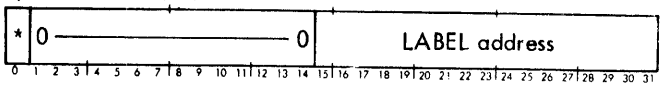
V specifies that variable length parameters are (V = 1) or are not (V = 0) present.

option {REL / SAVE} (P1)

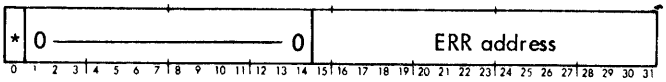


where OPT specifies REL/SAVE option (1 means release, 2 means SAVE). This option is not significant when closing a DCB opened with SHARE mode.

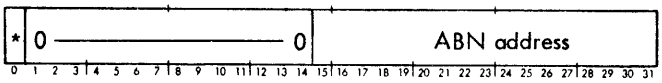
option LABEL (P2)



option ERR (P3)



option ABN (P4)



For further details on the action of the Open and Close functions, see Chapter 2, "Files and File Usage".

SET ERROR OR ABNORMAL ADDRESS

M:SETDCB The monitor SETDCB routine allows the user's program to set the error or abnormal address in a designated DCB; the call may be made while the DCB is either open or closed.

The M:SETDCB procedure call is of the form

M:SETDCB [*] dcb name[, (ERR, [*] address)];
[, (ABN, [*]address)] [,(CRPT, [*]address)]

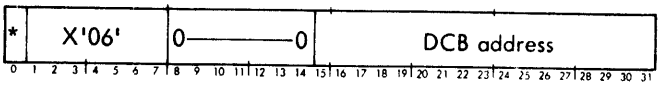
where the optional parameters are of the same form as those given for ERR and ABN in "M:DCB", earlier in this chapter. CRPT specifies the address of a word to be used as the seed for a data encryption process for keyed or consecutive files. If the effective encryption address is zero (e.g., CRPT,0 is specified), the encryption process is turned off. This option is only effective for open DCBs because the open process turns data encryption off. (This is done so that a user will not inadvertently get unwanted data encryption.)

Calls generated by the M:SETDCB procedure have the form

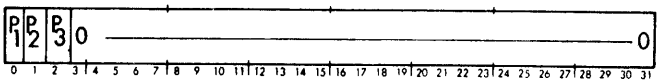
CAL1,1 fpt

where fpt points to word 0 of the FPT shown below.

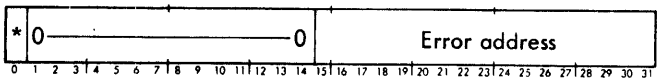
word 0



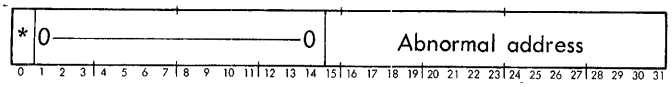
word 1



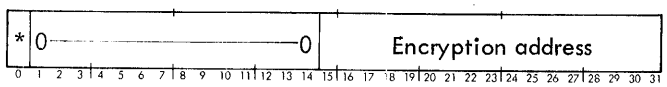
option ERR (P1)



option ABN (P2)



option CRPT (P3)



CHECK I/O COMPLETION

M:CHECK The monitor CHECK routine checks the completion-type indicator (TYC) of a specified DCB. If the completion type is other than normal and error or abnormal addresses were specified in the procedure call, an appropriate error or abnormal code is returned to the user's program via SR3 (i.e., system register 3, or general register 10). If the M:READ or M:WRITE procedure call specified an error or abnormal address, then a normal return to the user's program will be made by the CHECK routine. If I/O is currently active, it will be completed before control is returned to the user's program. If no error address or abnormal address was specified in the procedure call, no error or abnormal code is returned to the user's program. The check applies only to the most recent I/O operation done via the DCB (see the appendix titled "Monitor Error Messages"). The monitor waits for an M:CHECK or another use of the DCB before taking action on an I/O error.

The M:CHECK procedure call is of the form

M:CHECK [*]dcb name[, (option)]. . .

where dcb name specifies the name of the DCB to be checked for type of completion.

The options are as follows:

ERR, [*]address specifies the address of a user's routine that will handle error conditions for I/O operations performed via the DCB.

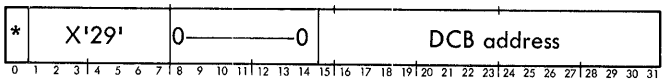
ABN, [*]address specifies the address of a user's routine that will handle abnormal conditions for I/O operations performed via the DCB.

Calls generated by the M:CHECK procedure have the form

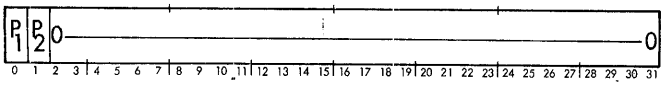
CAL1,1 fpt

where fpt points to word 0 of the FPT shown below.

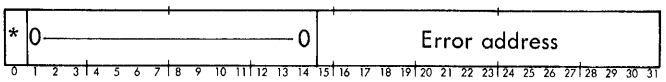
word 0



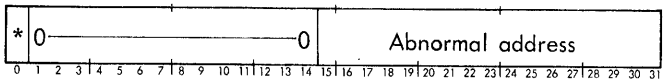
word 1



option ERR (P1)



option ABN (P2)



DECLARE A TEMPORARY FILE

M:TFILE The monitor TFILE routine causes the specified DCB's associated file to be registered with the monitor for release at the end of the job. Error and abnormal addresses may be specified, to allow the user's program to take appropriate action if the monitor is unable to register the file as temporary. The file should be closed and saved prior to the M:TFILE call. If the DCB is open, it will be closed with default options. Files declared by means of this call will be released at the end of the job, unless otherwise explicitly released. Thus, M:TFILE execution during a job step causes a file to be saved between subsequent job steps and yet be released on completion of the job. This procedure cannot be used for files on private disk packs.

The M:TFILE procedure call is of the form

M:TFILE[*]dcb name, (TFILE, [*]address);
[,(ERR, [*]address)][, (ABN, [*]address)]

where

[*]dcb name specifies the name of the DCB associated with the file to be declared temporary.

TFILE, [*]address specifies the address of the name of the file to be declared temporary. The name of the file must be in TEXTC format.

ERR, [*]address specifies the address of a user's routine that will handle error conditions for I/O operations performed via the DCB.

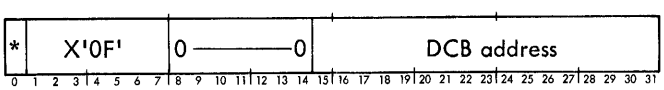
ABN, [*]address specifies the address of a user's routine that will handle abnormal conditions for I/O operations performed via the DCB.

Calls generated by the M:TFILE procedure have the form

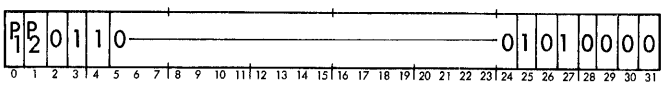
CAL1,1 fpt

where fpt points to word 0 of the FPT shown below.

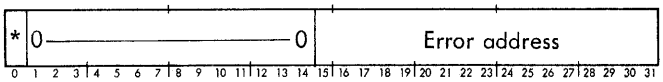
word 0



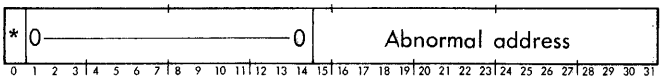
word 1



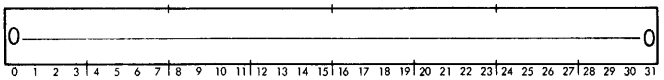
option ERR (P1)



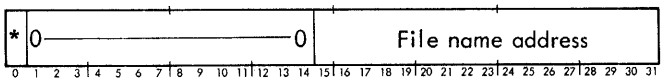
option ABN (P2)



required for TFILE



required for TFILE



DATA RECORD MANIPULATION

READ A DATA RECORD

M:READ The monitor READ routine causes a specified data record to be read into a buffer in core storage. If the record is larger than the specified buffer, part of the record is lost and this fact is communicated to the user's program (see the appendix titled "Monitor Error Messages"). All records of a length less than 18 bytes read from ANS formatted tapes are bypassed as noise records.

It is not necessary for the user's program to explicitly call the monitor OPEN routine prior to reading or writing a record, since the monitor generates such a call automatically if the DCB is not open. However, the options specified on the Read/Write call are used for the Read/Write only, and are not used as parameters for the OPEN call.

Both EBCDIC and binary decks may be used in the same job, but nonstandard binary information must be preceded by a BIN control command and must end with a BCD control command if the device is a card reader. On encountering a BIN control command, the monitor switches the device mode and automatically reads the next record in binary. Subsequent records are also read in binary until a BCD control command is encountered. The monitor then changes the device mode and automatically reads subsequent records in EBCDIC.

The mode flag (MOD), in the DCB associated with the read operation, is set to a 0 if a record is read in EBCDIC and is set to a 1 if a record is read in binary.

A BCD control command encountered when reading in the EBCDIC mode causes no change in the device mode. When the C device is read, any record having an 1 in column 1 (except for a BIN, BCD, or EOD control command) causes a code of 06 to be placed in byte 0 of SR3. The record is placed in the monitor's control command buffer and, if an attempt is made to read that record again via the same DCB, the job is aborted and the user is notified (via the LL device) of the reason for aborting the job.

Whenever an EOD control command is encountered (when reading from the C device), a code of 05 is returned to the user's program in byte 0 of SR3 if an abnormal address is specified.

The M:READ procedure call is of the form

M:READ [*]dcb name[,(option)]...

where dcb name specifies the name of the DCB to be associated with the read operation.

The options are as follows:

ABN, [*]address specifies the address of a user's routine that will handle abnormal conditions for the read operation (see the appendix titled "Monitor Error Messages"). The address specified must lie within the user's program.

BLOCK, [*]number applicable to random files only and specifies the granule number of the block at which the I/O transfer is to be made. Granule blocks within a random file are numbered 0 to n-1, where n equals the number of granules in the file.

The word pointed to by the KBUF field of the random file DCB is set equal to zero when a file is opened and is incremented by one after each granule is read or written, whether BLOCK is specified or not. When the BLOCK option is specified, it overrides the granule number in the word pointed to by the KBUF field and causes the word pointed to by KBUF to be reset equal to that value. If the BLOCK option is not specified, the granule number in the word pointed to by the KBUF field is used.

BTD, [*]value specifies the byte displacement (0-3), in the user's buffer, into which data is to be read; i.e., the byte into which the first data byte is to be read. This value is inserted into the BTD field of the DCB and will be default for subsequent read or write requests if the BTD option is not specified.

BUF,[*]address specifies the address of the user's buffer into which data is to be read. This value is inserted into the BUF field of the DCB and will be the default for subsequent read or write requests if the BUF option is not specified.

COC, [*]options specifies the options unique to a character oriented communications device, and will be ignored for any other device. If indirect addressing is specified, all option flags and values will come from the indirectly addressed word. The options are separated by commas. The options for the COC keyword are:

CONDITIONAL specifies that if no input is present (typed-ahead) when the read is issued an abnormal return (code X'24') will occur. Otherwise, the read proceeds normally.

DELETEIN specifies that all input present (typed-ahead) is to be deleted.

DELETEOUT specifies that all output present but not yet transmitted is to be deleted.

(TIMEOUT[,value]) specifies the timeout interval for the read, in 1.2 second units. If the interval expires without an activation condition being met, an abnormal condition occurs (code X'23'). Whatever input was typed will be transferred to the user's buffer. If a TIMEOUT of 0 is specified, the abnormal return will be taken after transferring any partial or complete input record.

(OACS, value) specifies an over-riding activation character set to be used in determining the end of the input message for this M:READ request only. Value must be in the range of 0 to 3. See M:CAC in the CP-V/TS Reference Manual, 900907.

REREAD specifies that the input message in the program's buffer is to be reread. The message will be transferred back to the monitor's input buffers, and echoed as if the user had just retyped the message. The user may then edit the message and again release it to the reading program. Reread handles the transfer in the following manner:

1. When a read is issued to the terminal, the user buffer is inspected. (ESC D forces the read to be reissued.)
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 reread characters.
5. The characters in the input buffer are echoed.

See ESCD in the CP-V/TS Reference Manual 900907.

ECB,[*]address specifies the address of a two-word event control block (ECB). (See the M:CHECKECB procedure description for an explanation of ECBs.) The ECB will be set to "in-use" status when the operation is started and will be posted on completion of the operation. Posted information consists of TYC in byte 0, and ARS in bytes 2 and 3 of Word 1 of the ECB. (See DCB description in Appendix A.) If an "insufficient or conflicting information" error or abnormal condition occurs (refer to Appendix B), the contents of the ECB are undefined.

ERR,[*]address specifies the address of a user's routine that will handle error conditions for the read operation (see the appendix titled "Monitor Error Messages"). The address specified must lie within the user's program.

FWD specifies that the record is to be read in the forward direction.

KEY,[*]address specifies the address containing the key (identifier) associated with the record to be read. The key may be up to 31 bytes in length and must be preceded by a byte that contains the length of the key in number of bytes. Indirect addressing can be made to a register; however, the key may not be in registers. The KEY option is valid for keyed files only.

REV specifies that the record is to be read in the reverse direction.

If neither FWD nor REV is specified, FWD is used in default.

SIZE,[*]value specifies the size, in bytes, of the user's buffer. If 0 is specified, a record is skipped. An asterisk may be used to indicate that the value is the address of a location containing the buffer size. If this option is not specified, the default value in the DCB (RSZ) will be used.

ULBL specifies a user trailer label. Bit 28 (f_1) in FPT word 1 will be set. If ULBL is specified and an end-of-volume (EOV) or end-of-file (EOF) sentinel is encountered on labeled tape, the trailer label written by a previous M:CVOL or M:CLOSE will be transferred to the user buffer in TEXTC format where the data record would have been transferred and an end-of-tape (1C) or end-of-file (06) abnormal code will be returned to the user. The user must explicitly request volume switch if ULBL is specified and an EOV sentinel is encountered. If label is larger than the area available for the label in the buffer, only the portion of the label that can be contained in the buffer will be transmitted. If ULBL is not specified, the end-of-volume (EOV) sentinel will cause automatic volume switching.

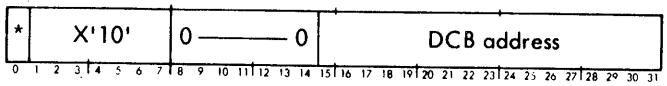
WAIT specifies that the operation is to be completed before control is returned to the user's program. WAIT is implied if either ERR or ABN is specified. If WAIT is neither specified nor implied, no wait is assumed.

Calls generated by the M:READ procedure have the form

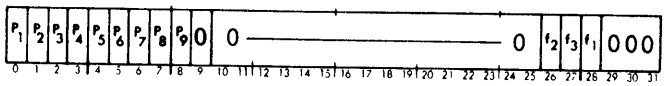
```
CAL1,1    fpt
```

where fpt points to word 0 of the FPT shown below.

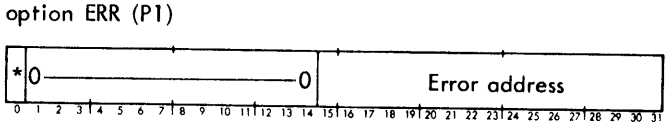
word 0



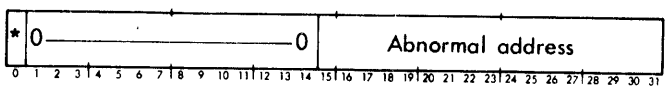
word 1



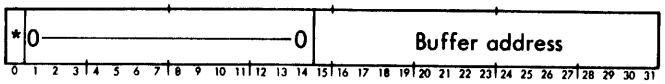
option ERR (P1)



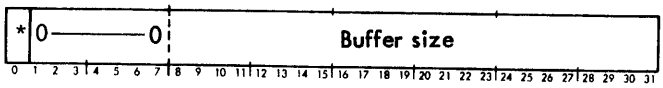
option ABN (P2)



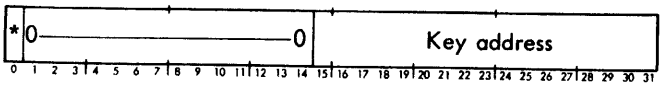
option BUF (P3)



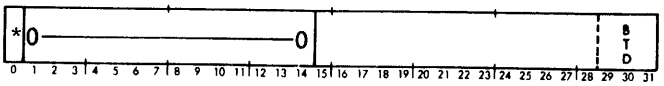
option SIZE (P4)



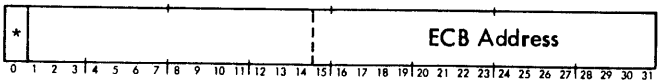
option KEY (P5)



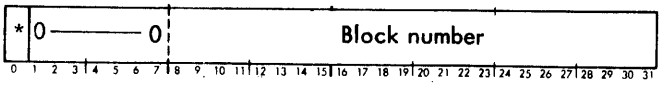
option BTD (P6)



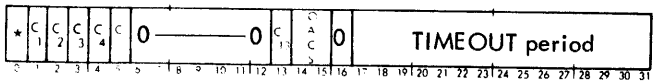
option ECB (P7)



option BLOCK (P8)



option COC (P9)



where

C_1 set indicates TIMEOUT is specified.

C_2 set indicates CONDITIONAL is specified.

C_3 set indicates DELETEIN is specified.

C_4 set indicates DELETEOUT is specified.

C_5 set indicates REREAD was specified.

C_{13} set indicates OACS was specified.

OACS specifies the OACS value.

Flag	Significance
f_2	0 means read in the forward direction. 1 means read in the reverse direction.
f_3	0 means return control to the user's program immediately. 1 means wait until I/O is complete before returning control to the user's program.
f_1	1 means ULBL option selected.

WRITE A DATA RECORD

M:WRITE: The monitor WRITE routine causes a specified data record to be written from a buffer in core storage. The format of the output depends on the type of physical device associated with the DCB.

If the DCB is assigned to a card punch, the monitor will cause !BIN and !BCD records to be punched on the card punch where appropriate.

For example, if the user's program needs to punch a binary record and the previous record was punched in EBCDIC, a !BIN record is punched automatically before the binary record is punched. Similarly, a !BCD record is punched automatically before a record is punched in EBCDIC, if the previous record was punched in binary.

On a binary record a maximum of 120 bytes are punched. On an EBCDIC record, a maximum of 160 bytes are punched, but the data is broken into two records, the first of which contains no more than 80 bytes.

For a line printer, vertical spacing is determined by the first output character in the vertical-format-control byte if the associated VFC flag in the DCB is set to a 1. A maximum of 132 characters per line may be printed on a line printer.

If the associated DCB is assigned to a typewriter (or to OC), a maximum of 256 characters per write operation is allowed. The user's program must include appropriate carriage return characters in the record to be written. If the DCB is assigned to LO, LL, or DO, a maximum of two lines per write operation is allowed.

If the DCB is assigned to PO or BO, the monitor will break the output data into two records. The first record will be 80 characters in length (EBCDIC) or 120 characters (BIN).

A request for output of an ANS tape record containing less than 18 bytes is written as an 18-byte record. The trailing bytes contain the data that follows the requested data in core. Output to an unlabeled tape is limited to 32767 bytes.

It is not necessary for the user's program to explicitly call the monitor OPEN routine prior to reading or writing a record, since the monitor generates such a call automatically if the DCB is not open. However, the options specified on the Read/Write call are used for the Read/Write only, and are not used as parameters for the OPEN call.

The M:WRITE procedure call is of the form

M:WRITE[*]dcb name[,(option)]...

where dcb name specifies the name of the DCB to be associated with the write operation.

The options are as follows:

ABN,[*]address specifies the address of a user's routine that will handle abnormal conditions for the write operation (see the appendix titled "Monitor Error Messages"). The address specified must lie within the user's program.

If an abnormal address is specified and an end-of-tape is encountered, a X'1C' abnormal code will be generated so that the user can issue a M:CVOL procedure. Also, for a Xerox or ANS labeled tape the record has not yet been written, and the user can issue another M:WRITE with no preceding M:CVOL to cause an automatic change of volume to be executed. For device tape, the record has already been written and automati M:CVOL is not performed.

BLOCK,[*]number applicable to random files only and specifies the granule number of the block at which the I/O transfer is to begin. Granule blocks within a random file are numbered 0 to n-1, where n equals the number of granules in the file.

The word pointed to by the KBUF field of the random file DCB is set equal to zero when a file is opened and is incremented by one after each granule is read or written whether BLOCK is specified or not. When the BLOCK option is specified, it overrides the granule number in the word pointed to by the KBUF field and causes the word pointed to by KBUF to be reset equal to that value. If the BLOCK option is not specified, the granule number in the word pointed to by the KBUF field is used.

BTD,[*]value specifies the byte displacement (0-3) in the user's buffer from which data is to be written. The value used is inserted into the BTD field of the DCB and becomes the default value for subsequent read/write operations for which the BTD option is not specified.

BUF,[*]address specifies the address of the user's buffer from which data is to be written. This value is inserted into the BUF field of the DCB and can be used on subsequent M:WRITE procedure calls.

COC,[*]options specifies the options unique to a character oriented communications device, and will be ignored for any other device. If indirect addressing is specified, all option flags and values will come from the indirectly addressed word. The options are separated by commas. The options for the COC keyword are:

DELETEIN specifies that all input present (typed-ahead) is to be deleted.

DELETEOUT specifies that all output present but not yet transmitted is to be deleted.

ECB,[*]address specifies the address of a two-word event control block (ECB). (See the M:CHECKECB procedure description for an explanation of ECBs.) The ECB will be set to "in-use" status when the operation is started and will be posted on completion of the operation. Posted information consists of TYC in byte 0, and ARS in bytes 2 and 3 of word 1 in the ECB. (See DCB description in Appendix A.) If an "insufficient or conflicting information" error or abnormal condition occurs (refer to Appendix B), the contents of the ECB are undefined.

ERR,[*]address specifies the address of a user's routine that will handle error conditions for the write operation (see the appendix titled "Monitor Error Messages"). The address specified must lie within the user's program.

KEY,[*]address specifies the address containing the key (identifier) associated with the record to be written. The key may be up to 31 bytes in length and must be preceded by a byte that contains the length of the key in number of bytes. Indirect addressing can be made to a register; however, the key may not be in registers. The KEY option is valid for keyed files only.

NEWKEY specifies that the KEY is a new key in the file index. That is, the key of the record to be written must not already exist; if it does exist, an abnormal return is given (see the appendix titled "Monitor Error Messages"). NEWKEY must be used for files in the output mode; if NEWKEY is not used, an X'17' abnormal condition code is returned.

ONEWKEY specifies that the NEWKEY option is to be overridden. That is, a record will be written with the specified key whether it existed previously or not.

SIZE,[*]value specifies the size, in bytes, of the user's buffer. If 0 is specified, the operation is ignored unless records are being written into a keyed file; the key is retained, but the record length is zero. If this option is not specified, the default value in the DCB (RSZ) is used.

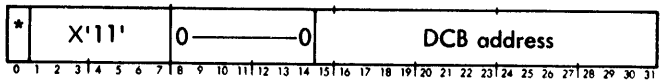
WAIT specifies that the operation is to be completed before control is returned to the user's program. WAIT is implied if either ERR or ABN is specified. If WAIT is neither specified nor implied, no wait is assumed.

Calls generated by the M:WRITE procedure have the form

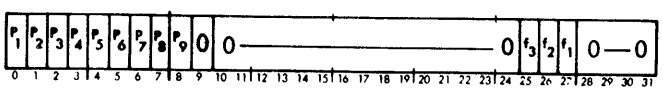
CAL1,1 fpt

where fpt points to word 0 of the FPT shown below

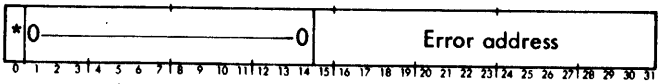
word 0



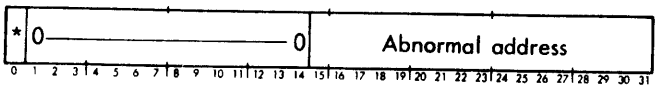
word 1



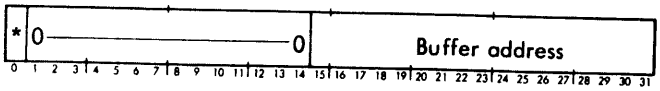
option ERR (P1)



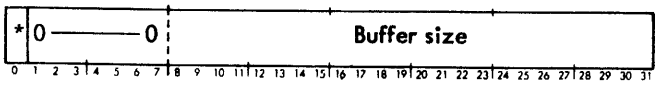
option ABN (P2)



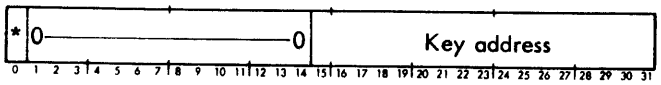
option BUF (P3)



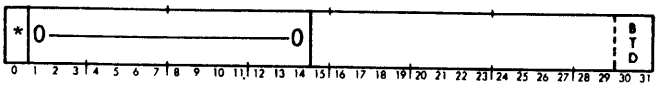
option SIZE (P4)



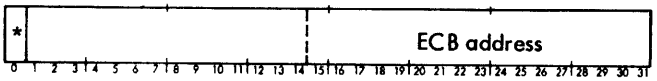
option KEY (P5)



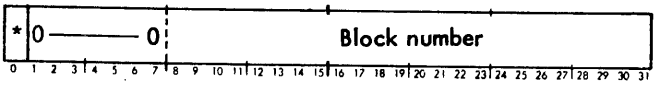
option BTD (P6)



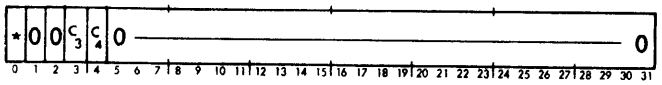
option ECB (P7)



option BLOCK (P8)



option COC (P9)



where

C_3 set indicates DELETEIN is specified

C_4 set indicates DELETEOUT is specified

Flag	Significance (when set to 1)
f_1	The WAIT option has been specified.
f_2	The NEWKEY option has been specified.
f_3	The ONEWKEY option has been specified.

COPY ALL DATA RECORDS

M:MOVE The monitor MOVE routine is designed to speed file copies. It treats a file as a logical unit rather than as a collection of records. This permits reduction in the number of CALIs required to copy a file, significantly reducing monitor overhead. The MOVE routine first reads a record through the input DCB (DCB1) and then writes it through the output DCB (DCB2). It then returns to the read cycle and continues this process until an error or abnormal condition occurs. When an error or abnormal condition occurs, control returns to the user at the user specified error/abnormal return location. The MOVE routine leaves the DCBs as they were so that the CAL may be reissued to continue where it left off. It is the user's responsibility to perform any necessary repositioning within the DCBs. The MOVE cycle always begins with a read. Therefore, if a write error occurs, both DCBs may need to be repositioned. On a read error, DCB2 is left in the state it was in at the end of the last successful write.

The M:MOVE procedure call has the form

M:MOVE [*]dcb1 name,(OUT,[*]dcb2 name),
(ERR,[*]address),(ABN,[*]address)[,(BUF,
[*]address)][,(SIZE,[*]value)]

where

dcb 1 name specifies the name of the input DCB. It must be a file or labeled tape DCB and must be opened IN or INOUT prior to execution on the MOVE CAL.

OUT, [*]dcb2 name specifies the name of the output DCB. It must be a file or labeled tape DCB and must be opened OUT or OUTIN prior to execution of the MOVE CAL.

ERR, [*]address specifies the address to which control is to be returned when an error condition occurs.

ABN, [*]address specifies the address to which control is to be returned when an abnormal condition occurs.

BUF, [*]address specifies the address of the buffer that is to be used for both input and output in the processing of the MOVE CAL. If this option is not used, the buffer specified in the input DCB will be used. The buffer is always common for input and output.

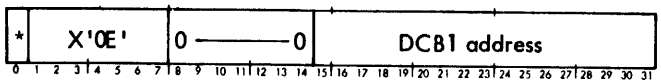
SIZE, [*]value specifies the size, in bytes, of the buffer. If this option is not used, the default record size of the input DCB will be assumed. The size of the output record is always set equal to the actual record size of the preceding read.

Calls generated by the M:MOVE procedure have the form

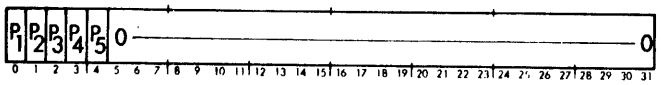
CAL1, 1 fpt

where fpt points to word 0 of the FPT shown below.

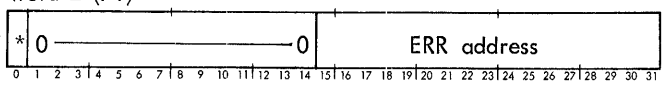
word 0



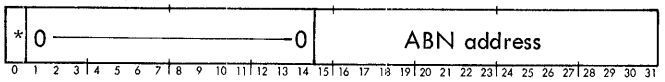
word 1



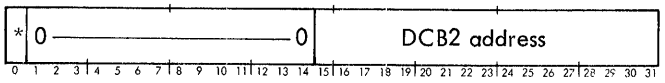
word 2 (P1)



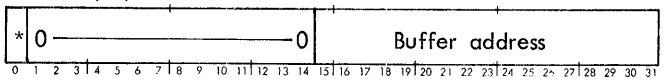
word 3 (P2)



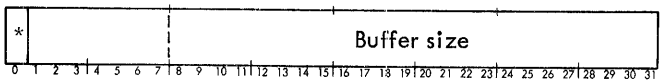
word 4 (P3)



word 5 (P4)



word 6 (P5)



DELETE A DATA RECORD

M:DELREC The monitor DELREC routine causes a data reocrd to be deleted from a keyed or consecutive file. The INOUT (update) function mode must be indicated in the DCB associated with the file.

The M:DELREC procedure call is of the form

M:DELREC [*]dcb name[,(KEY,[*]address)]

where

dcb name specifies the name of the DCB associated with the file containing the record to be deleted.

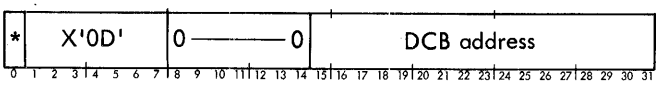
KEY,address specifies the address of the key that identifies the data record. The first byte of the key specifies the number of bytes in the key. A key may consist of up to 31 characters. If KEY is omitted, the last record read through the specified DCB is deleted.

Calls generated by the M:DELREC procedure have the form

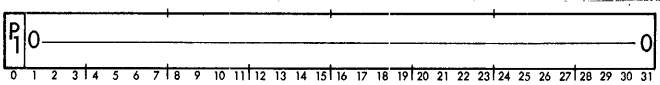
CAL1,1 fpt

where fpt points to word 0 of the FPT shown below.

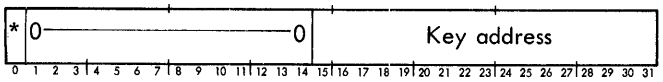
word 0



word 1



option KEY (P1)



TRUNCATE BLOCKING BUFFER

M:TRUNC The monitor TRUNC routine causes the monitor to wait for the completion of any outstanding I/O associated with a specified DCB and then to release the blocking buffer (if any is reserved for the DCB) back to the system for other use. The next read or write will be assigned a buffer automatically, as needed. This call applies only to DCBs assigned to files.

The M:TRUNC procedure call is of the form

M:TRUNC [*]dcb name

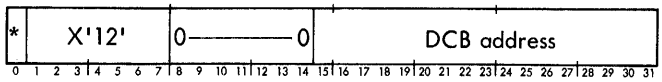
where dcb name specifies the name of the DCB associated with the blocking buffer to be released.

Calls generated by the M:TRUNC procedure have the form

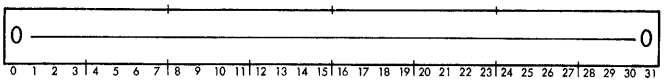
CAL1,1 fpt

where fpt points to word 0 of the FPT shown below.

word 0



word 1



FILE MANIPULATION

POSITION N RECORDS

M:PRECORD The monitor PRECORD routine causes a specified number of logical records of a keyed or consecutive file on secondary storage or magnetic tape to be skipped in the direction specified. M:PRECORD is not applicable for ANS tapes and is ignored.

The M:PRECORD procedure call is of the form

M:PRECORD [*]dcb name[,(N,[*]value)][,(option)];
[,(option)]

where

dcb name specifies the name of the DCB associated with the file (in secondary storage or on labeled or unlabeled magnetic tape).

N,[*]value specifies the number of records to be skipped. The default value is 1.

ABN,[*]address specifies the address of a user's routine to be entered if any of the following abnormal conditions occur: end-of-file, end-of-tape, beginning-of-file, beginning-of-tape. The number of records yet to be skipped is placed in the ARS field of the associated DCB.

FWD specifies that skipping is to take place in the forward direction.

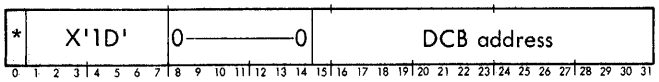
REV specifies that skipping is to take place in the reverse direction.

Calls generated by the M:PRECORD procedure have the form

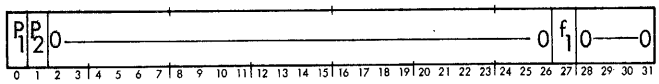
CAL1, 1 fpt

where fpt points to word 0 of the FPT shown below.

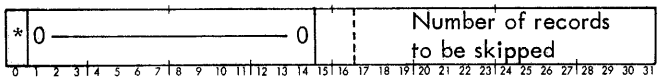
word 0



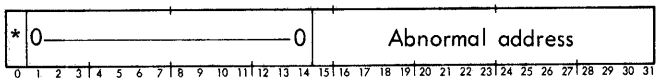
word 1



option N (P1)



option ABN (P2)



Flag	Significance
f_1	0 means skip in the forward direction. 1 means skip in the reverse direction.

POSITION FILE

M:PFIL The monitor PFIL routine causes the device associated with a specified DCB to move to the beginning or end of the current file (for keyed or consecutive files on disk storage or on labeled or unlabeled magnetic tape).

The M:PFIL procedure call is of the form

M:PFIL [*]dcb name, {(BOF) / (EOF)}

where

dcb name specifies the name of the DCB associated with the file that is to be positioned.

BOF specifies that the file is to be positioned at its beginning. For unlabeled magnetic tape, one end-of-file mark is skipped in the reverse direction and the tape is positioned immediately before that end-of-file mark. M:PFIL with the BOF option is ignored for an ANS labeled tape.

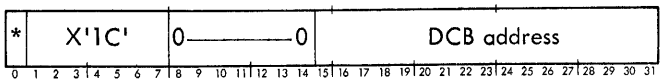
EOF specifies that the file is to be positioned at its end for Xerox labeled tapes or for ANS labeled tapes. For unlabeled magnetic tape, one end-of-file mark is skipped in the forward direction and the tape is positioned immediately after that end-of-file mark.

Calls generated by the M:PFIL procedure have the form

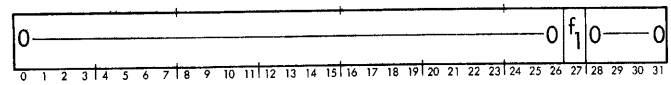
CAL1, 1 fpt

where fpt points to word 0 of the FPT shown below.

word 0



word 1



Flag	Significance
f_1	0 means position to the end-of-file. 1 means position to the beginning-of-file.

CLOSE VOLUME

M:CVOL The monitor CVOL routine causes the monitor to terminate the reading or writing of data in the magnetic tape reel currently associated with a specified DCB, and to advance to the next reel of the data set.

Unlabeled tapes are positioned at the beginning of the next input reel; output files are positioned at the beginning of a new scratch tape (or output reel, if any). The DCB is closed on the last reel.

For output files on labeled tape, end of volume and end of reel are written, and label and account sentinels are written on the next reel in the set.

For input tapes, the tape is advanced to the next reel of the data set and the file currently open is located on the next reel.

Volumes closed on labeled tapes cause the tape to be rewound and a DISMOUNT message to be output. For output, update, and scratch files, a SAVE message is also output. Volumes closed on unlabeled tape also cause the tape to be rewound. However, for unlabeled tape, the user's program must output any SAVE and DISMOUNT messages.

The M:CVOL procedure call is of the form

M:CVOL [*]dcb name[,(LABEL,[*]address)]

where

dcb name specifies the name of the DCB associated with the volume to be closed.

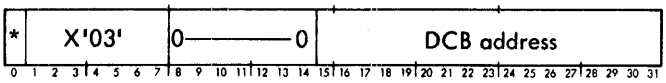
LABEL,[*]address specifies the address of a label to be added as a record following the :EOF or :EOV sentinel. The label must be in TEXTC format. An ANS label is 80 bytes long with 'UTL1' as the first four bytes.

Calls generated by the M:CVOL procedure have the form

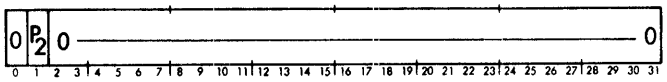
CAL1,1 fpt

where fpt points to word 0 of the FPT shown below.

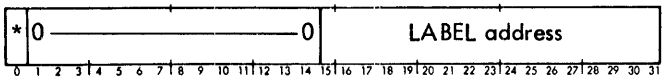
word 0



word 1



option LABEL (P2)



REWIND

M:REW The monitor REW routine causes the monitor to perform a rewind function under the following conditions:

- If the associated DCB is assigned to unlabeled tape, the DCB is opened (if it was closed) and the tape reel is rewound.
- If the associated DCB is open and is assigned to a keyed or consecutive file on a Xerox labeled tape, RAD or disk pack, the file is positioned to its beginning-of-file. However, if the associated DCB is closed, no action is taken.
- If the associated DCB is assigned to an ANS labeled tape, the DCB is closed with the PTV option; i.e., the tape is positioned as if an AVR sequence had been performed.

The M:REW procedure call is of the form

M:REW [*]dcb name

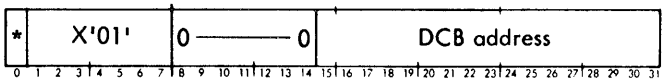
where dcb name specifies the name of the DCB associated with the file that is to be rewound.

Calls generated by the M:REW procedure have the form

CAL1,1 fpt

where fpt points to word 0 of the FPT shown below.

word 0



WRITE END-OF-FILE

M:WEOF The monitor WEOF routine causes an end-of-file to be written on the unlabeled tape associated with a specified DCB, an IEOD to be output to card punch, and a top-of-form to be output to the line printer.

The M:WEOF procedure call is of the form

M:WEOF [*]dcb name

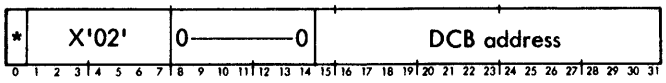
where dcb name specifies the name of the DCB associated with the tape on which the end-of-file is to be written.

Calls generated by the M:WEOF procedure have the form

CAL1,1 fpt

where fpt points to word 0 of the FPT shown below.

word 0



INSERT OR DELETE SYMBIONT FILE

M:JOB The Job Entry (JOBENT) procedure call is restricted to use by system processors. It permits a file (a block at a time) to be inserted into or deleted from an existing symbiont file residing on secondary storage. JOBENT operates under monitor control to perform the following functions:

- Insert a print block into an output symbiont.
- Insert a job block into an input symbiont.
- Check status of the particular block (job) that has been inserted.
- Delete a job waiting in the input symbiont.

The M:JOB procedure call is of the form

M:JOB [*]dcb address[,(option)]. . .

where dcb address specifies the address of the DCB that will be used to write the block. The DCB may be any system DCB (for example, M:EI or M:BO) or it may be a user DCB.

Options are as follows:

BUF, [*]address specifies the core storage address of the block to be inserted or deleted.

ABN, [*]address specifies an address in the user's program to be entered in the event this block cannot be inserted or deleted. If the block cannot be entered or deleted, the system returns an abnormal code 3F/39, 3A, 3B, 3C, 3D, 3E, or 3F to the user (see the appendix titled "Monitor Error Messages").

IN or OUT specifies the function mode to be performed. IN = insert into input symbiont (0, 1), OUT = insert into output symbiont (2).

DEL [*]address specifies either the system ID address or a pointer to the ID itself of the job to be deleted.

ACCT, [*]address specifies the address of a two-word area that contains the account number under which the job was submitted. This option is only meaningful in association with a DEL request. If not specified, the current user's account number is assumed. The user must have C0 privilege or greater to delete a job submitted under another account number.

LAST, [*]address specifies this is the last block of the file to be entered into the symbiont and the address is either the priority itself or a pointer to the priority to be used for the current job to be submitted.

If LAST is specified, zeros are placed in word 0 of the block, the previous block's disk address is placed in the last word of the block, and the block is written to the symbiont on secondary storage. Following insertion of this block, SR1 will contain the system identification (right-adjusted) and is available to the user when control is returned following M:JOB.

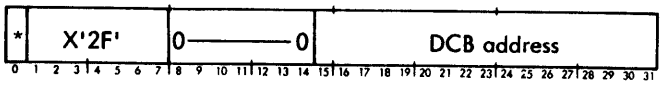
If LAST is not specified, the system determines the disk address of the next block to follow and places this address in word 0 of the present block and the previous block's disk address in the last word of the present block. The block is then written to the symbiont on the secondary storage.

Calls generated by the M:JOB procedure have the form

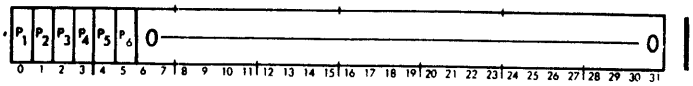
CAL1,1 fpt

where fpt points to word 0 of the FPT shown below.

