# Xerox Control Program-Five (CP-V)

Xerox 560 and Sigma 6/7/9 Computers

## System Programming Reference Manual

90 31 13A

Xerox Corporation
701 South Aviation Boulevard
El Segundo, California 90245
213 679-4511

XEROX

# Xerox Control Program-Five (CP-V)

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

## System Programming

## Reference Manual

90 31 13A
90 31 13A-1

May 1975

Price: $8.25

© Xerox Corporation, 1974, 1975

Printed in U.S.A.

# REVISION

This publication documents the C01 version of Control Program-Five (CP-V). The publication consists of the A edition of this manual (90 31 13A, dated October 1974) and the revision package numbered 90 31 13A-1(5/75). Vertical lines in the margins of pages labeled 90 31 13A-1(5/75) indicate changes that reflect the C01 version of CP-V.

# RELATED PUBLICATIONS

| Title | Publication No. |
|---|---|
| Xerox Control Program-Five (CP-V)/TS Reference Manual | 90 09 07 |
| Xerox Control Program-Five (CP-V)/TS User's Guide | 90 16 92 |
| Xerox Control Program-Five (CP-V)/OPS Reference Manual | 90 16 75 |
| Xerox Control Program-Five (CP-V)/BP Reference Manual | 90 17 64 |
| Xerox Control Program-Five (CP-V)/TP Reference Manual | 90 31 12 |
| Xerox Control Program-Five (CP-V)/RP Reference Manual | 90 30 26 |
| Xerox Control Program-Five (CP-V)/SM Reference Manual | 90 16 74 |
| Xerox Control Program-Five (CP-V)/Common Index | 90 30 80 |
| Xerox EASY/LN, OPS Reference Manual | 90 18 73 |
| Xerox BASIC/Reference Manual | 90 15 46 |
| Xerox Extended FORTRAN IV/LN Reference Manual | 90 09 56 |
| Xerox Extended FORTRAN IV/OPS Reference Manual | 90 11 43 |

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

| Title | Publication No. |
|---|---|
| Xerox Extended FORTRAN IV/Library Technical Manual | 90 15 24 |
| Xerox FORTRAN Debug Package (FDP)/Reference Manual | 90 16 77 |
| Xerox FLAG/Reference Manual | 90 16 54 |
| Xerox Meta-Symbol/LN, OPS Reference Manual | 90 09 52 |
| 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 Manage/Reference Manual | 90 16 10 |
| Xerox APL/LN, OPS Reference Manual | 90 19 31 |
| Xerox Sort-Merge/Reference Manual | 90 11 99 |
| Xerox 1400 Series Simulator/Reference Manual | 90 15 02 |
| Xerox Sigma 5/7 Mathematical Routines/Technical Manual | 90 09 06 |
| 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 CIRC-DC/Reference Manual and User's Guide | 90 16 97 |
| Xerox CIRC-AC/Reference Manual and User's Guide | 90 16 98 |
| Xerox CIRC-TR/Reference Manual and User's Guide | 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.

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

# CONTENTS


PREFACE ix

COMMAND SYNTAX NOTATION x

GLOSSARY xi

1. INTRODUCTION 1

CP-V Services 1
Time-Sharing and Batch Processing 1
Remote Processing 2
Transaction Processing 2
Real-Time Processing 2
System Programming Facilities 2

2. SYSTEM OVERVIEW 3

Introduction 3
Processors 3
Command Processors 3
Language Processors 5
Execution Control Processors 8
Service Processors 9
Application Processors 10
User Processors 12
Monitor 12
Scheduling and Memory Management 13
Scheduler Operation 15
System Integrity 19

3. BOOTSTRAP AND PATCHING OPERATIONS 23

System Tape Format 23
Patch Deck Structure 23
Delta Format Patches 23
Reconfiguration and Partitioning Commands 26
:GO 26
:SAVE 27
:TYPE 27
:REMOVE 27
:PART 28
:END 28
:GENDCB Command 32
GENMD Commands 33
GENMD 33
LIST 33
DELETE 33
GENMD Patches 33
GENMD Error Messages 33
Conditional Patch Control Commands 33
Comment Cards 34
Patch File Creation 35
Sequence of Operations 35
Booting From Disk 37
Bootstrap I/O Error Recovery 38
PASS0 Processor 38
PASS0 Messages 38

4. MONITOR DUMP ANALYSIS PROGRAM 39

Introduction 39
Ghost Mode 39
Batch and On-Line Modes 39
Commands 39
Input Command 40
INPUT 40
Display Commands 40
DISPLAY 40
RUN 40
ALL 40
Interactive Monitor Display Commands 40
loc 40
$loc_1$,$loc_2$ 43
LINE FEED 43
↑ 43
* 43
MONITOR 43
loc = value 43
Map Commands 43
MAP 43
UNMAP 43
Search Commands 43
COMPARE 43
SMASK 44
SEARCH 44
Output Commands 44
ROWS 44
LP 44
UC 44
PRINT 44
Debug Commands 44
BF 44
DELTA 44
NODELTA 44
Miscellaneous Commands 45
SYMBOLS 45
IS 45
SYMBOL 45
DUMP 45
CLOSE 45
HELP 45
Exit Commands 45
END 45
Output 45
ANLZ Messages 54
ANLZ Command Summary 54

5. ERROR MESSAGE FILE 59

Introduction 59
Format of Error Message File 59
Creating Error Message File 59
Card Reader Input 59
Terminal Input 60

6. SYSTEM ERROR LOG FILE 61

Introduction 61
ERR:FILL Program 61
Error Log Listing Processor 61
Starting Execution 61
Input/Output Assignments 61
SET 61
Input/Output Characteristics 63
Interrupting ELLA Execution 64
ELLA Commands 64
CLIS 64
SLIS 71
SUM 75
DISP 75
END 76
RSET 76
TIME 77
TYPE 78
DEV 78
MOD 78
DSPL 81
Predefined Tasks 81
ELLA Messages 84
ELLA Command Summary 84
Hardware-Error Diagnostic CALs 85
Read Error Log 86
Write Error Log 86
Initiate Ghost Job 86

7. SHARED PROCESSOR FACILITIES 87

Introduction 87
Public Programs 87
Shared Programs 87
Log-On Connection 87
Shared Processor Programming 88
Fixed Monitor Locations 88
Job Information Table (JIT) 88
Memory Control 89
Overlay Restrictions 89
Data Control Blocks 90
File Identification 91
TEL Scan 92
CCI Scan 93
Terminal I/O 93
File Extension 95
Shared File Use 95
Command Processor Programming 95
Public Libraries 98
CP-V Public Libraries 98
Creating Public Libraries 98
Loading Public Libraries 98
Shared Processor Maintenance 102
ENTER 102
REPLACE 103
DELETE 103
LIST 103
LISTALL 103
? 104
END 104
DRSP Limitations and Restrictions 104
DRSP Error Messages 104
DRSP Command Summary 104

8. ON-LINE PERIPHERAL DIAGNOSTIC FACILITIES 108

Introduction 108
Restrictions 108
PSECT Directive 108
System Procedures 109
Create Diagnostic Data Control Block 109
M:DDCB 109
Open Diagnostic Data Control Block 110
M:DOPEN 110
Close Diagnostic Data Control Block 111
M:DCLOSE 111
Build Command List 111
M:BLIST 111
Start I/O 112
M:SIO 112
Lock in Core 113
M:LOCK 113
Convert Address 113
M:MAP 113
Obtain Model Numbers and Type Mnemonics 113
M:DMOD# 113
Abnormal Codes and Messages 114
DDCB 114