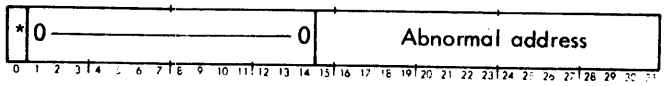
word 0



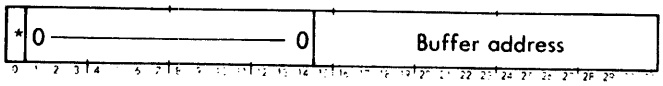
word 1



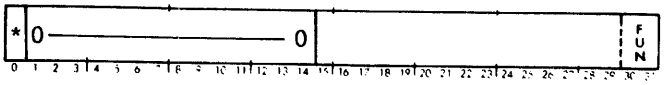
option ABN (P1)



option BUF (P2)

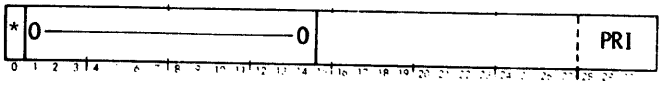


option {IN / OUT} (P3)



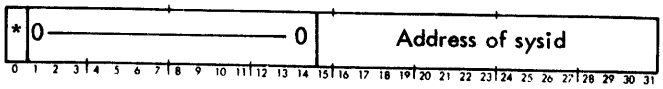
where FUN specifies function (0 or 1 means IN, 2 means OUT).

option LAST (P4)

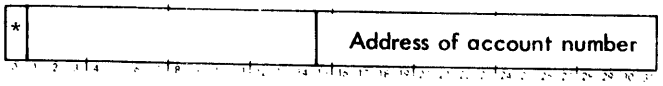


where PRI specifies priority only for output. The only valid values are 0-15.

option DEL (P5)



option ACCT (P6)



The organization of records and the format of the input and output file symbiont buffers is described in the appendix titled "Cooperatives and Symbionts".

If none of the first five parameters (P_1 to P_5) are specified in M: JOB, the M: JOB call is interpreted as a status check. The status of the file whose identification is specified in SR 1 is returned as a code in SR1, with

0 = completed.

1 = running.

2 = waiting for execution.

3 = never existed.

4 = waiting to output or job is waiting in the output queue.

If code 2 is returned, SR3 contains the number of jobs ahead of the checked job.

SPECIAL DEVICE PROCEDURES

M:DEVICE The monitor DEVICE routine is capable of performing a variety of functions. The function performed is determined by the keyword specified in the procedure call. In all cases where the M:DEVICE call is not compatible with the device associated with the specified DCB, the call is ignored and no error or abnormal return is given. (The DCB must be assigned to a DEVICE file.) For symbiont devices, all actions affecting any device (e.g., skip to top of form) are delayed until the I/O actually takes place.

SET LISTING TABS

This call allows the user's program to set listing tabs for designated columns of data output listed via a specified DCB.

The procedure call is of the form

M:DEVICE [*]dcb name, (TAB, value[,value]...)

where

dcb name specifies the name of the DCB associated with the device on which data is to be listed.

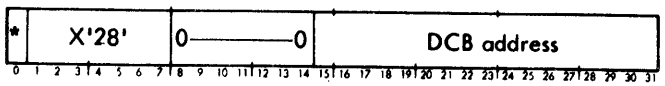
TAB, value[,value]... specifies the values (column numbers) of desired tab positions. As many as 16 tab values may be specified. The tab values are stored in the TAB fields of the specified DCB in the sequence in which they are specified in the procedure call. A value of 0 specified at TAB_i causes TAB_i through TAB_{16} to be set to 0, indicating null tabs.

Calls generated by the M:DEVICE (TAB) procedure have the form

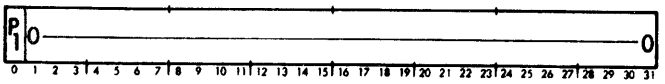
CAL1, 1 fpt

where fpt points to word 0 of the FPT shown below.

word 0

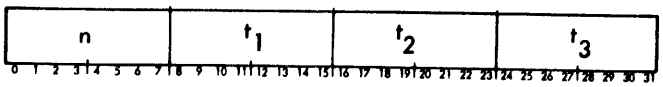


word 1

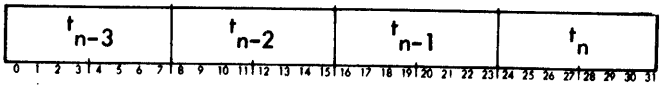


For this FPT, P_1 must be set to 1.

option TAB (P1)



last word



When the user's program requires tab spacing in the output buffer, this is indicated in the character string by an EBCDIC code of 05. The monitor responds to such a code by inserting the subsequent character (in the character string) at the column indicated by TAB_i (where TAB_{i-1} was the most recent tab setting used in formatting the current line).

Note that unless the value of $TAB_i > TAB_{i-1}$, data may be lost by being overlapped in the output buffer.

Example:

The procedure call

M:DEVICE M:LO, (TAB, 5, 20, 35)

would result in the following entries in word 15 of the associated DCB:

TAB1 = 5
TAB2 = 20
TAB3 = 35
TAB4 = unchanged

With these tab settings, the EBCDIC (hexadecimal) string

05C3D6D3E4D4D540F105C3D6D3E4D4D540F2

would result in the following typeout:

(col. 5) (col. 20)
↓ ↓
COLUMN 1 COLUMN 2

SKIP TO TOP OF FORM

This call allows the user's program to cause the printer or typewriter associated with a specified DCB to skip to the top of a new physical page. If the printer is already positioned at the top-of-form, no action takes place.

The procedure call is of the form

M:DEVICE [*]dcb name, (PAGE)

where

dcb name specifies the name of the DCB associated with the device that is to be positioned.

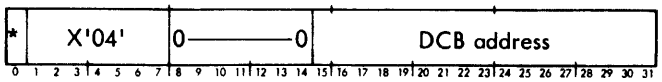
PAGE specifies that the device associated with the specified DCB is to skip to the top of the next page.

Calls generated by the M:DEVICE (PAGE) procedure have the form

CAL1,1 fpt

where fpt points to word 0 of the FPT shown below.

word 0



SET NUMBER OF PRINTABLE LINES

This call allows the user's program to set the number of printable lines per page, for the listing device associated with a specified DCB.

The procedure call is of the form

M:DEVICE [*]dcb name, (LINES, value)

where

dcb name specifies the name of the DCB associated with the device for which the number of printable lines is to be set.

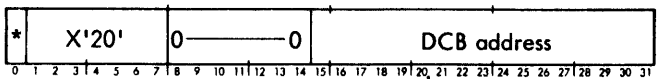
LINES, value specifies the number of printable lines per page. A maximum of 32,767 lines per page may be specified. The value includes any header and after-header spacing. A blank header line is used if no header is specified.

Calls generated by the M:DEVICE (LINES) procedure have the form

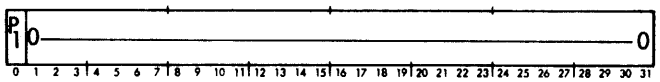
CAL1,1 fpt

where fpt points to word 0 of the FPT shown below.

word 0

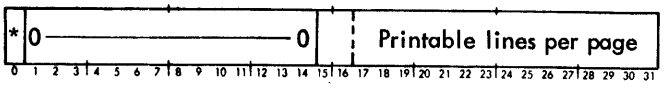


word 1



P_1 must be equal to 1.

option LINES (P1)



SET LINE SPACING

This call allows the user's program to set the number of spaces between lines and the number of spaces between the page header and the first line printed. It is valid only for listing devices. Between-lines spacing takes effect only if the VFC flag in the DCB is 0.

The procedure call is of the form

M:DEVICE

[*] dcb name, (SPACE, [*] value 1 [, [*] value 2])

where

dcb name specifies the name of the DCB associated with the device for which the line spacing is to be set.

value 1 specifies the number of lines to be spaced after printing a line. (A value of either 0 or 1 results in single spacing.)

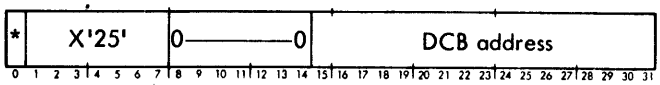
value 2 specifies the line number (with the header line number being 0) of the first line.

Calls generated by the M:DEVICE (SPACE) procedure have the form

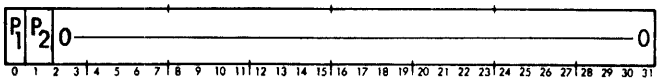
CAL1,1 fpt

where fpt points to word 0 of the FPT shown below.

word 0

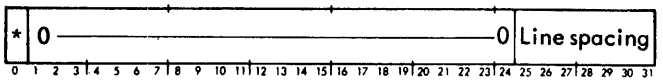


word 1

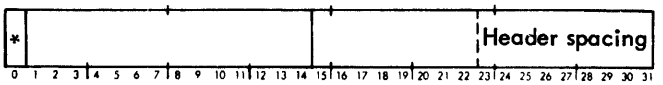


P_1 must be equal to 1.

option SPACE (P1)



option SPACE (P2)



SPECIFY DIRECT FORMATTING

This call allows the user's program to specify whether or not special record formatting is to be done by the monitor.

The procedure call is of the form

M:DEVICE [*]dcb name, {(DRC) / (NODRC)}

where

dcb name specifies the name of the DCB associated with the device for which the special formatting is or is not to be done.

DRC specifies that no special record formatting is to be done for the device associated with the designated DCB (inhibit monitor formatting).

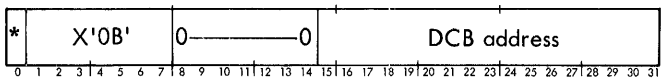
NODRC specifies that the normal mode of monitor formatting is to be reinstated for the device associated with the designated DCB.

Calls generated by the M:DEVICE (DRC/NODRC) procedure have the form

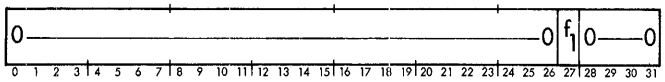
CAL1,1 fpt

where fpt points to word 0 of the FPT shown below.

word 0



word 1



Flag	Significance
f_1	0 means monitor formatting is not to be inhibited. 1 means monitor formatting (for card devices) is to be inhibited.

SPECIFY VERTICAL FORMAT CONTROL

This call allows the user's program to specify whether or not the monitor is to interpret the first character of each output image as a vertical format control character.

The procedure call is of the form

M:DEVICE [*]dcb name, {(VFC) / (NOVFC)}

where

dcb name specifies the name of the DCB associated with the listing device that is (or is not) to operate under vertical format control.

VFC specifies that the user has inserted a control character in his print image.

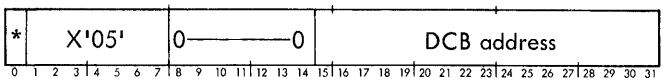
NOVFC specifies that the user has not inserted a control character in his print image.

Calls generated by the M:DEVICE (VFC/NOVFC) procedure have the form

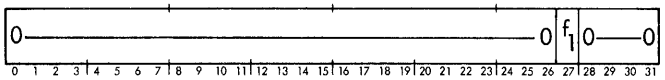
CAL1,1 fpt

where fpt points to word 0 of the FPT shown below.

word 0



word 1



Flag	Significance
f_1	0 means no vertical format control is to be performed. 1 means vertical format control is to be performed.

SPECIFY PAGE COUNT

This call allows the user's program to request that the monitor count output pages, and also to specify to which column this count is to be listed on the output device. The page count will appear at the top of the form, if no header has been specified (see "Specify Output Header"); otherwise, the page count will appear on the same line as the header. The count will be expressed in decimal form, from 1 to 9999.

The procedure call is of the form

M:DEVICE [*]dcb name, (COUNT, tab)

where

dcb name specifies the name of the DCB associated with the listing device on which the page count is to be listed.

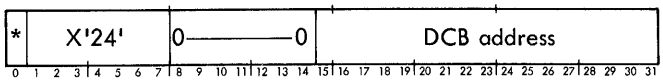
COUNT, tab specifies the column in which the most significant digit of the page count is to be listed. The value of "tab" must be appropriate for the physical device associated with the DCB.

Calls generated by the M:DEVICE (COUNT) procedure have the form

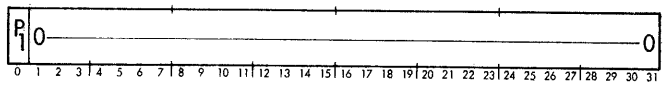
CAL1,1 fpt

where fpt points to word 0 of the FPT shown below.

word 0

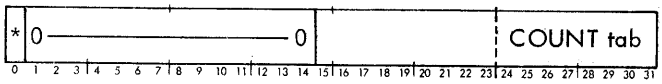


word 1



P_1 must be equal to 1.

option COUNT (P1)



CHANGE OUTPUT FORM

This call allows the user's program to request a change in the form used on the output device (e.g., card punch, typewriter, line printer, etc.). The monitor informs the operator of the change that is to be made. When the operator has changed the form, he informs the monitor by an appropriate key-in.

The procedure call is of the form

M:DEVICE [*]dcb name, (option)

where

dcb name specifies the name of the DCB associated with the device for which the change of form is to be requested.

FORM, [*]address specifies the address of the message (that is to be output to the operator) concerning a change of cards or paper. The first byte of the message must specify the number of bytes in the message.

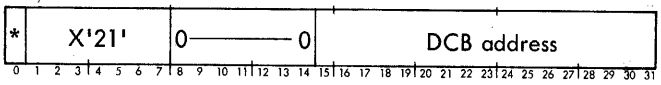
FNAME,'name' specifies the one- to four-character name of an installation-determined form or card stock. If 'NONE' is specified for 'name', the default form or card stock of the installation is required.

Calls generated by the M:DEVICE (FORM/FNAME) procedure have the form

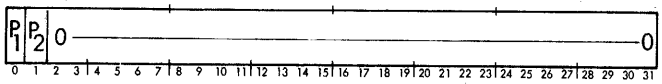
CAL1, 1 fpt

where fpt points to word 0 of the FPT shown below.

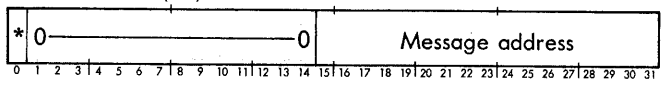
word 0



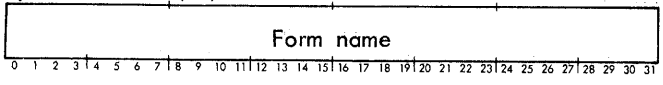
word 1



option FORM (P1)



option FNAME (P2)



CHANGE DEVICE MODE OR RECORD SIZE

This call allows the user's program to change the mode of the device associated with a specified DCB, or to change the logical record size entry (RSZ) in the specified DCB.

The procedure call is of the form

M:DEVICE [*]dcb name, (option)

where dcb name specifies the name of the DCB associated with the device for which the change in mode or record size is to be made.

The options are

BCD specifies the EBCDIC mode.

BIN specifies the binary mode.

FBCD specifies FORTRAN BCD conversion.

PACK specifies the packed binary mode (7-track tape) is to be used. PACK is not valid unless BIN is specified.

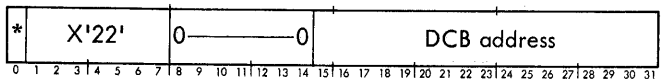
UNPACK specifies the unpacked binary mode (7-track tape) is to be used. UNPACK is not valid unless BIN is specified.

SIZE, value specifies the default record size, in bytes.

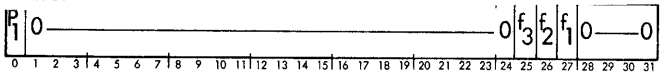
Calls generated by this procedure have the form

CAL1, 1 fpt

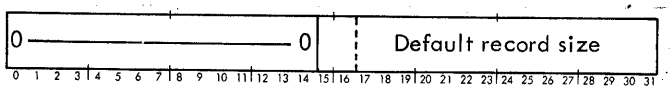
where fpt points to word 0 of the FPT shown below.



word 0



option SIZE (P1)



Flag	Significance
f_1	0 means BCD mode. 1 means binary mode.
f_2	0 means no FBCD. 1 means FBCD.
f_3	0 means packed. 1 means unpacked.

SPECIFY BEGINNING COLUMN

This call allows the user's program to specify that all data output by the card punch (EBCDIC only), typewriter, or other sting device associated with a designated system DCB is to begin in a specified column.

The procedure call is of the form

M:DEVICE [*]dcb name, (DATA, tab)

where

dcb name specifies the name of the DCB associated with the output device for which the beginning column is to be specified.

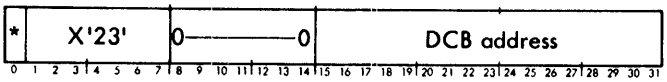
DATA, tab specifies the column in which the first character of the data output is to appear.

Calls generated by the M:DEVICE (DATA) procedure have the form

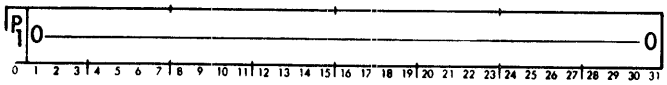
CAL1, 1 fpt

where fpt points to word 0 of the FPT shown below.

word 0

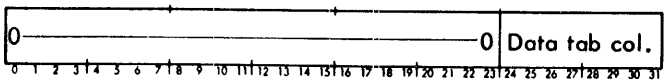


word 1



P_1 must be equal to 1.

option DATA (P1)



SPECIFY OUTPUT HEADER

This call allows the user's program to specify an output header (heading) that is to appear at the top of each form.

The procedure call is of the form

M:DEVICE [*]dcb name, (HEADER, tab, [*]address)

where

dcb name specifies the name of the DCB associated with the device on which the header is to appear.

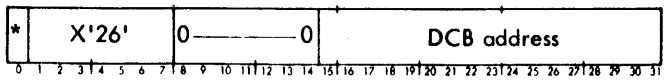
HEADER, tab, [*]address specifies the column number (tab) at which the header is to begin, and the address of the header. The first byte of the header must specify the number of bytes it contains.

Calls generated by the M:DEVICE (HEADER) procedure ave the form

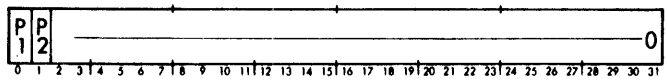
CAL1, 1 fpt

where fpt points to word 0 of the FPT shown below.

word 0

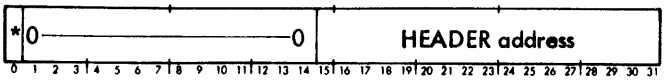


word 1

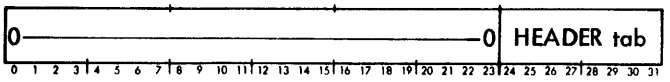


P_1 and P_2 must be equal to 1.

option HEADER address (P1)



option HEADER column (P2)



SPECIFY CARD PUNCH SEQUENCING

This call allows the user's program to specify that sequence numbers are to be punched on cards output by the card punch associated with a designated DCB.

The procedure call is of the form

M:DEVICE [*]dcb name, (SEQ[,'id'])

where

dcb name specifies the name of the DCB associated with the card punch that is to output cards with sequence numbers.

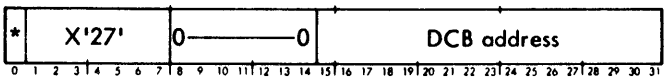
SEQ[,'id'] specifies that sequence numbers are to be punched in columns 77-80 of each card. If a user-defined id is specified, it will be punched in columns 73-76 of each card.

Calls generated by the M:DEVICE (SEQ) procedure have the form

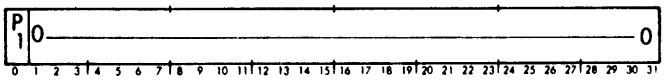
CAL1, 1 fpt

where fpt points to word 0 of the FPT shown below.

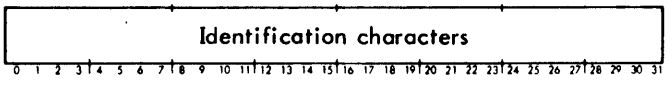
word 0



word 1



option SEQ (P1)



NUMBER OF LINES REMAINING

This call allows the user's program to determine the number of printable lines remaining on a page.

The procedure call is of the form

M:DEVICE [*]dcb name, (NLINES)

where

dcb name specifies the name of the DCB associated with the device for which the number of lines remaining on a page is to be obtained.

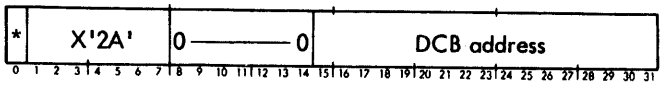
NLINES keyword designating what the procedure call is requesting.

Calls generated have the form

CAL1, 1 fpt

where fpt points to the FPT shown below.

word 0



Upon return to the caller, SR1 contains 0 if not applicable. SR1 contains the number of lines remaining on the current page only if the user indicated top of page and set the value of LVA with M:DEVICE [*] dcbname, (LINES, value).

CHECK CORRESPONDENCE OF DCB ASSIGNMENTS

This call allows the user's program a means of determining if two DCBs have been assigned to the same physical device. Both DCBs must have been opened.

M:DEVICE dcb_1name,(CORRES,dcb_2name)

where

dcb_1name specifies the name of a DCB which is to be checked for assignment correspondence with dcb_2name.

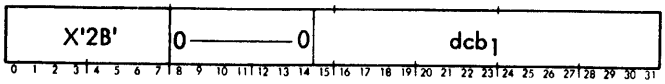
CORRES,dcb_2name specifies the name of a DCB which is to be checked for assignment correspondence with dcb_1name.

Calls generated have the form

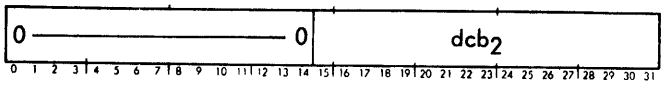
CAL1, 1 fpt

where fpt points to the FPT shown below.

word 0



word 1



If the assignments of the two DCBs correspond, a 1 is returned in SR1; otherwise, a 0 is returned.

6. PROGRAM LOAD AND EXECUTION

INTRODUCTION

There are three processors that can be used in the batch mode 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 has most of the capabilities of the overlay loader, Load, and also provides the same control over internal and global symbol table construction which is available in the linking loader, Link. 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 Load which actually performs the loading process.

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 the LYNX loader be used.

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.

The final sections of this chapter describe task control blocks, data control blocks, memory protection, virtual memory, and job accounting.

LOAD PROCESSOR

The purpose of the Load processor is to translate and unite its input (ROMs and libraries) into such a form that its output (a load module) may be executed under the CP-V operating system. The TREE, PTREE, and INCL control commands are used to provide overlay information to the loader. Program execution is initiated by the RUN command, which is described below after the description of the Load processor. The accounting summary generated at the end of each job is described at the end of this chapter.

The loader performs the following functions:

- Process relocatable object modules (ROMs) producing continuous sections of data, procedure, and DCBs (or static data) ensuring a page boundary for the three protection types (00, 01, and 10, respectively).
- Satisfy REFs among the ROMs.
- Access libraries to satisfy PREFs.
- Build Data Control Blocks (DCBs).
- Build a DCB name table for monitor use.
- Build Task Control Blocks (TCBs).
- Produce a load map printout for a newly built or previously formed load module.

The loader produces a load module, which is a keyed file having the format shown in Table 17.

Table 17. Standard Load Module Format

Key		Record Contents
C'HEAD'		Basic load module information
C'TREE'		Tree Table
LMN name concatenated with	X'00'	REF/DEF stack
	X'01'	Expression stack
	X'02'	00 Relocation Dictionary
	X'03'	00 Control Sections
	X'04'	01 Relocation Dictionary
	X'05'	01 Control Sections
	X'06'	10 Relocation Dictionary
	X'07'	10 Control Sections

A loader control command normally follows a processor command (and is read after all specified inputs have been received and processed) so that the processor's output will be translated into an executable load module.

The object modules or load modules may be input to the loader from one or more BI files, GO files, element files, or libraries.

Note that if the first few characters of any continuation to a LOAD, OVERLAY, OLAY, or TREE command match any of the input control commands or JOB, those commands must have at least one blank between ! and the first alpha character.

CONTROL COMMANDS

LOAD,OVERLAY,OLAY The loader which is invoked by a LOAD, OVERLAY or OLAY control command processes relocatable object modules, previously formed load modules, or a combination of both. The resulting load module is a keyed file which is placed in the user's account. Execution of the load module is triggered by the RUN control command which brings the load module into core storage and transfers control to it. (A load module may also be called internally by an executing program via the M:LINK procedure.)

The special characteristics of the Overlay Loader are as follows:

1. Overlaid programs. The overlaid program is one that has only one segment resident in core permanently. The other segments are called for by the M:SEGLD procedure call and brought in as needed. They may reside (at different times) in the same core area, thus reducing the amount of core storage required to house the entire program.

   Since a program may consist of three areas (protection types), each beginning on a page boundary, the Overlay Loader has the ability to create the three structures, each beginning on a page boundary.

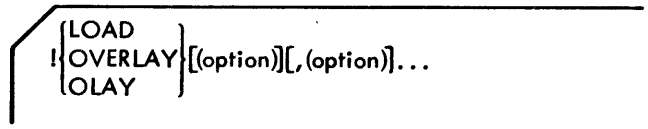
2. Reference Loading. If the user does not choose to maintain responsibility for calling in the segments of an overlaid program (by explicitly using the M:SEGLD), he may direct the loader to insert the M:SEGLD code into his program by specifying REF or BREF on the LOAD command. This code is built wherever there is a branch type instruction to a DEF in a higher segment (BREF mode) or wherever there is an expression involving a DEF from a higher segment (REF mode).

3. Load Module Libraries. It is desirable to maintain libraries of frequently used routines that are themselves already in load module form, since subsequent inclusion of a library module is faster than processing the original ROM language. Library load modules are of a less general nature since they must be of one protection type, relocatable, and not overlaid.

4. Relocatable Load Modules. The loader creates a relocation dictionary that allows subsequent placement of the load module into a core area other than the one at which it was originally biased. This is required for library load modules.

There is no functional difference between the LOAD, OLAY, and OVERLAY commands. The load parameters must be specified in either a LOAD, OVERLAY, or OLAY command and the overlay structure must then be specified in a TREE control command.

The forms of the loader control commands are



where the options are as follows:

Options that determine input to the loader

BI specifies that the BI input device 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. If neither BI, EF, nor GO are specified as input sources, BI is assumed by default. Normally, the BI and C operational labels are both assigned to the same device. If a control command is read, the monitor generates an end-of-file code and terminates the binary input.

GO specifies that data from the user's temporary GO file is to be included in the root of the load module (see TREE). If GO has been assigned to a labeled file, the GO option cannot be used to load the program (see the "EF" option below).

EF,(name[,account[,password]])[,...] specifies that the named modules (either object or load) from an element file of the designated account are to be included in the load module. If no account number is specified, that of the current job is assumed. If a password is associated with a named module, it and the account number must be included in this option. More than one module may be specified in an EF option. An invalid password will cause the job to be aborted. The named elements will be loaded in the order in which they are specified (if no TREE control command is used). The element file name must not be greater than 10 characters.

UNSAT,(account[,password])[,(:Pn)][,(account[,password])][,(:Pn)][,...] specifies that the library of the designated account is to be searched for external definitions required for the load module (i.e., corresponding to primary external references). More than one account may be specified in an UNSAT option. The library password (if any) for each account must be included, although listing of the password is suppressed. The total number of accounts must not exceed eight. :Pn may be used to specify a public library in the :SYS account. (See note 18 under overlay Loader Restrictions.)

NOSYSLIB specifies that the system library is not to be searched. If NOSYSLIB is omitted, the system library will be searched to satisfy any external references that are unsatisfied after loading has been accomplished from all other specified sources.

LMN,name[,password] specifies the name of a previously formed load module file which is to be mapped only. If this option is used to denote input, the only acceptable options are MAP, LIB, UDEF, LDEF, RDEF, VALUE, and NAME (see below).

Options affecting future access to the load module file

LMN,name[,password] specifies the name that is to be given to the load module. The name may consist of from 1 to 10 alphanumeric characters (except for shared processor names which may only have up to eight alphanumeric characters). If no name is specified for a load module, it is considered temporary, even if PERM is specified. A password to be associated with the load module may be specified.

If PERM and LIB are specified, the password specified for the first library load module entered in the library becomes the password of the library. Any subsequent load modules to be added to the account's library must specify the same password. The library password may be changed only by deleting files :LIB and :DIC and then reentering load modules with a new password.

PERM specifies that the load module is to be retained on the disk as a permanent element file. If PERM is omitted, and LIB (see below) is not specified, the load module will be a temporary file. If a previously formed load module of the same name (see LMN, above) exists, it will be replaced by the newly formed one. If PERM and LIB are specified, any external definitions or external references in the load module will be added to the account's library table of external definitions and the load module will be inserted into the account's element file library (:LIB). If LIB is specified, the load module must comprise a single control section of uniform memory access type. (See Note 18 under Overlay Loader Restrictions.)

LIB specifies that the input is a library load module. If LIB is specified in conjunction with the MAP option and PERM is omitted, the loader will print the DEF and DSECT names only. (See note 18 under Overlay Loader Restrictions.)

READ[,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., READ, ALL). The value NONE may be used to specify that no other account may read the file. If no value is specified, or is READ and EXECUTE 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 READ or WRITE specification must not exceed eight.

WRITE[,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 with the READ option (see above); and, if a conflict exists between READ and WRITE specifications, those of the WRITE option take precedence. NONE is assumed by default.

EXPIRE,{mm,dd,yy / ddd / NEVER} specifies either an explicit expiration date (mm, dd, yy), the number of days to retain the file (ddd), or that the file is never to expire (NEVER). If not specified, the default value as established in the authorization record for the user will determine the expiration date. Files will be automatically purged from the public file system if they have expired whenever secondary storage space passes below a SYSGEN established threshold.

The value specified may not exceed the maximum expiration period authorized for the user. If the maximum expiration period is exceeded or unspecified, the default expiration period authorized for that user will be used.

EXECUTE,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, READ, NONE is implied in the absence of any READ specification.

UNDER,name 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.

Options affecting the location of the program at execution time

BIAS, value specifies (in hexadecimal) the load bias, in word locations. 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'.

CORELIB 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.

CSEC1 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 (including library input).

M10 specifies that each control or dummy section is to be loaded at the next greater multiple of 10_{16}.

M100 same as M10, above, except that loading starts at the next greater multiple of 100_{16}.

Options determining how overlay segments will be brought into core at execution time

SEG 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.

REF[,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 "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.

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 BREF mode will assume that segment to be in core. BREF 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 "REF", 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).

If neither REF, BREF, nor SEG is specified, SEG is assumed. Only one may be specified.

Options concerning the loader-built Task Control Block

TSS,size specifies the maximum size, in hexadecimal number of words, of the Temporary Storage Stack (TSS) for the current job. 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.

ERTABLE,size specifies the size, in hexadecimal number of words, of the library error table (see the Mathematical Routines/Technical Manual, 90 09 06). The default is ten words.

ERSTACK,size specifies the size, in hexadecimal number of words, of the library error stack. The default is ten words.

NOTCB specifies that no Task Control Block (TCB) is to be created by the loader. This option should not be used for FORTRAN jobs, since FORTRAN requires a TCB.

Options concerning symbol tables

NI specifies that internal symbol tables are not to be built. (They are normally built by default.)

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).

Additional options

ABS specifies that a relocation dictionary is not to be formed for the load module.

REL 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 ABS nor REL is specified, ABS is assumed. Only one may be specified.

MAP[,NAME][,VALUE] specifies that a complete listing of external references and definitions for the load module is to be output on the LL device. VALUE specifies that the DEFs (and control sections) within each segment are to be sorted by value. NAME specifies that the DEFs within each segment are to be sorted by name, and that the control sections are to be sorted separately by value.

MAPONLY[,NAME][,VALUE] specifies that an existing load module is to be mapped. The output is the same as that described for MAP above.

LDEF is used in conjunction with the MAP 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 MAP and LDEF options 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.

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 through F. The default value is 4.

PAGE 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.

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 without 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.

Overlay Loader Restrictions

1. A load module acceptable for combination with ROMs to form a new load module must be of one protection type, relocatable, and not overlaid. DSECTs in such a load module are allowed only if the entire load module consists of one DSECT. Note that library load modules are subject to these restrictions.

2. If a DEF in a library load module is >11 characters, the corresponding entry in the :DIC file is forced to 11 characters. (The DEF entry in the library load module itself is not changed.)

3. The REL option will be overridden and the load module will be set ABS under any of the following conditions:

   a. REF or BREF has been specified on the LOAD card.

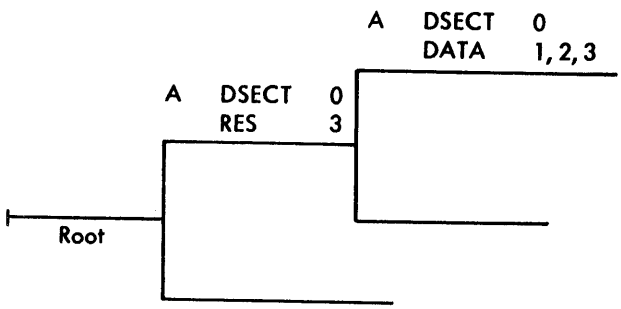
   b. The program contains a relocatable field not ending on a halfword boundary.

   c. It contains an expression of mixed resolution.

   d. The program is loaded in the extended memory mode. The loader enters this mode when it does not have enough core to build the core image of each segment in its entirety.

4. Segments may communicate with each other via REFs and DEFs only if they lie in the same path.

5. Load items of a DSECT are placed in the corresponding DSECT of the root segment. That is, there must be a DSECT by the same name in the root. The following case is not permitted.



6. MODIFY control cards will be ignored if a library load module is being formed or if extended memory mode is entered.

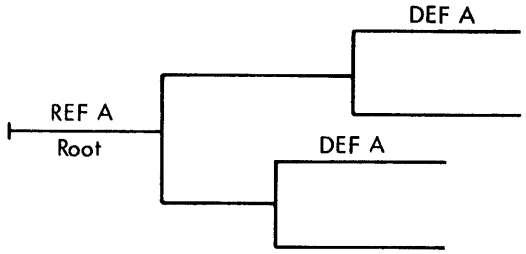
7. If a low segment references a DEF name that is both in a higher segment and a library, the library DEF will be used.

8. A program containing a relative address preceded by a minus sign (e.g., -BA(ADDR)) is not relocatable.

9. The load module name and input file names must be no greater than 10 characters in length. The element file names must be no greater than 8 characters in length.

10. No two segments on the TREE control command may begin with the same ROM name, since the first ROM named in a segment becomes the name of that segment.

11. If a low segment common to two or more paths references a DEF name that is in a higher segment of more than one path, that name will be doubly defined. The following case is not permitted:



12. If extended memory mode is entered, the load module being built must have no more than 256 segments.

13. Programs loaded in branch reference loading mode (see the BREF option) cannot contain BALs on register 0 because the loader uses register 0 in the BREF code which it supplies.

14. A library dummy section containing multiple defined locations must be loaded in a segment of an overlaid program below any segment referencing those locations.

15. Library DCBs may not be referenced solely from overlay segments.

16. DEFCOM output may not be included in a library load module.

17. Under certain rare conditions, it is not possible for the loader to accurately predict its core requirements by the end of its first pass. This may result in situations in which the loader will not automatically enter extended memory mode to produce a large load module, resulting in the "Insufficient Physical Memory" error message following the loader's allocation summary. The use of the PAGE option to force extended memory mode will alleviate this situation.

18. LYNX and LEMUR allow the assignment of a library name to the library that is being built. This is accomplished in LEMUR with the LIBRARY command and in LYNX with the (LIB, libname) option. Omission of either of these will default to the name :LIB for both library creation and usage. A library other than :LIB must be created by LYNX or LEMUR. Usage of a library other than :LIB must be invoked through LYNX (not LOAD).

Examples

```
!LOAD
```

This example specifies that loading is to be accomplished from the BI device. Default conditions are assumed.

```
!(TSS,3E8),(BIAS,10000),(BI),(M100)
!(LMN,MOD),(PERM),(WRITE,NONE),(SL,2);
!LOAD (EF,(FIL,ACCT123,PAS)),(UNSAT,(1235)),;
```

This example specifies that

1. No load information is to be taken from the GO file, since GO is not specified.

2. Element FIL, having the password PAS associated with it, is to be loaded from the account ACCT123.

3. The library of account 1235 is to be searched for external definitions corresponding to unsatisfied primary references (if any exist after loading has been accomplished from all other specified sources).

4. The name MOD is to be associated with the load module.

5. The load module is to be a permanent file in the user's account.

6. Assuming that the user was authorized with a default read access of 'ALL', any account may read the load module, but none may write into it.

7. Errors of severity level 2 are acceptable.

8. Up to $3E8_{16}$ words of temporary storage may be used.

9. A relocation bias of 10000_{16} is to be used.

10. No load map is to be output.

11. Relocatable object modules are to be loaded from the BI device.

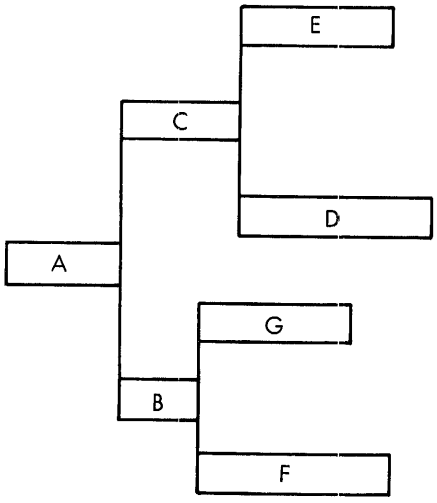
12. Each control section or dummy section is to be loaded starting at a multiple of 100_{16}.

TREE If a program is to be overlaid, a TREE control command must be the next control command following the associated OVERLAY (OLAY or LOAD) command. It must specify the overlay structure of the load module to be formed as a result of the preceding OVERLAY, OLAY, or LOAD command, 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 LOAD, OVERLAY, or OLAY 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 control 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. 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.

PTREE A PTREE control command may be used to obtain a TREE control command from the user's file (useful in jobs involving COBOL programs).

The form of the PTREE control command is

```
!PTREE (name[,account[,password]])
```

where

name specifies the name of the file containing the TREE control command.

account specifies the account containing the designated file.

password specifies the password associated with the designated file. If the file has an associated password, both it and the account number must be given in the command.

INCL An INCL (include) control command may be used, following a TREE or PTREE command, to include a named library routine in a specified overlay segment (e.g., to satisfy a secondary external reference).

The form of the INCL control command is

```
!INCL,segment name[,name]...
```

where

segment specifies the name of the segment to which the named library load module or ROM is to be appended. Each segment takes the name of the first element file named in the segment specified on the TREE card.

name specifies the name of a library load module or ROM that is to be appended to the specified segment.

Any number of library load modules and/or ROMs may be specified in a single INCL command.

Example:

An example of the control card sequence used to specify the structure of an overlay program is given below.

```
!TREE DEF - (GHI, JKL - ABC)
! 1234), (WRITE, NONE), (BI), (REF), (MAP)
! (1236), (1237)), (LMN, FILEX), (PERM), (READ, ;
!OVERLAY (EF, (ABC), (DEF), (GHI), (JKL)), (UNSAT, ;
```

The above example specifies that

1. Elements ABC, DEF, GHI, and JKL are to be loaded from the element file of the present account.
2. The libraries of accounts 1236 and 1237 are to be searched if unsatisfied primary references exist after loading has been accomplished for all other sources specified.
3. The name FILEX is to be associated with the load module being formed.
4. The load module is to be a permanent file in the user's account.
5. Account 1234 may read the load module, but no account (other than that of the current job) may write into it.
6. Relocatable object modules are to be loaded from the BI input device and placed in the root segment.
7. The overlay structure is to be set up for loading in the reference mode.
8. A load map is to be output.
9. The system library is to be searched for external definitions corresponding to unsatisfied primary external references (if any).
10. The overlay program is to consist of 3 segments, namely DEF, GHI, and JKL.

RUN The RUN control command specifies that a designated program (or the program most recently formed by the loader or Link) is to be executed, provided that the execution error severity level (see XSL option) has not been exceeded by the program (i.e., the load module).

The form of the RUN control command is

```
!RUN [(option)][,(option)]...
```

where the options are as follows:

LMN,name[,account[,password]] specifies the name (account number and associated password, if any) of the load module that is to be executed. The name may consist of from 1 to 10 alphanumeric characters (except for shared processor names which may only have up to 8 characters). If this option is omitted, the job's most recently formed load module will be executed.

START,address specifies the location at which program execution is to begin. The "address" may be either an external definition (optionally followed by a hexadecimal addend value) or a signed absolute hexadecimal address. This address overrides that specified in the load module. The external definition must not contain any embedded addend value (e.g., plus (+) or minus (-)).

If no start address is specified in the RUN command or in the load module, the program is entered at its lowest core location, which is register 0, and causes a trap and the job to abort.

XSL,value specifies a value to be placed in the Task Control Block (TCB) for examination at execution time by the user or run-time library routines. The default value is 8. XSL is used as the "current abort severity" by the FORTRAN IV run-time routines.

MODIFY The MODIFY control command allows the user to insert or modify words of a program in core storage. Library load modules cannot be modified by this command.

The form of the MODIFY control command is

```
!MODIFY[,segment] loc,word[,word]...
```

where

segment specifies the name of an overlay segment. This parameter is omitted if the load module is not overlaid.

loc specifies a relative hexadecimal location (i.e., an external definition followed by an optional hexadecimal addend value) or a signed positive absolute hexadecimal address where the modification is to be made. If an external definition is used, and the modification is to be made to an overlay segment, the definition must not have been referenced in a "lower" segment of the overlay tree. This restriction applies only if the MODIFY command appears after the OVERLAY, LOAD, or

OLAY control command. The total number of locations to be modified cannot exceed 255. The external definition must not contain any embedded addend value (e.g., plus (+) or minus (-)).

word specifies the word to be inserted (right-justified) at the designated location (see "loc", above). The word must be expressed as an unsigned hexadecimal (i.e., value + name). If it is desired to specify an address resolution for the external definition (following the value), the name of the external definition must be enclosed in parentheses (i.e., value + res (name)).

res	Resolution
BA	Byte
HA	Halfword
WA	Word
DA	Doubleword

If no resolution is specified, word resolution is assumed.

The MODIFY control command may be used either following a LOAD command or a RUN command. If used following a LOAD command, the inserted words become a permanent part of the program; otherwise, they are a temporary "patch" used only during the current execution of the program. If the load module is overlaid and the patch is to be permanent, the MODIFY command must follow the TREE command.

Example:

```
!MODIFY LOC1+A1,1234E
```

This example specifies that the hexadecimal value 1234E is to be inserted at a location whose address is 161 words higher than that of LOC1.

LIBRARIES

The purpose of a library is to collect frequently-used routines in a form that expedites their inclusion into other programs.

TYPES OF LIBRARIES

There are basically two types of libraries: public and user.

A public library can be viewed as code that resides in a fixed part of memory for usage by many concurrent users; it does not need to be loaded. Public library routines do not become a permanent part of a user's program.

A user library is a keyed file residing in an account. The file contains several modules, each of which is a named collection of routines. A library module becomes a permanent part of a user's program; consequently each user has a separate copy of a pertinent module. Collecting these modules into a single file (rather than making each module a separate file) minimizes the number of opens and closes that the loader must perform to process several such modules from one account.

PUBLIC LIBRARIES

The loader associates a public library with a program provided one of the following conditions exists:

1. The program contains an unsatisfied PREF to a module in the public library.

2. The public library (:Pn) is named in the UNSAT list on the LOAD card.

Either of these conditions causes the loader to allocate the public library's context area at the beginning of the user's virtual memory (normally X'A000').

To illustrate: the FORTRAN library subroutines reside in public libraries :P0 (FORTRAN library with FDP), :P1 (FORTRAN library without FDP), and :P4 (FORTRAN real-time library). A FORTRAN program contains an unsatisfied PREF to 9INITIAL or 9DBINIT which causes the loader to associate :P1 or :P0, respectively. If the real-time version of FORTRAN is required, :P4 is named in the UNSAT list. (The real-time system account, e.g., :SYSRT, must also be named.)

USER LIBRARIES

User libraries are formed by LYNX, Load, or LEMUR. The creation of a user library is triggered by the presence of the LIB option when LYNX or Load is used, or by the BUILD command when LEMUR is used. The name of the library is supplied by one of two methods: 1) by the LIBRARY command in LEMUR, or 2) by the secondary option libname, with LIB in LYNX. If neither of these are used, or if LOAD is used, the default library name is :LIB. If the named library does not exist in the running account, the "skeleton" of the file is created by opening to the library name.

STRUCTURE OF A USER LIBRARY

A user library can be viewed as having two sections: the dictionary section and the library module section.

The first part of a library is the dictionary section, comprising all the DEF names defined in that library and the names of the library modules in which those DEFs occur. The key of a dictionary record is a DEF name prefixed by a blank (X'40') to ensure its primary placement within the file. The dictionary record is the TEXTC name of the module within which the DEF occurs. (Table 18 shows the dictionary format.) The DEF is limited to 15 characters; the module name is limited to 10 characters.

The second part of the library contains the library modules, in either library load module or ROM form (as shown in Tables 19 and 20 respectively). The keys and records of a library load module are identical to those of a nonlibrary load module except that the keys HEAD and TREE are concatenated with the TEXT load module name to ensure the uniqueness of the record.

USER LIBRARY MODULES

Library modules are in library load module form if built by the overlay loader (via Load, LYNX, or LEMUR) or they are left as ROMs if built by LEMUR with the ROM option specified. The load module form offers the advantage that the user's target program can be built faster since the loader can process an input file in load module form much faster than in ROM form. Library load modules must be non-overlaid, relocatable, and of one protection type. The ROM module form does not have a restriction on protection type; the user may therefore include any ROM in a library without regard to protection type.

The loader creates library load modules by opening the library with "file name = LMN name" (where LMN name is the name of the library load module) and "synonym = library name". This synonymity allows inclusion of a library load module via explicit mention of its name in the element file list, rather than implicit inclusion via an unsatisfied reference.

When LEMUR is used to create a library ROM module, the name of the ROM module is supplied by the name field of the BUILD command. ROM modules are not synonymous to the library name.

Assuming the library does not already contain a module with the same name as the module being created, the loader or LEMUR writes a dictionary record for each DEF encountered in the new module. Depending on library type, this involves a scan of the REF/DEF stack (for load modules) or the ROM codes (for ROM modules). The library module records are then written.

If a module with the same name as the new one already exists within the library, the loader or LEMUR deletes all old dictionary entries containing DEFs that occurred in the old version. This is done by scanning the old version's REF/DEF stack (loader) or dictionary records (LEMUR). The new version (dictionary and module records) is then written.

(Determination of whether a module exists within a library is made by attempting to read a module record from the library using the name of the new module.)

Table 18. Library Dictionary Format

Key	Record Contents
X'40' {text of DEF}	textc name of library module

Table 19. Library Load Module Format

Key	Record Contents
LMN name concatenated with C'HEAD'	Basic information
LMN name concatenated with C'TREE'	Tree Table
LMN name concatenated with X'00'	REF/DEF stack
LMN name concatenated with X'01'	Expression stack
LMN name concatenated with X'0n'	00, 01, or 10 Relocation Dictionary
LMN name concatenated with X'0(n + 1)'	00, 01, or 10 Control Sections (same protection type as relocation dictionary)

Table 20. Library ROM Module Format

Key	Record
module name from LEMUR BUILD concatenated with X'0000'	library ROM module records
module name from LEMUR BUILD concatenated with X'0001'	
module name from LEMUR BUILD concatenated with .	
module name from LEMUR BUILD concatenated with .	
module name from LEMUR BUILD concatenated with .	
module name from LEMUR BUILD concatenated with X'FFFF'	

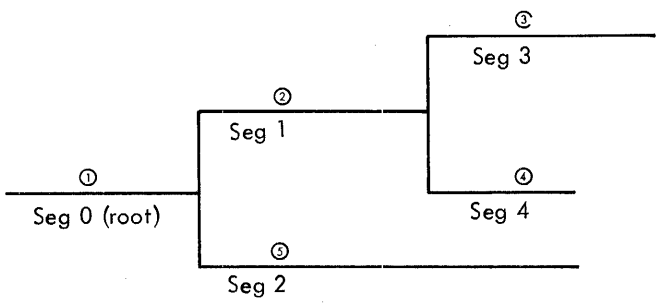
If a pre-existing module with a different name contains a DEF identical to a DEF in the new module being added or replaced, the dictionary record corresponding to this DEF is replaced by a record pointing to the new module. Thus, there is only one dictionary record per DEF, and it contains the name of the library module most recently entered that defines the corresponding DEF.

Note: Individual library modules cannot be accessed by PCL, since a library module is really a logically related set of records within a file. Functions such as copying and deleting library modules must be performed by LEMUR, not PCL.

USER LIBRARY REFERENCE PROCESSING

The loader associates a library module with the program the loader is processing if that module contains a DEF that satisfies an undefined PREF in the program. To accomplish this, the loader performs a library search by doing a keyed READ to the dictionary, using the name of an unsatisfied PREF as the key; a successful READ returns the name of the library module defining the PREF. The loader then reads the library module records into core and merges them with the program.

For an overlay program, the loader conducts a library search each time it finishes processing the external references in an entire segment. The segments are processed in the order specified in the following illustration:



Note that the loader attempts to satisfy all the PREFs in a lower segment before processing the DEFs to a higher segment, so that if a low segment has a PREF whose corresponding DEF is located in both a higher segment and one of the libraries specified in the UNSAT list, the library DEF will be used. (Otherwise, the high segment and all its backward path would be brought into core each time the lower segment needed that DEF.)

DIAGNOSTIC MESSAGES

Diagnostic (error) messages are output on the LL device. Table 21 lists the messages that are produced by the monitor when bringing a program into core storage for execution (running a load module).

Table 21. Monitor Error Messages

Message	Description
ABS CANNOT REL	The monitor cannot relocate program because it is absolute.
STACK OVERFLOW	The program will not fit in core.
IO ERROR or IO ABN	An I/O error or abnormal condition has occurred.
NO LOAD MODULE	The load module named is not available.

LOAD DIAGNOSTIC MESSAGES

The Load processor uses a keyed file provided by the Error Message File Write program (ERRMWR) for its error message records. Upon finding a load error, the Load processor obtains a message record from an ERRMSG file using an error-key.

The message record, the error-key, and additional information are printed according to the following format:

Line 1 error key message record
Line 2 INPUT FILE SEQ NO. CODE/SIZE/SL
Line 3 file name number var. data

where

file name specifies the name of the last input file processed by the loader.

number specifies the hexadecimal sequence number of the last binary record read.

var. data specifies variable information, the meaning of which depends on the particular error that occurred. When the contents of the variable data field is SR3, the message is preceded by the monitor error/abnormal code and its meaning.

The message records, their corresponding error-keys, and the significance of the CODE/SIZE/SL field are shown in Table 22.

If for any reason access to a message record is denied the loader, the following message is printed.

xxxxxx BAD ERROR MESSAGE FILE

where xxxxxx is a hexadecimal number for the key.

Table 22. Load Error Messages

Key	Message	Description	CODE/SIZE/SL Field
020001	UNEXPECTED EOF	An end-of-file was encountered before the end of an object module was reached (incomplete object module).	SR3
020002	ILLEGAL RECORD I.D.	The type of record read was neither X'3C' nor X'1C' (object module) nor X'81', X'82', or X'83' (load module).	Record I.D.
020003	SEQUENCE ERROR	The cards of an object module were out of sequence.	(None)
020004	ILLEGAL RECORD SIZE	The number of bytes in an object module card was less than five or greater than X'6C'.	Record Size
020005	CHECKSUM ERROR	A bit (or bits) was dropped in punching or reading the object module.	(None)
020006	ABNORMAL I/O	An abnormal return was encountered while reading a library load module or ROM.	SR3
020007	CANNOT OPEN E.F.	An element file could not be opened. (It does not exist, it has a password, etc.)	SR3
020008	STACK OVERFLOW	Insufficient memory in which to load. If no map has been partially printed, the module is too large. If a map has been partially printed, some unsatisfied primary references have caused the stack to grow to excessive size. (See messages with keys 020015 through 020023.)	SR3
020009	BIAS TOO LARGE	At the given bias, the load module will exceed 131K of memory.	Bias
02000A	ILL. ROM LANGUAGE	The object language in a relocatable object module was not translatable (assembler or compiler error).	Object module control byte
02000B	BAD START ADDRESS	A start address was given which is either not on a word boundary or is not within the load module.	Start address
02000C	UNEXPECTED ROM END	Module end was given on some card of the object module other than the last card (assembler or compiler error).	(None)
02000D	REPEAT LOAD IS ZERO	An assembler or compiler generated a repeat load item with a 0 count (assembler or compiler error).	(None)
02000E	IMPROPER BOUND	A short- or long-relocatable item was not on a word boundary.	Byte address of load relocatable item
02000F	ILLEGAL ORG	An origin was generated having no resolution or was not within the load module (assembler or compiler error or violation of loader DSECT restrictions).	SR4 (for debugging purposes)

Table 22. Load Error Messages (cont.)

Key	Message	Description	CODE/SIZE/SL Field
020010	BAD I/O RETURN FROM M:LM DCB	The load module file could not be opened.	SR3
020011	SEV. LEV. EXCEEDED	The severity level specified in the LOAD card was less than that encountered in some object module or that generated by the Loader (a DDEF yields a severity level of 4, a PREF yields 7).	Computed severity level
020012	ILL. LIB. LOAD MOD.	(PERM) and (LIB) were specified and the load module had one of the following: 1. More than one protection type. 2. No relocation dictionary (ABS was specified or forced by the Loader due to nonstandard relocatable fields). 3. More than one segment.	(None)
020013	NO ROOM TO ROUND DCBs TO PAGE BOUNDARIES. TRY FORCING XMEM.	The DCBs and DCB Name Table exceed 1024 words.	High address of DCBs
020014	ILL. DSECT	Two dummy sections having the same name but different protection types were encountered.	First 4 characters of DSECT name
020015	ROOT SEGMENT TOO LARGE TO LOAD		Number of words exceeding available core
020016	TOO MANY CORE LIBRARIES	Only one core library (named :P0, :P1, :P2, etc.) is permitted.	(None)
020017	CANNOT ENTER XMEM. STACKS TOO LARGE		Number of words that stacks exceed available core
020018	NOT ENOUGH ROOM TO CONCATENATE XMEM PAGES		Number of words exceeding available core
020019	NO ROOM TO READ LIBRARY CORE IMAGE		Size of library lmn's core image
02001A	NO ROOM TO READ LIBRARY RELOCATION DICTIONARY		Size of relocation dictionary
02001B	NO ROOM FOR NEW EXPRESSION		(None)
02001C	NO ROOM TO BUILD DCB TABLE. TRY FORCING XMEM.		(None)

Table 22. Load Error Messages (cont.)

Key	Message	Description	CODE/SIZE/SL Field
02001D	NO ROOM TO BUILD DCB		Size of DCB table
02001E	LIBRARY LOAD MODULE REF/DEF STACK TOO LARGE TO UPDATE		(None)
02001F	INSUFFICIENT PHYSICAL MEMORY	See "Stack Overflow" description (Key 020008).	R0 (for debugging)
020020	BAD ASSIGN/MERGE RECORD		SR3
020021	NO ROOM TO ADD LIBRARY LOAD MODULE TO ROM TABLE		Top of REF/DEF Stack
020022	NO ROOM TO READ LIBRARY REF/DEF STACK		Size of library lmn's REF/DEF Stack
020023	NO ROOM TO UPDATE LIBRARY		REF/DEF Stack size of old version of this lmn.
020024	INVALID KEY SUPPLIED FOR DELETE RECORD ON M:DIC	Cannot update :DIC for this library load module.	Key size
020025	IO ERROR ON M:DIC IN WRITESEG	Cannot update :DIC for this library load module.	SR3
020026	ILLEGAL LIBRARY LOAD MODULE NAME	The name is > 12 characters.	Number of characters in name
020028	INVALID DECLARATION NUMBER REFERENCE (BAD ROM)	An expression in a relocatable object module contains a reference to an unassigned declaration name number (assembler or compiler error).	The bad declaration number
020029	INVALID KEY SUPPLIED FOR WRITE RECORD ON M:DIC	A DEF name in a library load module was not in the legal range of 1-63 characters.	Key size
02002A	ILLEGAL LOADER TRAP	Loader error. When such errors occur, the loader takes memory snapshots for use in identifying the error.	Register 0
02002B	ABNORMAL I/O IN WRITELIB	The :LIB file could not be opened.	SR3
02002C	CANNOT FIND REF/DEF NAME IN STACK	The loader encountered a new REF/DEF name during its second pass.	Byte count and first 3 characters of name
02002D	LIB. LOAD MODULE TOO BIG – CANNOT USE EXTENDED MEMORY	Extended memory mode has been entered (because the core image is too large to be formed in one piece) and the load module has been forced ABS (illegal for library lmn's).	(None)

Table 22. Load Error Messages (cont.)

Key	Message	Description	CODE/SIZE/SL Field
02002F	ABNORMAL I/O READING LIB LMN	An abnormal return was encountered while reading a library load module during the loader's second pass.	SR3
020030	PAGED LMN MUST NOT HAVE MORE THAN 256 SEGMENTS		Number of segments specified
020031	LMN'S SIZE TOO BIG	The size (in doublewords) of a protection type of the load module does not fit in the halfword allowed for it in the tree.	(None)
020032	THAT'S NOT A (MAPPABLE) LOAD MODULE	Specified file has no 'HEAD' or 'TREE' record.	Byte count and first three characters of name
020033	BAD ENTRY IN LIBRARY REF/DEF STACK	The loader detected a malformed library REF/DEF stack. (The user may have violated rules for library load modules.)	(None)
020034	BAL TO AN OVERLAY ON REGISTER ZERO DETECTED WHILE IN BREF MODE	BAL, 0 to an overlay segment is not allowed in BREF mode.	(None)

LYNX PROCESSOR

LYNX is a load processor that is available in both the online and batch modes. LYNX has the capabilities of the overlay loader, Load, and also provides the same control over internal and global symbol table construction which is available in the linking loader, Link. 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. Therefore LYNX is a load module as shown in Table 17.

The batch made LYNX processor recognizes two commands, LYNX and :TREE. Since the LYNX command is a control command which calls the LYNX processor, it must be preceded by a ! character. These two commands will be described in detail later.

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, or the

```
!RUN
```

command in the batch 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 parenthesis 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 options 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 now be specified as element files. However, the presence of either 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 RUN command.

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 :SYS.

password specifies the password for the library if one exists.

options specify 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 the command verb. All options must be specified within parentheses.

As with the LINK command, the options may actually appear anywhere in the command string and must be preceded by a left parenthesis or enclosed within parentheses. The options are described below.

Options that determine input to the loader

BI specifies that the BI input device 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.

Options affecting future access to the load module file

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 account's element file 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, {mm,dd,yy / ddd / NEVER} 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

{OS / SEG} 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.

{OR / SEG} [,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.

Additional 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 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.

{MOVN / MONV} 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. NBS and MNV cannot both be specified.

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 DESECT 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 without 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 allocation.
- 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.

MAPPING EXISTING LOAD MODULES

In order to produce a map of an existing load module, the format of the LYNX command must be:

!LYNX fid {(MO) / (MOV) / (MONV)}

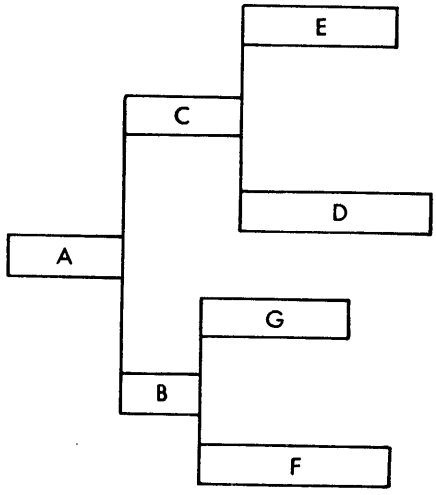
where fid specifies the file identification of the load module. The LDEF and UDEF options are also valid in this context. All other options will be ignored.

:TREE If a program is to be overlaid, a :TREE command must be the next command following 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

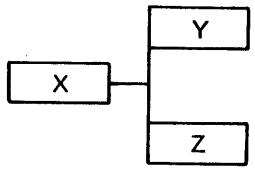
The following is an on-line example of LYNX usage.

!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 23.

Table 23. 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 on the LYNX command.
*** CONFLICTING OPTIONS	The user specified two conflicting options (e.g., I, NI)or the same option twice.
*** 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	The file name must be no more than 10 characters in length.

Table 23. LYNX Error Messages (cont.)

Message	Description
*** 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.
*** ROOT WOULD BE OVERLAID-BAD TREE STRUCTURE	The user misplaced a parenthesis or misused the :TREE specification.
*** 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 PROCESSOR

The Link Processor operates in the batch mode or in the on-line mode. It 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. Program execution is initiated by the RUN command described below under "Control Commands". (Note: The batch-mode RUN command has a different format than the on-line RUN command used to initiate execution. The batch mode RUN command is described in the Load processor description. However, it may be used to execute a load module formed by either the Link or Load processors. The accounting summary generated at the end of each job is described at the end of this chapter.

As previously mentioned, Link is a one-pass linking loader that makes full use of mapping hardware. It is not an overlay loader. The Load processor must be used if an overlay loader is needed.

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).

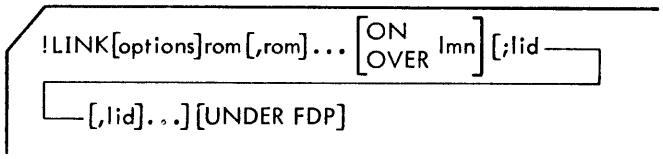
03 no read or write permitted (no access).

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.

LINK CONTROL COMMAND

LINK The loader that is invoked by a LINK control command processes relocatable object modules. The resulting load module is a keyed file that is placed in the user's account. Execution of the load module is triggered by the RUN control command (described below) which brings the load module into core storage and transfers control to it. (A load module may also be called internally by an executing program via the M:LINK procedure.)

The LINK control command has the form



where

rom specifies a relocatable object module and may be either a file identification (fid) or a dollar sign. (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.) The dollar sign designates the most recent compilation or assembly. Parentheses enclosing roms cause merge of symbol tables

lmn (load module name) specifies where the load module is to be placed and may be a file identification (fid) or dollar sign. (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.) If lmn is omitted, the resulting load module is placed in a special file and is available for subsequent execution.

lid specifies a library file identification. 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 (lid), separated by commas, is appended to the list of modules in the LINK command and is separated from the module list by a semicolon.

codes are optional codes used to specify a library search, a display, or inclusion of a symbol table. The optional codes are described below; they may be entered anywhere in the command except between a preposition and its object.

Options specifying library search

(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.

(Pi)(Ji) specifies that the ith public core library is to be associated with the program to satisfy external references. Only one public library of each type (J or P) may be associated with a program. P0, P1, and P4 are supplied by Xerox; P1 contains a subset of the FORTRAN library subroutines; P0 includes P1 and the FORTRAN Debug Package; P4 includes P1 and the FORTRAN real-time features. J0 contains JIT definitions and J1 contains the monitor definitions. (These two libraries are generally only used by system analysts.) Additional public libraries must be named P2, P3, P5-P9, and J2-J9. (P1 is a default option.)

(FDP) equivalent to (P0).

(NP) specifies that a public core library is not required.

The sequence of the library search is as follows: User libraries are searched first, the public library is associated, and the system library is searched. In the absence of any other specifications, public library P1 is associated with the load module to satisfy external references, and the system (ROM) library is searched if necessary.

Options affecting end actions and error displays

(D) specifies that all unsatisfied internal and external symbols are to be displayed at the completion of the linking process (including library searches, if specified). The unsatisfied symbols are identified as to whether they are internal or external and to which module they belong. (This is a default option.)

(ND) specifies that the unsatisfied internal and external symbols are not to be displayed.

(C) specifies that all conflicting internal and external symbols are to be displayed. The symbols are displayed with their source (module name) and type (internal or external). (This is a default option.)

(NC) specifies that the conflicting symbols are not to be displayed.

(M) specifies that the load map is to be displayed upon completion of the linking process. The symbols are displayed by source with type resolution and value.

(NM) specifies that the load map is not to be displayed. (This is a default option.)

Options affecting inclusion of the symbol table.

(I) include symbol table with LM.

(NI) do not include symbol table with LM. (This is a default option.)

Option affecting execution of the load module

(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 when no EX option is specified). The value NONE may be used to specify that no other account may execute the lmn.

Examples

1. Assume that a load module, F, is to be created from ROMs A, B, C, and D. The internal symbols for files D and A are to be merged. The internal symbols for B and C are not to be included in load module F.

```
!LINK (A,D), (NI) B,C ON F
```

2. Assume that a load module, F, is to be created from files A, B, C, and D. Internal symbols for files B and C are not to be included in the load module; internal symbols for files D and A are to be merged. Two user libraries, G and H, are to be searched to satisfy external references. Public library P1 is to be associated with the load module but no search of the system library is required.

```
!LINK (D,A), (NI)B,C ON F;G,H
```

3. Assume the same problem as in the previous example except that the system library is to be searched for external references and public library P2 is to be associated with the load module.

```
!LINK (L)(P2)(D,A), (NI)B,C ON F;G,H
```

4. Assume the same conditions as in the second example except that no libraries are to be searched.

```
!LINK (NL)(NP)(D,A),(NI)B,C ON F
```

5. 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
```

If the Link processor needs additional information, the job will be aborted with the appropriate message output to the line printer. For instance, using the same example, suppose that Link cannot find MFL2 because it was supposed to be MFL3. The job will be aborted and the following message will be output to the line printer:

CANT FIND: RETYPE MFL2

CONTINUED COMMANDS

The LINK command may be continued from one card to the next by putting a 'less than' symbol (<) in column 80 of the card to be continued. This symbol cannot be embedded within a word or between a preposition and its object.

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:.

   Output is via the M:LO DCB. 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.) All the DCBs cannot exceed two pages when the Link processor is used.

   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 later in this chapter under "Data Control Blocks".

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 constructued 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.

   Four public libraries are available: 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, respectively. (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 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 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 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 provide internal symbols with internal resolution and type classification. The loaders retain this information in processing object language code.

DIAGNOSTIC MESSAGES

Diagnostic (error) messages are output on the LL device. Table 14 lists the messages that are produced by the monitor during a link operation. Some of these messages are for syntax errors, and others are for errors arising out of the link operation. Most of these errors terminate the link operation prematurely.

Table 24. 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 LL device 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 overlays 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.

Table 24. Link Error Messages (cont.)

Message	Description
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. The load module is still formed.
ON FILE fid ILLEGAL	ON was specified and the output file (fid) already exists.
SEQUENCE ERROR	A sequence error has occurred.
STACK OVERFLOW	An internal storage overflow has occurred.
UNEXPECTED END OF ROM DATA	EOF encountered before last card of ROM.
Note: All errors, except CANT FIND and NO PROGRAM START ADDRESS cause abnormal termination of Link.	

LEMUR PROCESSOR

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) that 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 specified ROMs.
- 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 one library to another.
- Copy a library to another library.

CALLING LEMUR

LEMUR is invoked in the batch mode by the control command

```
!LEMUR
```

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 10 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 on 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 module 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 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 any 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
LIBRARY LIB5
BUILD LR1 FROM R1 (ROM) (M)
BUILD LR2 FROM R2, R3 (ROM)
BUILD LR3 FROM R3 (SL,4) (C1)
BUILD LR4 FROM R5, R6 (SL,4) (M)
END
```

2. Assume that the user is logged on in account A55 and that the user wishes to replace LR3 in the example above with a load module rebuilt from R4.OTHERACT:

```
!LEMUR
LIBRARY LIB5
BUILD LR3 FROM R4.OTHERACT(SL,4)(C1)
END
```

DELETE The DELETE command deletes either the destination library (i.e., the library named on the LIBRARY command or :LIB by default) or deletes one or more named modules from the destination library. In the latter case, all entries in the dictionary that 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
DELETE X,Y,Z
LIBRARY LIB6
DELETE A
END
```

2. Assume that the user is logged onto account A55 and wishes to delete the library :LIB.A55.

```
!LEMUR
DELETE
END
```

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 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
LIBRARY LIBB
COPY LIBA.1234
END
```

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
COPY .B36
END
```

CARRY The CARRY command copies a library module from one library (source library) to another library (destination library). The format of the command is

CARRY $name_1$ FROM lib-id[/$name_2$]

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 name 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 same 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
LIBRARY NEWLIB
CARRY ZAP FROM NEWLIB.:SYS
END
```

(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
LIBRARY LIB2
CARRY MAP FROM LIB2.ACN2/ZAP
END
```

3. The user wishes to carry module SQRT from :LIB.:SYS to SQRT in :LIB.A55.

```
!LEMUR
CARRY SQRT FROM .:SYS
END
```

(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
CARRY ZAP FROM :LIB/MAP
END
```

An error message is issued and the command is aborted.

END The END command terminates LEMUR and returns control to CCI. The format of the command is

END

ERROR MESSAGES

Error messages for LEMUR are listed in Table 25.

Table 25. 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 with 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 COMMENT	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.
LIBRARY NAME TOO LONG	A library load module name exceeds 10 characters.
MALFORMED OPTION	Self-explanatory.
MAXIMUM SEVERITY LEVEL EXCEEDED	The severity level specified by the SL option has been exceeded.

Table 25. LEMUR Error Messages (cont.)

Message	Meaning
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 eight 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 26.

Table 26. 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.

TASK CONTROL BLOCK

The format of the Task Control Block (TCB) generated by a loader for the user's program is shown in Figure 9.

The fields of the TCB are as follows:

TSTACK is the address of the current top of the user's temp stack.

TSS indicates the size, in words, of the user's temp stack (maximum size is 7FFF).

TSA is the address of the temp stack used by the library error package.

TSASIZ indicates the size, in words, of the temp stack used by the library error package.

ERTSIZ indicates the size, in words, of the error table used by the library error package.

ERT is the address of the error table used by the library error package.

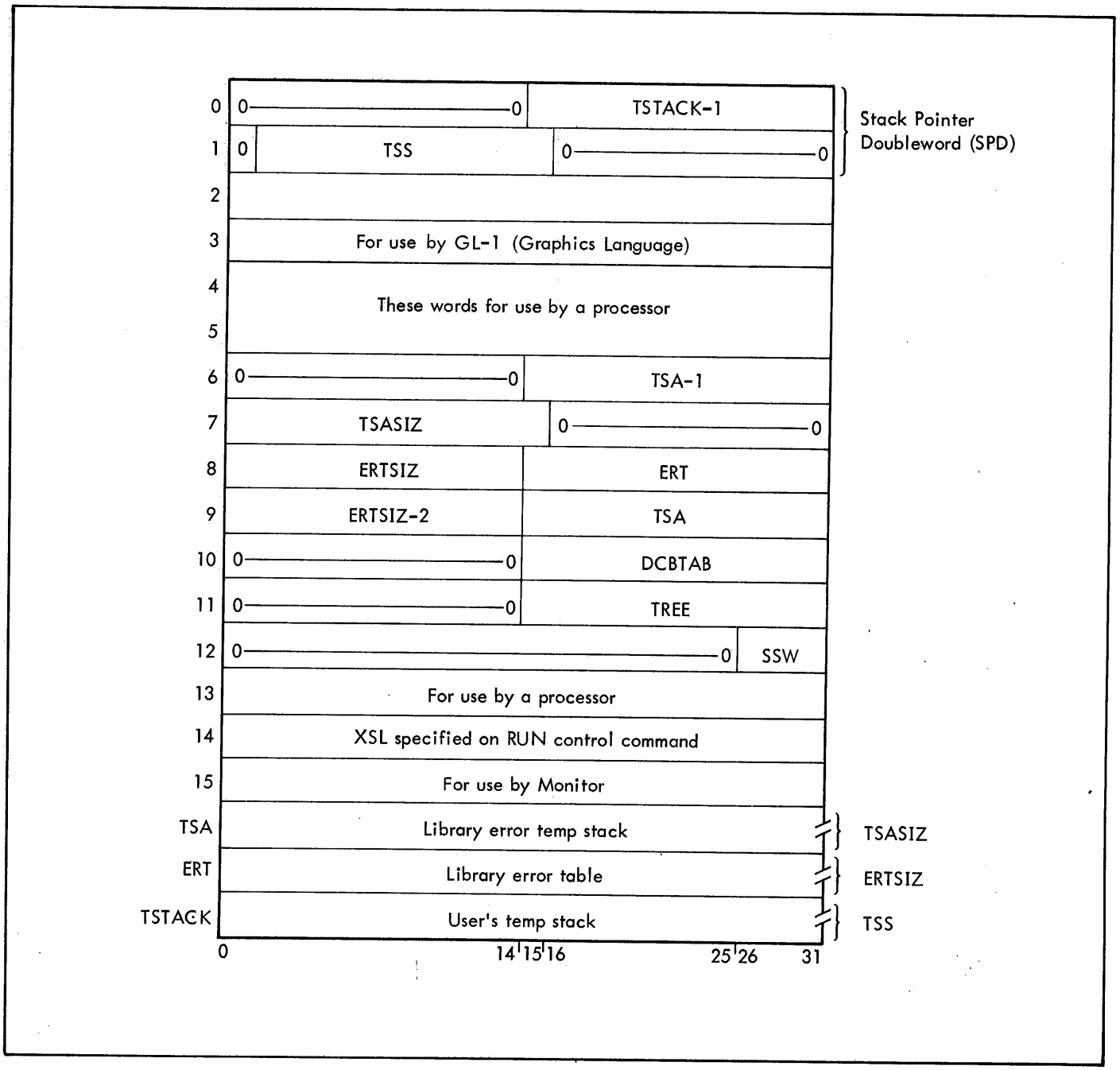


Figure 9. Task Control Block Format

DCBTAB is the address of a table of names and addresses of all of the user's DCBs. This table has the form shown in Figure 10.

TREE is a pointer to the location of the user's overlay structure.

SSW contains the user's sense switch settings (bits 26-31 contain the settings of switches 1-6).

On transferring control to a user's program or to a processor, the monitor communicates the TCB address through general register 0.

DATA CONTROL BLOCKS

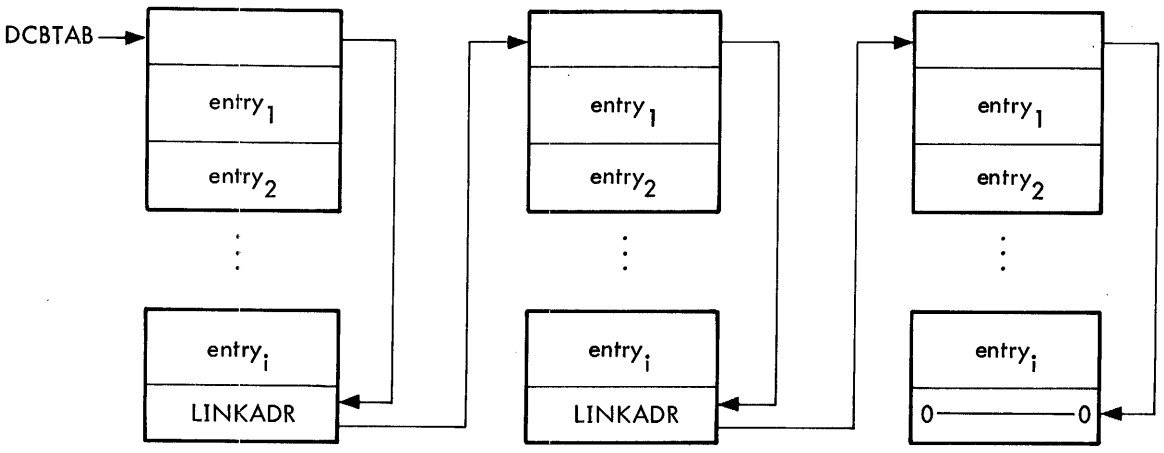
The loader constructs DCBs to be included in the load modules. The Load and Link processors build a DCB only if any PREFs exist that begin with either M: or F:. DCBs are not built, however, if the LIB option was specified in the LOAD command. Specifically, the Link and Load processors build DCBs when

- The Control Card Interpreter (CCI) assign/merge record contains an F: (for example, F:108) entry.
- The user has a REF DCB name and has no relocatable object modules (ROMs) or libraries in the element file list which satisfied this REF. The load processor does not search libraries of accounts in the UNSAT list to satisfy PREFs to M: and F: names. To include a library DCB in the output load module built by the LOAD processor, put the library load module name containing the DCB in the EF list.
- An M:DO DCB is generated in the absence of a NOTCB option. (The Link processor will always build an M:DO DCB.)
- An M:SEGLD is generated if a TREE control command is present.

Library DCBs are summarized in Table 27 together with the composition of DCBs generated by the loader and by COBOL.

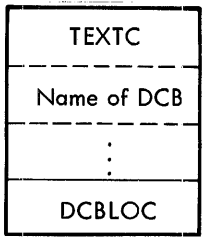
The detailed format of DCBs for files, devices, and labeled tape is shown in the appendix titled "Data Control Block Formats".

All loader generated system DCBs are 51 words long. The first 22 words (0 through 21) are standard and allocated for the fixed portion of the DCB. Each variable length parameter (words 22 through 40) is preceded by a control word, three words for file name, two words for account, two words for password, two words for an expire date, three words for INSN (SN), and three words for OUTSN (SN). Words 41-48 are reserved.



Notes:

1. Each entry contains a variable length DCB name and DCBLOC word as follows:



2. LINKADR is the location of another block of the DCB name table. If LINKADR contains zero, the current block is the last one of the DCB name table.

3. DCBLOC is the address of the first word of the DCB.

Figure 10. DCBTAB (Name Table)

Table 27. Data Control Block Size

DCB	Device	Name	Account	Password	Expiration Date	Read Accounts	Write Accounts	INSNs	OUTSNs	Synony-mous Name	Key Buffer	Total Words
M:C	22											22
M:OC	22											22
M:BI	22	9	3	3	3			4			8	52
M:CI	22	9	3	3	3			4			8	52
M:SI	22	9	3	3	3			4			8	52
M:EI	22	9	3	3	3			4		9	8	61
M:BO	22	9	3	3	3	17	17	4			8	86
M:CO	22	9	3	3	3	17	17		4		8	86
M:SO	22	9	3	3	3	17	17		4		8	86
M:PO	22	9	3	3	3				4		8	52
M:LO	22	9	3	3	3				4		8	52
M:LL	22	9	3	3	3				4		8	52
M:DO	22	9	3	3	3				4		8	52
M:GO	22	9	3	3	3						8	48
M:EO	22	9	3	3	3	17	17		4	9	8	45
M:SL	22	4	3	3	3						8	43
M:AL	22	4	3	3	3						8	43
Loader	22	4	3	3	3			4	4		8	51
COBOL	22	9	3	3	3	17	17	4	4	9	8	99

If the user requires a DCB with any field larger than those constructed by a loader, he must use the LOAD processor. He can then either construct the DCB with an M:DCB procedure call or, if it is an M: type DCB, request it explicitly from the :SYS account by including the DCB name as an element file in a LOAD, OVERLAY, or OLAY control command. For example:

(EF,(M:EO,:SYS),(M:LL,:SYS))

Total space for DCBs and buffers for a single job step is limited to 10,752 words (21 pages) including a one-word link for the DCB name chain and enough words to carry the DCB names. Each DCB must be contained entirely within one page to facilitate unmapped access. This usually results in less space due to "breakage". However, the amount of space provided is adequate in nearly all situations. When the allocated space is insufficient, the job is aborted with a code of X'C0'.

If a file is opened in the output mode through a system DCB, a flag is set in the Job Information Table (JIT). If the DCB is not reassigned before the DCB is opened again in the output mode for the same job, all records output through this DCB are appended to the end of this file.

MEMORY PROTECTION

Monitor pages and unallocated virtual pages are protected against access by user programs with the map access protection. Thus, programs that either deliberately or inadvertently access the monitor (by reading it or branching into it) will trap. The same restriction also applies to other areas of the machine that were not owned by the program (e.g., read access to unobtained common or dynamic data pages). The first page of core memory is an exception to these rules; its access is always set to read only. A trap

will occur on a conditional branch command for which the condition is not satisfied and the address of the branch is indirect through a protected memory address.

VIRTUAL MEMORY

The user's 96K words of virtual space are divided as follows:

1. 8K words for monitor overlays and user context (JITs and buffers).

2. 88K words for user procedure, DCBs, and data unless the user program requires the use of a special shared processor or a public library. In this case, the user area is 72K words and the special processor area is 16K words.

With the exception of a fixed minimum requirement of six pages for monitor overlays, one page for JIT, and three pages for the file buffers, the 96K words of user area is demand allocated.

The Link and Load loaders place ROM data, including any data overlays, in memory beginning at 40K then directly follow this with the DCBs, procedure, and procedure overlays. When a BIAS is specified, the load module is created at the specified location even though it may not be possible to run the load module there.

Load modules are constructed from ROMs composed of control sections. A control section is of type 00, 01, or 10. All control sections of type 00 are gathered together by the loader and designated as DATA. Similarly, all control sections of type 01 and type 10 are gathered together and designed as PROCEDURE and STATIC DATA, respectively.

Except for DCBs, DSECTs or CSECTs with value 2 or 3 are changed to 1. That is, no-access and read-only data are loaded with pure procedure. Any DSECT that has a name beginning with M: or F: is assumed to be a DCB and is removed to the DCB area and listed in the DCB table.

DCBs and the DCB name table are allocated in the user context area (10 protection) rather than in the root procedure area (01 protection). Because of the differences in allocation (DCBs) and the HEAD format, load modules formed under the BPM overlay loader will not execute (RUN) under CP-V and vice versa.

Internal symbol tables are generated for use by a debug processor (e.g., Delta) if a program is assembled with the SD option. An internal symbol table is built for each load module and is included in the load module as a keyed record consisting of the element file name appended with an X'10'. A symbol table can be loaded by a debug processor for an overlay or a nonoverlay program by specifying its element file name. If the element file contains more than one ROM then only the symbol table for the last ROM is produced. A symbol table that is generated during a load from the GO file cannot be accessed by a debug processor. No internal symbol tables are generated for library load modules.

VIRTUAL MEMORY LAYOUT

Figure 11 gives the layout of virtual memory for a program loaded by Load or Link. Ordinary shared processors follow this layout.

Figures 12 and 13 show the actual background memory layout at execution time for the Load and Link processors.

LOAD MAPS

If a listing of a load map is specified – MAP option of LOAD (OVERLAY or OLAY) control command or M option of LINK control command – a listing of external references and definitions for the load modules is output on the LL device.