9. REAL-TIME PROCEDURES 119

Interrupt Connection and Control Services 119
M:GJOBCON 120
Connect User Program to Interrupt 120
M:CONNECT 120
Disconnect User Program or Ghost Job from Interrupt 121
M:DISCONNECT 121
Control an Interrupt 122
M:INTCON 122
General Interrupt Inhibit 122
M:INHIBIT 122
Return from Interrupt Processing 123
M:INTRTN 123
Queue for Interrupt 123
M:QFI 123
Obtain Interrupt Status 124
M:INTSTAT 124
Lock in Core Service 124
M:HOLD 124

Clock Service ____ 125
M:CLOCK ____ 125
Device Preemption Services ____ 125
Preempt Device ____ 125
M:STOPIO ____ 125
Return Preempted Device ____ 127
M:STARTIO ____ 127
Direct I/O Services ____ 127
IOEX Services ____ 127
M:IOEX (SIO) ____ 127
M:IOEX (HIO/TIO/TDV) ____ 129
Execute Privileged Instruction Service ____ 129
M:EXU ____ 129
Enter Master Mode ____ 129
M:MASTER ____ 129
Enter Slave Mode ____ 129
M:SLAVE ____ 129
PSECT Directive ____ 130
Virtual/Physical Address Conversion ____ 130
M:MAP ____ 130
Miscellaneous Real-Time Services ____ 130
Get or Free Physical Page ____ 130
M:GPP ____ 130
M:FPP ____ 130
Initiate Ghost Job ____ 130
M:GJOB ____ 130
Get and Release Disk Granule ____ 131
M:GDG ____ 131
M:RDG ____ 131
Report User Event ____ 131
M:RUE ____ 131
Check Interrupt Status ____ 132
M:CHKINT ____ 132
I/O Services ____ 132
M:EXCP ____ 133
M:NEWQ ____ 133
M:QUE ____ 135
M:COC ____ 136
Dynamic Physical Page Allocation for Real-Time Processing ____ 136
Introduction ____ 136
SYSGEN Considerations ____ 136
Initialization ____ 137
The Physical Page Stealer Ghost Job (PPS) ____ 137
DISPLAY ____ 138
GET ____ 138
FREE ____ 138
DYNRESDF ____ 139
RESDF ____ 139
END ____ 140
Monitor DEFs ____ 140
RESDF Memory CAL ____ 140

10. TRANSACTION PROCESSING FACILITIES 141

M:QUEUE Procedure Format ____ 141
M:QUEUE Function Parameter Tables (FPTS) ____ 142
Queue UNLOCK Request ____ 142
Queue DEFINELIST Request ____ 142
Queue PUT Request ____ 142
Queue GET Request ____ 143
Queue STATS Request ____ 143
Queue PURGE Request ____ 143
Queue LOCK Request ____ 143
List Formats ____ 143
DEFINELIST or STATS List ____ 143
GET Message ____ 144
PUT List ____ 144
M:QUEUE Procedure Output Parameters ____ 144
SR1 Information ____ 144
ECB Information ____ 145
Queue Error Codes ____ 145

INDEX 213

## APPENDIXES

A. OPERATIONAL LABELS 147

B. PHYSICAL DEVICE NAMES 148

C. CP-V SCREECH CODES 149

D. XEROX 560 REMOTE ASSIST STATION 165

Introduction ____ 165
Hardware Interface ____ 165
Software Interface ____ 165
Processor Restrictions ____ 165
Communications Restrictions ____ 165

E. ERRFILE Formats 168

F. XEROX STANDARD OBJECT LANGUAGE 185

Introduction ____ 185
General ____ 185
Source Code Translation ____ 185
Object Language Format ____ 186
Record Control Information ____ 186
Load Items ____ 187
Declarations ____ 187
Definitions ____ 189
Expression Evaluation ____ 190
Formation of Internal Symbol Tables ____ 193
Loading ____ 194
Miscellaneous Load Items ____ 195
Object Module Example ____ 195

G. XEROX STANDARD COMPRESSED LANGUAGE 201

H. XEROX STANDARD SYMBOLS, CODES AND CORRESPONDENCES 202

Xerox Standard Symbols and Codes ____ 202
Xerox Standard Character Sets ____ 202
Control Codes ____ 202
Special Code Properties ____ 202

## FIGURES


1. CP-V Operating System ____ 3
2. Typical User Program – Virtual Memory Layout (not to scale) ____ 20
3. Typical Memory Layout for Sigma Computers (not to scale) ____ 20
4. Typical Memory Layout for the Xerox 560 (not to scale) ____ 21
5. Format of Master System Tape ____ 24
6. Segment Patching Order ____ 25
7. Device Resource Configuration from SYSGEN ____ 29
8. Reconfiguration and Partitioning Commands that were Ignored ____ 29
9. Reconfiguration and Partitioning Commands that were Used ____ 30
10. Device Resource Configuration for the Booted System ____ 30
11. Special Processors – Virtual Memory ____ 87
12. Locations Common to all Monitors ____ 88
13. Public Library Creation Process ____ 99
14. Generalized Library Load Process (Link) ____ 100
15. Generalized Library Load Process (Load) ____ 101
16. Format of the DDCB ____ 115
17. I/O Operation Codes for Device Handler (M:QUE) ____ 135


## TABLES


1. Event Inputs Received by Scheduler ____ 14
2. Service Request Input to Monitor ____ 15
3. Scheduler Status Queues ____ 16
4. Swap-In and Swap-Out Queues ____ 17
5. Reconfiguration and Partitioning Messages ____ 30
6. GENMD Error Messages ____ 34
7. PASS0 Messages ____ 38
8. INPUT Command Options ____ 40
9. DISPLAY Command Options ____ 41
10. RUN Command Options ____ 42
11. Displays ____ 45
12. Trap and Interrupt Locations for XPSD Instructions ____ 48
13. User Table Headings ____ 48
14. Additional User Table Headings ____ 49
15. Resource Wait Queues ____ 49
16. Swap Table Terms ____ 49
17. Partition Table Headings ____ 49
18. Processor Table Headings ____ 50
19. ALLYCAT Headings ____ 50
20. I/O Table Headings ____ 51
21. Device Control Table Headings ____ 51
22. IOQ Table Headings ____ 52
23. COC Line Table Headings ____ 52
24. AVR Table Headings ____ 53
25. Symbiont Table Headings ____ 53
26. TSTACK Headings ____ 53
27. ANLZ Messages ____ 54
28. ANLZ Command Summary ____ 55
29. ELLA On-Line I/O Functions ____ 63
30. ELLA Batch I/O Functions ____ 63
31. ELLA Ghost I/O Functions ____ 64
32. Error Log Entry Headings ____ 66
33. Error Log Entry Types ____ 71
34. ELLA Messages ____ 84
35. ELLA Command Summary ____ 85
36. Partial Contents of JIT ____ 89
37. Standard DCBs ____ 91
38. Routines in :LIB Library File ____ 99
39. DRSP Error Messages ____ 104
40. DRSP Information Messages ____ 106

41. DRSP Command Summary ____ 107
42. On-Line Diagnostics Abnormal Messages ____ 114
43. Register Settings for End-Action Routines ____ 128
44. M:QUEUE Error Subcodes ____ 145
A-1. Standard Operational Labels and Default Device Assignments ____ 147
A-2. Batch Assignment of Operational Labels ____ 147
A-3. On-Line Assignment of Operational Labels ____ 147
B-1. Standard I/O Device Type Codes ____ 148
B-2. Sigma IOP Designation Codes ____ 148
B-3. Xerox 560 Cluster/Unit Matrix ____ 148
B-4. Device Designation Codes ____ 148
D-1. ASCII to EBCDIC Translate Table for Remote Assist Station ____ 166
E-1. Error Record Terminology ____ 168
E-2. Xerox 7670 RBT – RP1, RP3 and RP4 ____ 179
E-3. Xerox 7670 RBT – RP2 ____ 180
E-4. IBM 2780 RBT – RP1 and RP4 ____ 180
E-5. IBM 2780 RBT – RP2 and RP3 ____ 180
E-6. IRBT – RP1 and RP4 ____ 181
E-7. IRBT – RP2 and RP3 ____ 182
H-1. CP-V 8-Bit Computer Codes (EBCDIC) ____ 203
H-2. CP-V 7-Bit Communication Codes (ANSCII) ____ 204
H-3. CP-V Symbol-Code Correspondences ____ 205
H-4. ANSCII Control-Character Translation Table ____ 209

## EXAMPLES

1. Batch Operation of ELLA ____ 62
2. On-Line Operation of ELLA ____ 62
3. Use of the CLIS Command ____ 65
4. Use of the SLIS Command ____ 73
5. Use of the SUM Command ____ 75
6. Use of DISP Command ____ 76
7. TIME Command Usage ____ 77
8. Use of the MOD, DEV, and TYPE Commands ____ 79
9. Use of the MOD, DEV, and TYPE Commands ____ 79
10. Use of the MOD, DEV, and TYPE Commands ____ 80
11. Use of the MOD, DEV, and TYPE Commands ____ 80
12. Use of the MOD, DEV, and TYPE Commands ____ 81
13. Parameter Display ____ 82
14. Listing the Entire Error File ____ 82
15. Listing Errors for the Current Day ____ 83
16. Listing Start-Ups, Configuration, and Device Partitioning Activity ____ 83

# PREFACE

This manual describes the CP-V features that are designed to aid the system programmer in the development, maintenance, and modification of the CP-V system.

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

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

- The CP-V Time-Sharing Reference Manual, 90 09 07, is the principal source of information for the time-sharing features of CP-V. 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 Remote Processing Reference Manual, 90 30 26, is the principal source of information about the remote processing features of CP-V. All information about remote processing for all computer personnel (remote and local users, system managers, remote site operators, and central site operators) is included in the manual.

- The CP-V 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 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 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.<br><br>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.<br><br>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.<br><br>[KEYM] means the term "KEYM" may be entered. |
| { } | Elements placed one under the other inside a pair of braces identify a required choice.<br><br>{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.<br><br>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.<br><br>MASK2 DATA,2 X'1EF'<br>⋮<br>BYTE DATA,3 BA(L(59))<br><br>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.<br><br>(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.<br><br>$sysid_1$,$sysid_2$,$sysid_3$ means that three successive values for sysid should be entered, separated by commas. |
| Superscripts | Superscripts indicate shift keys to be used in combination with terminal keys. c is control shift, and s is case shift.<br><br>$L^{cs}$ means press the control and case shift (CONTROL and SHIFT) and the L key. |
| Underscore | All terminal output is underscored; terminal input is not.<br><br>!RUN means that the exclamation point was sent to the terminal, but RUN was typed by the terminal user. |
| (ESC) (RET) (LF) | These symbols indicate that an ESC ((ESC)), carriage return ((RET)), or line feed ((LF)) character has been sent.<br><br>!EDIT (RET) means that, after typing EDIT, a carriage return character has been sent. |

# GLOSSARY

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

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

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

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

control 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 the information used by the monitor in the performance of an I/O operation.

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

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

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

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

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

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

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

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

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

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

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

language processor — a program that translates a user's source language program into an object language program.

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 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 modules (LMs) as input information.

logical device — a peripheral device that is represented in a program by an operational label (e.g., 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 language 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 Link processor.

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

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

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.

patch — a symbolic representation of a correction to the system that is used to temporarily correct the system without necessitating a reassembly.

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

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

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

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".)

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

reentrant — an attribute of a program that allows the program to be shared by several users concurrently. Shared processors in CP-V are 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.

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).

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 during SYSGEN or via DRSP.

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

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

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

stream-id the name of a logical device stream.

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

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

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

SYSGEN see "system generation", below.

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

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

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

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

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

# 1. INTRODUCTION

## CP-V SERVICES

Control Program-Five (CP-V) is a comprehensive operating system designed for use with Sigma 6/7/9 and Xerox 560 computers and a variety of peripheral equipment. CP-V offers:

- On-line time-sharing, batch processing, remote processing, transaction processing, and real-time services.
- Ability to handle a large number of concurrent users.
- High efficiency due to hardware relocation map, shared reentrant processors, multiple I/O processors, and device pooling.
- A complete recovery system coupled with preservation of user files to provide fast restart following hardware or software malfunction.
- For on-line users: highly efficient and extensive software, file saving feature, fast response time.
- For batch users: on-line entry, local and remote entry to an efficient multiprogramming batch job scheduler.
- For installation managers: thorough system monitoring and reporting, control and tuning ability, extensive error checking and recovery features.
- For all users: comprehensive accounting and a complete set of powerful processors.

### TIME-SHARING AND BATCH PROCESSING

CP-V allows multiple on-line terminal users to concurrently create, debug, and execute programs. Concurrent to time-sharing, CP-V allows up to 16 batch processing jobs to execute in its multiprogramming environment. An efficient multi-batch scheduler selects batch jobs for execution according to priority, job requirements, and availability of resources. Batch jobs may be submitted to this scheduler from a local batch entry device such as a card reader, from an on-line user's terminal, or from a remote site such as a remote batch terminal or another computer.

Time-sharing and batch users have access to a variety of powerful and comprehensive language processors and facilities. These processors and facilities are listed below.

| Processor | Function |
|---|---|
| TEL | Executive language control of all terminal activities. (On-line only.) |
| EASY | Creation, manipulation, and execution of FORTRAN and BASIC programs and data files. (On-line only.) |
| Edit | Composition and modification of programs and other bodies of text. (On-line only.) |
| FORTRAN IV | Compilation of Extended FORTRAN IV programs. |
| COBOL | Compilation of ANS COBOL programs. |
| Meta-Symbol | Assembly of high-level assembly language programs. |
| BASIC | Compilation and execution of programs or direct statements written in an extended BASIC language. |
| APL | Interpretation and execution of programs written in the APL language. |
| FLAG | Compilation of fast "load-and-go" FORTRAN programs. |
| FDP | Debugging of Extended FORTRAN IV programs. |
| Delta | Debugging of programs at the assembly language level. (On-line only.) |
| COBOL On-line Debugger | Debugging of ANS COBOL programs. (On-line only.) |
| PCL | Transfer (and conversion) of data between peripheral devices. |
| Link | Linkage of programs for execution. |
| Load | Linkage of programs for execution. (Batch only.) |
| Batch | Submission of batch jobs via an on-line terminal or another batch job. |
| Manage | File retrieval, updating, and reporting. |
| SL-1 | Compilation of programs written in a language designed specifically for digital or hybrid simulation. |
| CIRC | Analysis of electronic circuits. |

| Processor | Function |
|---|---|
| EDMS | Organization, storing, updating, and deletion of information in a centralized data base. |
| Sort/Merge | Sorting and/or merging of records in one or more files. |
| GPDS | Experimentation with and evaluation of system methods, processes, and designs. (Batch only.) |

### REMOTE PROCESSING

The remote processing system is an extension of the CP-V symbiont system. Its purpose is to provide for flexible communication between CP-V and a variety of remote terminals. These terminals can range from a simple card reader, card punch, and line printer combination to another computer system with a wide variety of peripheral devices. Any CP-V user (batch, on-line, ghost) can communicate with any number of devices at one or several remote sites. Because CP-V can act as a central site to some remote sites and simultaneously as a remote terminal to other computers, the remote processing facilities encourage the construction of communication networks.