A general allocation summary, indicating the total amount of memory allocated to each protection type for the entire program, appears after the LOAD or LINK control command. Next appears the severity level for this load module if it is nonzero. This severity level is actually the maximum of any severity levels inherited from the ROMs and those generated by the loader. Internal loader-generated severity levels are as follows:

Type	Severity
PREF	7
DDEF	4
REF or BREF load table exceeded	F
Nonbranching REFs found while in BREF mode	3

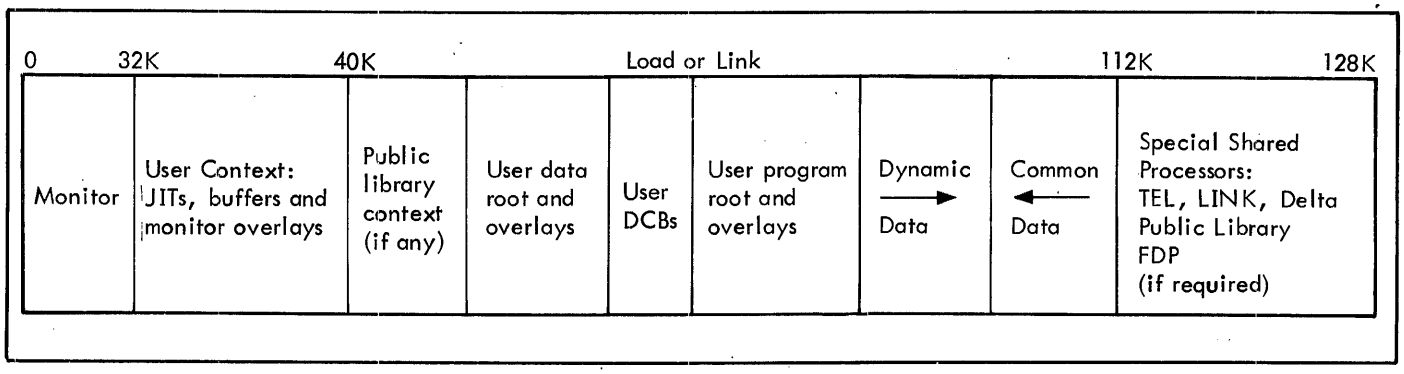
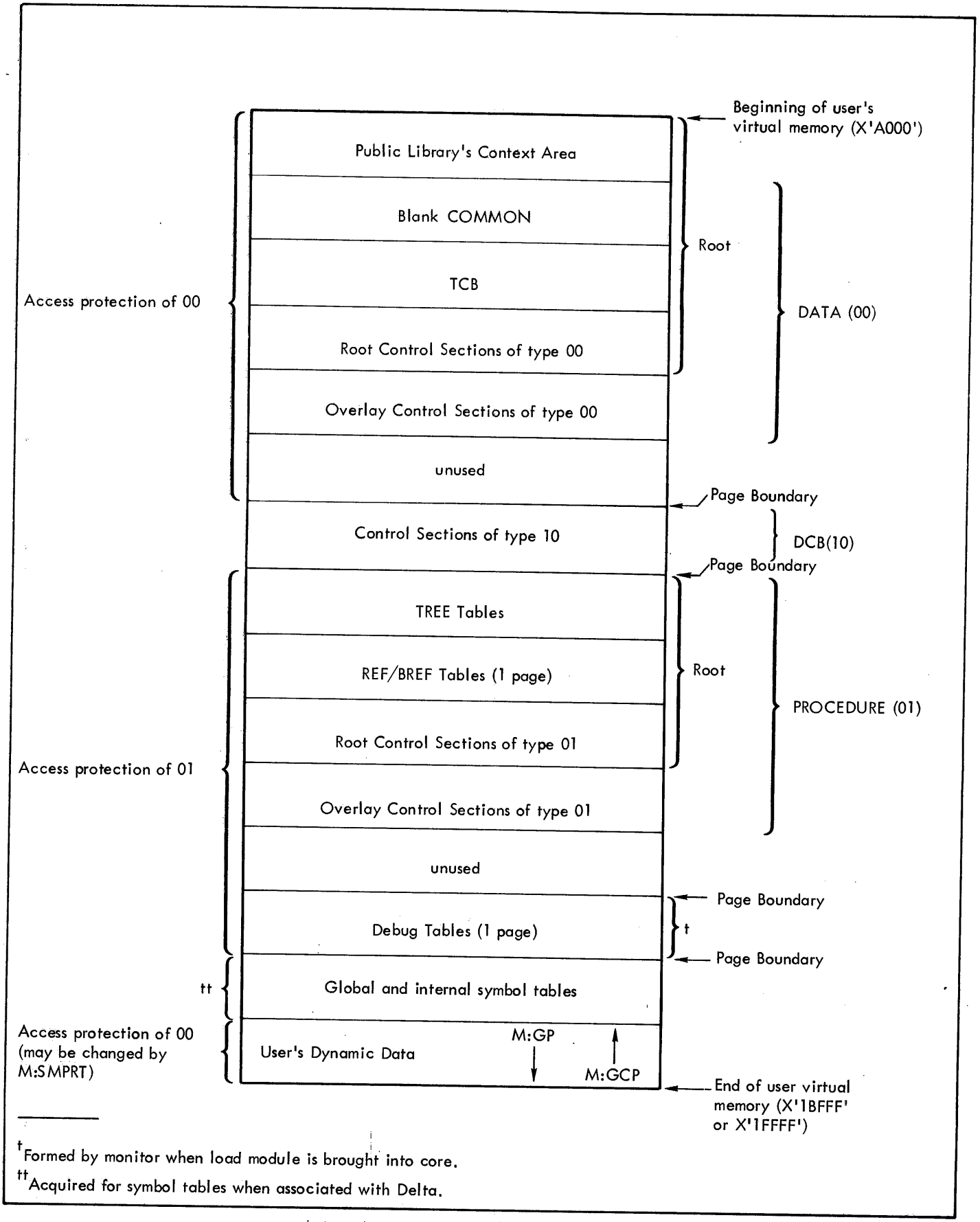


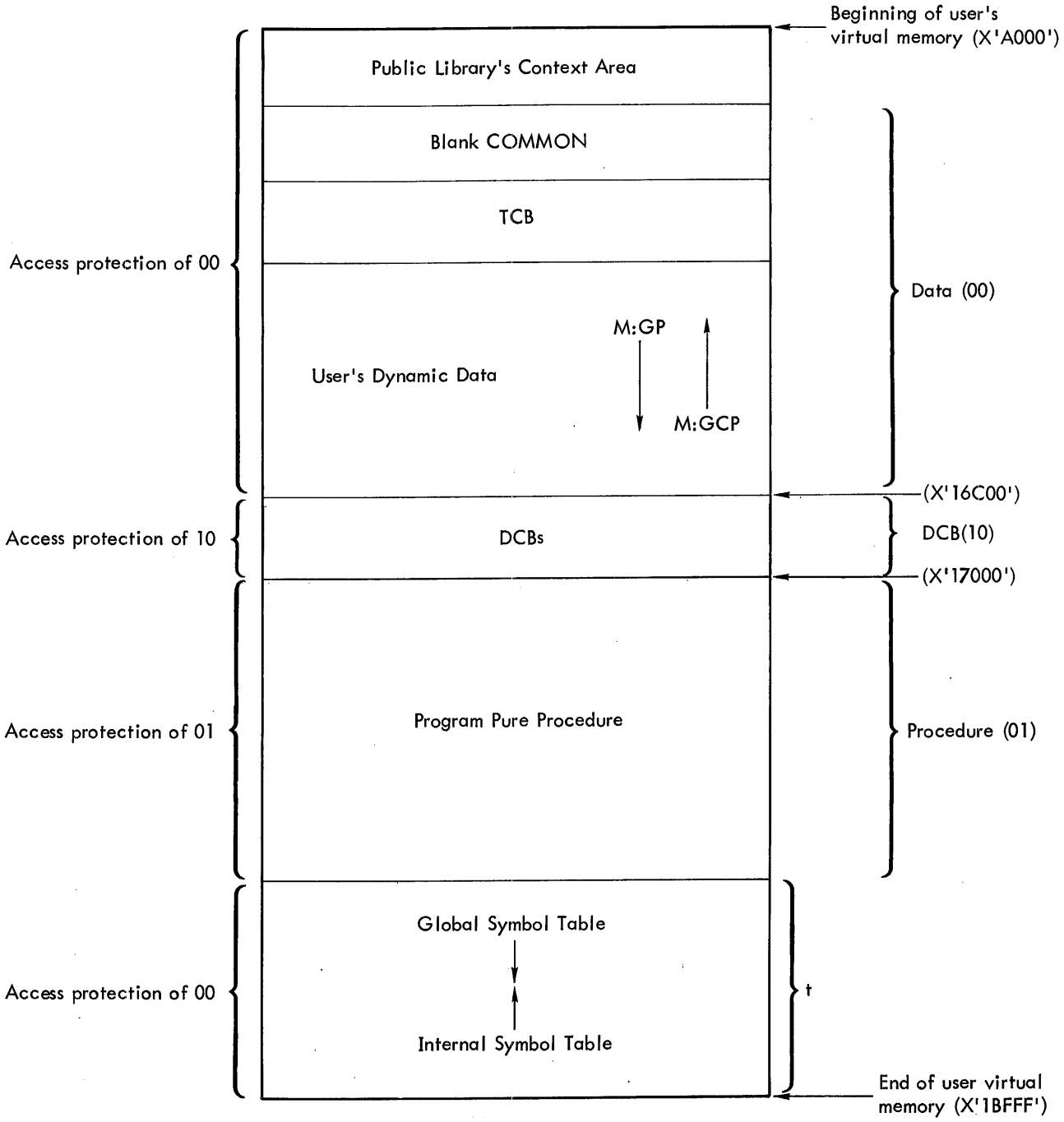
Figure 11. Virtual Memory Layout



[t] Formed by monitor when load module is brought into core.

[tt] Acquired for symbol tables when associated with Delta.

Figure 12. User Virtual Memory Layout, Load Processor



[t]Acquired for symbol tables when associated with Delta.

Figure 13. User Virtual Memory Layout, Link Processor

Figures 14 and 15 show sample load map printouts for the Link and Load processors respectively. The sample Link processor load map (Figure 14) is simple and self-explanatory. The sample Load processor load map (Figure 15), however, is more complex and requires further explanation.

```
!LINK LINKBO,VDCB (M) OVER LMN
     LINKING  LINKBO
SEVERITY 0
     LINKING  VDCB
SEVERITY 0
'P1' ASSOCIATED.
YOU DO NOT NEED
  P1
PREF              J:CCBUF
PREF              J:AMR
PREF              J:JIT
DEF      8C2D 0   M:UC
DEF     16C00 0   VDCB
DSEC    16C00 0   F:LINK
DEF     16C28 0   M:DO
DEF     16C5C 0   F:LINKIN
DEF     16C90 0   M:GO
DEF     16CC4 0   M:LO
DEF     16CF8 0   M:C
DEF     16C2C 0   M:LL
DEF      A400 0   VLC
UDEF     A400 0   DATA ORG
DSEC     A52E 0   LINK
UDEF     A730 0   DATA MAX(TCB ORG)
DEF     17000 0   VPP
DSEC    17000 0   PLSECT
UDEF    17000 0   PURE PROC ORG
DSEC    17044 0   PPLINK
UDEF    17DD9 0   PURE PROC MAX
DEF     1C000 0   VDP
```

Figure 14. Sample Load Map Printout for the Link Processor

For the Load processor, the load map for each segment starts on a new page. The map consists of a header, an allocation summary for this segment, and a series of lists of external definitions and control sections. Separate lists of PREFs, SREFs, DDEFs, and ADEFs (absolute DEFs) are generated only if such items exist in this segment. Then the relocatable DEFs (i.e., an external DEF whose value is an address as opposed to a constant) and control sections are listed. If (MAP) or (MAP,NAME) was specified on the LOAD card, the control sections (and the first DEF in each section) are listed first, then the relocatable DEFs are generated in alphanumeric order. Library DEFs will not be listed unless LDEF and/or UDEF are specified on the command in conjunction with the MAP option. For the (MAP,VALUE) option, the DEFs and control sections are listed in the order of increasing value, with the information for each control section serving as a header for the DEFs within that section.

Control sections are presented in the format

value type prot. type

where

value specifies the hexadecimal address of the beginning of the section.

type specifies LDCB (loader-built DCB), DSECT, or CSECT.

prot. type specifies 0 (Data), 1 (Procedure), or 2 (Static).

For DEFs, the format is

value byte disp. name

where

value specifies the hexadecimal value of the DEF.

byte disp. specifies the byte displacement (0, 1, 2, or 3) for relocatable DEFs only.

name specifies the symbolic name of the DEF.

All addresses (and control section sizes) are expressed in word resolution.

ACCOUNTING

A comprehensive accounting summary is generated at the end of each job. This summary includes both a detailed list of facility usage and an item called "Charge Units", which is a weighted sum of the other accounting variables, (e.g., total CPU Time, Cards Read, etc.). The weighting for each accounting variable are installation dependent and can be set or modified by the installation manager. I/O wait operations are not charged to the user, nor are they accrued as part of the time specified on a LIMIT control command. Table 28 shows the accounting summary for batch jobs. Items for which the value is zero are not printed in the summary.

```
LOAD (LMN,EXAMP),(EF,(ROM)),(UNSAT,(C8908314)),(MAP)①,(LDEF)②,(UDEF)③,(NOSYSLIB),(ABS),(SL,F)

* * ALLOCATION SUMMARY * *

  PROTECTION      LOCATION   PAGES

DATA (00)          A000       1
PROCEDURE (01)     A400       1
DCB (10)           A200       1
④ SEV. LEV.           7       1
                        .
                        .
                        .

************************************ SIGMA 5/6/7/9  LOAD MODULE MAP ************************************
************************************ ACN= F5608307   ROOT  START= A06A ************************************
************************************  LMN= EXAMP             BIAS= A000  ************************************
************************************  SGN= EXAMP             SIZE=000.6K  ************************************
                                + =DBLE DEF   - =LIB DEF   * =UNUSED DEF

 ******** PROTECTION TYPES:  00 DATA        01 PROCEDURE   10 STATIC

                           SEGHI-0  A08D  SEGHI-1  A417  SEGHI-2  A3FF ⑤
                           SEGLO-0  A000  SEGLO-1  A400  SEGLO-2  A200

                           00 SIZE=   8E  01 SIZE=   18  10 SIZE=  200

 ********* PREF - PRIMARY REFERENCES NOT LOCATED *********
                      FINISH              MODE                TYPE

 ********* SREF - SECONDARY REFERENCES NOT LOADED *********
                      OPTION

 ********* ADEF - ABSOLUTE SYMBOL VALUES          *********
                    7 *ROMSIZE         3E8 *SEGSIZE          2 *TREEDIS
                      ⑥

 ********  SECT - PROGRAM SECTIONS MAP             *******                                    -   34 WDS
⑦A206 LDCB  2   A206 0  M:DO                                                                -    6 WDS
 A064 DSECT 0   A064 0  DUM                                                                 -   14 WDS
 A06A CSECT 0   A06A 0 *START                                                               -   10 WDS
 A07E CSECT 0   A07E 0 -D                                                                   -    A WDS
 A40C CSECT 1

 ********  RELOCATABLE DEFINITIONS SORTED BY NAME  *******
               A074 0  B           A07E 0 -D           A064 0  DUM         A089 0 -G
               A206 0  M:DO        8C3C 0 -M:UC        8CF6 0 -M:XX        A06A 0 *START
               A07C 0  Y58                ⑥
```

Notes

1. Specification of "MAP" implies "MAP, NAME"; i.e., the DEFs are sorted by symbolic name.
2. The used library DEFs for the load module are listed on the map.
3. The unusued library DEFs defined in the load module are listed on the map.
4. The loader assigned a severity level of 7 to this load module because it contained unsatisfied PREFs.
5. The protection type boundaries and sizes are expressed in word resolution.
6. The external definitions D, G, M:UC, and M:XX are defined in library load modules. START, ROMSIZE, SEGSIZE, and TREEDIS are unused DEFs (i.e., the loader did not encounter an SREF or PREF corresponding to any of these names).
7. M:DO (a loader-built DCB) has protection type 10 (2_{10}) and begins at X'A206'.

Figure 15. Sample Load Map Printout for the Load Processor

Table 28. Accounting Printout for Batch Jobs

Printed Format			Explanation
(Time and Date)			
ELAPSED JOB TIME		hh:mm:ss	Clock time in hours, minutes, and seconds for job or terminal session.
PARTITION NUMBER		xx	Partition number in which the job ran.
TOTAL CPU TIME		x.xxxx	Sum of all execution time (in minutes).
	PROCESSOR EXECUTION TIME	x.xxxx	Shared processor execution time (e.g., FORTRAN)(in minutes).
	PROCESSOR SERVICE TIME	x.xxxx	Monitor time for CALs issued by shared processors (in minutes).
	USER EXECUTION TIME	x.xxxx	User program execution (in minutes).
	USER SERVICE TIME	x.xxxx	Monitor time for user issued CALs (in minutes).
CARDS:	CARDS READ	xxxx	Number of cards read.
	CARDS PUNCHED	xxxx	Number of cards punched.
PAGES:	PROCESSOR PAGES	xxxx	Number of pages printed by shared processors.
	USER PAGES	xxxx	Number of pages printed by user programs.
	DIAGNOSTIC PAGES	xxxx	Number of pages printed through M:DO.
TAPES:	DRIVES ALLOCATED	xx	Number of tape drives allocated.
	TAPES MOUNTED	xx	Number of tapes mounted.
PACKS:	SPINDLES ALLOCATED	xx	Number of disk spindles allocated.
	PACKS MOUNTED	xx	Number of disk packs mounted.
CORE:	PEAK CORE (PAGES)	xxx	Maximum number of core pages used at any one time.
	PAGE. MINUTES	xxxxxx	Amount of core time used.
I/O:	OPERATIONS	xxxxx	Number of physical I/O actions except terminal and swap I/O.
	CALS	xxxxxx	Number of CAL, 1 operations.
FILE SPACE			
PEAK RAD TEMPORARY		xxxx	Peak value of temporary RAD granules used.
NET RAD PERMANENT		xxxx	Net change in accumulated RAD storage (in granules).
AVAILABLE RAD PERMANENT		xxxx	Amount of RAD space available for permanent storage (in granules).

Table 28. Accounting Printout for Batch Jobs (cont.)

Printed Format		Explanation
PEAK DISK TEMPORARY	xxxx	Peak value of temporary public disk pack granules used.
NET DISK PERMANENT	xxxx	Net changes in accumulated public disk pack storage (in granules).
AVAILABLE DISK PERMANENT	xxxx	Amount of public disk pack space available for permanent storage (in granules).
NUMBER OF SWAPS	xxxx	Number of times user was swapped.
RESOURCES ALLOCATED		
CO=xxx 9T=xxx 7T=xxx (etc.)		Values for resources allocated.
CHARGE UNITS	xxxxxxxx	Total charge units.

PROGRAM DEBUGGING AIDS

INTRODUCTION

Batch program errors are reported via either a default mechanism or through explicit dump and snap commands supplied by the user in his JCL deck or internally within his program. These debug commands are described in this chapter.

Errors occurring during the execution of a batch user program are reported to the user via the error codes and subcodes detailed in Appendix B. If the user does not choose to handle these errors himself (i.e., does not use the debug commands), the monitor aborts the job and interprets the codes for him by accessing the error message file for an appropriate message. This message is printed together with the location of the error, the PSD, the general registers, and, if the error is DCB-related, the contents of the DCB. For example:

4000 CAN'T READ AN OUTPUT FILE

AT C065

ON DCB M:EO

WHICH CONTAINS

(contents of DCB)

USER's PROGRAM STATUS DOUBLEWORD

(contents of PSD)

USER's GENERAL REGISTERS

(contents of registers)

The memory dumps performed by debug commands may be either conditional (dependent on whether errors occurred during program execution) or unconditional. All dumps are taken before the DCBs are closed and are output through the user's M:DO DCB. If M:DO does not exist or cannot be opened, a postmortem dump is output to the LP device; the user's program is aborted if there is no M:DO DCB for a snapshot dump.

Postmortem dump (PMD) and snapshot (SNAP) control commands may only be used following a RUN control command for the program to which they apply. They are not allowed for load modules created by the LINK command. The PMD and SNAP control commands may appear in any sequence.

The dump routines list the current Program Status Doubleword (PSD) and registers, followed by the requested memory areas. In the case of PMD and PMDI, any pages gotten by M:GP and M:GCP are also listed. Figure 16 shows the format of a dump printout.

Only one page of storage is reserved for debug control command Functional Parameter Tables (FPTs) generated by the program loaders. If this limit is exceeded, the following error message is listed:

TOO MANY DEBUG COMMANDS

A location appearing in a debug control command may be listed as a hexadecimal address, an external definition, or an object module name. Addresses relative to external definitions consist of the label of the definition optionally followed by a signed hexadecimal addend value, for example: LOC+B.

All dynamic debug commands (i.e., SNAP, SNAPC, IF, AND, OR, COUNT) cause the specified instruction to be replaced by a monitor call. The replaced instruction will be executed after the specified action takes place.

POSTMORTEM DUMPS

A postmortem dump control command requests the monitor to dump a selected area of memory. Such a dump is termed "postmortem" because it is performed after the program has been executed or terminated due to error (i.e., "errored"). If an error is detected during program execution, the monitor lists an appropriate error message on the LL device, in addition to listing the dump output on the DO device.

Any number of separate program areas may be specified for a program, in one or more postmortem dump commands. If no program areas are specified, all areas having a protection code of 00 will be dumped. If a single job includes several programs to be loaded and executed separately, each such program may have one or more associated postmortem dump control commands. The dump printout is in hexadecimal and BCD.

Postmortem dumps are requested by the PMD control commands, PMD, PMDE, and PMDI.

USERS PROGRAM STATUS DOUBLEWORD

xxxxxxxx xxxxxxxx *eeeeee*

where

x equals the hexadecimal representation of the PSD.

e equals the EBCDIC representation, if printable.

USERS GENERAL REGISTERS

xxxxxxxx xxxxxxxx ... xxxxxxxx *eeeeee*

xxxxxxxx xxxxxxxx ... xxxxxxxx *eeeeee*

where

x equals the hexadecimal representation (eight words per line) of the general registers.

e equals the EBCDIC representation, if printable.

THE FOLLOWING SEGMENTS ARE PRESENTLY IN CORE

List of the segments in core if the program is an overlaid program.

ALL USERS DCBS FOLLOW

List of user's DCBs.

SYSTEM CFU FOR ABOVE DCB

List of the CFU if the PMDE control command was used and the DCB is open to a file.

SYSTEM INDEX BUFFER FOR ABOVE DCB

List of the index buffer if the PMDE control command was used and the DCB has an index buffer assigned.

SYSTEM BLOCKING BUFFER FOR ABOVE DCB

List of the blocking buffer if the PMDE control command was used and the DCB has a blocking buffer assigned.

USER-SPECIFIED DUMP LIMITS FOLLOW

List of any user-specified dump limits or protection types.

USERS DYNAMIC PAGES FOLLOW

List of any presently allocated pages obtained by an M:GP procedure call.

USERS COMMON DYNAMIC PAGES FOLLOW

List of any presently allocated pages obtained by an M:GCP procedure call.

Figure 16. Format of a Dump Printout

PMD The PMD control commands cause a specified dump to occur. These control commands must follow the RUN control command for the program to which they apply.

The form of the PMD control command is

```
!{PMD  }
 {PMDE } [,segment] [(from, to),...] [,(pp)]
 {PMDI }
```

where

PMD causes the PSD, registers, the segment names presently in core (if an overlaid program), all DCBs, the specified areas, user's dynamic pages, and user's common dynamic pages to be dumped.

PMDE causes (in addition to the information dumped by PMD) the system Job Information Table (JIT) and for each DCB that is open to a file, the system Control File Unit (CFU), and the system FPOOLs to be dumped.

Both PMD and PMDE cause a specified dump to occur only if an error occurred during program execution or if the program has returned control to the monitor through an M:ERR or M:XXX procedure.

PMDI causes a specified dump to occur whether or not any errors have been detected. All areas to be dumped must lie within the designated overlay segment.

segment specifies the name of an overlay segment containing the areas to be dumped. If the segment name is omitted, the specified area currently in core will be dumped. To dump only the root, the name of the root segment is specified. If the specified segment is not in core, no dump will occur.

from, to specifies the location of the beginning (from) and end (to) of an area to be dumped. Either "from" or "to" may be expressed as a relative hexadecimal location (i.e., an external definition followed by an optional hexadecimal addend value) or a positive (preceded by a "+" character) absolute hexadecimal address.

pp specifies the memory-access class that is to be dumped.

where

00 = Read, write, or access from.

01 = Read or access from.

10 = Read only.

and where

Read = Program can obtain information from.

Write = Program can store information into.

Access = The computer can execute instructions stored in the protected area.

Examples:

```
!PMD
```

This example specifies that the data areas of the program currently in core are to be dumped. It is equivalent to !PMD (00).

```
!PMD, UNO (10)
```

This example specifies that all areas of overlay segment UNO that have a memory-access code of 10 (i.e., memory access 10) are to be dumped.

```
!PMD, EIN (LOC1+5, LOC2-A)
```

This example specifies that the area to be dumped is that part of overlay segment EIN beginning five words higher than location LOC1 and ending ten words lower than LOC2.

```
!PMD (10), (00)
```

This example specifies that all areas of the program root segment that have a memory-access code of 10 or 00 are to be dumped.

SNAPSHOT DUMPS

A memory snapshot dump provides an instantaneous "picture" of program conditions existing at a particular point in time during program execution. Such a dump can be obtained just prior to the execution of any specified instruction in a user's program. Six control commands and six equivalent procedures are provided for specifying the circumstances that will produce a snapshot dump and the portion of memory that the dump will include.

Differences between using a control command and an equivalent procedure are: (1) there is no limit to the number of program areas that can be dumped by a single control command, while the procedures can dump only one; (2) the segment and loc fields are used only for control commands;

and (3) the loc specified in the control commands must contain an executable instruction that is not altered or replaced during program execution. This final restruction is necessary because the monitor replaces the contents of the location with a trap to the monitor's snapshot dump routine. The initiating location may contain any type of executable instruction (e.g., a BAL, LAD, MSP, FDS, or EXU instruction). In addition, the procedures may not be the target of an EXU instruction because word 6 of the FPT contains a branch instruction to the location following the procedure CAL. For example, following execution of the instruction at EP, control would be returned to the user at TAB+1 instead of at EP+1.

```
EP     EXU   TAB
       :
       :
TAB    M:SNAP 'SNAP1', (HERE+14, THERE-1)
```

If the M:DO DCB has not been REFed or defined prior to an M:SNAP or M:SNAPC procedure reference line, it will be REFed by the procedure.

SNAP The SNAP control command (and M:SNAP, the corresponding procedure) requests the monitor to take an unconditional memory snapshot.

The form of the SNAP control command is

```
!SNAP[,segment]loc,com[,(from[,to])]...
```

where

segment specifies the name of an overlay segment containing the location initiating the dump and also the areas to be dumped. If omitted, the root is assumed.

loc specifies the location at which the dump is to be initiated. That is, the specified dump is to occur just prior to the execution of the instruction at "loc". Note that "loc" (and either "from" or "to", below) may be expressed as a relative hexadecimal location (i.e., an external definition followed by an optional hexadecimal addend value) or a positive absolute hexadecimal address preceded by a + character.

com specifies a string of up to eight alphanumeric comment characters that are to be printed with the dump output. Note that at least one such comment character must be specified in the command.

from specifies the location of the beginning of an area to be dumped.

to specifies the location of the end (i.e., highest core location) of an area to be dumped.

The form of the M:SNAP procedure is

```
M:SNAP 'com' , (from[,to])[,NREGS]
```

where com, from and to are as specified above in the SNAP control command.

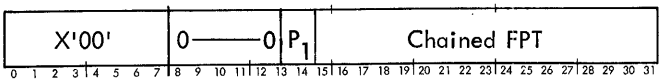
NREGS, if specified, suppresses the printing of the PSD and the registers in the map; this feature is only available with the M:SNAP procedure.

Calls generated by the M:SNAP procedure have the form

```
CAL1,3   fpt
```

where fpt points to word 0 of the FPT shown below.

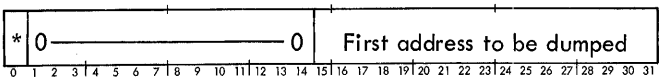
word 0



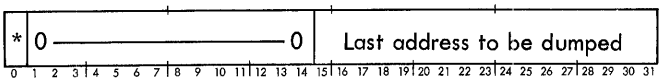
Chained FPT is the address of an FPT for some other CAL1,3 that is to be executed immediately following the current CAL1,3. If it is zero, FPT is not chained.

P_1 specifies whether the snap of the user's PSD and registers is to be suppressed; if set, they are not printed.

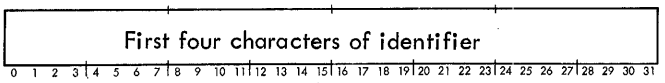
word 1



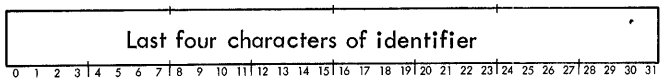
word 2



word 3

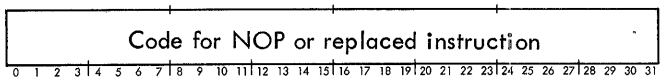


word 4



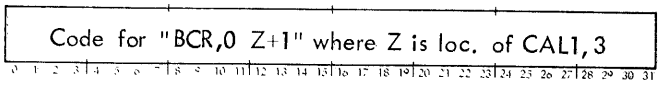
This is an optional 1-8 characters the user wants printed with his dump, if it occurs.

word 5



If the CAL1,3 is to be executed as the result of a debug control command (SNAP, etc.), this is the instruction from the user's program that was replaced by the CAL1,3.

word 6



Examples:

```
!SNAP TAB, SNAP1, (HERE+14, THERE-1)
```

This example specifies that the area beginning twenty word locations higher (in address) than location HERE and ending one word location lower than THERE is to be dumped just prior to the execution of the instruction at location TAB. The message "SNAP1" is to be printed with the dump. Since segment is not specified, the root is assumed.

```
M:SNAP 'SNAP1',(HERE+14,THERE-1)
```

The call generated by this procedure, if located at TAB in the user's program, would produce the same dump as the SNAP control card example above.

SNAPC The SNAPC control command (and M:SNAPC, the corresponding procedure) requests the monitor to take a conditional memory snapshot.

The form of the SNAPC control command is

```
!SNAPC[,segment] flag,specification
```

where

segment specifies the name of an overlay segment (see SNAP).

flag specifies the name of the test identifier. It may consist of any character string from one to eight characters in length. Since the monitor does not associate the flag with the user's program, no confusion with program symbols can arise. The normal state of the flag bit associated with a flag (in a table established and maintained by the monitor) is the set state. It is set and reset by means of the IF, AND, OR, and COUNT control commands. Unless the flag bit is set, the specified dump cannot take place.

specification must include both of the required parameters of a SNAP control command (i.e., initiating location and comment string) and may also include any or all of the optional specifications (see SNAP).

The form of the M:SNAPC procedure is

```
M:SNAPC flag, 'com', (from,to) [,NREGS]
```

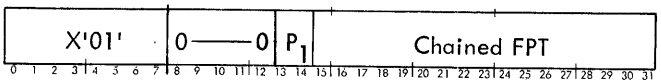
where com, from,to, and NREGS are specified above in the SNAP control command, and flag is as specified above in the SNAPC control command, except that the test identifier must be the name of a location within the user's program.

Calls generated by the M:SNAPC procedure have the form

```
CAL1,3   fpt
```

where fpt points to word 0 of the FPT shown below.

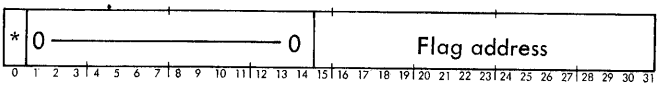
word 0



where P_1 has the same meaning as in M:SNAP above.

words 1 through 6 of the FPT have the same form as shown above for the M:SNAP procedure.

word 7



Examples:

```
!SNAPC, NIM AT5, LUP+1, TP33
```

This example specifies that, if flag AT5 is in the set state just prior to the execution of the instruction whose memory address is one (word) greater than that of LUP (in overlay segment NIM), then all general registers and the PSD are to be dumped. If the dump occurs, the message "TP33" is printed with the dump.

```
M:SNAPC AT5,'TP33'
```

The call generated by this procedure, if located at LUP in the user's program, would produce the same dump as the SNAPC control card example above.

IF The IF control command (and M:IF, the corresponding procedure) may be used in conjunction with a conditional snapshot command (see SNAPC). It requests the monitor to make a specified test at a designated location and, if the test condition is found to be true, to set the flag bit associated with the conditional snapshot. If the test condition is found to be false, the flag bit is reset by the monitor.

Since the IF control command may be used in conjunction with other dynamic debug commands (see AND and OR), the relative sequence of such commands may affect the performance or inhibition of the dump. It is the user's responsibility to sequence such commands in the order dictated by the logical requirements of the conditional dump.

Note that the instruction at the test location specified in a dynamic debug command must be executed prior to the execution of the instruction at the location that initiates the dump.

The IF control command is of the form

```
!IF[,segment] flag,loc,(l1,x1[,b1],r,l2,x2[,b2])
```

where

segment specifies the name of an overlay segment.

flag specifies the name of the test identifier (see SNAPC).

loc specifies the absolute or relative (external definition [±addend]) hexadecimal location at which the test is to take place. That is, the specified test is to occur just prior to the execution of the instruction at "loc".

l_1 and l_2 specify locations that are to be compared as specified by "r" (see r option). They may be either absolute or relative and may be indirect ($*l_i$).

x_1 and x_2 specify index registers to be used to modify the addresses specified by l_1 and l_2, respectively. A zero may be used to specify that indexing is not to be used.

b_1 and b_2 specify the number of bytes to be compared. The permissible values and their meanings are

Value	Meaning
1	Byte 0
2	Halfword 0
4	Fullword
8	Doubleword

The values of b_1 and b_2 are normally the same but may be different. If omitted, the value 4 (i.e., fullword) is assumed.

specifies the type of comparison to be made. The permissible values and their meanings are

Value	Meaning
GT	Greater than
LT	Less than
EQ	Equal to
GE	Greater than or equal to
LE	Less than or equal to
NE	Not equal

The form of the M:IF procedure is

M:IF flag, (l_1,x_1,b_1,r,l_2,x_2,b_2)

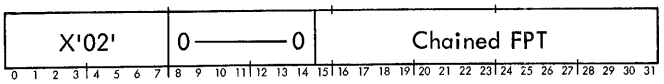
where flag, l, x, b, and r are as specified above in the IF control command.

Calls generated by the M:IF procedure have the form

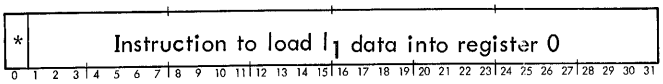
```
CAL1,3   fpt
```

where fpt points to word 0 of the FPT shown below.

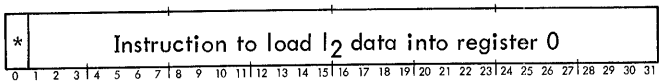
word 0



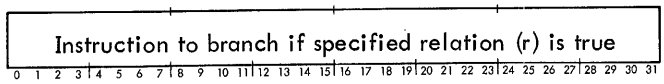
word 1



word 2

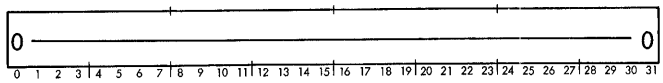


word 3

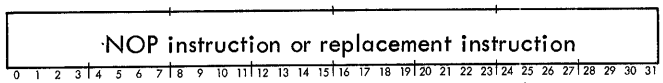


r	Instruction	
GT	BCS, 1	0
LT	BCS, 2	0
EQ	BCR, 3	0
GE	BCR, 2	0
LE	BCR, 1	0
NE	BCS, 3	0

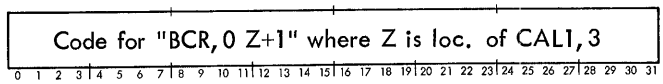
word 4



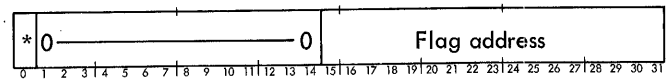
word 5



word 6



word 7



Examples:

```
!IF TAU, ETA+1, (RHO+A, 4, EQ, PSI-5, 5)
```

This example specifies that two words in core storage are to be tested to determine whether they are equal (i.e., identical). One of these two words has an address that is ten word locations greater than that of external definition RHO, plus the contents of index register 4. The other word to be compared has an address that is five word locations less than that of external definition PSI, plus the contents of index register 5. The example also specifies that the test is to occur just prior to the execution of the instruction that is one word location higher than external definition ETA. If the specified test gives a true result, the flag named TAU is to be set; otherwise, the flag is to be reset.

M:IF TAU, (RHO+A, 4, EQ, PSI-5, 5)

The call generated by this procedure, located at ETA+1, would result in the same test as the IF control command above.

AND The AND control command (and M:AND, the corresponding procedure) may be used in conjunction with a conditional snapshot command. It requests the monitor to make a specified test at a designated location, but only if the flag bit for the associated snapshot is in the set state when the test is to be made. If the test condition is found to be true, the flag bit remains set; otherwise, the flag bit is reset. If the flag bit is in the reset state when the test is to be made, the test is not performed and, unless the flag bit is set as a result of some subsequent command, the associated snapshot does not occur.

The AND control command has the form

```
!AND[,segment] specification
```

where

segment specifies the name of an overlay segment.

specification (see IF control command).

The M:AND procedure has the form

M:AND specification

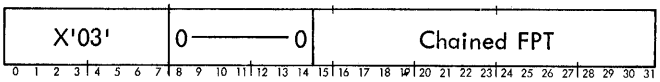
where specification is the same as M:IF.

Calls generated by the M:AND procedure have the form

CAL1,3 fpt

where fpt points to word 0 of the FPT shown below.

word 0



Words 1 through 7 of the FPT have the same form as shown above for the M:IF procedure.

OR The OR control command (and M:OR, the corresponding procedure) may be used in conjunction with a conditional snapshot command. It requests the monitor to make a specified test at a designated location, but only if the flag bit for the associated snapshot is in the reset state when the test is to be made. If the test condition is found to be true, the flag bit is set; otherwise, the flag bit remains reset and, unless the flag bit is set as a result of some subsequent command, the associated snapshot does not occur.

The OR control command has the form

```
!OR[,segment] specification
```

where

segment specifies the name of an overlay segment.

specification (see IF control command).

The form of the M:OR procedure is

M:OR specification

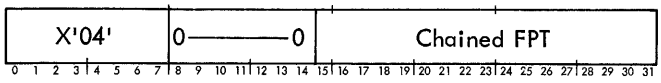
where specification is the same as M:IF.

Calls generated by the M:OR procedure have the form

CAL1,3 fpt

where fpt points to word 0 of the FPT shown below.

word 0



Words 1 through 7 of the FPT have the same form as shown above for the M:IF procedure.

COUNT The COUNT control command (and M:COUNT, the corresponding procedure) allows the user to specify an iteration range (and steps within that range) in which a designated test identifier (i.e., a flag for a snapshot dump)

will be set. A separate internal counter is established by the monitor for each COUNT command and the count is incremented by one whenever (i.e., just before) an instruction at a specified location is executed. The iteration count is then tested to determine whether the flag for the specified dump will be set or reset. COUNT operates independently of any OR, IF, or AND commands.

The flag for the designated dump will be set if the current count is within the range of the specified start and end count, and if the quotient "(count-start)/step" is an integer. Otherwise, the flag will be reset.

The COUNT control command has the form

```
!COUNT[,segment] flag, loc, start, end, step
```

where

segment specifies the name of an overlay segment.

flag specifies the name of the test identifier (see SNAPC).

loc specifies the absolute or relative (external definition [±addend]) hexadecimal location at which the count is to be incremented by one.

start specifies the decimal count at which the testing of the count is to begin. When the count equals "start", the flag is set (even if "start" is equal to zero).

end specifies the decimal count at which the incrementing of the count is to cease. A maximum value of 2,147,483,647 may be specified.

step specifies the decimal count increment that determines the intervals (within the range designated by "start" and "end") at which the flag can be set so that conditional dumps can be taken. Both "step" and "start" must be less than "end".

The format of the M:COUNT procedure is

```
M:COUNT flag, start, end, step
```

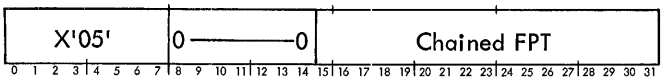
where flag, start, end and step are as specified above in the COUNT control command.

Calls generated by the M:COUNT procedure have the form

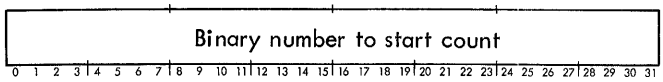
```
CAL1,3   fpt
```

where fpt points to word 0 of the FPT shown below.

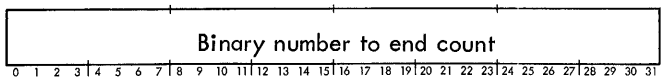
word 0



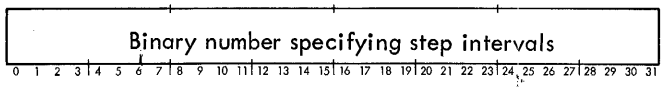
word 1



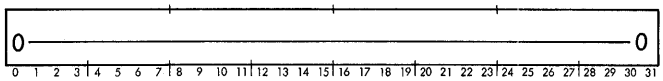
word 2



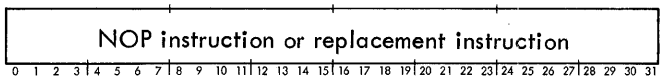
word 3



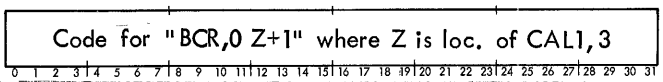
word 4



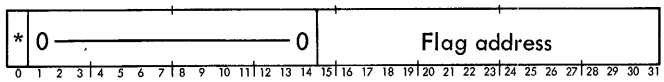
word 5



word 6



word 7



Examples:

```
!COUNT FLG26, HERE+6, 1, 10, 1
```

This example specifies that the name of the flag to be set is FLG26, the count is to be incremented by one just prior to executing the instruction located six word locations higher than external definition HERE, the count range within which the count is to be tested (to determine if "count-start/step" is an integer) is from 1 to 10, and the flag is to be set whenever the count is incremented by one. Note that, in this example, the flag is set when the count reaches 1.

```
M:COUNT FLG26, 1, 10, 1
```

The call generated by this procedure, if located at HERE+6, would produce the same results as the COUNT control command in the prior example.

DEBUG ERROR MESSAGES

Table 29 lists the self-explanatory error messages that may occur when debug commands are used.