### 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. As necessary, reports may then be created and sent to an appropriate terminal.

### 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.

## SYSTEM PROGRAMMING FACILITIES

This manual 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 facilities described in this manual aid the system programmer in the following areas:

- Modification of the CP-V operating system at the instruction level at boot-time.
- Reconfiguration of peripheral devices at boot-time.
- Analysis of crash dumps to determine the cause of a system crash.
- Creation and modification of the error message file for the CP-V monitor overlays while the system is operational.
- Listing and analysis of hardware and software malfunctions occurring during system operation.
- Development of shared processors such as compilers, assemblers, command language processors, and debuggers.
- Replacement, creation, and deletion of shared processors and monitor overlays while the system is operational.
- Development of peripheral hardware diagnostic programs.
- Development of real-time programs.
- Support of transaction processing facilities.
- Implementation of remote diagnostics for the Xerox 560.

# 2. SYSTEM OVERVIEW

## INTRODUCTION

The CP-V operating system consists of a monitor and a number of associated processors (Figure 1). The monitor provides overall supervision of program processing. The associated processors provide specific functions such as compilation, execution, and debugging.

## PROCESSORS

The CP-V system is illustrated in Figure 1 at two levels. The upper level lists the various monitor routines. The lower level lists the various processors. The 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.

LOGON/LOGOFF

LOGON admits on-line users to the system and connects the user's terminal either to TEL or to an alternative processor, such as BASIC, that has been selected by the user. LOGOFF disconnects a user from the system and does the final cleanup and accounting.

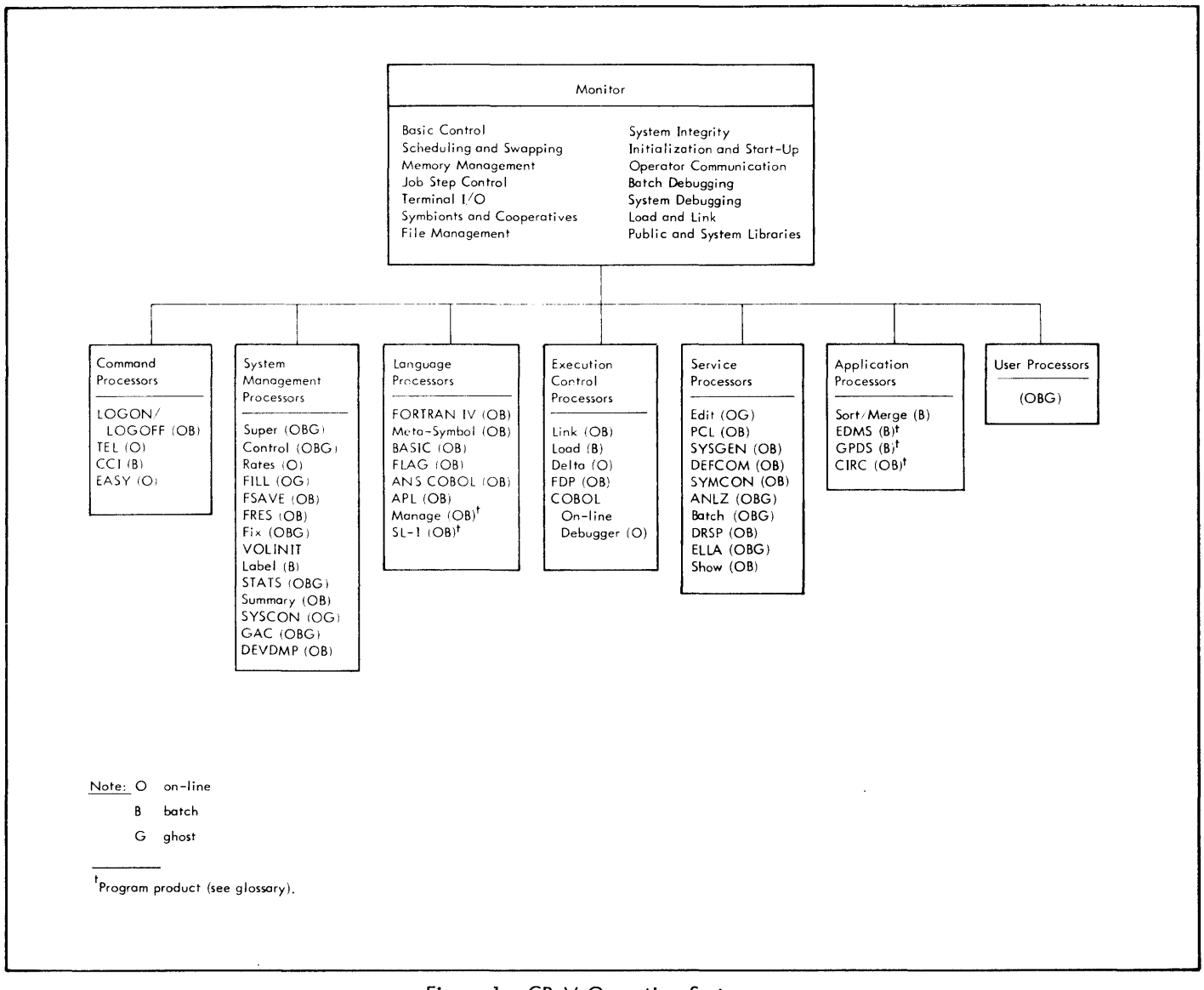



Figure 1. CP-V Operating System

### EASY

EASY is a shared processor that enables the user to create, edit, execute, save, and delete program files written in BASIC or FORTRAN. EASY also allows the user to create and manipulate EBCDIC data files. Although intended primarily for Teletype® operations, EASY can be used with any type of on-line terminal supported by the system. (Reference: EASY/LN,OPS Reference Manual, 90 18 73.)

### TERMINAL EXECUTIVE LANGUAGE

The Terminal Executive Language (TEL) is the principal terminal language for CP-V. Most activities associated with FORTRAN, COBOL and assembly language programming can be carried out directly in TEL. These activities include such major operations as composing programs and other bodies of text, compiling and assembling programs, linking object programs, initiating execution, and debugging programs. They also include such minor operations as saving and restoring core images of programs for which execution was interrupted, determining program status, and setting simulated tab stops. (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. (Reference: CP-V/BP Reference Manual, 90 17 64.)

## SYSTEM MANAGEMENT PROCESSORS

System management processors furnish the manager of a CP-V installation with on-line control of the system. Fourteen system management processors are supplied.

### SUPER

Super gives the system manager control over the entry of users and the privileges extended to users. Through the use of Super commands, the system manager may add and delete users, specify how much core and disk storage space a user will have, specify how many central site magnetic tape units a user will have, grant certain users, such as system programmers, special privileges, (e.g., the privilege of examining, accessing, and changing the monitor), and individually authorize or deny access to the various processors for each user. Super is also used to create and delete remote processing workstations. (Reference: CP-V/SM Reference Manual, 90 16 74.)

®Registered trademark of the Teletype Corporation

### CONTROL

The Control processor provides control over system performance. CP-V has a number of performance measurements built directly into the system. Commands of the Control processor enable the system manager to display these measurements and to "tune" the system as needed by setting new values for the parameters that control system performance. (Reference: CP-V/SM Reference Manual, 90 16 74.)

### RATES

The Rates processor allows the system manager to set relative charge weights on the utilization of system services.

Specific items to which charge weights may be assigned include

1. CPU time.
2. CPU time multiplied by core size.
3. Terminal interactions.
4. I/O CALs.
5. Console minutes.
6. Tapes and packs mounted.
7. Page-date storage.
8. Peripheral I/O cards plus pages.

(Reference: CP-V/SM Reference Manual, 90 16 74.)

### FIX

The Fix processor enables the system manager to repair or delete damaged file directories. It also provides HGP reconstruction for private disk pack sets and the public file system. (Reference: CP-V/OPS Reference Manual, 90 16 75.)

### FILL

The FILL processor performs three basic file maintenance functions:

1. It copies files from disk to tape as a backup.
2. It restores files from tape to disk.
3. It deletes files from disk.

(Reference: CP-V/OPS Reference Manual, 90 16 75.)

FSAVE

The Fast Save (FSAVE) processor is designed to save disk files on tape at or near tape speed. The processor is faster than any other file saving procedure under CP-V. (Reference: CP-V/OPS Reference Manual, 90 16 75.)

FRES

The File Restore (FRES) processor is designed to restore to disk files that were saved on tape by FSAVE or Fill. (Reference: CP-V/OPS Reference Manual, 90 16 75.)

VOLINIT

VOLINIT provides for the initialization of public and private disk packs. It is used to establish serial numbers and ownership, to write headers and other system information in selected areas of the volumes, and to test the surface of the disks and select alternate tracks to be used in place of flawed tracks. (Reference: CP-V/OPS Reference Manual, 90 16 75.)

LABEL

The Label processor initializes ANS tapes by writing ANS formatted labels. It may also be used to create "unlabeled" tapes from new tapes to be used as scratch tapes and to print the contents of the header and trailer labels of labeled tapes or the first 80 bytes of each block on unlabeled tapes. (Reference: CP-V/OPS Reference Manual, 90 16 75.)

STATS

The STATS processor displays and collects performance data on a running system and produces snapshot files to be displayed by the report generator Summary. (Reference: CP-V/SM Reference Manual, 90 16 74.)

SUMMARY

The Summary processor provides a global view of system performance by formatting and displaying the statistical data collected by STATS. (Reference: CP-V/SM Reference Manual, 90 16 74.)

SYSCON

SYSCON is a system control processor that can be used to partition resources from the system, to return resources to the system, and to display the status of the various system resources. SYSCON can also be used to build, update, or display the M:MODNUM file, a file which contains device and controller model numbers. (Reference: CP-V/SM Reference Manual, 90 16 74.)

GRANULE ACCOUNTING CLEANUP PROCESSOR (GAC)

The Granule Accounting Cleanup (GAC) processor correlates information between the file DISKPOOL and the account authorization file, :USERS. DISKPOOL is created by the FSAVE processor and contains specific account information. Each account record in DISKPOOL contains an entry for accumulated public disk pack granules and an entry for accumulated RAD granules. When GAC is run, these accumulated values are compared against the maximum values for the corresponding accounts in the :USERS file and the user's entry in the :USERS file is updated to reflect the latest accumulated values for RAD and disk. When the accumulated RAD or disk granules exceed the corresponding maximum values, this fact is noted in the report that is produced by the GAC processor. (Reference: CP-V/OPS Reference Manual, 90 16 75.)

DEVDMP

The Device Save/Restore processor (DEVDMP) is a standalone utility program designed to dump entire disk volumes to magnetic tapes for restoration at a later time. Restoration may only be made to an identical storage unit. (Reference: CP-V/OPS Reference Manual, 90 16 75.)

## LANGUAGE PROCESSORS

Language processors translate high-level source code into machine object code. Eight processors of special importance are described below. All of these can be used in both online and batch mode.

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 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.

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

- Extended language features to reduce programming effort and increase range of applications.
- Extensive meaningful diagnostics to minimize debugging time.

- In-line symbolic code to reduce execution time of critical parts of the program.
- Overlay organization for minimal core memory utilization.
- Compiler produced reentrant programs.

(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 hardware capabilities.

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

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

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

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

Meta-Symbol users are provided with an extensive set of directives. These directives, which are commands intrinsic to the assembly, fall into three classes:

1. Directives that involve manipulation of symbols and are not conditionally executed.
2. Directives that allow parametric programming.
3. Directives that do not allow parametric programming.

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

## 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.

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

Statements may be entered via a terminal and immediately executed. The principal benefit of direct execution is on-line development of programs and short simple computations. During execution, programs may be investigated for loop detection, snapshots of variables may be obtained, values of variables may be changed, flow of execution may be rerouted, and so on. This unique capability allows an on-line terminal to be used as a "super" desk calculator.

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

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

An easy-to-use feature is provided to allow the user to read, write, and compare variable alphanumeric data. This is particularly important for conversational input processing.

Chaining permits one BASIC program to call upon another for compilation and execution without user intervention. Thus, programs that would exceed user core space may be segmented, and overlay techniques may be employed via the chaining facility. (Reference: BASIC/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 commerical applications. The language specifications fully conform to the proposed ANSI standard for the various functional processing modules. Only those language elements that cause ambiguities or are seldom used have been deleted. The compiler's design takes full advantage of the machine's 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.

Xerox ANS COBOL contains many facilities that are either not found in other systems or, if available, are provided only at greater cost in terms of equipment required. Some of the facilities that provide more flexibility and ease of use in program development include

1. Implementation of table handling mode.
2. Sort/merge linkage.
3. Sequential access.
4. Random access linkage.
5. Segmentation.
6. Report writer.
7. Library utilization.
8. Calling sequence for FORTRAN, Meta-Symbol, etc.
9. Packed decimal as well as floating-point arithmetic formats.
10. Data name series options for ADD, SUBTRACT, MULTIPLY, DIVIDE, and COMPUTE verbs.

The system provides the user with a comprehensive set of aids to minimize the time required to print "bug-free" programs in the form of listings. These listings include

1. The source language input to the compiler with interspersed English language diagnostic messages.
2. An optional listing of the relocatable binary output, printed in line number sequence identical to the source language listing.
3. A cross-reference listing, indicating by line number where each data name or paragraph name is defined in the COBOL program and where each reference is located.

In addition, at run time, the user may use TRACE and EXHIBIT to follow execution of the procedure division.

The compiler is designed to take full advantage of high-speed, random access secondary storage (e.g., RAD storage). This feature means faster job execution because of minimized I/O delays, and smaller core memory requirements because of rapid overlay service. (Reference: ANS COBOL/LN Reference Manual, 90 15 00.)

## APL

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

Because APL is powerful, concise, easy to learn, and easy to use, it is widely used by universities, engineers, and statisticians. It also has features that make it attractive for business applications where user interaction and rapid feedback are key issues. One of APL's major strengths is its ability to manipulate vectors and multidimensional arrays as easily as it does scalar values. For example, a matrix addition that might require a number of statements and several loops in other languages can be accomplished as A+B in APL. This type of simplification exemplifies APL's concise power. (Reference: APL/LN,OPS Reference Manual, 90 19 31.)

### 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.)

### SIMULATION LANGUAGE (PROGRAM PRODUCT)[†]

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

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

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

## EXECUTION CONTROL PROCESSORS

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

[†]See "program product" in glossary.

### 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 and the job must be entered in the batch stream. (Reference: CP-V/TS Reference Manual, 90 09 07.)

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

### DELTA

Delta is designed to aid in the 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. The main purpose of Delta, however, is to facilitate the activities of debugging by

1. Examining, inserting, and modifying such program elements as instructions, numeric values, and coded information (i.e., data in all its representations and formats).
2. Controlling execution, including the insertion of breakpoints into a program and requests for breaks on changes in elements of data.
3. Tracing execution by displaying information at designated points in a program.
4. Searching programs and data for specific elements and subelements.