Table 29. Debug Error Messages

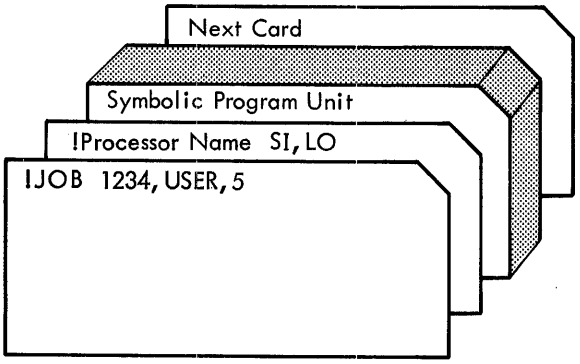
Key	Message
040358	BAD DEBUG LOCATION NAME –
040359	BAD IF/AND/OR TEST NAME –
04035A	BAD SNAP FROM/TO NAME –
04035B	BAD PMD FROM/TO NAME –
04035C	BAD MODIFY NAME –
04035E	TOO MANY DEBUG COMMANDS
04035F	INVALID DEBUG RECORD TYPE
040360	RUNNER RECEIVED ABOVE I/O ERROR READING LOAD MODULE WITH KEY =
040362	RUNNER RECEIVED ABOVE I/O ERROR READING DEBUG FILE
040363	BAD START ADDRESS NAME –
040364	MODIFY LOCATION NOT WITHIN PROGRAM –
040365	START ADDRESS NOT WITHIN PROGRAM –
040366	BAD SEGMENT NAME IN DEBUG COMMAND –
040367	CAN'T GET PAGE AFTER PURE-PROCEDURE FOR DEBUG AND CLOBBER TABLE
040368	PMD FROM/TO ADDRESS NOT WITHIN USER AREA –
040369	DEBUG LOCATION NOT WITHIN PROGRAM –
04036A	SNAP FROM/TO ADDRESS NOT WITHIN USER AREA –
04036C	PMDS AND DEBUGS NOT ALLOWED FOR LOAD MODULES BUILT BY LINK
04036D	NOT ENOUGH CORE TO PROCESS DEBUG COMMANDS
04036E	MALFORMED LOAD MODULE HEAD OR TREE RECORD

8. PREPARING THE PROGRAM DECK

INTRODUCTION

The following examples show some of the ways that program decks may be prepared for monitor operation. Standard system assignments are normally assumed.

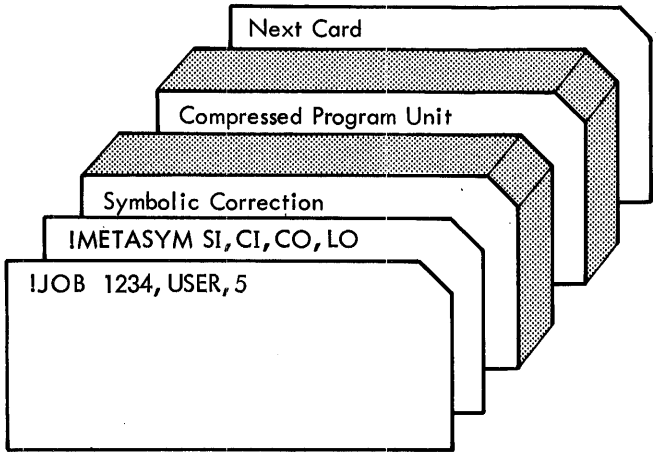
SYMBOLIC DECK TO PROGRAM LISTING



As indicated in the example, a program listing can be obtained even though no binary (object) output is requested.

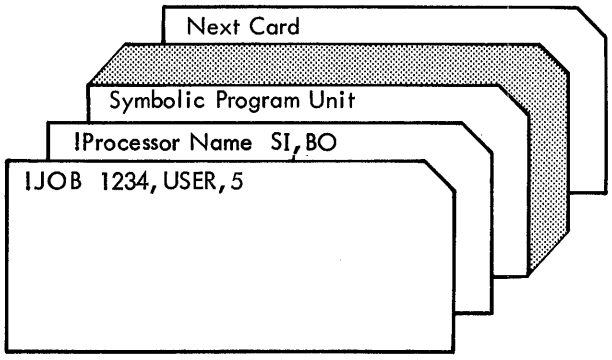
In this and subsequent examples of program decks, the "next card" could be any appropriate control command card such as JOB or FIN.

COMPRESSED DECK UPDATE



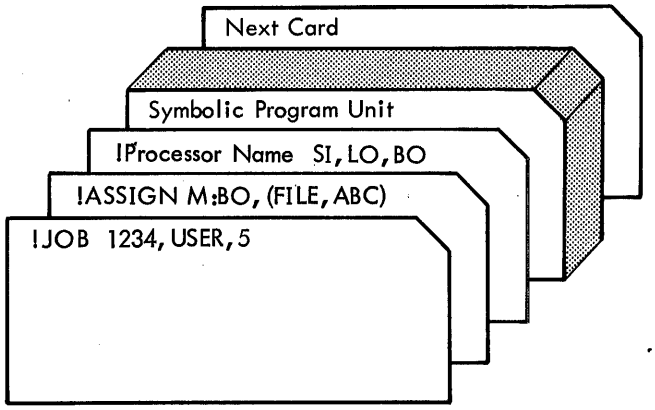
This example shows how a compressed symbolic deck can be updated using uncompressed symbolic corrections.

SYMBOLIC DECK TO BINARY DECK



This example shows how a binary deck can be output using an uncompressed symbolic deck as input. A deck produced in this way will be in standard Sigma object language and may be loaded from a card reader, subsequently, as a binary object module.

SYMBOLIC DECK TO BINARY FILE ON DISK

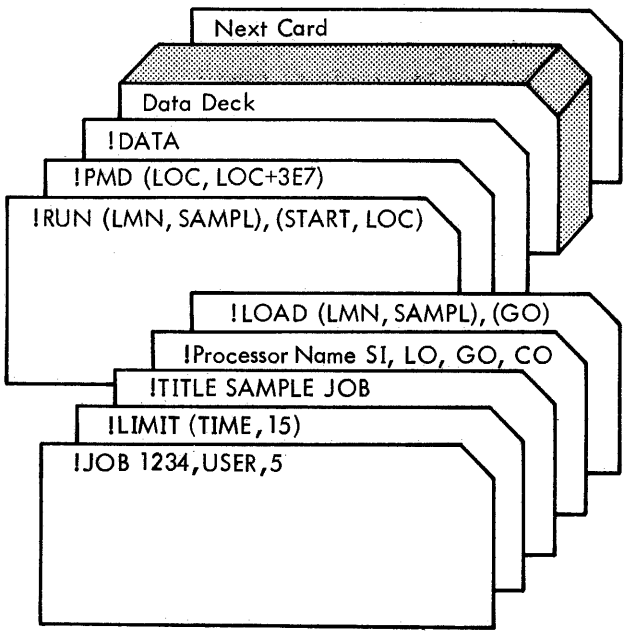


This example shows how a binary (object) file on disk can be formed from the output of a processor. The disk file thus formed may be accessed thereafter by means of the file name ABC. The example specifies that a program listing is to be output also; but this is optional.

The file becomes a permanent user's file. That is, it is placed in the disk area allocated to permanent storage, since all processor output is treated by the monitor as a SAVE file (see ASSIGN control command).

PROCESS, LOAD, AND EXECUTE

This example shows how a program can be input in symbolic form, then processed, loaded, and executed under monitor control. If assembly and run-time are in excess of 15 minutes, the job is aborted. The title "SAMPLE JOB" is printed at the top of each page of the assembly listing.

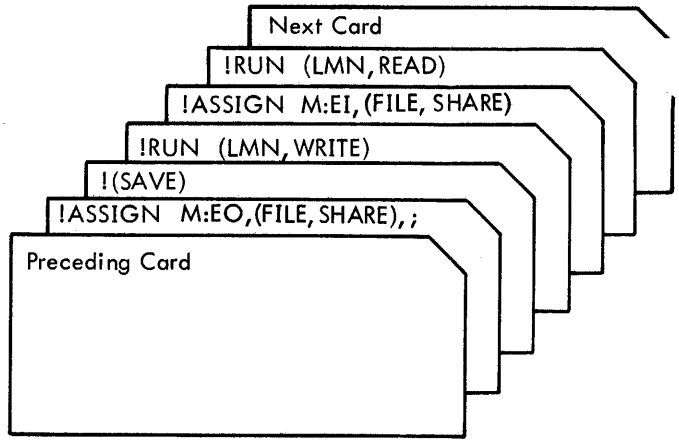


The processor accepts symbolic input from the SI device (in this example, a card reader) and produces compressed symbolic output as well as binary object code and a program listing on the system devices to which such functions are currently assigned (by standard system assignments).

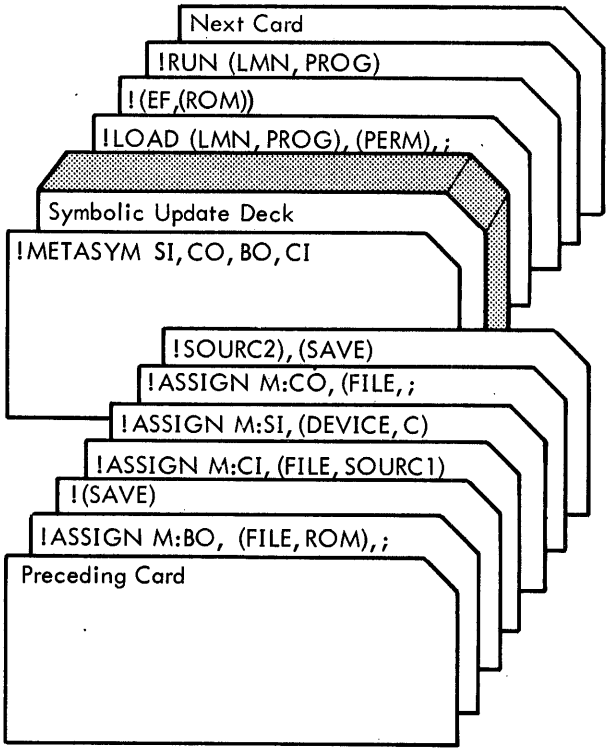
The name SAMPL is associated with the load module, and no load map is output. After being loaded into core storage, the program is executed beginning with the instruction at symbolic location LOC. If an error is detected during execution of the user's program, a post-mortem dump is taken of the 1000 words of the program starting at location LOC.

CREATE FILE FOR USE BY ANOTHER PROGRAM

This example shows how an output file (SHARE) can be created by one executing program (WRITE) in a user's job and then used by another executing program (READ) in the same job.



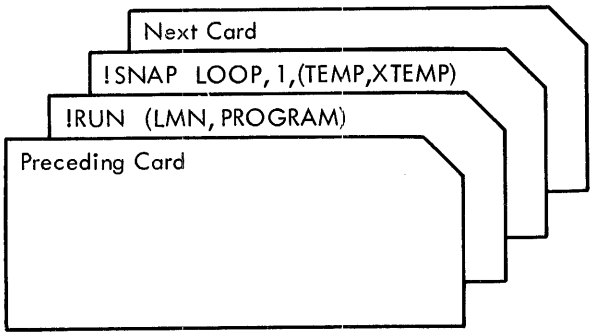
UPDATE FILE, OBJECT MODULE, AND LOAD MODULE OF USER'S PROGRAM



This example shows how a user can accomplish the following:

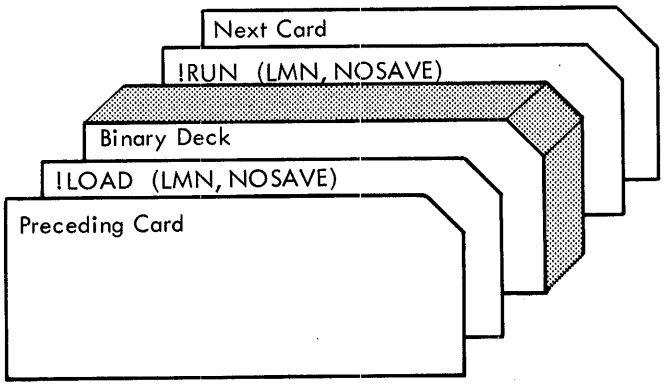
1. Update a symbolic file (SOURC1 becoming SOURC2) in the user's account.
2. Create a new relocatable object module (ROM) and place it in the user's account.
3. Execute the program so formed.

EXECUTE PROGRAM FROM USER'S ACCOUNT, USING DEBUG FEATURE



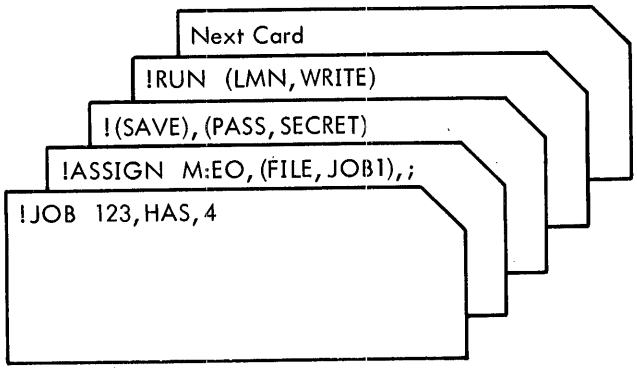
This example shows how a program (PROGRAM) from the user's account can be executed, and a snapshot dump of a specified program area (from TEMP to XTEMP) taken.

CREATE AND EXECUTE A TEMPORARY PROGRAM



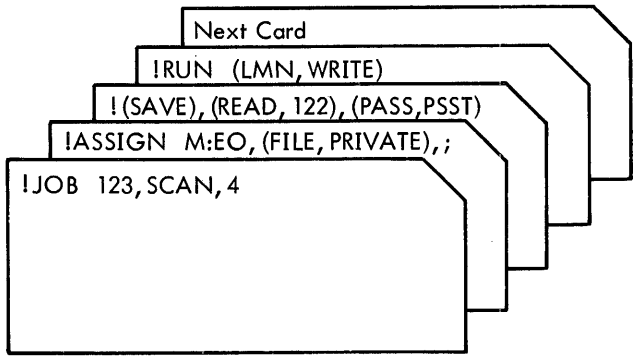
This example shows one way in which a user can create a temporary load module (NOSAVE) that can be executed but is released at the end of the job, because the PERM option is absent from the LOAD control card.

CREATE A FILE WITH PASSWORD



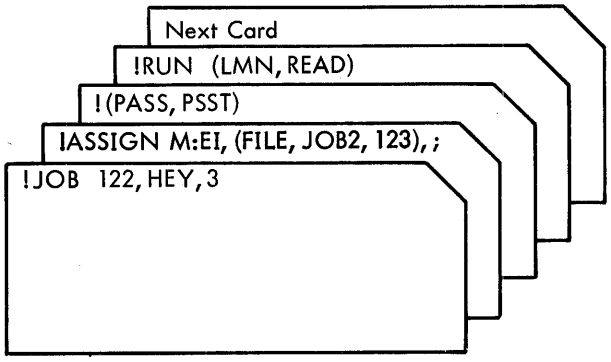
This example illustrates how a user can create a file (JOB1) having a password (SECRET). It assumes that a user's program (WRITE) capable of writing into the file (e.g., via an M:WRITE procedure call referencing the M:EO DCB) has been loaded previously.

CREATE A FILE HAVING PRIVILEGED READ ACCESS



Here, a user creates a file (PRIVATE) having a password (PSST), specifying also that other jobs with account number 122 may read the file (but may not write on it).

READ A FILE HAVING PRIVILEGED READ ACCESS



This illustrates how a user can read a file (JOB2) having a password (PSST) for which privileged read access has been established. Note that the account number associated with the file (123) must be given as well as a password.

9. PROCESSORS

INTRODUCTION

A variety of processors are available to batch users. Four of these processors are described in this chapter. Others are described in their own reference manuals (see Chapter 1). All are called, if available in a particular installation, by the processor control commands described below.

PROCESSOR CONTROL COMMANDS

Processor control commands indicate to the monitor that control is to be transferred to a specified processor. They specify the types of input to be accepted and the types of output to be produced.

There are no restrictions as to how many or what kind of processors may be added to the operating system. Processors may be created, updated or deleted under normal batch operations. System processors (programs) must be inserted in the operating system under the monitor's :SYS system account number.

All processors in the :SYS account may be called by !name, where name is the name of a processor or is the name of the load module specified in a processor control command. User programs (processors) are called by !RUN (LMN,name).

The form of a monitor (:SYS) account control command is

```
!name [options]
```

where

name is the name of a processor in the :SYS account.

options is a listing of input and output options for the processor.

For an explanation of the options available for a specific processor not described in this chapter, see the separate manual for the particular processor.

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.

The following description of PCL is oriented toward the batch user. However, descriptions of PCL command options include on-line as well as batch options.

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. 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 command that assigns M:SI to the file of commands.

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 PCL command may be continued from one card to the next by ending the continued card with a semicolon (i.e., a semicolon must be the last nonblank character.) There is no limit on the number of continued cards; 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#;
   A#B#C#D
   ```

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 30. 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.

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 2.

Table 30. 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.

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 mangetic 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 specifications. 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 31.

Table 31. PCL Organization Types

Organization Type	Description
DC	RAD storage. (See Default Disk Pack, page 184.)
DP	Disk pack storage. (See Default Disk Pack, page 184.)
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.[t] A file name consists for PCL of 1 to 31 characters,[tt] which in general may be any characters except the following PCL delimiters:

Blank . ( ) ; / ,

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 current job account.
name.account	file in specified account.
name..password	file in current job 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 contained in quotes. An ANS tape serial number may be up to six alphanumeric or blank 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 job 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 account.

Therefore, in subsequent command descriptions, the following convention is adopted. If a volume identifier 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 output 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. 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.

Default Disk Pack. 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

[t] An ANS tape file identifier consists of a name only.

[tt] 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.

is random, the absence of a reel identifier on a 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. 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 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, "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 whose volume ids 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 an ANS tape.

   AT#'123␣␣␣'#'456␣␣␣' - BT/ MYFILE

4. The same file on a private disk pack 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[t] or INTO 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 job 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 copy 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 RAD 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.

PRINT sends any waiting output for symbiont 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 tape.

REM remove designated tape or disk pack.

TABS define tab settings for tab expansion.

MOUNT causes designated tape or disk pack to be mounted.

MODE OPTION COMPATIBILITY

In the current version PCL, 7T and 9T are resource types and are no longer used as mode options. However, for

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

compatability 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 mode options, it will be treated exactly as though it had been specified as a resource type.

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

COPY source[,source...] [TO / OVER / INTO] destination

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.

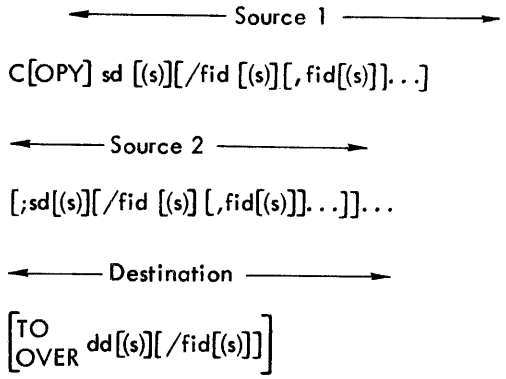
If the purpose of the COPY is to replace a RAD or disk file currently existing in the user's account directory, PCL requires that the preposition OVER be used in the command. That is, COPY TO, OVER, or INTO creates a file, but for the user's protection only COPY OVER can replace an existing file. After 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.

Note that the TO or OVER 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.

File extension may also be accomplished by using the INTO preposition in the command.

COPY COMMAND FORMAT (SPECIFIC)

The specific format of the COPY command is



where

sd represents the device portion of a source specification and may be an input device type (Table 30), 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

name[. [account]. password / . 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 30 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 it is the last nonblank character of a card.)

(s) represents specifications for data encoding: data codes (Table 32), formats (Table 33), modes

(Table 34), record sequencing (Table 35), accounts (Table 36), ANS tape options (Table 37), expiration time, 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 a destination specification and may be an output device type (Table 29), 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 line printer. The command could be coded as

```
COPY SOURCE TO LP
```

This command could also be written as

```
C SOURCE TO LP
```

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
```

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
```

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

C FT#7236

C FT#7236
```

or

```
C FT#7236;FT#7236;FT#7236 TO CP
```

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 32) describe the source or destination data types to be expected or produced for devices only.

Table 32. Data Codes

Code	Meaning
E	EBCDIC (default data code)
H	Hollerith (FORTRAN BCD conversion)

Data Formats. Data formats (Table 33) describe the source or destination record formatting to be expected or produced.

Table 33. 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 one to eight hexadecimal characters which specify the seed for data encryption for keyed and consecutive files. Separate algorithms are employed for keyed and consecutive files. A keyed file that is encrypted cannot be decrypted if its keys are stripped. Data encryption is described in Chapter 2.

Modes. Mode codes dictate the control modes for the specified files or devices. They are shown in Table 34.

Table 34. 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.
LC,UC	Translate alphabetic characters to lower (LC) or upper (UC) case.
NF	No formatting. PCL does not produce any output that is not in the input data (e.g., file name to LP or separation of files to LP/UC).

Table 34. Mode Codes – COPY Command (cont.)

Mode	Description
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 precedes 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	Specifies that the file is temporary; it is to be kept across JOB steps but released at job termination. If the second and third characters of the file name are colons (e.g., A::AFILE), the double colons are replaced by the user's sysid.

Examples:

1. Assume that file A is to be copied to labeled tape No. 4 on a dual density drive at 1600 bpi with exactly the same attributes it had on RAD storage.

```
C A TO LT#4/A (DEN(1600))
```

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)
```

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)
```

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)
```

Record Sequencing. Insertion or deletion of sequence identification for output data records is accomplished by using record sequencing specifications (Table 35). These specifications are available only as output options. All of these options are mutually exclusive.

Table 35. 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 columns73-80 from each output dota 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.

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))
```

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 A(LN(5, 5))
```

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)
```

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))
C B INTO C(LN(2,2))
```

Assignment of Accounts. A combined list (not to exceed 16 entries) of read accounts, write accounts, execute accounts, and the name of a processor under which the file is to be run may be added as attributes of the output files as shown in Table 36.

Table 36. 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.

Table 36. Account Options – COPY Command (cont.)

Code	Description
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 UN is not specified, the file is presumed to be a load module and may be executed by any user running under an EXecute account but not under Delta.

Examples:

1. Assume that 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))
```

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),;
EX (ONE,TWO,THREE),UN(BETA))
```

ANS Tape Options. Special options for ANS tapes are described in Table 37. These options pertain to 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 37. 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.

Table 37. ANS Tape Options – COPY Command (cont.)

Code	Description
BLK(n)	Block size. The value n specifies the maximum block size to be built for FMT(F), FMT (D), and FMT(V), where $1 \leq n \leq 32,767$ 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 $1 \leq n \leq 32,767$ 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 a card deck is to be copied to file X on ANS tape number 123456. The tape is to be on a BT drive. Only the first 72 characters of each card are to be copied and the block size is to be 720.

```
C CR TO AT#123456-BT/X(FMT(F),;
BLK(720),REC(72))
```

2. Assume that four files named A are to be copied from ANS tape numbers 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#5/A(CAT(4)) TO B
```

Expiration Option. The expiration option specifies an expiration time for the output file of the COPY command. It has the format

EXP({mm,dd,yy / ddd / NEVER})

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);
TO N3
```

Valid Option Combinations. Not all combinations of source and destination devices, data types, formats, modes, or sequencing codes are valid. Table 38 shows the valid combinations, the invalid combinations, and the default provisions for the various possible combinations that are checked by PCL. Other combinations are 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 current job account 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' 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 name 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

Table 38. Valid Option Combination

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	RD	-	-	-	-	-	-	-	-	x	x	x	-	-	-	-	-	-
	WR	-	-	-	-	-	-	-	-	x	x	x	-	-	-	-	-	-
	EX	-	-	-	-	-	-	-	-	x	x	x	-	-	-	-	-	-
	UN	-	-	-	-	-	-	-	-	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.

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 a fid (see COPY command) 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 31 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.

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

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 the correct password is specified in the range specification.

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. 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 line printer, each file copy is preceded by the name of the file.

If there are no files present in the specified account, the following message prints:

NO FILES IN DIRECTORY

As each file is copied, its name is listed through M:LL. If a file cannot be copied, the file name is followed by an 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 during execution of the command and ssssss is the number of files that could not be copied.

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

   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

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

4. Assume that all keyed files on disk pack #5 are to be written to a scratch tape.

   COPYALL DP#5 (KEY) TO LT

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))
```

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:

- [DP/]
- [DC/]fid
- DP#serial no.[-rt]/fid
- LT#serial no.[-rt]/fid

output may be one of the following:

- DP
- DC
- 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 job account and the same file names as appear in the control 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 within 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 except that options are not allowed. 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 line printer, each file copy is preceded by the name of the file.

As each file is copied its name is listed through M:LL. If a file cannot be copied, the file name is followed by an error or abnormal code and subcode.

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 filed copied during execution of the command including the control file itself, and ssssss is the number of files that could not be copied.

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
```

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 the 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

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

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] { [DP/] / [DC/] / DP#serial no.[-rt]/ } fid[,fid]...

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 3I 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 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 job account with password PLEASE.

D SOURCE..PLEASE

The message

.. 1 FILES DELETED, 2 TOTAL GRANULES

is printed on the LO device following execution of the command.

DELETEALL Another delete command deletes all files, or a specified range of files, in the job account. The form of the command is

DELETEAL[L] { [DP/] / [DC/] / DP#serial no.[-rt]/ } [range]

where

rt is the two-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 COPY-ALL command.

Each of the PCL identification codes listed in Table 3I is a reserved word for this command 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.

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.

If there are no files in the job account. PCL responds to the command with the following message:

NO FILES IN DIRECTORY

As each file is deleted, its name is listed through M:LL. If a file cannot be deleted, the file name is followed by an error or abnormal code and subcode.

After the delete function is performed, the following message is printed:

. .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 job account are to be deleted.

   DELETEALL

   A message such as

   .. 8 FILES DELETED

   is printed on the LO device following execution of the command.

2. Assume that all files in the inclusive range B through H are to be deleted.

   DELETEALL B, H

   A message such as

   .. 4 FILES DELETED

   is printed on the LO device following execution of the command.

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][(s)]                            ]
```

All listed output goes through the M:LO DCB.

The first two formats of this command allow a 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 specified in the S specification.

1. {DP, LT, AT}#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, random, 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').

   Read 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 tape (AT), options A and EA are equivalent and list:

   Format (F, D, V, or U)
   Block length
   Record length (F format)
   Block count
   File sequence number

   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 $1 \leq n \leq 9$ specifies the number of four-character columns to be occupied by each name. The default is C3. Names longer than the allotted space will occupy more than one space.

   The R option specifies that a range specification will be given in the command.

   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. {LT / DP / AT}#serial no.[-rt][(s)]/fid[(s)][,fid[(s)]]...
(list file attributes)

This is a request for the attributes of 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)][,fid[(s)]]... (list file attributes)

This is a request for the attributes of the one or more RAD or public disk pack files named. Option (s) 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. 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.

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 current job account are to be listed.

L

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#4.:SYSGEN-7T

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.

L ALPHA,BETA

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)

5. Assume that a tape requires identification. The fake serial no. X is used in the command.

L FT#X

6. Assume that the files A through B of account X are to be listed with attributes.

L (R,A) A. X,B

REVIEW In the batch mode, this command functions identically to LIST. The format of this command is:

REV[IEW] [DC/ / DP#serial no.[-rt]/ / .account] [(s)][range]

where

rt is the two-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 COPY-ALL command.

s may be either A or EA (as in LIST command).

Example:

REV N,X

Each file name within the inclusive range N through X is listed.

If a file has a password or is open by another user, this is noted by an appropriate message.

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

ERR[ORS] { SAV[E] / 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 hexidecimal error code whose value is to be printed.

SPF
SPR These command(s) 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.

SPT FT#2076,2

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

SPE { LT / FT / AT } #serial no.[-rt]

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, PCL aborts the job after printing the message

PCL ABORT

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

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 the line printers. The form of the command is

WEO[F] [FT# serial no. [-rt] / LP / CP / PP]

(Note that only one output will be open at a time.)

If no output device is specified, the current output device is used.

REW This command rewinds the specified magnetic tape reel. It has the form

REW { [LT / FT] #serial no. [-rt] / AT[#serial no.] [-rt] [/filename] }

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

Example:

Assume that magnetic tape reel No. 205 is to be rewound.

```
REW#205
```

MOUNT This command mounts a magnetic tape or disk pack. The form of the command is:

```
MOU[NT) { [LT]# serial no.[-rt]          } [(RING)]
        { [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 and RING specifies that the device is to be mounted with write access.

Example:

Assume that a tape reel #2075 is to be mounted with a write ring.

```
MOUNT #2075 (RING)
```

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

```
REM[OVE] { [LT]#serial no.[-rt]          }
         { [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
```

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. The maximum value for s is 255.

Example:

Assume that tabs are to be set for expansion in the standard Meta-Symbol list format.

```
TABS    10,19,37
```

TERMINATION OF PCL

PCL operations are terminated by the END command. This command returns control to the monitor. The form of the command is

```
{E[ND]}
{X    }
```

When PCL is terminated by an END command, the following message is output

```
PCL PROCESSING TERMINATED
```

PCL 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 39.

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.

PCL COMMAND SUMMARY

Table 40 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 39. 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
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

[t] RS signifies record selection.

Table 39. PCL Error Codes (cont.)

Hexadecimal Code	Message	Severity Level
11C00	PK/BIN/7T COMBINATION NOT VALID.	2
11D00	NULL ARGUMENT (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.	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 INCREMENT ON CS OR LN OPTION.	2
13000	TX OPTION USED WITHOUT TABS COMMAND.	2
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 40. 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] LT#serial no.[-rt] AT[#serial no.][-rt] FT#serial no.[-rt] where rt identifies 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, LC, UC, NF, TX, DEOD, 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, JOB); or selection (x-y). dd may be DC, CP, LP, ME, operational label, stream-id or: DP#serial no.[-rt] LT[#serial no.][-rt] AT[#serial no.][-rt] FT[#serial no.][-rt] where rt identifies a device that is defined 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 identifies a device that is defined to be a resource. a may be COPY output options.
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.

Table 40. PCL Command Summary (cont.)

Command	Description
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. The hhhhhh option is a hexadecimal error code whose value is to be printed.
L[IST] [LT#reel-id[-rt][(s)][range] [DC][.acct][(s)][range] DP#reel-id[-rt][(s)][range] DP#serial no.[-rt]/fid[(s)][,fid[(s)]]... LT#serial no.[-rt][(s)]/fid (s)][,fid[(s)]]... fid[(s)][,fid[(s)]]... FT#serial no.[-rt][(s)]]	List 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, or Cn.
MOU[NT] {[LT / FT]# serial no.[-rt] / AT# serial no.[-rt]... [/filename] / 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 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 40. PCL Command Summary (cont.)

Command	Description
REV[IEW] [DC/ | DP#serial no.[-rt]/ | .account] [(s)][range]	Reviews all 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.
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 (Xerox) 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 | PP]	Writes an end-of-file on the specified output device.
X	Returns control to the monitor.

(This page intentionally left blank.)

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 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 the on-line, batch, or ghost 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 on 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 of CP-V/TS Reference Manual, 90 09 07.

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 interchangeably. It also allows a replacement string to be a null string.

Examples:

In the examples below, 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 for one file. There may be no more than 470 characters in the data replacement requests for one file (including the left and right sides, 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,FILE2,;(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 FiBATCH 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)
:
```

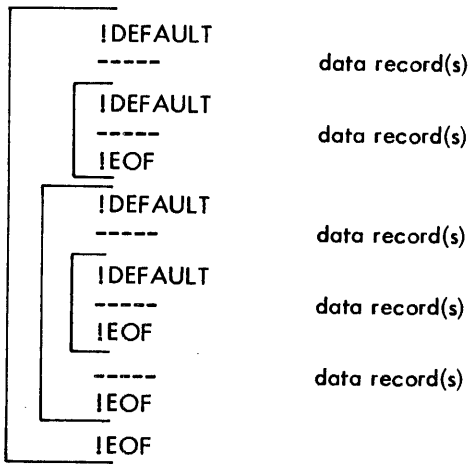
DEFAULT The DEFAULT command allows data replacement requests to be made within the input file. The DEFAULT command may appear any number of times and anywhere within the file being submitted and is effective on subsequent records of that file. If a data replacement request on a DEFAULT command is made 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:



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 41. These messages are output through the F:BATCH DCB. 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.

Table 41. 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 255 bytes.
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 41. 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 is specified.
*****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 parenthesis in an extended accounting field. (The extended accounting field must be terminated by a right parenthesis or the end of the command.)

SHOW PROCESSOR

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. Show is called by a !SHOW command or by CCI when a job is aborted for exceeding a limit. The values displayed by Show are the maximum values that a user can legally request on a !LIMIT command.

The form of the SHOW command is:

SHOW[option[,option]...]

The legal options are:

USER — displays the log-on account, name, auto-call processor, 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, service limits, resource limits, and device and feature authorization for both batch and on-line operation.

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, and DCBS options and is assumed if no options are specified.

If for any reason SHOW is not able to access the system default tables, only the user specific values are displayed and the message

CAN'T GIVE YOU SYSTEM DEFAULT VALUES

is output. Show's output is directed through the M:LO DCB.

DEFCOM PROCESSOR

The DEFCOM processor provides users a means of accessing core resident data and routines in one load module by another load module. This is accomplished by using a specified load module as input and producing another load module that contains only the DEFs and their values, which can then be combined with other load modules to give them access to core resident data and routines.

Thus the DEFCOM processor may be used to provide the user with a common data pool or library. This can be done in the following way. A load module consisting of DEFs for the data or the actual routines is produced by the loader, and the DEFCOM processor is run using the load module as input, producing another load module which contains only the DEFs and the DEF values of the input load module. When other load modules are created by the loader, the load module containing only the DEFs can be included by specifying the name of the DEF load module on the EF option of the LOAD control command.

The format of the control command is

```
!DEFCOM
```

The input load module name is specified by assigning the DCB of M:EI to the file containing the load module. The name of the load module to be formed by DEFCOM is specified by assigning the DCB of M:EO to a file with that name.

Example:

```
!ASSIGN    M:EI,(FILE,M:MON)
!ASSIGN    M:EO,(FILE,MONSTK)
!DEFCOM
```

The input load module must not contain any REFs or dummy sections and must not have been generated with the PERM, LIB option in the LOAD command used to generate it.

SYMCON PROCESSOR

INTRODUCTION

The Symbol Control Processor (SYMCON) provides a means of controlling the external symbols in a load module. Its primary function is to give the programmer a means of preventing double definitions of external symbols. A programmer who is working on one section of a large system need be concerned only with the external symbols that will eventually be used to communicate with other sections of the system. He need not be concerned with symbols internal to a group of ROMs once they have been loaded together. If someone else uses the same name in another section, either one or both of the programmers can delete the name with SYMCON before the sections are combined.

Thus, with the aid of SYMCON, a programmer need only decide what symbols are to be referenced by external programs. Then except for these symbols, he may use any symbols he wants in the various relocatable object modules that make up the load module.

Another use of SYMCON is to reduce external symbols. If certain load modules cannot be combined because their tables of control information are too large, the tables may be reduced in size by deleting all but the essential external symbols.

SYMCON may also be used to provide the load module with a global symbol table in Delta symbol table format for use by Delta during a run. Conversely, a global symbol table may be discarded from a load module.

Printed output from SYMCON goes to the LO device.

CONVENTIONS

Blanks may be used within SYMCON commands but may not be embedded within a command verb or symbol. A command is terminated by the end of the input record or by a period, and may be continued from record to record by use of a semicolon, in which case the continuation record begins with the first character.

CALLING SYMCON

SYMCON is called by the following processor control command.

```
!SYMCON
```

However, before SYMCON can be called, the load module file must be assigned to the element input DCB. This is done by an ASSIGN control command.

```
!ASSIGN M:EI,(FILE,lmn),(INOUT)
```

where

M:EI specifies the element input DCB.

lmn is the name of the load module.

INOUT specifies file use in the update mode.

SYMCON reads the load module, processes each command independently, then rewrites the load module, providing no major errors are encountered. Note that the old load module is overwritten unless an abort occurs.

SYMCON may be used as an on-line processor by including it in the :SYS account. When used in this way, it may be entered with a SYMCON command in response to a TEL prompt. When entered, SYMCON will type SYMCON HERE and accept commands from the terminal, prompting for each with an asterisk. The identify of the load module

is established prior to calling SYMCON, with a SET command of the form

```
!SET  M:EI DC/lmn
```

The END command terminates SYMCON.

SYMCON COMMANDS

There are eight SYMCON commands: LIST, DELETE, KEEP, RETAIN, CHANGE, BUILD, DISCARD, and END. The function of these commands is to

1. Produce a load map (LIST).
2. Delete specified symbols (DELETE).
3. Delete all symbols but those specified (KEEP).
4. Delete all but a specified range of symbols (RETAIN).
5. Rename a symbol (CHANGE).
6. Build a Delta-format global symbol table (BUILD).
7. Discard a Delta-format global symbol table (DISCARD).
8. Exit from SYMCON (END).

The execution of each command is independent of any other command. Thus, after the configuration of the load module after the execution of one command is what is acted on by the next command. This serial nature of operation is useful for certain kinds of symbol manipulation, such as that for the DELETE command.

Five SYMCON commands – LIST, DELETE, KEEP, RETAIN, CHANGE – do not operate on the global symbol table built by the BUILD command. Hence a BUILD command must be executed after a DELETE, KEEP, RETAIN, or CHANGE command is executed in order for the global symbol table to accurately reflect the load module.

LIST

This command lists the external symbols of the load module in the same format as the load map. The ordering of items will usually be somewhat different from that produced by the loader and there may be some additional control sections (CSECs) listed corresponding to items (such as DCBs) obtained from the library. Forward references do not appear in the load map. The LIST command has the form

```
LIST
```

DELETE

This command deletes the specified symbols. Any DEF symbol in the module load map may be deleted unless it enters into the definition of a DEF symbol or a forward reference that is not yet completely defined.

The general form of the DELETE command is

```
DELETE  name[,name]...
```

where name is the name of a symbol to be deleted.

Example:

Assume the following Meta-Symbol code.

```
        DEF     A,B
        REF     C
        :
        :
A       EQU     B+C
B       EQU     2
```

If the external reference C is not satisfied when the load module is formed, then A is not completely defined. Thus any attempt to delete B will be ignored and an error message will result.

If A can be deleted, the DELETE B will work because A no longer exists after the DELETE A has been executed.

KEEP

This command deletes all DEFed symbols except those that fall into the following categories.

1. DEFs listed in the command.
2. DEFs that help define the symbols listed in the command.
3. DEFs defined in terms of unsatisfied references (and used).

The form of the KEEP command is

```
KEEP  name[,name]...
```

where name is the name of a DEF symbol to be deleted.

Example:

Assume two ROMs are loaded to form a load module. They are

```
        DEF     A, B, C
        REF     D, E, F
        .
        .
A       EQU     297
B       EQU     E + 3
C       EQU     F - 1
```

and

```
        DEF     D, E
        REF     A, C
        .
        .
D       EQU     A
E       EQU     6
        LW, R7  C
        .
        .
```

After these ROMs are in the form of a load module, the following command is issued.

```
KEEP  B, A
```

DEF symbols A and B are listed in the command, E is used to define B, and C is defined in unsatisfied terms (namely F). Thus, A, B, C, and E are not deleted but D is. DEFs in the unsatisfied reference category are not deleted if they help define a core location in the object code (i.e., they are used) but are deleted otherwise. In the above example, C would have been deleted had it not been used in the LW instruction.

RETAIN

This command deletes all but a specified range of DEFed symbols with the constraints specified for KEEP. The form of the RETAIN command is

RETAIN $name_1$,$name_2$

The symbols $name_1$ and $name_2$ delimit a range of symbols as they appear within the load module's REF/DEF stack. Note that they do <u>not</u> refer to an alphabetical range of symbols, but rather to the actual physical order in which symbols appear within the REF/DEF stack.

<u>Note:</u> This command is intended primarily for use in system development and modification and should be used with caution.

CHANGE

This command renames symbols. Unlike DELETE and KEEP, the CHANGE command may be used to operate on any item with a name (DEF, SREF, PREF, DSEC). The form of the command is

CHANGE $name_1/name_2[,name_1/name_2]\ldots$

where

$name_1$ is the name of the symbol to be changed.,

$name_2$ is the name to be given to the symbol identified by $name_1$.

The only restriction is that $name_1$ must be in the module and $name_2$ must not.

BUILD

This command builds a Delta-format global symbol table. The form of the command is

BUILD [(LIB)]

If the (LIB) option is specified, library DEFs are included in the global symbol table along with the load module DEFs. Since Delta symbols are truncated to seven characters in length, any set of symbols that are alike in the first seven characters are treated as a multiple DEF and only the one that appears first is retained in the Delta symbol table. The Delta symbol table type associated with the symbol is "constant" for DEFs with constant values and is "instruction address" for all others.

DISCARD

This command is used to discard a Delta-format global symbol table from a load module that includes one. The form of the command is

DISCARD

END

This command terminates SYMCON. The form of the command is

END

SYMCON ERROR MESSAGES

SYMCON checks for a number of error conditions. Table 42 lists SYMCON error messages.

Table 42. SYMCON Error Messages

Message	Description
name ALREADY IN STACK, CHANGE NOT MADE	An attempt was made to change the name of an item to a name currently used by another item.
name APPEARS AS TYPE OTHER THAN DEF, NO ACTION	The symbol was a PREF, SREF, or DSEC and could not be deleted.
CAN'T USE SYMCON ON LINK OR LIBRARY LOAD MODULES	An attempt was made to use SYMCON on a load module library or a Link-built load module.
COMMAND CONTAINS ILLEGAL CHARACTER	The command contained a character not in the character set defined for Meta-Symbol. The job is aborted.
DELTA SYMBOL TABLE ALREADY IN LOAD MODULE, NO ACTION TAKEN	A BUILD command was given and a global symbol table is already included in the load module. The command is ignored.
EH?	SYMCON does not recognize the command.
ILLEGAL OPTION	An option other than (LIB) was specified on a BUILD command.
ILLEGAL SYNTAX	Command syntax was incorrect. The job is aborted.
INCOMPLETE COMMAND LOAD MODULE UNCHANGED	This message indicates that a continuation was specified (with a semicolon) but the end of the file was encountered when an attempt was made to read another card. The job is aborted.
INPUT M:EI FILE IS NOT A LOAD MODULE	The M:EI file is either not keyed or is not a properly formed load module.
M:EI I/O ERR: xxxx xxxx	An I/O error occurred accessing M:EI. The content of SR3 is displayed following this message.
NO DELETIONS RESULTED FROM THIS COMMAND	None of the symbols listed caused any deletions. The load module is unchanged.
NO DELTA SYMBOL TABLE TO DISCARD, NO ACTION TAKEN	A DISCARD command was given, but there is no global symbol table included in the load module. The command is ignored.
NO SYMBOLS FOR DELTA SYMBOL TABLE, TABLE NOT BUILT	A BUILD command was given, but there are no nonlibrary DEFs in this load module, hence no Delta symbol table can be built. The command is ignored.
name NOT FOUND IN REF/DEF STACK	The identified symbol did not exist as an external symbol in the load module.
OVERLAY PROGRAM, DELTA SYMBOL TABLE BUILT FOR ROOT ONLY	A symbol table was built only for the root of the overlay.
REQUIRED CORE SPACE NOT AVAILABLE	This message indicates that the M:GP procedure failed to supply enough operating space for the processor. The job is aborted.
THESE SYMBOLS WERE DELETED name,name...name	This message includes all deleted symbols, including deletions caused by other deletions.
name USED IN UNEVALUATED EXPRESSION, NOT DELETED	This message indicates that the symbol was used to define an item that depended on an external reference. The item may have been a DEF, a forward reference, or a core location of the object code.