Although Delta is specifically tailored to machine language programs, it may be used to debug any program. Delta is designed and interfaced to the system in such a way that it may be called in to aid debugging at any time, even after a program has been loaded and execution has begun. (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 are 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.

In addition to the debugging commands, the debugger has a few automatic debugging features. One of these features is the automatic comparison of standard calling and receiving sequence arguments for type compatibility. When applicable, the number of arguments in the standard calling sequence is checked for equality with the receiving sequence. These calling and receiving arguments are also tested for protection conflicts. Another automatic feature is the testing of subprogram dummy storage instructions to determine if they violate the protection of the calling argument. (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 his program and the contents of data-items during on-line execution. The debugger also allows the COBOL source program to be examined and modified.

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

## SERVICE PROCESSORS

The processors in this group perform general service functions required for running and using the CP-V system.

### EDIT

The Edit processor is a line-at-a-time context editor designated 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.

Edit functions are controlled through single line commands supplied by the user. The command language provides for insertion, deletion, reordering, and replacement of lines or groups of lines of text. It also provides for selective printing, renumbering records, and context editing operations of matching, moving, and substituting line-by-line within a specified range of text lines. File maintenance commands are also provided to allow the user to build, copy, merge, and delete whole files. (Reference: CP-V/TS Reference Manual, 90 09 07.)

### PERIPHERAL CONVERSION LANGUAGE

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

PCL is controlled by single-line commands supplied through on-line terminal input 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. (References: CP-V/TS Reference Manual, 90 09 07 and CP-V/BP Reference Manual, 90 17 64.)

### SYSGEN

SYSGEN is made up of several processors. These processors are used to generate a variety of CP-V systems that are tailored to the specific requirements of an installation. The SYSGEN processors are PASS2, LOCCT, PASS3, and DEF. PCL is used to select from various sources the relevant modules for system generation. PASS2 compiles the required dynamic tables for the resident monitor. LOCCT and PASS3 file away and execute load card images to produce load modules for the monitor and its processors. DEF writes a monitor system tape that may be booted and used. (Reference: CP-V/SM Reference Manual, 90 16 74.)

### 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 monitor and the shared run-time libraries. (Reference: CP-V/BP Reference Manual, 90 16 64.)

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

### ANLZ

ANLZ provides the system programmer with a menas of examining and analyzing the contents of dumps taken during system recovery. It is called automatically by the Automatic Recovery Procedure and is executed as a ghost job. It may also be called by the operator to analyze tape dumps when recovery is not possible, or by an on-line user to examine crash dumps or the currently running monitor. (Reference: Chapter 4.)

### 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 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 the batch mode. (Reference: CP-V/TS Reference Manual, 90 09 07.)

### DRSP

DRSP (Dynamic Replacement of Shared Processors) enables the system programmer to dynamically add, replace, or delete processors during normal system operation with other users in the system. (Reference: Chapter 7.)

### ELLA

The Error Log Listing program (ELLA) provides an efficient tool to list and sort the error data base which is automatically generated and updated by the CP-V system. (Reference: Chapter 6.)

## APPLICATION PROCESSORS

The application processors are intended for use for specific types of applications.

### 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 1 to 16 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.

The principal highlights of Sort are as follows:

1. Sorting capability allows either magnetic tapes, disks or both.
2. Linkages allow execution of user's own code.
3. Sorting on from 1 to 16 key fields in ascending or descending sequence is allowed. Keys may be alphanumeric, binary, packed decimal, or zoned decimal data.
4. Records may be fixed or variable length.
5. Fixed length records may be blocked or unblocked.
6. Disks may be used as file input or output devices, or as intermediate storage devices.
7. Sort employs the read backward capability of the tape device to eliminate rewind time.
8. User-specified character collating sequence may be used.
9. Buffered input/output is used.

(Reference: Sort-Merge/Reference Manual, 90 11 99.)

## 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.

The principal features of EDMS are as follows:

- The user can describe data in various data structures. Using sets, any element can be related to any other element. The data structures include lists and hierarchies (trees). The two relationships can be combined to form extensive networks of data.
- Access techniques include random, direct, indexed, and indirect (relative to another record).
- An EDMS data base may consist of up to 64 monitor files.
- Multiple secondary indexes can be defined by the user to allow records to be retrieved via any combination of secondary record keys.
- Users may construct any number of logical files or data bases within an EDMS file.
- Data is described separately from the user program to facilitate management of the data base.
- Comprehensive security exists at all levels of a file.
- Journalization provides an audit trail for backup and recovery.
- A dynamic space inventory is maintained to facilitate rapid record storage and to optimize the use of available storage space.
- Detailed data description is provided for inclusion into the user's application program to reduce programming effort.
- File I/O logic is performed for the user program including
    1. Logical or physical record deletion.
    2. Record retrieval on random or search basis.
    3. Record insertion or modification.

(Reference: EDMS/Reference Manual, 90 30 12.)

## 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. Potential applications include

- Advanced management planning.
- Analysis of inventory or financial systems.
- Studies of message switching and communications networks.
- Risk and capital investment studies.
- Evaluation and data processing systems.
- Job shop and queuing studies.

Although GPDS is compatible with other simulator systems, it has a number of salient features not usually found in competitive versions. (Reference: GPDS/Reference Manual, 90 17 58.)

## CIRC (PROGRAM PRODUCT)[†]

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

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

- The on-line mode offers several advantages since it provides true conversational interaction between the user and computer. Following CIRC start-up procedures, CIRC requests a control message from the user. After the control message is input (e.g., iterate a cycle of calculations with changed parameters) the computer responds (via CIRC) with detailed requests 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

[†]See "program product" in glossary.

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.

(References: 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 Chapter 7.

## 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 programs 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. Resource Management.
7. Job Step Control.
8. Terminal I/O Handling.
9. Symbionts.
10. Cooperatives.
11. System Integrity.
12. Initialization and Start-up.
13. Operator Communications.
14. Batch Debugging.
15. Load-and-Link.
16. 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 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 multibatch job scheduling routines select jobs to be run from the waiting input queue depending on priority and resource and partition availability.

Resource management facilities keep track of the number of resources of each kind (i.e., tape drives, disk spindles, core) that are in use. For a batch job, the multi-batch scheduler compares the resources required with the available resources and does not start the job until sufficient resources are available. Once the job is started, the resources that are required by the job are reserved for the exclusive use of the job, thereby guaranteeing that they will be available for the duration of the job.

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 associated shared processors or to load program modules.
4. Merging DCB assignments for execution.
5. Checking user authorization for individual processors.
6. Fetching program load modules into core.

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

Symbiont routines transfer data from the card reader to logical device streams on disk storage and from logical device streams on disk storage to the card punch or line printer.

Cooperative routines intercept read, print, or punch commands in user programs and transfer data from or to logical device streams residing on disk storage. The input cooperative simulates card reading from a logical device stream. The output cooperative builds a logical device stream using intercepted program output directed by the user program to a line printer or card punch.

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. Three debugging routines are available. They are

1. Executive Delta: This is a stand-alone processor and is essentially the same as on-line Delta. Executive Delta is optionally loaded at boot time along with the root of the monitor and monitor system tables.
2. Analyze: This program is intended for debugging CP-V crash dumps. To accomplish this, it performs two major functions.
   a. It summarizes the complete software environment at the time of the crash in a series of tables.
   b. It permits on-line interactions similar to Delta.
3. Recover: This program provides the "bail-out" exit from the monitor. The error code that is transmitted to RECOVER defines the problem and the module that discovered the problem.

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 types of FORTRAN libraries. One type is a public library and the other is a system library. In the standard release of CP-V, there are three FORTRAN public libraries. One library (P1) contains a useful set of Extended FORTRAN IV run-time library routines; another (P0) contains P1 and the FORTRAN Debug Package; the third (P4) contains P1 and the FORTRAN real-time features. These three 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.