APPENDIX A. DATA CONTROL BLOCK FORMATS

This appendix contains the formats for the three kinds of DCBs created by the monitor: files, devices, and labeled tape. Following each format, the parameter fields of the DCB are described in alphabetical sequence by their mnemonic. All referenced addresses have word resolution unless otherwise specified.

FILE DCB

Figure A-1 shows the format of the DCB for consecutive, keyed, and random files. All single fields are applicable to the three kinds of files. Fields shown with a heavy border depict differences between consecutive, keyed, and random. Shaded fields are not used by the DCB.

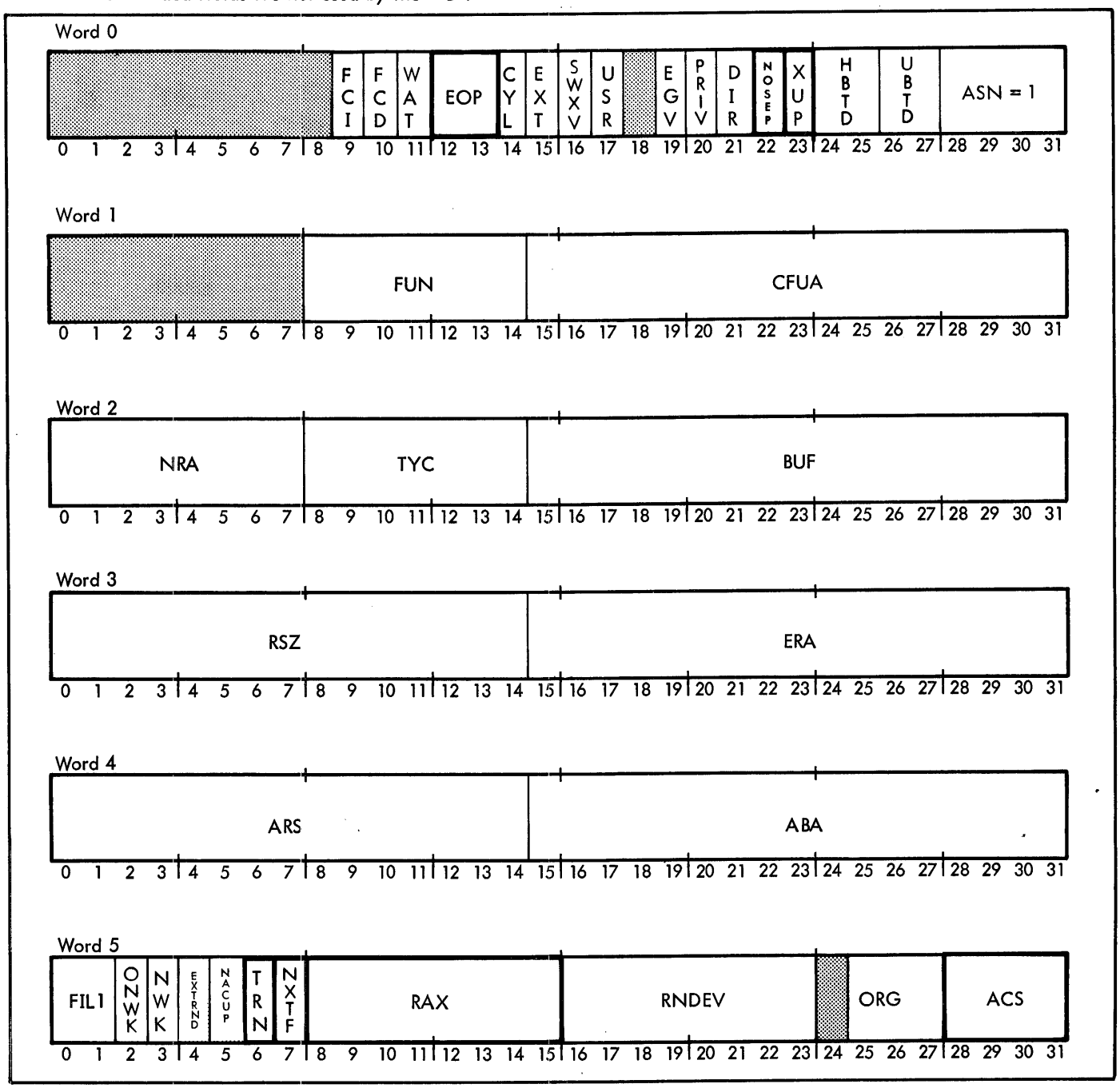


Figure A-1. Format of File DCB

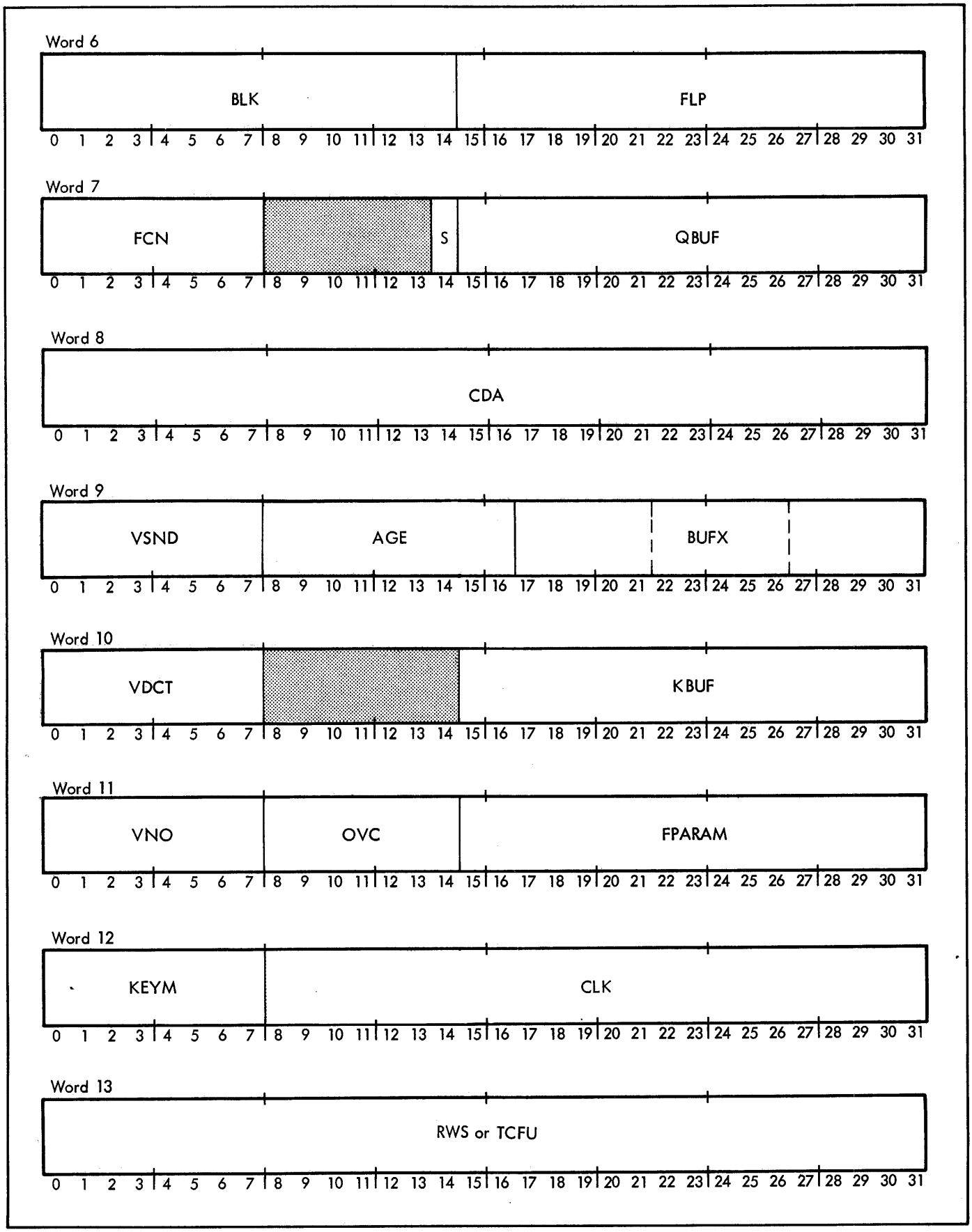
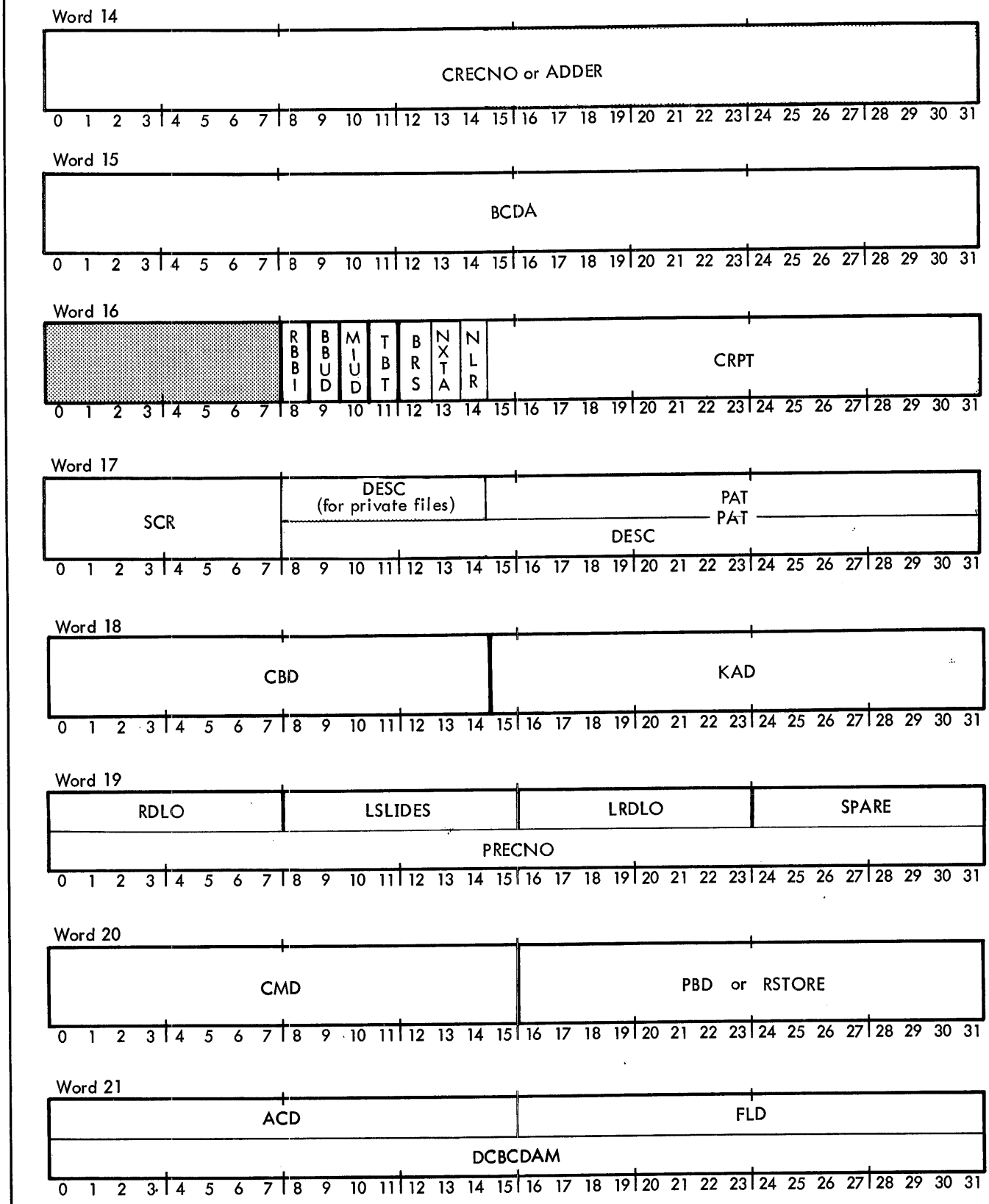


Figure A-1. Format of File DCB (cont.)



Words 22 → n are used for variable length parameters.

Figure A-1. Format of File DCB (cont.)

In the following field descriptions, the Control column signifies who specifies the contents of the field – the monitor (M) or the user (U).

Field	Description	Word	Control
ABA	Contains the address of the user's routine that will handle abnormal conditions resulting from insufficient or conflicting information. (The monitor returns to ABA in the FPT if the abnormal condition is the result of a device abnormality.)	4	U
ACD	Contains the word displacement to the user's account number in the DCB relative to the start of the variable length parameters. (FLP+ACD = FWA of the EBCDIC account number.) (Meaningful only during an open or close.)	21	M
ACS	is the file access indicator (0 = none specified and is treated as sequential, 1 = sequential, 2 = direct). ACS is only meaningful when a file is first written in the OUT or OUTIN mode. If a file has keyed organization and sequential access is specified, the keys written must be in ascending order. However, if the organization is keyed and direct access is specified, the keys can be written in any order (the monitor sorts them into ascending order). ACS is not used by random files.	5	U
ADDER	contains the size of a single entry in the master index structure or directory for operations on keyed files or directories.	14	M
AGE	is used to measure the most recent activity on the DCB so that buffer truncation can be made more efficiently.	9	M
ARS	contains 1. the actual number of data bytes transferred to or from the user following a read or write. 2. the number of records remaining to be skipped following a PRECORD operation that has terminated due to an end-of-file or a beginning-of-file condition.	4	U, M
ASN	indicates the assignment type currently in effect for the DCB (0 = null, 1 = file, 2 = Xerox labeled tape, 3 = device, X'A' = ANS labeled tape).	0	U
BBUD	indicates whether or not the blocking buffer (BUF1) has been changed since it was last read or initialized (0 = unchanged, 1 = changed). This flag is used to determine whether or not BUF1 needs to be written out to the data granule specified in BCDA before truncating the buffer. BBUD is not used by random files.	16	M
BCDA	contains the disk address of the data granule currently in the blocking buffer (BUF1). BCDA is not used by random files.	15	M
BLK	contains 1. the byte count of the record segment pointed to by either CBD or PBD, depending upon the point in time. Not applicable to random files. 2. the number of bytes to be transferred by the I/O routines whenever called.	6	M
BRS	indicates whether or not the record segment pointed to be CBD or PBD, depending upon the point in time, is blocked (0 = unblocked, 1 = blocked). BRS is not used by random files. During an open BRS, indicates whether the 'TEST' option was indicated in the open FPT (0 = not test, 1 = test).	16	M

Field	Description	Word	Control
BUF	contains the address of the user's buffer where the data record is to be read or written.	2	U
BUFX	contains three 5-bit subfields used to index into the table of pooled buffers available to the file management system. These indexes have varying significance depending on the current operation being performed.	9	M
CBD	contains the current byte displacement within the blocking buffer (BUF1). CBD specifies where the record segment associated with the key pointed to by CMD begins. When writing on the file, CBD = 0 if a data granule other than the last is being updated. CBD is not used by random files.	18	M
CDA	contains 1. the disk address to be used by the I/O routines whenever called. 2. a counter indicating the number of records to skip. Not applicable to random files.	8	M
CFUA	contains the address of the CFU associated with the file. During open or close operations, CFUA contains the address of the ACNCFU and FILCFU.	1	M
CLK	contains 1. the net number of data and Master Index granules allocated to or released from the file during this OPEN. Applicable to keyed and consecutive files. The field is a 23-bit signed integer with a guard bit 8 that is used to prevent overflow into the KEYM field. 2. the number of granules allocated to the file. Applicable to random files.	12	M
CMD	contains 1. the byte displacement to the current key entry in the Master Index Buffer (BUF2) for keyed files. CMD, along with TRN and DCBCDAM, points to the current position in the file. For consecutive files, CMD contains a word position in the granule pointed to by DCBCDAM. None of this is applicable to random files. 2. the byte displacement to the current entry in the Account Directory or File Directory index buffer (BUF2) when the file is being opened or closed.	20	M
CRECNO	contains the current record number. It is set to 1. 0 if at the beginning of the file. 2. the number of records in the file (obtained from TDA in the CFU) if at the end of the file. 3. the sequential record number of the record most recently read or written. CRECNO is only used for consecutive files.	14	M
CRPT	specifies the address of a word to be used as the seed for a data encryption process. This field applies to keyed and consecutive files only.	16	U
CYL	specifies whether the file assigned to the DCB is to be allocated by granules or cylinders (0 = granule allocation, 1 = cylinder allocation). Only meaningful for public files.	0	M

Field	Description	Word	Control
DCBCDAM	is used when CFUA points to a user CFU for keyed or random files and contains the disk address of the current index granule in the Master Index Buffer (BUF2). If CFUA points to the Account or File Directory CFU, CDAM in FILCFU or ACNCFU contains the disk address of the current granule in BUF2. For consecutive files, DCBCDAM contains a disk address of a granule, reflecting (in conjunction with CMD) the location in the file at which the most recent data transfer operation took place.	21	M
DESC	is used as storage for file descriptors. For private files, DESC resides in bits 8-14.	17	M
DIR	indicates the direction of the read operation (0 = forward, 1 = reverse). DIR is not used by random files.	0	U
EGV	is the event-given flag and indicates whether or not the completion code posted in the TYC field has been communicated to the user's program by the CHECK routine (0 = no, 1 = yes). The CHECK routine is called either directly by the user or indirectly by the monitor, depending upon the WAIT, ERR, and ABN options in the FPT.	0	M
EOP	is the ending operation indicator (0 = other, e.g., rewind, 1 = read, 2 = write). Specifies the type of I/O operation currently or last performed. EOP is not used by random files.	0	M
ERA	contains the address of the user's routine that will handle error conditions resulting from insufficient or conflicting information. (The monitor returns to the ERA in the FPT if the error condition is the result of a device failure.)	3	U
EXT	is the file extension flag and indicates whether OPEN is to position to the beginning or end of a specified file (0 = beginning-of-file, 1 = end-of-file).	0	M
EXTRND	is set to one if the RAX field is to be logically appended to the RSTORE field (RAX being the most significant field) for a random file. Otherwise, it is set to zero.	5	M
FCD	indicates whether the DCB is opened or closed (0 = closed, 1 = opened).	0	M
FCI	indicates whether the DCB has ever been closed. This flag is set when the DCB is first closed and then never reset (0 = DCB has never been closed, 1 = DCB has been previously opened and closed).	0	M
FCN	indicates the current number of I/O operations that have been initiated but not completed, for this DCB.	7	M
FIL1	indicates the file option last specified (0 = none specified and is treated as release, 1 = release, 2 = save, 3 = JOB).	5	U
FLD	contains the word displacement to the file name in the DCB relative to the start of the variable length parameters (FLD + FLP = FWA of the EBCDIC file name). (Meaningful only during open and close.)	21	M
FLP	contains the address of the start of the variable length parameters in the DCB (called the file list-pointer).	6	M
FPARAM	contains the receiving address of the user's 90-word buffer to which the variable length parameters from the file's FIT are to be passed.	11	U
FUN	indicates the file mode function (0 = null, 1 = IN, 2 = OUT, 4 = INOUT, 8 = OUTIN).	1	U

Field	Description	Word	Control
HBTD	is the I/O handler's byte displacement indicator and is used whenever the I/O routines are called to specify the byte displacement within QBUF into which the data transfer is to begin.	0	M
KAD	contains	18	
	1. the address of the key specified by the user in the read or write FPT.		U
	2. the address of the account number or file name when opening or closing the file.		M
KBUF	contains	10	U
	1. the address of the buffer containing the key most recently accessed in the Master Index or File Directory. The field is set up by the M:DCB procedure and points to an 8-word buffer following the VLPs. Not applicable to random files except during open.		
	2. the address of the word buffer containing the relative granule number of the first sector to be used in the I/O transfer. Applicable to random files only.		
	3. the address of an 8-word buffer in the DCB that contains the TEXTC key of records read sequentially from a keyed file.		
KEYM	contains the maximum length, in bytes, of the keys in the file pointed to by the DCB. Applicable to keyed files. Maximum value is 31.	12	U
LRDLO	contains the limiting number of contiguous index granules that can be allocated in level 0 and not be reflected in level 1 before the flag, which signals CLOSE to reconstruct the higher level index structure, is set (i.e., before SLIDES in the CFU is set equal to 255).	19	U
	LRDLO is only used for keyed files.		
LSLIDES	only has meaning if a multilevel index exists and contains	19	U
	1. the limiting number of index granules that can be allocated in level 0 and not be reflected in level 1 before the flag, which signals CLOSE to reconstruct the higher level index structure, is set.		
	2. the value 255, which means that once a higher level index structure exists, it is not to be reconstructed.		
	LSLIDES is only used for keyed files.		
MIUD	indicates whether or not the Master Index Buffer (BUF2) has been changed since it was last read or initialized (0 = unchanged, 1 = changed). This flag is used to determine whether or not BUF2 needs to be written out to the granule specified in either DCBCDAM or CDAM in FILCFU or ACNCFU before truncating the buffer.	16	M
NACUP	indicates whether the file's descriptors indicate that the last access date is not to be updated (0 = may be updated, 1 = may not be updated).	5	M
NLR	indicates whether or not the record segment pointed to by CBD is the first record in a continued data record (0 = second or nth record segment, 1 = first or only record segment). NLR is only meaningful during a WRITE operation.	16	M
NOSEP	specifies whether or not granules are to be allocated from RAD (0 = no, 1 = yes). Normally, granules are allocated on DP. However, if all the devices of the normally allocated type are saturated, the system attempts to allocate on an alternate device. The order of allocation is DP and RAD if the NOSEP flag is reset. If the	0	U

Field	Description	Word	Control
NOSEP (cont.)	NOSEP flag is set, granules will be allocated from RAD if DEVICE, DC was specified. This flag has no meaning for private files. NOSEP is not used by random files.		
NRA	indicates the number of recovery tries that may be attempted before a device error message is to be logged.	2	U
NWK	indicates whether or not NEWKEY was specified in the M:WRITE FPT (0 = replace an existing key, if the key does not exist, take an abnormal return; 1 = write a new key, if the key already exists, take an abnormal return). If ONWK is set, the NWK flag is ignored. NWK is only used for keyed files.	5	U
NXTA	is the next account indicator and specifies whether this account (i.e., the account number in the DCB/JIT) or the next account in the Account Directory (i.e., the one following the account named in the DCB) is to be assigned to the DCB at OPEN (0 = this account, 1 = the next account). If an account number is not specified in the DCB and the NXTA indicator is set, the first account in the Account Directory is put in the DCB and nothing more is done unless NXTF is also set. After a file is open, the bit is set to 1 if the DCB is open to a star file (see Glossary); otherwise, it is set to 0.	16	U
NXTF	is the next file indicator and specifies whether this file (i.e., the file named in the DCB/FPT) or the next file in the File Directory (i.e., the one following the file named in the DCB) is to be assigned to the DCB at OPEN. If a file name is not specified (in either the DCB or FPT), the first name in the File Directory is put in the DCB and assigned (0 = this file, 1 = next file).	5	U
ONWK	indicates whether or not ONEKEY was specified in the M:WRITE FPT (0 = check NWK flag, 1 = if the key already exists, replace the corresponding record, otherwise write a new record). ONWK is only used for keyed files.	5	U
ORG	is the file organization indicator (0 = none specified and is treated as consecutive, 1 = consecutive, 2 = keyed, 3 = random).	5	U
OVC	is the open volume count and only has meaning for private files. 1. for consecutive private files, OVC indicates whether or not the volume pointed to by VNO is opened or not (0 = no, 1 = yes). 2. for keyed or random private files, OVC contains a count of the numbers of volumes that have been opened.	11	M
PAT	contains the allocation table address of the private volume pointed to by VNO. Only has meaning for private files.	17	M
PBD	is the previous buffer displacement indicator, specifying at which byte in the blocking buffer (BUF1) the previous record segment begins. PBD is not used by random files.	20	M
PRECNO	contains the direction (+ or -) and the number of records that must be skipped from the position indicated in CRECNO prior to a data transfer operation (read, write, or delete). PRECNO is only used for consecutive files.	19	M

Field	Description	Word	Control
PRIV	indicates whether the file assigned to the DCB is public or private (0 = public, 1 = private). Public files reside on public devices and private files reside on private volume sets.	0	M
QBUF	contains 1. the buffer address to be used by the I/O routines whenever called. 2. the address within the user's buffer where the next record segment begins. QBUF, 2 is not applicable to random files.	7	M
RAX	controls read ahead. If set to X'FF', no read ahead is possible. If set to zero, no read ahead is in progress. Otherwise, RAX contains an index into read ahead tables.	5	M
RBBI	is the release blocking buffer inhibit flag and indicates whether or not the blocking buffer (BUF1) should be released during end-action after the data granule has been read into (BUF1) and the record segment has been transferred to the user's buffer. (0 = release BUF1, 1 = do not release BUF1.) RBBI is not used by random files.	16	M
RDLO	contains a tally (up to 255) of the number of index granules that are read or inserted at level 0 to locate the position of a user-specified key entry at level 0. If RDLO is greater than LRDLO, the flag, which signals CLOSE to reconstruct the higher level index structure, is set. RDLO is only used for keyed files.	19	M
RNDEV	contains the type of device requested for file allocation (0 = none specified and for private files gets changed to X'B', 7 = RAD, and X'B' = DP).	5	U
RSTORE	contains the number of granules to be allocated to the file. RSTORE is used by random files only. If RSTORE value is zero when a random file is created, an abnormal return is made with a code of X'14'. Bits 8-15 of word 5 are used by random files as a high order extension of this field if the EXTRAND bit is set.	20	U
RSZ	indicates the default record size, in bytes.	3	U
RWS	indicates 1. the requested number of bytes to be read or written from the user's buffer (BUF). During the I/O operation, RWS is decremented by the value in BLK each time that a record segment is either output or blocked. At the termination of the I/O operation, RWS is set equal to ARS. Applicable to keyed and consecutive files. 2. the requested number of bytes to be read or written from the user's buffer (BUF). At the termination of the I/O operation, RWS is set equal to ARS. Applicable to random files.	13	M
S	contains the value of the S field from the mode specification in the Open Cal FPT. S = 1 means SHARE; S = 0 means EXCLUSIVE.	7	U
SCR	indicates the byte length of the key portion of the entries in the Master Index currently referenced by the DCB. This can be the Master Index for the Account Directory, the File Directory, or the user's file.	17	M

Field	Description	Word	Control
SPARE	contains the number of spare byte positions to be left unused in the end of the current index granule in the event that the key to be added is the last key in the file. SPARE is only used for keyed files.	19	U
SWXV	is the switch volume flag and indicates whether or not the current volume is to be switched to the next volume after all updated buffers have been output to the current volume (0 = no, 1 = yes). Only used for consecutive private files.	0	M
TBT	not meaningfully used for files; however, the flag does get set and reset.	16	M
TCFU	contains the address of the user CFU during CLOSE.	13	M
TRN	indicates, for keyed files, whether the file is positioned before or after the data record whose key entry is pointed to by CMD (0 = after, 1 = before). For consecutive files, this bit is set only if the most recently executed operation on the file was a read backwards.	5	M
TYC	indicates the type of completion of an I/O operation.	2	M

TYC Code	Corresponding Error/ Abnormal Code	Meaning
0	0	normal without device I/O transfer
1	0	normal with a device I/O transfer
2	7	lost data
3	1D	beginning-of-tape
4	4	beginning-of-file
5	1C	end-of-reel
6	5	end-of-data
7	6	end-of-file
8	41	read error
9	45	write error
A	57	public devices/private volume-set saturated
B	0	SLIDES is 255
C	0	partial higher level index built

UBTD	is the byte displacement indicator, specifying at which byte in the user's buffer (BUF) the data record begins.	0	U
USR	indicates whether the JOB account number is the same as the account number specified in the DCB (0 = yes, 1 = no).	0	M
VDCT	contains the DCT index of the device on which the volume (in a private volume set) pointed to by VNO is mounted. Only meaningful for private files.	10	M

Field	Description	Word	Control
VNO	contains the volume number of the private volume currently being referenced via the DCB. Volume number is the position (starting with one) of a volume within the DCB's SN list. The SN list in the DCB has a fixed order and comes from the serial number table on the primary volume of a private volume set. Only meaningful for private files.	11	M
VSND	contains the word displacement to the serial number table of the private volume set (i.e., the SN list) in the DCB relative to the start of the Variable Length Parameters (FLP + VSND = the control word of the SN list).	9	M
WAT	is the wait flag and indicates whether or not WAIT was specified in the FPT (0 = no, 1 = yes).	0	U
XUP	indicates whether or not a higher level index structure is in the process of being reconstructed or constructed (0 = either that there is no higher level index or that the higher level index is complete, 1 = that the higher level index is being built). Only meaningful for keyed files.	0	M

VARIABLE LENGTH PARAMETERS 22→n

Each variable length parameter entry is preceded by a control word of the following form:

Byte 0 = a code number (see Table A-1) identifying the parameter which follows.

Byte 1 = code for the entry position (00 = more parameter entries to follow, 01 = last parameter entry).

Byte 2 = number of significant data words in the parameter entry.

Byte 3 = total number of words reserved for the entry, not including the control word (that is, maximum entry length).

Table A-1. Variable Length Parameter Codes

Code	Parameter Type
01	File name (in TEXTC format).
02	Account number.
03	Password.
04	Expiration date.
05	READ account numbers.
06	WRITE account numbers.
07	SN/INSN serial numbers.
08	OUTSN serial numbers.
09	File information (see Figure A-2).
0A	Modification date.
0B	SYNON name.

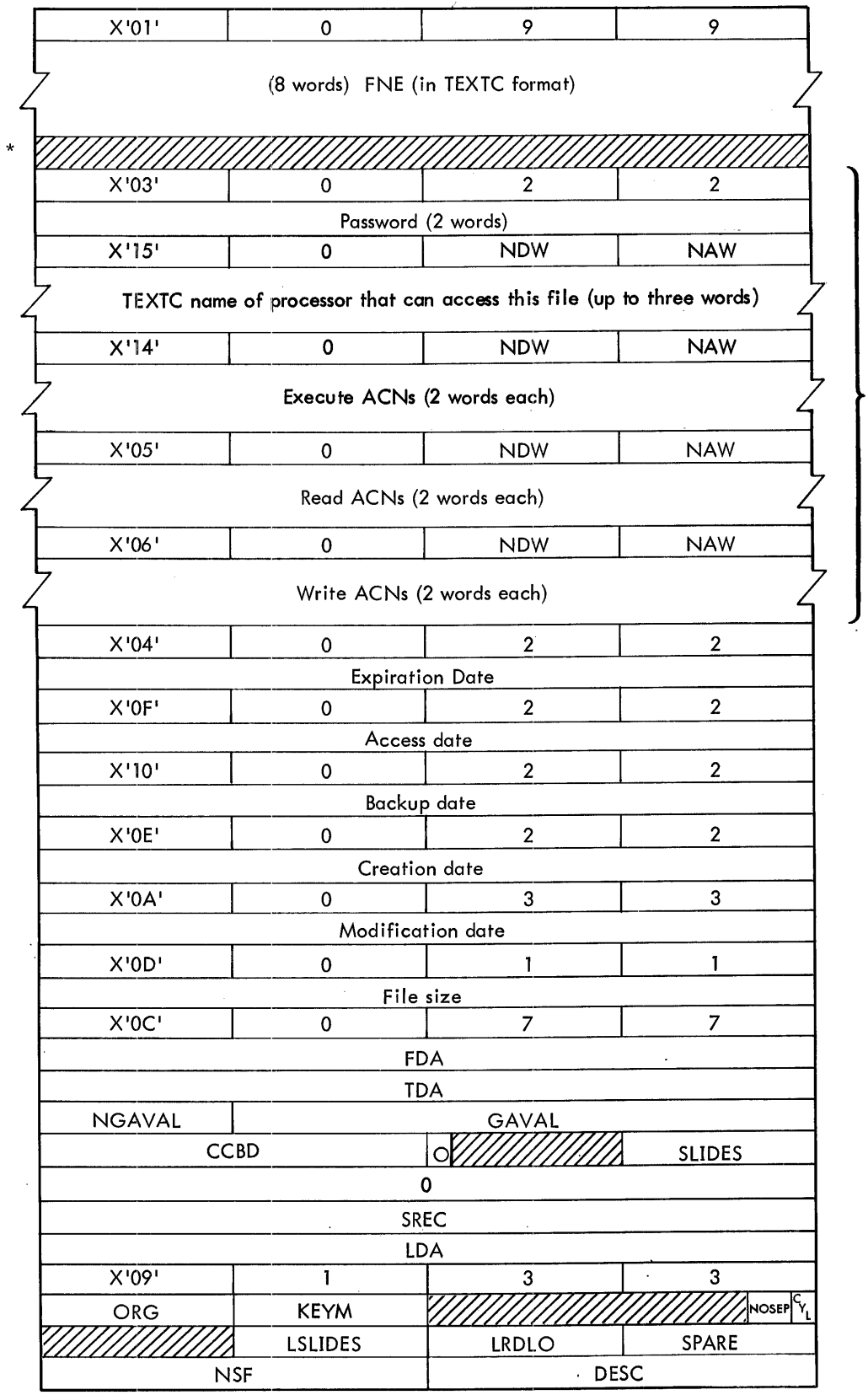
Table A-1. Variable Length Parameter Codes (cont.)

Code	Parameter Type
0C	File information (see Figure A-2).
0D	File size.
0E	Creation date.
0F	Last access date.
10	Backup date.
11	The X'11' VLP is used to control disk file status. It consists of one data word. The meanings of the bits are: Bit — Meaning 8 — If set, bits 12-15 will be moved from the data word into the file descriptor. 12 — If set, the file has been modified since last backed up by Fill. 13 — If set, the file has been modified since the last INCREMENTAL. 14 — If set, the file has been modified since the last SAVEALL. 15 — If set, the file has been modified since the last FILL. 20 — If set, the file is not to be backed up. 21 — If set, the access date is not to be updated. 22 — If set, the file is not to be deleted by the PURGE operation of Fill. This bit is only looked at if the user has a privilege that is greater than or equal to X'A0' or is a ghost. 23 — Must be set if bits 20-22 are to be looked at.
12	On line diagnostics; used to hold user's I/O command list.
14	Execute account numbers.
15	Names of the processors that may access this file. The names are in TEXTC format. Each name begins three words beyond the beginning of the previous name.

FIT FILE PARAMETERS (FPARAM TABLE)

The format of the file parameters that are passed from the FIT to the memory location specified by the FPARAM parameter of M:OPEN is given in Figure A-2, "Format of the FPARAM Table". A description of the fields of the table follows. Note that each variable length parameter is preceded by a control word of the form described in the section above, "Variable Length Parameters".

Field	Description
ACN	is an account number. There can be a maximum of 16 total Read and Write ACNs. Each ACN is an eight-byte EBCDIC entry with trailing blanks. If there is no Read ACN entry, any ACN can read the file. If there is no Write ACN entry, no one can write in the file except the ACN that created the file.



*For synonymous files opened via the NXTF option, the nine words immediately following the unused tenth word contain an X'0B' entry specifying the name of the primary file.

Figure A-2. Format of FPARAM Table

Field	Description
CCBD	contains, for keyed files, either the byte displacement to the next available byte in the last data granule of the file (SREC), which means that the blocking buffer was truncated; or 0, which means that the last data granule in the file (SREC) contains 512 words.
CYL	specifies whether the file assigned to the DCB is to be allocated by granules or cylinders (0 = granule allocation, 1 = cylinder allocation). It is only meaningful for public files.
Date	is of the form mmddhhyy, where mm is numerical month. dd is day of month. hh is hour of day. yy is last two digits of the year, all in EBCDIC bytes. Expiration date may contain the word NEVER followed by three blanks, which indicates that the file does not have an expiration date. The modification date contains three words. The third word is of the form hhmm, where hh is a repeat of the hour. mm is the minute.
DESC	contains the settings of the file descriptions.
FDA	contains the disk address of the file's first index granule at level 0.
File size	contains the current number of 512-word granules allocated to the file.
FNE	is the EBCDIC name of the file in TEXTC format.
GAVAL	contains the disk address of the next available granule in the last cylinder allocated to the file; zero if none.
KEYM	contains 1. the maximum length, in bytes, of the keys in the file. Applicable to keyed files. Maximum value is 31. 2. the type of device that the random file is to be allocated on (0 = allocate on either RAD or DP, X'7' = allocate on RAD, X'B' = allocate on DP). Applicable to random files.
LDA	contains the disk address of the file's last index granule at level 0.
LRDLO	contains the limiting number of contiguous index granules that can be allocated in level 0 and not be reflected in level 1 before the flag, which signals CLOSE to reconstruct the higher level index structure, is set (i.e., before SLIDES in the CFU is set equal to 255). LRDLO is only used for keyed files.
LSLIDES	has meaning only if a multilevel index exists and contains 1. the limiting number of index granules that can be allocated in level 0 and not be reflected in level 1 before the flag, which signals CLOSE to reconstruct the higher level index structure, is set. 2. the value 255, which means that once a higher level index structure exists, it is not to be reconstructed. LSLIDES is only used for keyed files.
NAW	is the number of available words in the entry (not including the control word).

Field	Description
NDW	is the number of significant data words in the entry (not including the control word).
NGAVAL	is the number of available granules in the last cylinder allocated to the file.
NOSEP	specifies whether or not granules are to be allocated from RAD (0 = no, 1 = yes). Normally, granules are allocated on DP. However, if all the devices of the normally allocated type are saturated, the system attempts to allocate on an alternate device. The order of allocation is DP and RAD if the NOSEP flag is reset. If the NOSEP flag is set, granules will be allocated from RAD if DEVICE, DC was specified. This flag has no meaning for private files. NOSEP is not used by random files.
NSF	is the number of files synonymous with this file.
O	is a level 1 flag indicating whether or not a level 1 index exists in a keyed file (0 = no, 1 = yes).
ORG	is the file organization indicator (0 = none specified and is treated as consecutive, 1 = consecutive, 2 = keyed, 3 = random).
Password	is an eight-byte EBCDIC entry with trailing blanks.
SLIDES	contains, for keyed files, either 1. a tally of the number of index granules allocated at level 0 since the current multilevel index structure was created, or if none exists, since the file was first opened. 2. a tally of the number of index granules allocated at the current level while the multilevel index structure is being (re)created. 3. the value 255, which means that a new multilevel index structure should be built when the file is closed (unless LSLIDES in the DCB equals 255 and a level-1 index exists).
SPARE	contains the number of spare byte positions to be left unused in the end of the current index granule in the event that the key to be added is the last key in the file. SPARE is only used for keyed files.
SREC	contains the disk address of the last data granule in the file. It is only used in the output mode.
TDA	contains, for keyed files, either 1. the disk address of the first index granule at the top of the multilevel structure, if one exists. 2. the disk address of the middle index granule, if there are three level-0 index granules and the file is keyed. 3. 0, which means that either the file is consecutive, or that the file is keyed and there are at the most two index granules. For consecutive files, TDA contains the number of records in the file.

DEVICE DCB

Figure A-3 shows the format of the DCB for a device. Shaded fields are not used by the DCB.