## SCHEDULING AND MEMORY MANAGEMENT

Scheduling and memory management routines control the overall operation of the system. Inputs to these routines, together with the current status of users as recorded by the

scheduler, are used to change the position of each user in the scheduling status queues. It is from these queues that selections are made for both swapping and execution. Swaps are set up by the selection of a high priority user that is to be brought into core and by pairing this user with one or more low priority users that are to be transferred to disk storage. Similarly, the highest priority user in core is selected for execution.

## SCHEDULER INPUTS

System activities are reported by direct entry to the scheduler, which makes changes to user status queues through a logical signaling table. The scheduler records inputs by changing the user status queues and other information associated with the user. In general, a table-driven technique is used. The received signal is on one coordinate and the current state of the user is on the other. The table entry thus defined names the routine to be executed in response to the given signal-state combination. Since the number of signals and states is large, the table technique aids in debugging by forcing complete specification of all the possibilities. Inputs to the scheduler are listed in Table 1. The scheduler also receives control at execution of each CAL issued by a user program that is requesting monitor service. These entries (Table 2), the special entries from the executive language processors, and entries from internally reported events drive the scheduling of the system.

Table 1. Event Inputs Received by Scheduler

| Event | Meaning |
|---|---|
| E:ABRT | Operator aborted user. |
| E:AP | Associate shared processor with user. |
| E:ART | Activate real-time user. Interrupt has occurred. |
| E:CBA | COC buffer available. |
| E:CBK | Break signal received. |
| E:CBL | Number of output characters > SL:TB. |
| E:CEC | TEL request (ESC ESC, ESC Y, or Y^C). |
| E:CFB | Cannot find COC buffer. |
| E:CIC | Terminal input message complete. |
| E:CRD | Read terminal command received. |
| E:CUB | Number of output characters = SL:UB. |
| E:DPA | RAD page available. |
| E:ERR | Operator errored user. |
| E:IC | I/O complete. |

Table 1. Event Inputs Received by Scheduler (cont.)

| Event | Meaning |
|---|---|
| E:IIP | I/O started and now in progress. |
| E:IP | Request permission to start I/O. |
| E:KO | User removed from core. |
| E:NC | Cannot get requested core pages. |
| E:ND | Cannot get requested disk page. |
| E:NOCR | User allowed to open or close file. |
| E:NSYMD | No symbiont disk space. |
| E:NSYMF | No symbiont file entry. |
| E:NQR | Enqueue release – resource available. |
| E:NQW | Enqueue – wait for resource. |
| E:OCR | Request permission to open or close file. |
| E:OFF | User has hung up telephone. |
| E:QA | User queued for access (e.g., for access to tape or disk pack). |
| E:QE | Quantum end. |
| E:QFAC | No granules available for use. |
| E:QFI | Real-time user. Queue for interrupt. |
| E:QMF | Queue for I/O master function count too high. |
| E:SL | Sleep time for user. |
| E:SYMF | Symbiont file now available. |
| E:SYMD | Symbiont disk space now available. |
| E:UQA | User dequeued for access (e.g., for access to tape or disk pack). |
| E:UQFAC | ALLOCAT has refreshed granule stacks. |
| E:WU | Wake-up time for user. |

Table 2. Service Request Input to Monitor

| Source of Inputs | Service Request Entries |
|---|---|
| User program (through monitor service calls) | 1. Terminal input/output request.<br>2. Input/output service calls for RAD, disk pack, or magnetic tape.<br>3. Wait request.<br>4. Program exit (complete).<br>5. Core request (for common, dynamic, or specific pages).<br>6. Real-time services.<br>7. Program overlay (load and link, load and transfer).<br>8. Debug requests.<br>9. Miscellaneous service requests. |
| Executive language processor | 1. Name of system programs to be loaded and entered (implies deletion of any current program).<br>2. Continuation signal.<br>3. Special continuation address.<br>4. Link load-and-go-exit. |

## SCHEDULER OUTPUT

The scheduling routine performs two major functions during the time it is in control of the computer. The first function consists of setting up swaps between main core memory and secondary disk storage in such a way that high priority users are brought into core to replace low priority users that are transferred to disk storage. The actual swap is controlled by an I/O handler according to specifications prepared by the scheduler. These specifications are prepared according to the priority state queues described in the next section. Given a suitably large ratio of available core to average user size (greater than 4), the scheduler can keep swaps and computing close to 100 percent overlapped.

The second function the scheduler performs consists of selecting a user for execution according to the priority state queues and the rules for batch processing. The rule is simple: the highest priority user whose program and data are in core is selected.

## USER STATUS QUEUES

Status queues form a single priority structure from which selections for swapping and execution are made. The status queues form an ordered list with one and only one entry for each user. The position in queue is an implied bid for the services of the computer. As events are signaled to the scheduler, individual users move up and down in the priority structure. When they are at the high end, they have a high priority for swapping into core and for execution. When they are at the low end, they are prime candidates for removal to secondary storage. This latter feature — that of having a defined priority for removal of users to disk storage — is an important and often overlooked aid to efficient swap management. It avoids extraneous swaps by making an intelligent choice about outgoing as well as incoming users.

In addition to these primary functions, user status queues have other functions such as

1. Synchronizing the presence in core of the user program and data with the availability of I/O devices.
2. Queuing user programs to be "awakened" at a preestablished time.
3. Queuing requests for entry and use of processors.
4. Managing core memory.
5. Queuing requests for buffers either in core or on disk.
6. Queuing requests for nonresident monitor services.

A list of the status queues is given in Table 3.

## SCHEDULER OPERATION

To select users for execution, the scheduler searches down a list of the status queues for the first user in core memory. The highest priority user is served first. Interrupting users are served before those with an active input message (both of these take precedence over users with unblocked terminal output), then come on-line compute-bound users and finally, compute-bound batch jobs. Note that users in lower states have no current requests for CPU resources. Note also that as each user is selected for execution, the status queue of the user is changed to CU. When the quantum is complete, the highest priority queue the user can enter is the compute queue. Users that enter any of the

Table 3. Scheduler Status Queues

| State | Meaning |
|---|---|
| SRT | Real-time execute (0 ≤ priority ≤ X'BF'). |
| SC0 | Background execute (X'C0' ≤ priority ≤ X'F5'). |
| SC1 | Background execute (priority = X'F6'). |
| SC2 | Background execute (priority = X'F7'). |
| SC3 | Background execute (priority = X'F8'). |
| SC4 | Background execute (priority = X'F9'). |
| SC5 | Background execute (priority = X'FA'). |
| SC6 | Background execute (priority = X'FB'). |
| SC7 | Background execute (priority = X'FC'). |
| SC8 | Background execute (priority = X'FD'). |
| SC9 | Background execute (priority = X'FE'). |
| SC10 | Background execute (priority = X'FF'). |
| STOB | Terminal output blocked in core. (More characters than the system limit are ready for typing.) |
| STOBO | Terminal output blocked. Not in core. |
| SIOW | I/O wait. Users waiting for an I/O that is in progress to complete. |
| SIOMF | Users blocked because I/O master function count (number of I/O operations in progress) has reached the system limit. |
| SW | Users waiting for a specified "wake-up" time. |
| SQA | Users waiting for service by RBBAT, the symbiont ghost. |
| SQR | Users in core and blocked for dynamic resource such as swapper page, COC buffer, symbiont disk page, symbiont table space, enqueued resource, service by ALLOCAT (for file granules), or file open or close. |
| SQRO | Same as SQR but not in core. |
| STI | Typing input and in core. |
| STIO | Typing input and not in core. |
| SQFI | Foreground time user waiting for interrupt. |

three highest priority states receive rapid response but only for the first quanta of service. Thereafter, they share service with others in the compute queue.