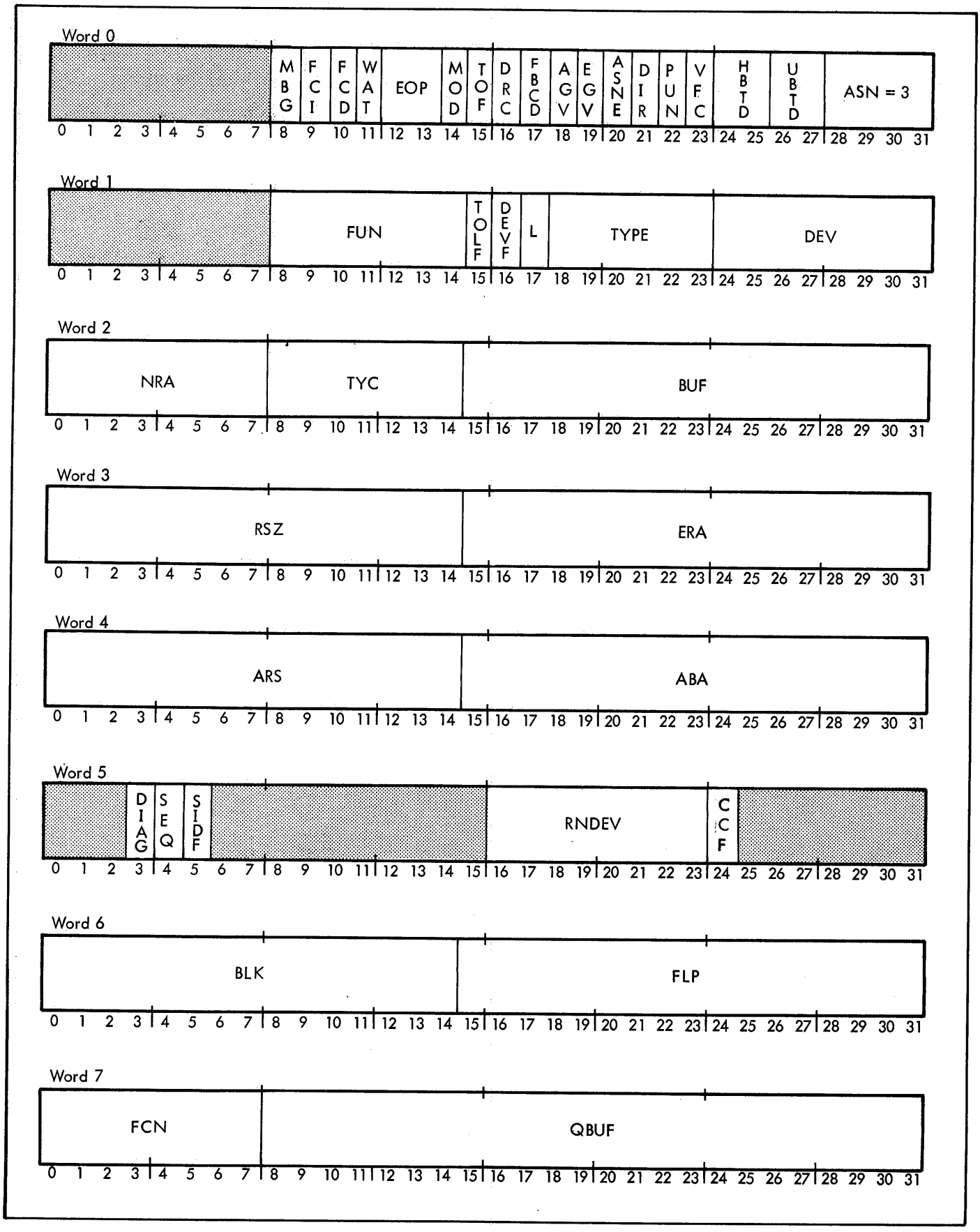


Figure A-3. Format of Device DCB

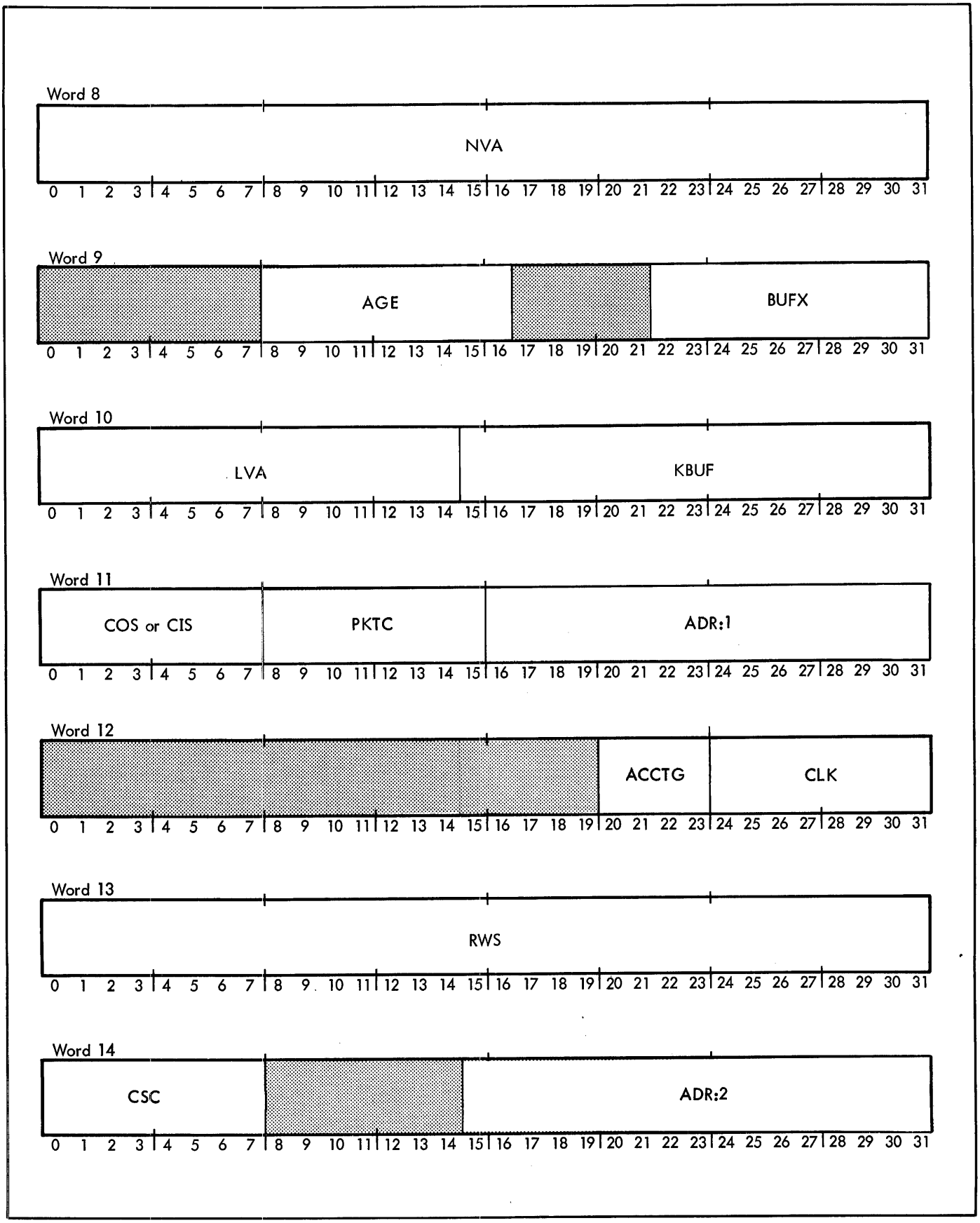


Figure A-3. Format of Device DCB (cont.)

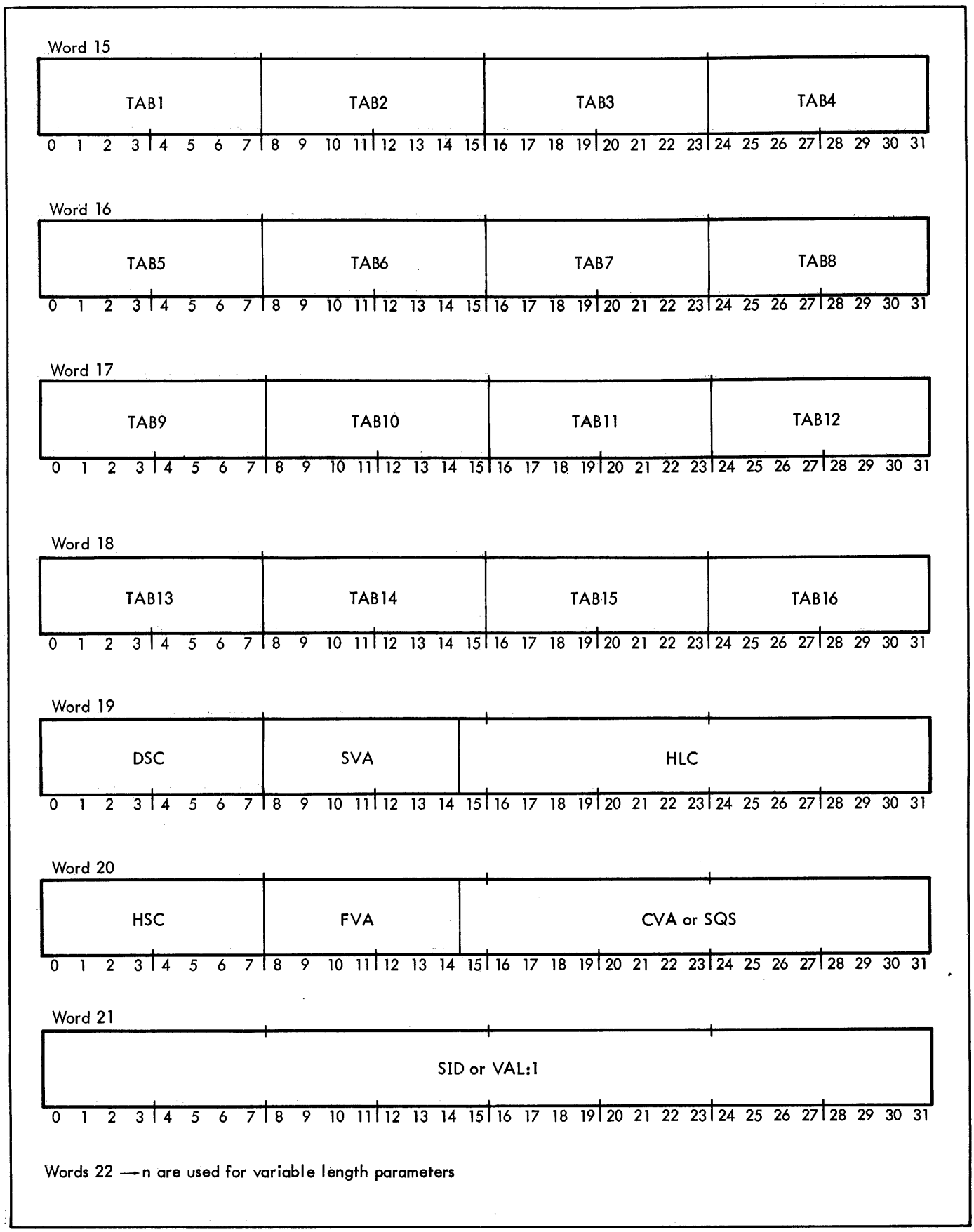


Figure A-3. Format of Device DCB (cont.)

In the following field descriptions, the Control column signifies who specifies the contents of the field – the monitor (M) or the user (U).

Field	Description	Word	Control
ABA	contains the address of the user's routine that will handle abnormal conditions resulting from insufficient or conflicting information. (The monitor returns to ABA in the FPT if the abnormal condition is the result of a device abnormality.)	4	U
ACCT	contains an index signifying the accounting type of the DCB corresponding to the service limit options on the !LIMIT command. (0 - no accounting, 1 - D0, 2 - P0, 3 - U0, 4 - L0.)	12	M
ADR:1	contains an address used internally by the monitor in transaction processing.	11	M
ADR:2	contains an address used internally by the monitor in transaction processing.	14	M
AGE	is a field common to all DCBs and contains J:CALCNT/4 which determines how recently the user used this DCB.	9	M
AGV	is the abnormal given flag and indicates whether or not an end-of-file completion code has been returned to the user because a control command was encountered when reading from the C device, (0 = no, 1 = yes).	0	M
ARS	contains the actual number of data bytes transferred to or from the user in the I/O operation.	4	M
ASN	indicates the assignment type currently in effect for the DCB (0 = null, 1 = file, 2 = Xerox labeled tape, 3 = device, X'A' = ANS labeled tape).	0	U
ASNE	is an ASN extension bit and is used internally by the monitor in transaction processing.	0	M
BLK	contains the number of bytes to be transferred by the I/O routines whenever called.	6	M
BUF	contains the address of the user's buffer where the data record is to be read or written.	2	U
BUEX	is a field common to all DCBs and is cleared to zero by the monitor for device DCBs.	9	M
CCF	specifies whether code conversion is to take place between ASCII on tape and EBCDIC in core (0 = no, 1 = yes).	5	U
CIS	contains the relative position of the serial number (in the SN list) of the magnetic tape reel used for current file input. When the DCB is open, this field is always zero if not assigned to tape.	11	M
CLK	for a nonsymbiont device, contains 0. For a symbiont device, contains the accounting type in bits 20-23 (0 = none, 1 = DO, 2 = PO, 3 = UO, 4 = LO) and the logical device index in bits 24-31.	12	M
COS	contains the relative position of the serial number (in the SN list) of the magnetic tape reel used for current file output. When the DCB is open, this field is always zero if not assigned to tape.	11	M
CSC	indicates the number of the column at which the page count is to begin (for printer or typewriter). The most significant digit of the count will be printed in this column on the page.	14	U
CVA	indicates the current value of the page count (for printer or typewriter).	20	M
DEV	contains the DCT index of the device assigned to the DCB. DEV is only meaningful if DEVF equals 1.	1	M

Field	Description	Word	Control
DEVF	indicates whether the DCB is assigned to a device or an operational label (0 = operational label, 1 = device).	1	M
DIAG	signifies that the DCB is being used for diagnostic purposes.	5	M
DIR	indicates the direction of the read operation (0 = forward, 1 = reverse).	0	U
DRC	is the format control flag and indicates whether or not the monitor is to do special formatting of records on read or write operations (0 = yes, 1 = no).	0	U
DSC	indicates the column number at which the output record is to begin (for a card punch, typewriter, or printer).	19	U
EGV	is the event-given flag and indicates whether or not the completion code posted in the TYC field has been communicated to the user's program by M:CHECK (1 = yes, 0 = no). M:CHECK is called either directly by the user or indirectly by monitor, depending upon the WAIT, ERR, and ABN options in the FPT.	0	M
EOP	is the ending operation indicator (0 = other, e.g., rewind, 1 = read, 2 = write). Specifies the type of I/O operation currently or last performed.	0	M
ERA	contains the address of the user's routine that will handle error conditions resulting from insufficient or conflicting information. (The monitor returns to the ERA in the FPT if the error condition is the result of a device failure.)	3	U
FBCD	is the FORTRAN BCD flag and indicates whether or not BCD is to be converted to EBCDIC on input, or EBCDIC is to be converted to BCD on output. (0=no conversion, 1=conversion.) On write operations, conversion is performed in the user's buffer.	0	U
FCD	indicates whether the DCB is opened or closed (0 = closed, 1 = opened).	0	M
FCI	indicates whether the DCB has ever been closed. This flag is set when the DCB is first closed, and then never reset (0 = DCB has never been closed, 1 = DCB has been previously opened and closed).	0	M
FCN	indicates the current number of I/O operations that have been initiated but not completed, for this DCB.	7	M
FLP	contains the address of the variable length parameters in the DCB (called the file list-pointer) or zero if no space was reserved.	6	U
FUN	contains the file mode function (0 = null, 1 = IN, 2 = OUT, 3 = IN and OUT, 4 = INOUT, 8 = OUTIN).	1	U
FVA	indicates the first line on which printing is to begin (for printer or typewriter).	20	U
HBTD	is the I/O handler's byte indicator and is used whenever the I/O routines are called to specify the byte displacement within QBUF into which the data transfer is to begin.	0	M
HLC	contains the address of the user's page header that is to be output at the beginning of each listing page (the first byte of the page header contains the byte count).	19	U
HSC	indicates the column number at which the user's page header is to begin (for printer or typewriter).	20	U
KBUF	contains the address of buffer for the DCB which is reserved beyond the end of the variable length parameters (8 words). If no space was reserved, KBUF contains zero.	10	U

Field	Description	Word	Control
L	indicates whether or not the user specified that the DCB was assigned to a listing type device. (0 = no, 1 = yes.) This flag is only used by the FORTRAN I/O routines. The monitor automatically sets this flag when the DCB is assigned to a listing type device (such as the line printer).	1	U
LVA	indicates the number of printable lines per logical page (for printer or typewriter). The value = 0 if the stream default is selected.	10	U
MBG	is the monitor buffer-flag and indicates whether or not a 34-word output buffer has been allocated to the DCB from the monitor's buffer pool. (0 = the actual I/O operation will take place directly from the user's buffer, 1 = the output record will be transferred from the user's buffer to the monitor's buffer and that the actual I/O operation will take place using the monitor's buffer.)	0	M
MOD	is the mode flag and indicates the device mode to be used in the I/O operation. (0 = EBCDIC, 1 = binary.) This flag is only used when 1. the DCB is assigned to a card punch or 7-track magnetic tape. 2. the DCB is assigned to a card reader and DRC has been specified.	0	U
NRA	indicates the number of recovery tries that may be attempted before a device error message is to be logged.	2	U
NVA	contains a counter indicating the number of records to skip on magnetic tape. It is also used as an indicator. If NVA is negative, the last operation performed was a rewind.	8	M
PKTC	is used internally by the monitor to handle line cornering and unit record devices. Line cornering is the simulation of a typewriter wherein one record is broken into small records which fit on the platen.	11	M
PUN	indicates whether a 7-track tape is to be read or written in the packed or unpacked mode (0 = unpacked, 1 = packed). PUN is only meaningful when MOD is set.	0	U
QBUF	contains the buffer address to be used by the I/O routines whenever called.	7	M
RNDEV	same as TYPE field.	5	U
RSZ	indicates the default record size, in bytes.	3	U
RWS	indicates the requested number of bytes to be read or written from the user's buffer (BUF).	13	M
SEQ	is the sequence option flag and indicate whether or not punched output is to have sequencing in columns 77-80 (0 = no, 1 = yes).	5	U
SIDF	1. if the DCB is not assigned to tape, SIDF is the sequence identification (ID) flag and indicates whether or not punched output is to have sequence identification in columns 73-76 (0 = no, 1 = yes). 2. if the DCB is assigned to tape, SIDF is the density selection flag for dual density tape drives (0 = 1600 bpi, 1 = 800 bpi).	5	U
SID	contains the 4-byte EBCDIC identification to be output in the sequencing identification field (columns 73-76) of punched card output.	21	U
SQS	indicates the next sequence number to be output in columns 77-80 (for punched card output).	20	M

Field	Description	Word	Control
SVA	indicates the number of lines to be spaced between printed lines (for typewriter or printer). A 0 means SPACE was not specified; the output will be single spaced.	19	U
TAB1-16	indicates the column numbers for the tab-stop settings (for output devices).	15-18	U
TOF	is used by the monitor to remember that the last operation through this DCB occurred at the top-of-page.	0	M
TOLF	if 1, bits 16-31 of DCB are TEXT OPLABEL. If 0, DEVF is meaningful.	1	U
TYC	indicates the type of completion of an I/O operation.	2	M

TYC Code	Corresponding Error/ Abnormal Code	Meaning
0	0	normal without device I/O transfer
1	0	normal with device I/O transfer
2	7	lost data
3	1D	beginning-of-tape
4	4	beginning-of-file
5	1C	end-of-reel
6	5	end-of-data
7	6	end-of-file
8	41	read error
9	45	write error

TYPE	contains the device-type code assigned to the DCB. This field is set whether the DCB is assigned directly to a device or indirectly through an operational label.	1	U
UBTD	is the type displacement indicator, specifying at which byte in the user's buffer (BUF) the data record begins.	0	U
VAL:1	contains a value used internally by the monitor in transaction processing.	21	M
VFC	is the vertical format control-flag and indicates whether or not the first byte of the output is a format control character (0 = no, 1 = yes). This flag is only used for printer output.	0	U
WAT	is the wait flag and indicates whether or not WAIT was specified in the FPT (0 = no, 1 = yes).	0	U

VARIABLE LENGTH PARAMETERS 22→n

Each variable length parameter entry is preceded by a control word of the form shown for File DCB and in Table A-1.

XEROX LABELED TAPE DCB

Figure A-4 shows the format of the DCB for Xerox labeled tape files. Shaded fields are not used by the DCB.

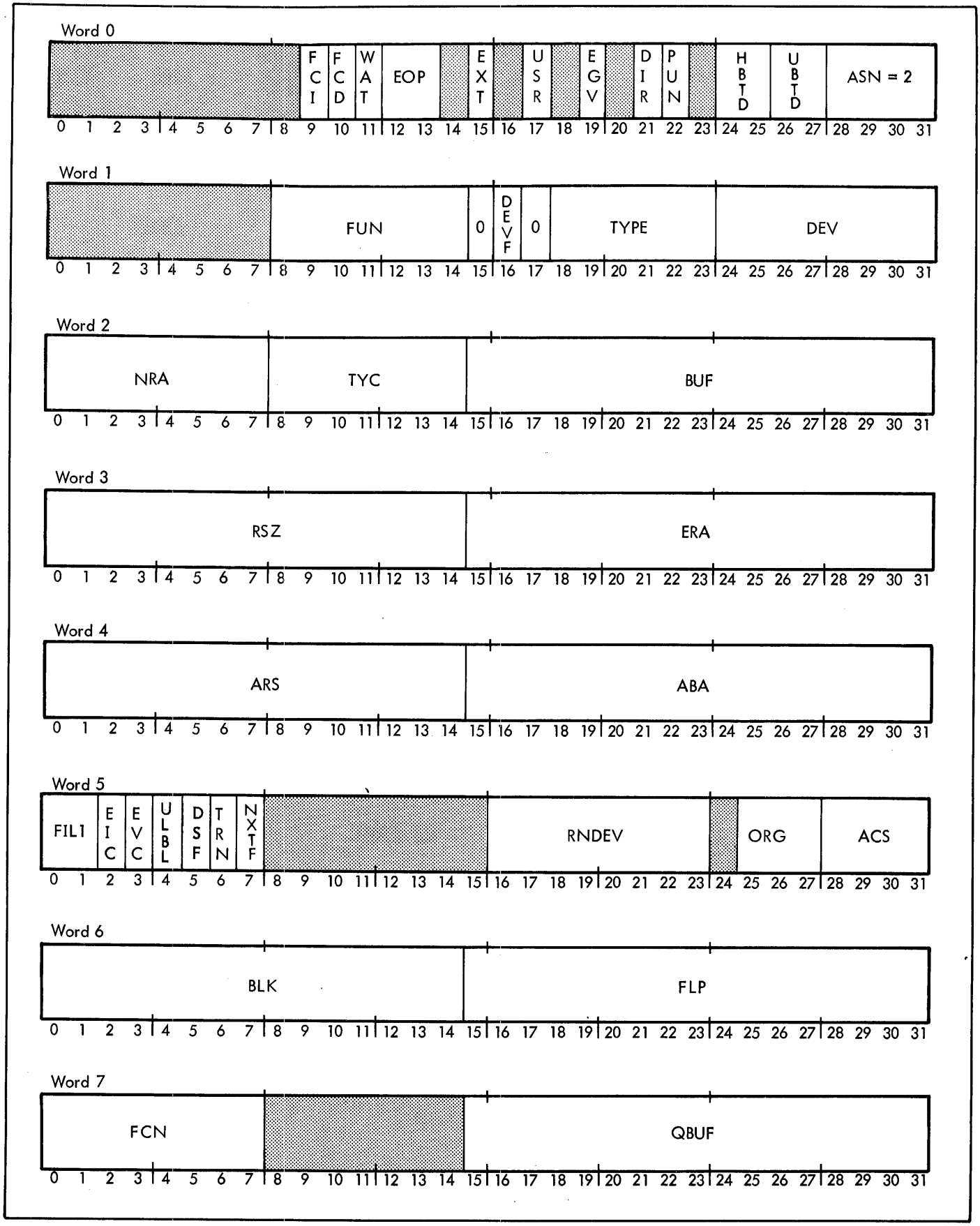


Figure A-4. Format of Xerox Labeled Tape DCB

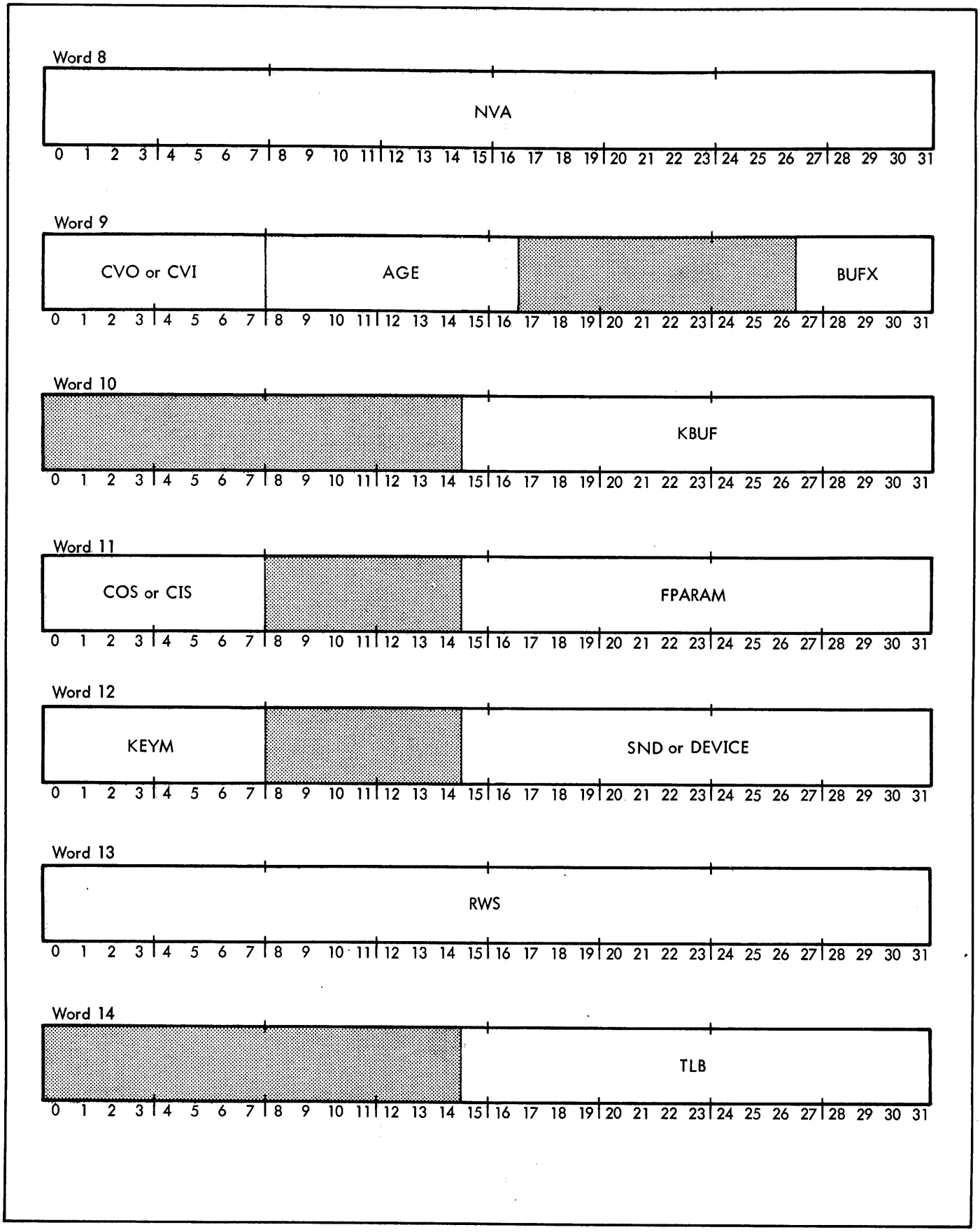


Figure A-4. Format of Xerox Labeled Tape DCB (cont.)

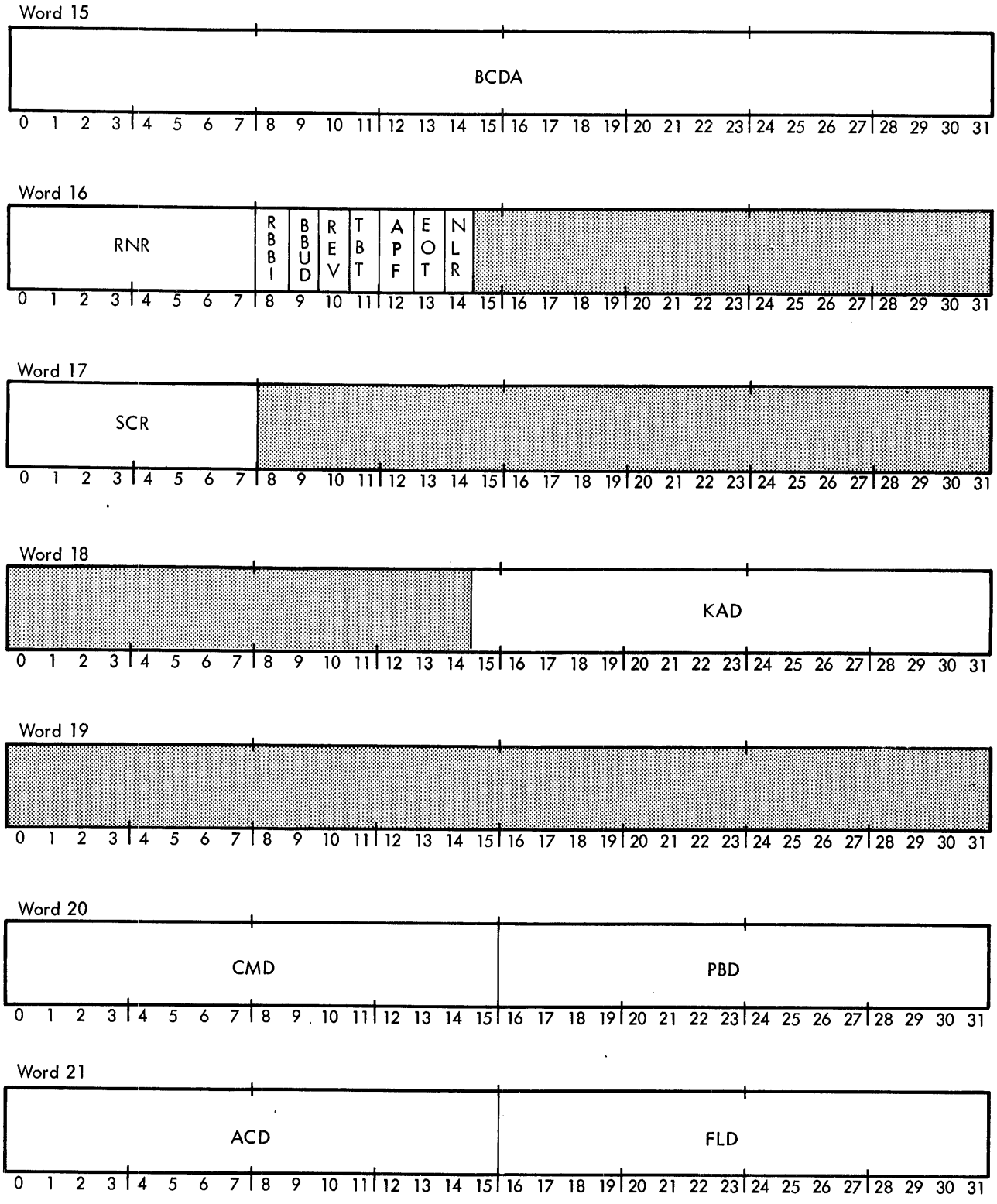


Figure A-4. Format of Xerox Labeled Tape DCB (cont.)

In the following field descriptions, the Control column signifies who specifies the contents of the field – the monitor (M) or the user (U).

Field	Description	Word	Control
ABA	contains the address of the user's routine that will handle abnormal conditions resulting from insufficient or conflicting information. (The monitor returns to ABA in the FPT if the abnormal condition is the result of a device abnormality.)	4	U
ACD	contains the word displacement to the users account number in the DCB relative to the start of the variable length parameters. (FLP + ACD = FWA of the EBCDIC account number.)	21	M
ACS	is the file access indicator (0 = none specified and is treated as sequential, 1 = sequential, 2 = direct). If a file has keyed organization, the keys written must be in ascending order regardless of the access specified.	5	U
AGE	is used to measure the most recent activity on the DCB so that buffer truncation can be made more efficiently.	9	M
APF	contains the ANS post flag. If set, it indicates that post processing of a block accessed record has not yet been done.	16	M
ARS	contains 1. the actual number of data bytes transferred to or from the user following a read or write. 2. the number of records remaining to be skipped following a PRECORD operation that has terminated due to an end-of-file or a beginning-of-file condition.	4	U,M
ASN	indicates the assignment type currently in effect for the DCB (0 = null, 1 = file, 2 = Xerox labeled tape, 3 = device, X'A' = ANS labeled tape).	0	U
BBUD	indicates whether or not the blocking buffer (BUF1) has been changed since it was last read or initialized (0 = unchanged, 1 = changed). The monitor uses this flag to determine whether or not BUF1 needs to be written out to the data granule specified in BCDA before truncating the buffer.	16	M
BCDA	contains the number of either the current or last accessed entry in the blocking buffer (BUF1), depending upon the point in time. An entry in a Labeled Tape block consists of a key, control information, and the associated record segment. Entries are numbered from 1 to n.	15	M
BLK	contains 1. the byte count of the record segment pointed to by either CBD or PBD, depending upon the point in time. 2. the number of bytes to be transferred by the I/O routines whenever called.	6	M
BUF	contains the address of the user's buffer where the data record is to be read or written, or where user trailer labels are to be read.	2	U
BUFX	contains the index of the blocking buffer.	9	M
CIS	contains the relative position of the serial number (in the SN list) of the magnetic tape reel used for current file input.	11	M
CMD	contains the byte displacement to the current entry in the blocking buffer (BUF1). An entry in a Labeled Tape block consists of a key, control information, and the associated record segment.	20	M

Field	Description	Word	Control
COS	contains the relative position of the serial number (in the SN list) of the magnetic tape reel used for current file output.	11	M
CVI	indicates the relative volume number of the current input tape within the current file. CVI is taken from the beginning-of-file sentinel, which appears at the beginning of file and at the beginning of each reel, if the file is continued on more than one reel.	9	M
CVO	indicates the relative volume number of the current output tape with respect to the current file. CVO is recorded in the beginning-of-file sentinel which is written at the beginning of the file and at the beginning of each reel, if the file is continued on more than one reel.	9	M
DEV	contains the DCT index of the device assigned to the DCT. DEV is only meaningful if DEVF = 1. When DEVF = 0, the field is defined as OPLB.	1	U
DEVF	indicates whether the DCB is assigned to a device or an operational label. (0 = operational label, 1 = device.)	1	U
DEVICE	contains the EBCDIC name specified on the DEVICE option in the M:OPEN call. This use is only transient, and the field is later overlaid by SND.	12	U
DIR	indicates the direction of the read operations (0 = forward, 1 = reverse).	0	U
DSF	indicates whether a dual density tape drive is to be written at 1600 bpi or 800 bpi. (0 = 1600, 1 = 800.)	5	U
EGV	is the event-given flag and indicates whether or not the completion code posted in the TYC field has been communicated to the user's program by the CHECK routine (0 = no, 1 = yes). The CHECK routine is called either directly by the user or indirectly by the monitor, depending upon the WAIT, ERR, and ABN options in the FPT.	0	M
EIC	indicates whether or not the last block read from a consecutive file was in error and that a validity check on the control information revealed inconsistencies (0 = no, 1 = yes).	5	M
EOP	is the ending operation indicator (0 = other, e.g., rewind, 1 = read, 2 = write). Specifies the type of I/O operation currently or last performed.	0	M
EOT	indicates whether or not the physical end-of-tape mark has been encountered (0 = no, 1 = yes).	16	M
ERA	contains the address of the user's routine that will handle error conditions resulting from insufficient or conflicting information. (The monitor returns to the ERA in the FPT if the error condition is the result of the device failure.)	3	U
EVC	indicates whether or not the last block read from a consecutive file was in error but a validity check on control information revealed no inconsistencies (0 = no, 1 = yes).	5	M
EXT	is the file extension flag and indicates whether OPEN is to position a tape at the beginning or end of a specified file (0 = beginning-of-file, 1 = end-of-file).	0	M
FCD	indicates whether the DCB is opened or closed (0 = closed, 1 = opened).	0	M
FCI	indicates whether the DCB has ever been closed. This flag is set when the DCB is first closed and then never reset (0 = DCB has never been closed, 1 = DCB has been previously open and closed).	0	M

Field	Description	Word	Control
FCN	indicates the current number of I/O operations that have been initiated but not completed, for this DCB.	7	M
FILI	indicates the file option specified when the DCB was last opened (0 = none specified, 1 = release, 2 = save).	5	U
FLD	contains the word displacement to the file name in the DCB relative to the start of the variable length parameters (FLD + FLP = FWA of the EBCDIC file name).	21	M
FLP	contains the address of the variable length parameters in the DCB (called the file list-pointer).	6	M
FPARAM	contains the receiving address of the user's 90-word buffer to which the variable length parameters from the file's FIT are to be passed.	11	U
FUN	indicates the file mode function (0 = null, 1 = IN, 2 = OUT, 4 = INOUT, 8 = OUTIN).	1	U
HBTD	is the I/O handler's byte indicator and is used whenever the I/O routines are called to specify the byte displacement within QBUF into which the data transfer is to begin.	0	M
KAD	contains the address of the key specified by the user in the read or write FPT. If a consecutive file is being written, KAD points to the dummy key. If a consecutive file is being read, KAD contains 0.	18	U
KBUF	contains the address of the buffer containing the key associated with the data record last accessed in the blocking buffer.	10	M
KEYM	contains the maximum length, in bytes, of the keys in the file pointed to by the DCB. Only meaningful for keyed files. Maximum value is 31.	12	U
NLR	indicates whether or not the record segment pointed to by CMD is the first record in a continued data record (0 = second or nth record segment, 1 = first or only record segment). NLR is only meaningful during a write and is reset to zero when the first record segment is output.	16	M
NRA	indicates the number of recovery tries that may be attempted before a device error message is to be logged.	2	U
NXTF	is the next file indicator and specifies whether this file (i.e., the file named in the DCB/FPT) or the next file in the File Directory (i.e., the one following the file named in the DCB) is to be assigned to the DCB at OPEN. If a file name is not specified (in either the DCB or FPT), the first name in the File Directory is put in the DCB and assigned (0 = this file, 1 = next file).	5	U
NVA	contains a counter indicating the number of records to skip. It is also used as an indicator. If NVA is negative, the last operation performed was a rewind.	8	M
ORG	is the file organization indicator (0 = none specified, and is treated as consecutive, 1 = consecutive, 2 = keyed).	5	U
PBD	contains 1. a counter used by M:OPEN to determine how many volumes remain to be searched for the specified file. 2. the number of bytes in the previous labeled tape block. PBD is only meaningful on a read operation and is taken from the PBS field of a labeled block.	20	M

Field	Description	Word	Control
PUN	indicates whether a 7-track tape is to be read/written in the packed or unpacked mode (0 = unpacked, 1 = packed).	0	U
QBUF	contains 1. the buffer address to be used by the I/O routines whenever called. 2. the address within the user's buffer where the next record segment begins.	7	M
RBBI	indicates whether or not the blocking buffer should be released at end-action (0 = release blocking buffer, 1 = do not release blocking buffer because the buffer will be reused to read in the next block). RBBI is set during a read operation when a data record is continued and more than one read request will be initiated.	16	M
REV	indicates whether the Labeled Tape block currently in the blocking buffer (BUF1) was read in the forward or reverse direction (0 = forward, 1 = reverse).	16	M
RNDEV	contains the type of device specified (0 = none specified, 8 = 9T, 9 = 7T, X'A' = MT).	5	U
RNR	is a transient flag used by the system to defer error reporting for a tape block read by the monitor in anticipation of a read not yet requested by the user (0 = user requested read, 1 = user read not requested).	16	M
RSZ	indicates the default record size, in bytes.	3	U
RWS	indicates the requested number of bytes to be read or written from the user's buffer (BUF). At the termination of the I/O operation, RWS is set equal to ARS.	13	M
SCR	indicates the byte length of the key portion of the entries in the Labeled Tape block.	17	M
SND	contains the word displacement to the tape serial number (SN list) in the DCB relative to the start of the variable length parameters (FLP + SND = FWA of the EBCDIC serial numbers).	12	M
TBT	indicates whether or not the Labeled Tape blocking buffer has been truncated (0 = no, 1 = yes). Truncation means that monitor has taken the blocking buffer and, if necessary, written the block on tape.	16	M
TLB	contains the address of a user's label that is to be written on a tape file when the file is output.	14	U
TRN	indicates whether the file is positioned before or after the data record whose key entry is pointed to by CMD (0 = after, 1 = before).	5	M
TYC	indicates the type of completion of an I/O operation.	2	M

TYC Code	Corresponding Error/ Abnormal Code	Meaning
0	0	normal without device I/O transfer
1	0	normal with a device I/O transfer
2	7	lost data
3	1D	beginning-of-tape
4	4	beginning-of-file
5	1C	end-of-reel
6	5	end-of-data
7	6	end-of-file
8	41	read error
9	45	write error

Field	Description	Word	Control
TYPE	contains the device-type code for the tape assigned to this DCB.	1	U
UBTD	is the byte displacement indicator, specifying at which byte in the user's buffer (BUF) the data record begins.	0	U
ULBL	indicates whether or not the ULBL option was specified in the FPT of M:READ (0 = no, 1 = yes).	5	U
USR	indicates whether or not the job account number is the same as the account number specified in the DCB (0 = yes, 1 = no).	0	M
WAT	is the wait flag and indicates whether or not WAIT was specified in the FPT (0 = no, 1 = yes).	0	U
VARIABLE LENGTH PARAMETERS		22→n	

Each variable length parameter entry is preceded by a control word of the form shown for File DCB and in Table A-1.

ANS LABELED TAPE DCB

Figure A-5 shows the format of the DCB for ANS Labeled Tape files. Shaded fields are not used by the DCB.

In the following field descriptions, the Control column signifies who specifies the contents of the field – the monitor (M) or the user (U).

Field	Description	Word	Control
ABA	contains the address of the user's routine that will handle abnormal conditions resulting from insufficient or conflicting information. (The monitor returns to ABA in the FPT if the abnormal condition is the result of a device abnormality.)	4	U
ABCERR	indicates whether or not block count errors are to be accepted; i.e., whether or not processing is to continue in the case of inconsistency between the tape-specified and system-accumulated block counts (0 = no, 1 = yes).	0	U
ACS	is the file access indicator (only 3, block, is possible for ANS tape).	5	M
APF	contains the ANS post flag. If set to 1, it indicates that ANS post-processing of an I/O operation has not yet been done.	16	M
ARS	contains 1. the actual number of data bytes transferred to or from the user following a read or write. 2. the number of records remaining to be skipped following a PRECORD operation that has terminated due to an end-of-file or a beginning-of-file condition.	4	U,M
ASN	indicates the assignment type currently in effect for the DCB (0 = null, 1 = file, 2 = Xerox labeled tape, 3 = device, X'A' = ANS labeled tape).	0	U
BCERR	indicates whether or not a block count error has been detected during EOF/EOT processing (0 = no, 1 = yes). Always cleared before returning to user.	0	U
BLK	contains the number of bytes to be transferred by the I/O routines whenever called.	6	M
BLKCNT	specifies the number of blocks in the file.	17	U

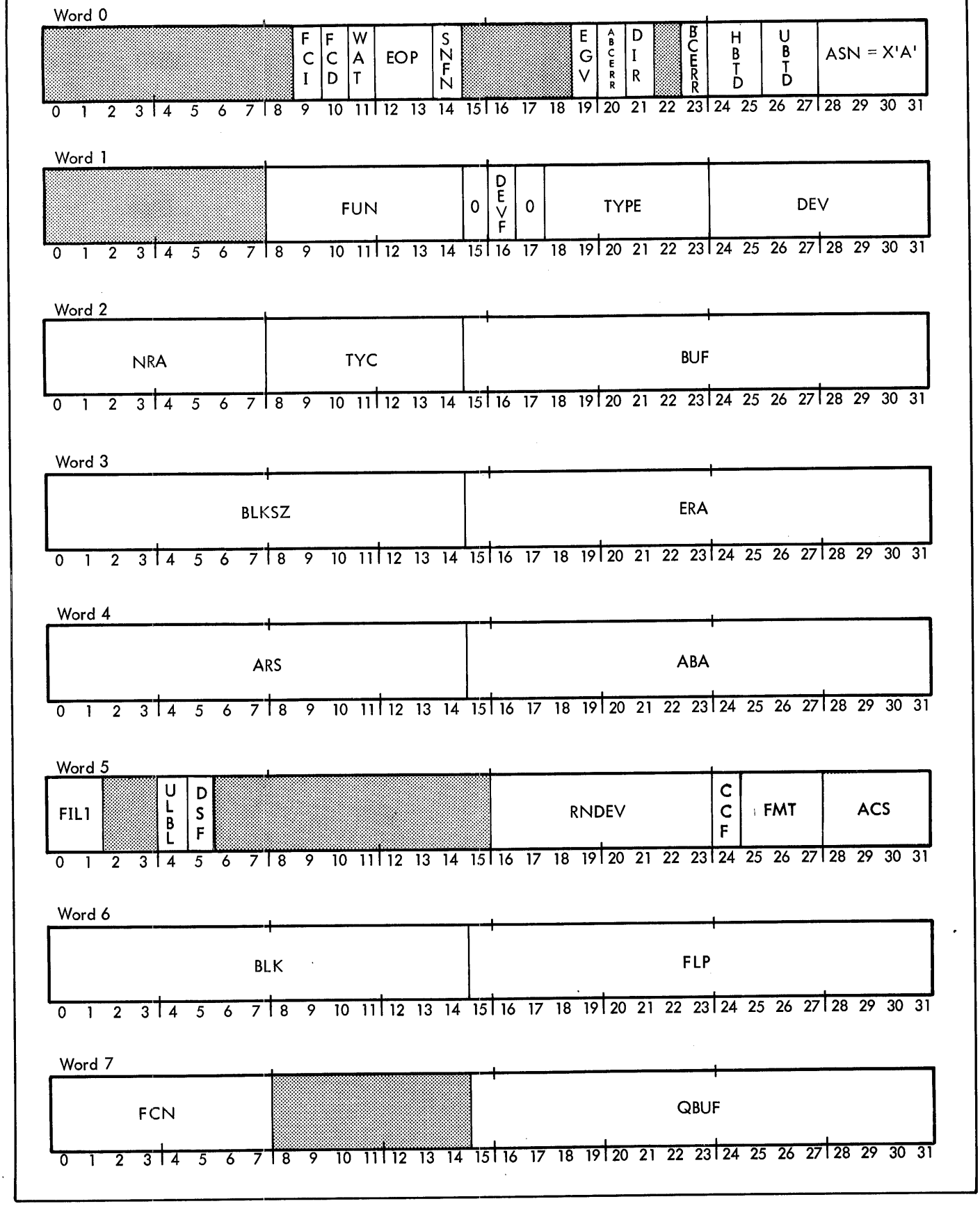


Figure A-5. Format of ANS Labeled Tape DCB

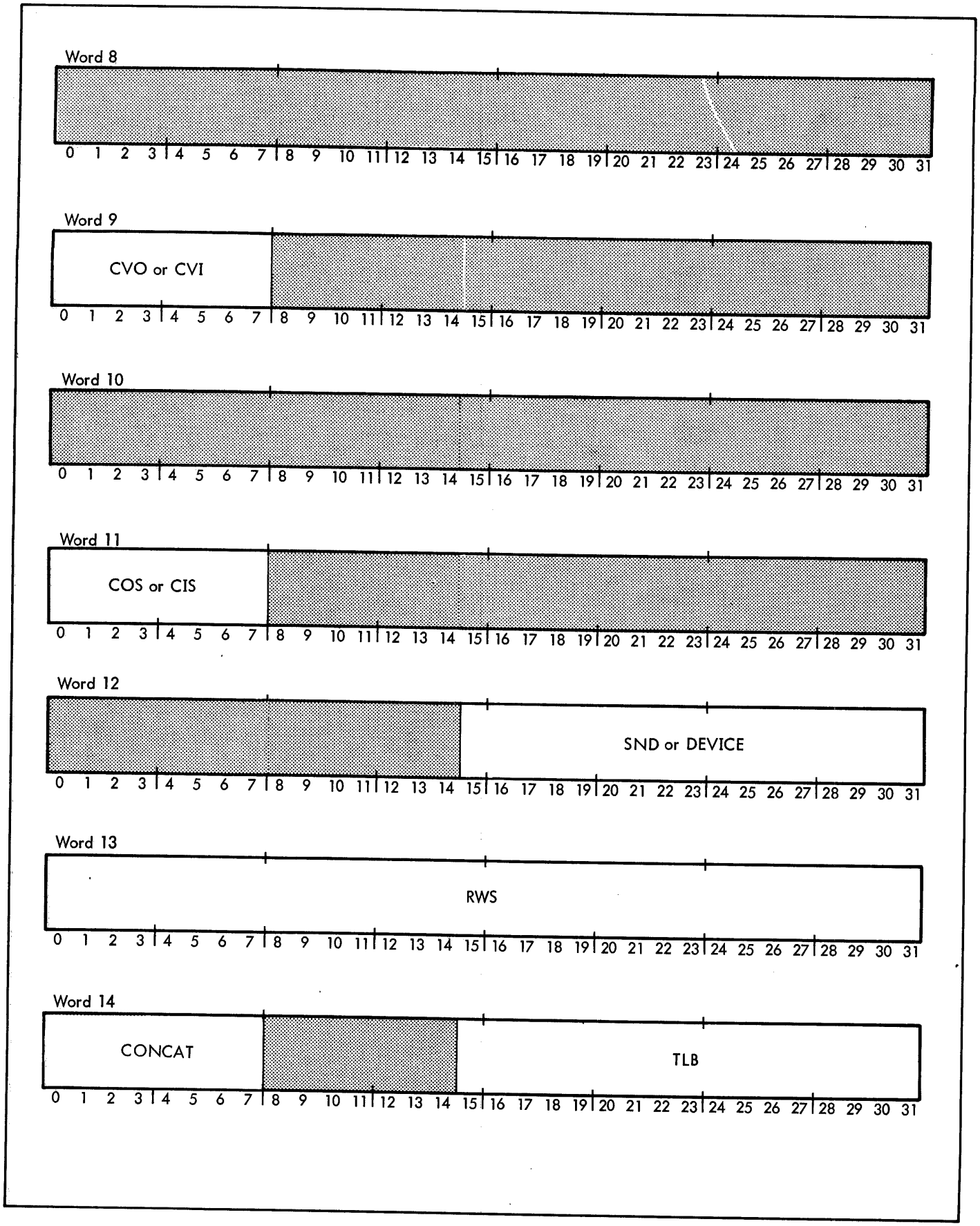


Figure A-5. Format of ANS Labeled Tape DCB (cont.)

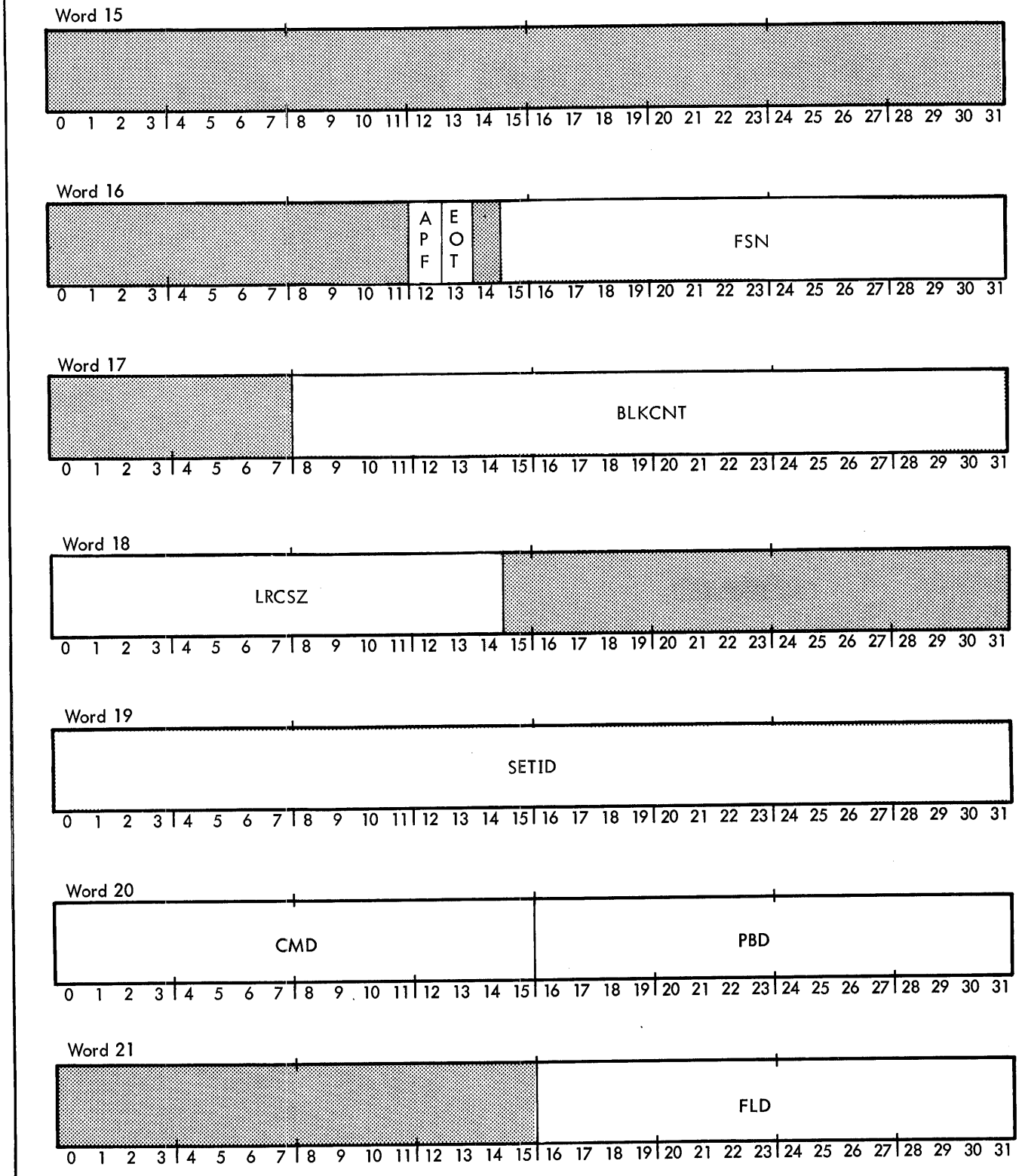


Figure A-5. Format of ANS Labeled Tape DCB (cont.)

Field	Description	Word	Control
BLKSZ	specifies the block size in bytes.	3	U
BUF	contains the address of the user's buffer where the data record is to be read or written, or where user trailer labels are to be read.	2	U
CCF	specifies whether code conversion is to take place between ASCII on tape and EBCDIC in core (0 = no, 1 = yes).	5	U
CIS	contains the relative position of the serial number (in the SN list) of the magnetic tape reel used for current file input.	11	M
CMD	contains the number of tape marks that may be passed during an OPEN while searching the last tape of a set.	20	M
CONCAT	specifies the number of identically named files that are to be read as one logical file (concatenation.)	14	U
COS	contains the relative position of the serial number (in the SN list) of the magnetic tape reel used for current file output.	11	M
CVI	indicates the relative volume number of the current input tape within the current file. CVI is taken from the beginning-of-file sentinel, which appears at the beginning of file and at the beginning of each reel, if the file is continued on more than one reel.	9	M
CVO	indicates the relative volume number of the current output tape with respect to the current file. CVO is recorded in the beginning-of-file sentinel which is written at the beginning of the file and at the beginning of each reel, if the file is continued on more than one reel.	9	M
DEV	contains the DCT index of the device assigned to the DCT. DEV is only meaningful if DEVF = 1. When DEVF = 0, the field is defined as OPLB.	1	U
DEVF	indicates whether the DCB is assigned to a device or an operational label. (0 = operational label, 1 = device.)	1	U
DEVICE	contains the EBCDIC name specified on the DEVICE option in the M:OPEN call. This use is only transient, and the field is later overlaid by SND.	12	U
DIR	indicates the direction of the read operations (0 = forward, 1 = reverse).	0	U
DSF	indicates whether a dual density tape drive is to be written at 1600 bpi or 800 bpi (0 = 1600, 1 = 800).	5	U
EGV	is the event-given flag and indicates whether or not the completion code posted in the TYC field has been communicated to the user's program by the CHECK routine (0 = no, 1 = yes). The CHECK routine is called either directly by the user or indirectly by the monitor, depending upon the WAIT, ERR, and ABN options in the FPT.	0	M
EOP	is the ending operation indicator (0 = other, e.g., rewind, 1 = read, 2 = write). Specifies the type of I/O operation currently or last performed.	0	M
EOT	indicates whether or not the physical end-of-tape mark has been encountered (0 = no, 1 = yes).	16	M
ERA	contains the address of the user's routine that will handle error conditions resulting from insufficient or conflicting information. (The monitor returns to the ERA in the FPT if the error condition is the result of the device failure.)	3	U
FCD	indicates whether the DCB is opened or closed (0 = closed, 1 = opened).	0	M

Field	Description	Word	Control
FCI	indicates whether the DCB has ever been closed. This flag is set when the DCB is first closed and then never reset (0 = DCB has never been closed, 1 = DCB has been previously open and closed).	0	M
FCN	indicates the current number of I/O operations that have been initiated but not completed, for this DCB.	7	M
FIL1	indicates the file option specified when the DCB was last opened (0 = none specified, 1 = release, 2 = save).	5	U
FLD	contains the word displacement to the file name in the DCB relative to the start of the variable length parameters (FLD + FLP = FWA of the EBCDIC file name).	21	M
FLP	contains the address of the variable length parameters in the DCB (called the file list-pointer).	6	M
FMT	indicates the record format, where 1 = F (fixed length) 2 = D (variable, expressed in decimal) 3 = V (variable, expressed in binary) 4 = U (undefined)	5	U
FSN	specifies the file sequence number.	16	U
FUN	indicates the file mode function (0 = null, 1 = IN, 2 = OUT, 4 = INOUT, 8 = OUTIN).	1	U
HBTD	is the I/O handler's byte indicator and is used whenever the I/O routines are called to specify the byte displacement within QBUF into which the data transfer is to begin.	0	M
LRCSZ	specifies the logical record size in bytes.	18	U
NRA	indicates the number of recovery tries that may be attempted before a device error message is to be logged.	2	U
PBD	contains 1. a counter used by M:OPEN to determine how many volumes remain to be searched for the specified file. 2. the block count according to EOF1 or EOV1.	20	M
QBUF	contains 1. the buffer address to be used by the I/O routines whenever called. 2. the address within the user's buffer where the next record segment begins.	7	M
RNDEV	contains the type of device specified (0 = none specified, 8 = 9T, 9 = 7T, X'A' = MT).	5	M
RWS	indicates the requested number of bytes to be read or written from the user's buffer (BUF). At the termination of the I/O operation, RWS is set equal to ARS.	13	M

Field	Description	Word	Control
SETID	specifies the file set identification.	19	U
SND	contains the word displacement to the tape serial number (SN list) in the DCB relative to the start of the variable length parameters (FLP + SND = FWA of the EBCDIC serial numbers).	12	M
SNFN	indicates the access method (0 = serial number, 1 = filename).	0	M
TLB	contains the address of a user's label that is to be written on a tape file when the file is output.	14	U
TYC	indicates the type of completion of an I/O operation.	2	M

TYC Code	Corresponding Error/ Abnormal Code	Meaning
0	0	normal without device I/O transfer
1	0	normal with a device I/O transfer
2	7	lost data
3	1D	beginning-of-tape
4	4	beginning-of-file
5	1C	end-of-reel
6	5	end-of-data
7	6	end-of-file
8	41	read error
9	45	write error

TYPE	contains the device-type code for the tape assigned to this DCB.	1	M
UBTD	is the byte displacement indicator, specifying at which byte in the user's buffer (BUF) the data record begins.	0	U
ULBL	indicates whether or not the ULBL option was specified in the FPT of M:READ (0 = no, 1 = yes).	5	U
WAT	is the wait flag and indicates whether or not WAIT was specified in the FPT (0 = no, 1 = yes).	0	U

VARIABLE LENGTH PARAMETERS 22→n

Each variable length parameter entry for ANS labeled tapes is preceded by a control word of the following form:

Byte 0 = a code number (see Table A-2) identifying the parameter which follows.

Byte 1 = code for the entry position (00 = more parameter entries to follow, 01 = last parameter entry).

Byte 2 = number of significant data words in the parameter entry.

Byte 3 = total number of words reserved for the entry, not including the control word (that is, maximum entry length).

Table A-2. Variable Length Parameter Codes for ANS Labeled Tapes

Code	Parameter Type
01	File name (the first byte of which contains the number of characters in the name).
04	Expiration date.
07	SN/INSN serial numbers. (ANS serial numbers are encoded to fit in 32 bits.)
08	OUTSN serial numbers. (ANS serial numbers are encoded to fit in 32 bits.)

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:

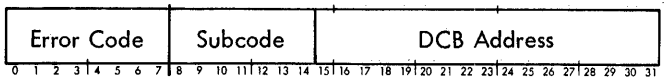
1. Those associated with insufficient or conflicting information.
2. Those associated with device failures or end-of-data conditions.

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



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	OPND	Conflicting or missing DCB information. Probably either no file name is specified or the file name TEXTC count is illegal.
14	07	OPND	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 B1. 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 MOVE CAL is not allowed for device or ANS DCBs.
1A	05	MOVECAL (RDL)	The MOVE CAL 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 or READ	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, a bad command has been sent to the terminal, or a 0 byte COC read has been issued.
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	Time parameter too large on M:CHECKECB.
4A	02	IOCHECK	ECB in wrong state.
4A	03	IOCHECK	Infinite wait condition.
4A	04	IOCHECK	No monitor work space.
4A	05	IOCHECK	Wrong access code for ECB address.
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	ANS block count error and no ABCERR specified.
4E	01	READ or CVOL	A volume sequence number error occurred on an ANS tape.
4E	04	LBLT	A 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	User should not simulate that trap.
A4	00		User is trapped.
A4	01	TRAP	Trap 40 - Nonexistent instruction.
A4	02	TRAP	Trap 40 - Nonexistent memory reference.

Table B-5. Other Monitor Error Codes (cont.)

Error Code	Sub-code	Originating Monitor Routine	Meaning of Code
A4	03	TRAP	Trap 40 - Privilege 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.
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	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.

Table B-5. Other Monitor Error Codes (cont.)

Error Code	Sub-code	Originating Monitor Routine	Meaning of Code
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.)
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 area 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 A6) subcodes and I/O error codes.

Table B-5. Other Monitor Error Codes (cont.)

Error Code	Sub-code	Originating Monitor Routine	Meaning of Code
B5	62	LDLNK	M:LINK and M:LDTRC are not permitted when a shared processor is associated with the user program.
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 aquired 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.

Table B-5. Other Monitor Error Codes (cont.)

Error Code	Sub-code	Originating Monitor Routine	Meaning of Code
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.
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. 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 C-1. CP-V 8-Bit Computer Codes (EBCDIC)

Least Significant Digits: Hexadecimal	Binary	Most Significant Digits: 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
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	L	ESC Z	⌡	≤	d	m	u	[[1]	D	M	U	4
5	0101	HT	LF NL	EM	ESC (	€	T			e	n	v	] [1]	E	N	V	5
6	0110	ACK	SYN [8]	/	ESC)		o	ω	≥	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 Data [6]				
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 C-2. CP-V 7-Bit Communication Codes (ANSCII)

Least Significant Digits: Decimal (rows)	Binary	Most Significant Digits: 0 (x000)	1 (x001)	2 (x010)	3 (x011)	4 (x100)	5 (x101)	6 (x110)	7 (x111)
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 ∧. It also interprets the [ ] characters as | ¬ .)

5 On the Xerox 7670 Remote Batch Terminal, the symbol

! is | (2-1)
[is ¢ (5-11)
] is ! (5-13)
^ 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 C-3. CP-V Symbol-Code Correspondences

EBCDIC[t] Hex.	EBCDIC[t] 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/ Printers Models 7012, 7020, 8091, and 8092.
09	9	ENQ	12-9-8-1	0-5	enquiry	
0A	10	NAK	12-9-8-2	1-5	negative acknowledge	
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 output only. These codes are duplicates of the label entries that caused activation. The 20-2F entries output a single code only and are not affected by any special COC functional processing.
21	33	FS	0-9-1	1-12		
22	34	GS	0-9-2	1-13		
23	35	RS	0-9-3	1-14		
24	36	US	0-9-4	1-15		
25	37	EM	0-9-5	1-9		
26	38	/	0-9-6	2-15		
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 special functions.
31	49	CANCEL	9-1		delete all input and output	
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 C-3. CP-V Symbol-Code Correspondences (cont.)

EBCDIC† Hex.	EBCDIC† 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	∈	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	On 2741 APL, ¢ is ⊂ (subset).
4C	76	<	12-8-4	3-12	less than	
4D	77	(	12-8-5	2-8	left parenthesis	
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	81		12-11-9-1			51, 57, 58, and 59 are unassigned.
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	T	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 ∪ (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	⍵	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 ⍴.
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 C-3. CP-V Symbol-Code Correspondences (cont.)

EBCDIC† Hex.	EBCDIC† Dec.	Symbol	Card Code	ANSCII††	Meaning	Remarks
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 !.
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			B0 and B7 through BB are unassigned.
B8	184		12-11-0-8			
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).

†Hexadecimal and decimal notation.

††Decimal notation (column-row).

Table C-3. CP-V Symbol-Code Correspondences (cont.)

EBCDIC† Hex.	EBCDIC† 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
C2	194	B	12-2	4-2		uppercase alphabet.
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 C-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 C-4. ANSCII Control-Character Translation Table (cont.)

Input					Output	
ANSCII	TTY Key	Echoed	Prog. Receives (EBCDIC)	Process	EBCDIC	Transmitted (ANSCII)
SYN (16)[t]	V[c]	SYN	SYN	None	SYN[t] (16)	SYN (not transmitted if last character in user's buffer).
ETB (17)[t]	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.	

[t]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 C-5. Substitutions for Nonexistent Characters on 2741 Keyboards

EBCDIC Character	APL Keyboard	Selectric Keyboard	EBCD Keyboard
>	>	, (upper case)	>
<	<	. (upper case)	<
^	↑	¢	¢
|	|	° (degree)	|
¬	~	±	¬
#	≠	#	#
%	ρ	%	%
¢	⊂	¢	¢
@	α	@	@
"	∇	"	"
!	∘	!	!
&	∩	&	&
$	∪	$	$

APPENDIX D. USE OF TEMPORARY STORAGE BY LIBRARY ROUTINES

All standard system library routines are entered by a BAL instruction, using current general register 11 as a link register. Arguments are passed to the library routine through current general registers 6 through 9. Current general register 0 contains a pointer to a Task Control Block (TCB) established and maintained by the monitor. The first two words of a TCB comprise a stack pointer doubleword, and the subsequent words contain additional information used by the monitor to control the current task.

The library routine can make register contents available for use by pushing them into the temp stack set by the monitor for the current job. Other kinds of temporary data also can be saved in the temp stack (e.g., trap return information).

One means of storing data in the temp stack is to push the data into the stack, by means of push instructions (PSM and PSW); another is to set aside a block of storage in the stack, by using an MSP instruction, so that data can then be stored in the block by the use of store instructions.

All storage used in the temp stack must be released before the associated routine exits. Storage can be released by a pull (PLM or PLW) or MSP instruction. All registers except those used to return output information must remain unchanged. Note that all push, pull, and MSP instructions must be done indirectly, through current general register 0.

The following examples show two different methods of placing data in temporary storage.

```
R0      EQU     0
R6      EQU     6
        :
        :
        LCI     4
        PSM,R6  *R0
```

The preceding example causes the contents of current general registers 6, 7, 8, and 9 to be pushed into the temp stack.

```
R7      EQU     7
BLOCK1  EQU     100
        :
        :
        LI,R6   BLOCK1
        MSP,R6  *R0
        LW,R7   *R0
```

The preceding example reserves a 100-word block in the temp stack. The address of the top of the block is contained in R7. When the block has been reserved, data can then be stored in it, using the method illustrated below.

```
R3      EQU     3
BLW1    EQU     -99
        :
        :
        STW,R3  BLW1,R7
```

In the example given above, the contents of current general register 3 are stored in the first word of the reserved block.

The area reserved for use by the temp stack is established by a LOAD control command TSS specification or is set at 64 words by default.

APPENDIX E. COOPERATIVES AND SYMBIONTS

In CP-V the routines to perform peripheral operations for unit-record peripherals operate concurrently with the jobs being run. The peripheral system is composed of a "cooperative" and a "symbiont" or symbionts. The cooperative is a monitor routine called as result of a user's I/O request, whereas a symbiont is a monitor routine that is initiated either by the action of the cooperative or by operator command from the system console. The cooperative is used to transfer information between the user's program and secondary (disk) storage, and symbionts are used to transfer information between secondary storage and peripheral devices (see Figure E-1 and Table E-1).

The symbiont-cooperative system provides for complete buffering between I/O devices and the user's program. Therefore, a user's program never has to wait for an I/O device to complete an action. Also, the current job may be running while the output of the previous job and the job file for the following job are being handled by symbiont operation.

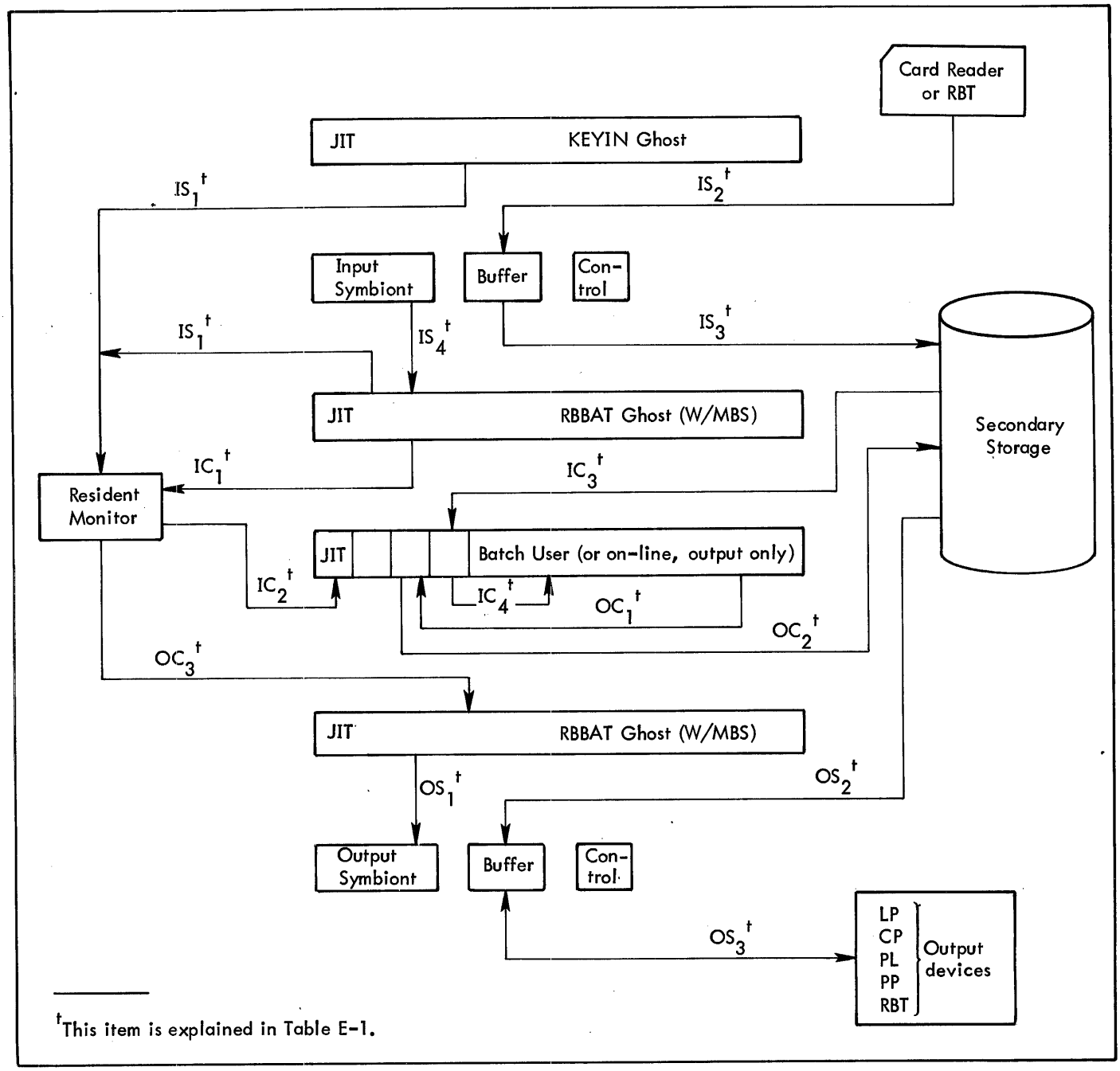


Figure E-1. Information Flow Through Cooperative and Symbionts

Table E-1. Cooperative and Symbionts Descriptions

Name In Figure E-1	Type	Description
IS_1	Input Symbiont	Input symbiont activated by KEYIN or RBBAT via resident tables.
IS_2	Input Symbiont	Card images read into symbiont buffer.
IS_3	Input Symbiont	When buffer is full, contents are written to secondary storage and linked to previous blocks of the same file.
IS_4	Input Symbiont	Continues until end-of-file or IFIN, then notifies RBBAT and terminates.
IC_1	Input Cooperative	RBBAT/MBS selects job to run by putting job information into resident tables.
IC_2	Input Cooperative	When JOB card read, job information is transferred to JIT.
IC_3	Input Cooperative	Input symbiont file blocks are read into user's cooperative buffer.
IC_4	Input Cooperative	Records are transferred, one at a time, in response to user reads.
OC_1	Output Cooperative	User symbiont output is intercepted and put into cooperative buffers.
OC_2	Output Cooperative	When buffer is full, contents are written to secondary storage.
OC_3	Output Cooperative	User issues 'superclose' and file is queued by RBBAT.
OS_1	Output Symbiont	RBBAT initiates output symbiont.
OS_2	Output Symbiont	Output symbiont blocks are read into symbiont buffer.
OS_3	Output Symbiont	Records are transferred, one at a time, to output device. Continues processing blocks until end-of-file, then starts another file or terminates.

COOPERATIVE

A single cooperative is provided for handling both user input and output files. It is reenterable and can handle any number of device-type files (printer, punch, etc.) per job.

SYMBIONTS

A symbiont is a small, reenterable routine that controls the action of a symbiont dedicated I/O device having a lower transfer rate than secondary storage. Core storage as well as secondary storage will be used by the symbiont to produce a continuous flow of information to or from these devices. Symbionts will transfer information from a peripheral device to the disk and from the disk to a peripheral device.

Unlike the user's program, which is directed primarily by control commands, symbionts - once initiated - receive all their control from the operator's console. An input symbiont device can be initiated only by the console operator, while an output symbiont device can be initiated either by the operator or the cooperative.

For each device, a symbiont performs only one I/O operation at a time (chaining is not used) and is inactive from the time that it initiates a request for I/O until the I/O operation is complete. The symbiont regains control by stipulating an I/O end-action return to itself.

Since symbionts are reenterable, a single symbiont may drive several types of devices. For example, the same symbiont may be used to drive many printers and card punches. All the peripheral-dependent information is contained in a context buffer. The location of this buffer is made known to the symbiont whenever it is operating on the associated device.

Two symbionts are provided in the monitor system: one for driving all standard input devices, and one for driving all standard output devices.

SYMBIONT-COOPERATIVE HOUSEKEEPING

Two monitor subroutines are provided for automatic maintenance of core storage. One is used to release a core buffer after use by the symbionts, and the other is used to obtain a core buffer.

If a core buffer is requested by a symbiont and none is available, an entry is made in a symbiont core-buffer queue. When one becomes available for symbiont use, control is returned to the requesting symbiont.

As each buffer is emptied, either by reading from or storing into secondary storage, it is released. This procedure allows for efficient utilization of core buffers.

The symbiont routines themselves will be executed in resident monitor space. After starting an I/O operation on a peripheral device, with an end-action return specified, the symbiont relinquishes control to the monitor system.

An area of secondary storage is set aside for symbiont files. The size of this area is an installation variable set up at System Generation time. A secondary storage allocation table is maintained by the monitor to indicate which disk areas are available. Two monitor subroutines are also provided for maintenance of secondary storage. One of these requests storage; the other releases it. If secondary storage is requested and none is available, an entry is made in a symbiont secondary storage queue. Where subsequent I/O information is read by the cooperative or a symbiont (operating on another device), secondary storage is released and control is returned to the symbiont requesting secondary storage.

Secondary storage holds the files produced by, or committed to a peripheral device. Each disk block of secondary storage contains the disk address of the next disk block in the file, and a table of job files is maintained by the monitor. Monitor subroutines are provided for symbiont input and output file maintenance. One removes a file; the other inserts a new file into the file table.

When preparing to output a file, the output cooperative places an appropriate entry in the file directory.

SYMBIONT BUFFERS

There are two symbiont file buffers:

1. Input symbiont file buffer.

2. Output symbiont file buffer.

In CP-V, both input and output symbiont file buffers have the same format (Figure E-2). Each such block contains 256 words, and two blocks reside in a granule of file storage. Word 0 of the block is used for the forward link address that is inserted by the system when the file is created. A value of zero implies no forward address (i.e., end of file). Word 255 is used for the backward link address, again inserted when the file is created. A value of zero implies no backward address (that is, beginning of file). Each record in the block is preceded by four bytes of control information. Neither the record nor the control information need start on a word boundary except the first control string. Each control string must immediately follow the preceding record. The first two bytes of a control string are the byte count (BC) of the following record. BC must be greater than zero and less than 1008. No record may be split between blocks. If a block does not have space remaining for a block end control string, a record control string, and a record, the next record must begin in a new block. The third byte of a control string is the record control character (RCC) which defines the record. RCC may have the following values:

0 = a BCD record (e.g., card).

1 = an EOD record (e.g., !EOD).

2 = a binary record (BIN).

4 or 5 = a PRINT record without a vertical format control character.

6 or 7 = a PRINT record, the first byte of which is a vertical format control character.

X'40' = a block ending control string (e.g., no more records this block).

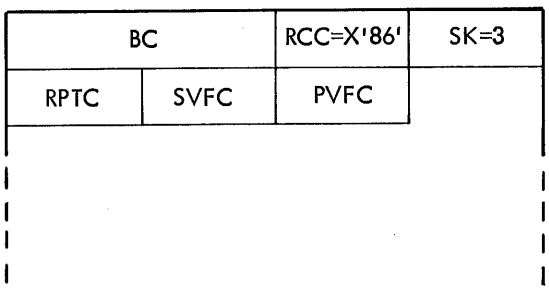
X'86' = a nonbatch banner. (The record will usually be repeated enough times to fill two print pages.) When the RCC value is X'86' the symbiont file buffer has three additional fields that are not shown in Figure E-1. They are each one byte in length and are:

RPTC specifies the number of times this record is to be printed plus one.

SVFC specifies the secondary VFC character. The character is used for the second and each succeeding repeated print (e.g., X'C1' for doublespace).

PVFC specifies the primary VFC character to be used on the first print (e.g., X'F1' for top-of-form).

For Example:



Other values for RCC are reserved for future enhancement and should not be used. The fourth byte of a control string is the skip byte (SK) defined for the convenience and efficiency of the block encoder. SK may have the value 1 through 4 inclusive.

The next SK-1 bytes following the control sequence have no significance and are skipped before the start of data. The skipped bytes are provided to allow a byte-aligned MBS instruction (the most efficient execution) to move the bytes into the symbiont block or to allow placement of the record on a word boundary for record construction ease. The final control string of a block must have BC=0, RCC=X'40', and SK=0.

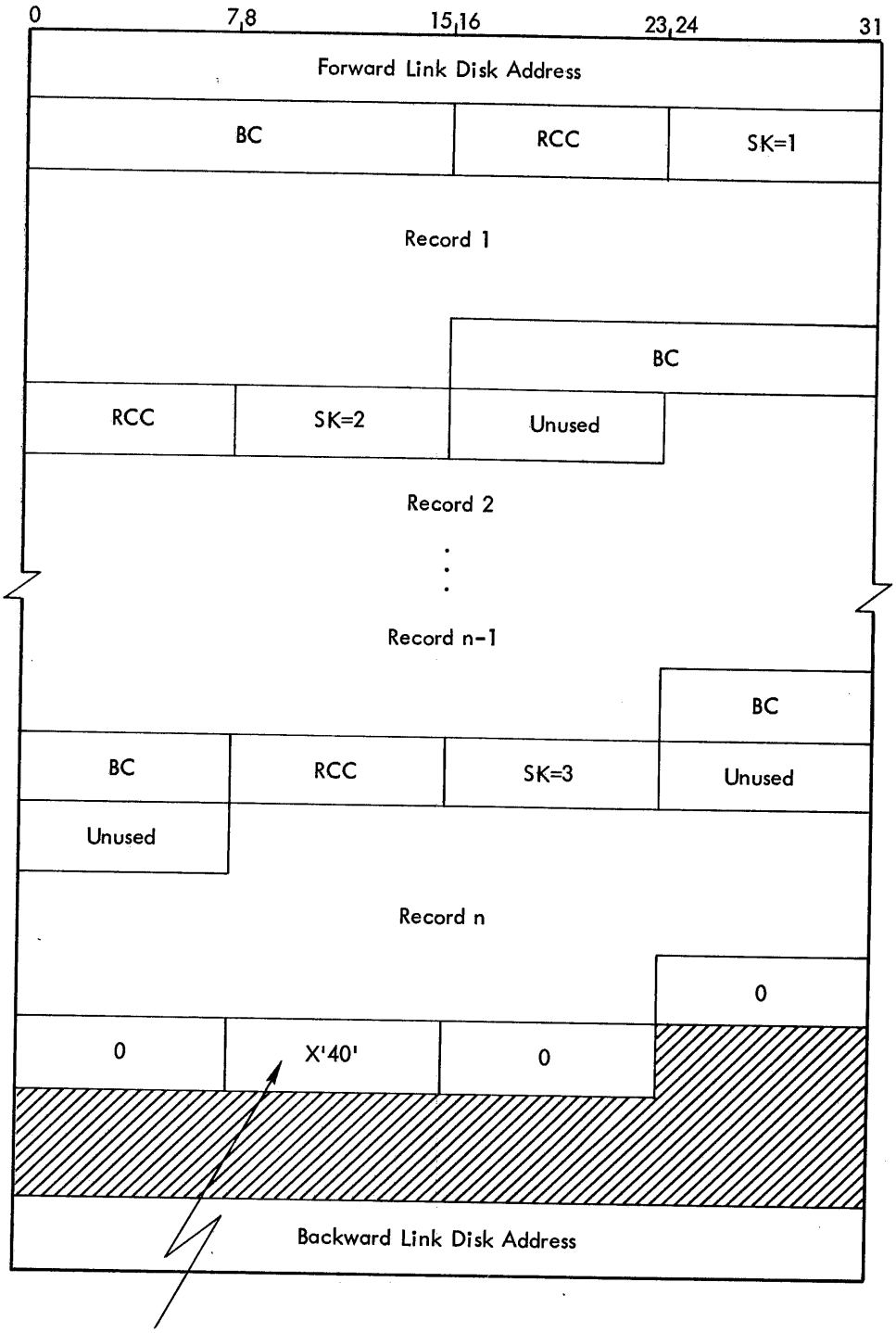


Figure E-2. Symbiont File Buffer Format

Appendix F has been deleted. Information regarding simultaneous file usage is contained in Section 2, Files and File Usage.

(This page intentionally left blank.)

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.

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.

D

E

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.

F

G

H

I

J

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.

K

L

M

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.

N

O

P

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.

R

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.

S

T

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.

U

V

W

X

Y

XEROX

Publication Revision Sheet

September 1978

CORRECTIONS TO CP-V/BATCH PROCESSING REFERENCE MANUAL

PUBLICATION NO. 90 17 64H-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, 21/22, 23/23.1, 23.2/24, 25 through 40, 47/48, 51 through 54, 57 through 62, 65/66, 75/76, 83 through 90, 91 through 108, 111/112, 115 through 118, 121 through 124, 127 through 134, 137/138, 143 through 150, 153/154, 183 through 202, 203/203.1, 203.2/204, 205/206, 221/222, 233/234, 251 through 262, 279/280, and 283 through 286. (Pages 23.1, 23.2, 203.1, and 203.2 are new pages.)

Pages that are to be inserted are: 90.1/90.2.

Revision bars in the margins of replacement pages identify changes. Pages without the publication number 90 17 64H-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.

23408
1579
Printed in U.S.A.

File No.: 1X13
XL89A, Rev. 0

XEROX

Reader 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

FOLD ALONG LINE

PLEASE FOLD AND TAPE—
NOTE: U. S. Postal Service will not deliver stapled forms

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: 2025 Sheppard Avenue East, Willowdale, Ontario M2J 1W5
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.

27994, 3.5C880, Printed in U.S.A.

XL89, Rev. 0