A similar selection procedure is used to set up users for swapping. First, the highest priority user in the execution queue who is not in core is selected and his size requirement (including the requirement for shared processors not in core) is determined. Second, users are selected from the swapout queue until enough space is freed by these users and their shared processors to provide for the user selected for swap-in. If a single user in a state below SC10 (Table 3) can be found to swap out, then a single rather than a multiple swap is chosen. No swaps occur until a user that is not in core enters a high priority queue.

Two lists resulting from this selection are presented to the swapper. One list contains the user (or users) to be swapped out and the other contains the user to be swapped in. This latter list also contains the shared processors that must accompany the user and the current free core page list. When the scheduler selects users for swapping, it picks a high priority user to load into core and the lowest priority user to remove from core. Priorities are arranged from high to low, in order of increasing expected time before the next activation. This ensures that the users that are least likely to be needed are swapped out first, while the users most likely to require execution are retained in core. The swap algorithm operates so that compute users remain in core and use all available compute time, while the interactive users are swapped through the third core slot whenever the following three conditions exist:

1. There is room in core for three user programs.
2. Two users are computing steadily.
3. Many other users are doing short interactive tasks.

Table 4 shows the queue used for selection of users to be brought in for execution and the queue used for selection of users to be moved to disk.

Note that the queues CU, IOW, QRO, TOBO, TIO do not appear in either list. Thus, the users in these states are not selected either for execution or for swapping.

Two examples of typical interactive use are illustrative of the scheduling operation.

The first example traces scheduling operations for a simple, short interactive user request. At the time the request is typed, the user is in the ST1 queue. His program, which has probably been swapped to disk storage, remains there until the COC routines receive an activation character. Receipt of this character is reported to the scheduler and causes a change in state of the user to the appropriate executable state (SC0-SC10). The scheduler finds a high priority user not in core and initiates a swap to

Table 4. Swap-In and Swap-Out Queues

| Swap-In (and Execution) Queue | Swap-Out Queue |
|---|---|
| SRT | SW |
| SC0 | STI |
| SC1 | STOB |
| SC2 | SQFI |
| SC3 | SQA |
| SC4 | SC10 |
| SC5 | SQR |
| SC6 | SC9 |
| SC7 | SC8 |
| SC8 | SC7 |
| SC9 | SC6 |
| SC10 | SC5 |
| | SC4 |
| | SC3 |
| | SC2 |
| | SC1 |
| | SC0 |
| | SRT |

remove a low priority user (if necessary) and to bring in the one just activated. On completion of the swap, the scheduler is again called and now finds a high priority user ready to run. The user's state is changed to CU, the program is entered, and the input command is examined by the reading program. The cycle in this example is completed by preparation of a response line and a request to the monitor for more input which changes the user's state to TI again, making him a prime candidate for removal to disk.

The second example illustrates an output-bound terminal program. This program moves through the state cycle STOB-SC-SCU as output is generated by the program. The COC routines signal when the output limit has been reached, thus causing the program to be delayed while output is transferred to the terminal. In a typical operation, four to six seconds of typing is readied in buffers each time the user program is brought into core and executed. During this typing time, the program is not required in core and the CPU resources can be given to other programs.

## I/O SCHEDULING

I/O scheduling is designed to provide good service to I/O-bound users while keeping the CPU busy with compute-bound users. The intent is to make the fullest possible utilization of both the CPU and the I/O devices. The manner in which this is accomplished is described below.

A user that has been waiting for an I/O to complete (SIOW) is changed to an executable state at a priority slightly higher than a similar compute-bound user when the I/O completes. At that time, the execution scheduler interrupts the execution of the compute-bound user so that the I/O-bound user can execute. The I/O-bound user requires comparatively little CPU time before initiating another I/O request and returning to the SIOW state. The compute-bound user then resumes execution.

It should be noted that the scheduler automatically adapts to jobs that alternate between bursts of computing and bursts of I/O.

## SWAP HARDWARE ORGANIZATION

Users are removed from core to a dedicated area of disk storage (or to several disks in large configurations) when core is required for higher priority users.

Bit tables are used to keep track of the availability of each sector on the disks. In these tables, a zero is used to indicate the sector is in use (usually assigned to a user) and a one is used to indicate the sector is available. Users are assigned a sufficient number of page-size sectors to accommodate their current use. The assignment is done in such a way that command chaining of the I/O can order the sectors to be fetched for a single user with minimum latency. That is, each user's pages are spread evenly over the set of available sectors on the disk to which he is dedicated so that data will be transmitted in every disk sector passed over when the user is swapped.

The records of disk sectors associated with each user are kept in the user's job information table (JIT), which is kept on disk when the user is not in core. The disk location of the JIT and the user's disk address are kept in core by the scheduler. The disk layout is such that sufficient time is available after the user's JIT arrives from the disk for the system to set up the I/O commands for the remainder of the user.

The amount of disk storage assigned to swapping is a parameter of SYSGEN. The number of on-line users that the system can accommodate is limited by the size of disk space allocated for swapping and the total size of active on-line users.

The allocation scheme for systems which have file space allocated on both RADs and disk packs is described in the following paragraphs.

For the sake of overall performance, the RAD is preferred for frequently accessed system information and temporary files used by the major processors. Special users who need high performance on special files may specify RAD preference.

All of the account directory and all files from :SYS are assigned to the RAD. The first granule of each file directory is assigned to disk pack but any additional granules are assigned to RAD. All star or id files and all scratch files (opened OUT or OUTIN with REL) prefer RAD. Random files with no user stated preference and all other files and their indexes prefer pack. These pack preferences may be overridden either by the operator keyin 'PREFER' for all files or by the user specification of NOSEP and DEVICE for individual files.

Briefly, the effect of authorization and defaults upon the allocation is: If not enough space is available on the preferred device, the other device will be used if space is available there. The exception to this is random files with user specified preference. In this case, if space is not available on the user specified device, the file is not allocated and an error is returned to the user. Also, within the authorized limits, temporary files may use only temporary authorization and permanent files may use only permanent authorization.

In general, the rule for authorization should be: A large amount of temporary RAD and disk pack space should be authorized for all users and the amount of permanent disk space should be individually authorized by need. Very few users should be authorized permanent RAD space.

There are four in-core buffers for types of space to be allocated. Three are for granule allocated devices:

1. RAD PFA (permanent file storage).
2. Pack PFA (permanent file storage).
3. PER (peripheral symbiont storage).

The fourth is for cylinder requests. These buffers are used to satisfy requests for all purposes except directories, random files, and PSA (permanent system storage and swapping).

Due to the system configuration and SYSGEN, at most six sets of devices can be created:

1. RAD all PFA (PFA RAD first).
2. Pack all PFA (PFA pack first).
3. All PER (PER first).
4. RAD PFA plus PER and/or PSA (PFA RAD second, PER second).
5. Pack PFA plus PER and/or PSA (PFA pack second, PER third).
6. PER plus PSA (PER fourth).

Granules are selected for the in-core buffers from one of the six sets of devices starting each device at sector zero and allocating from all the devices within the set simultaneously (i.e., round-robin). The preference in choice of sets is noted above in parentheses. All devices of a set will be depleted before the next set is chosen.

Cylinders for the in-core buffer are allocated starting at cylinder zero of the first (lowest DCT index) cylinder allocated device. Each device will be depleted before the next is used.

Random files are allocated starting at the last sector of the last (highest DCT index) device of the proper type. The cylinder allocated devices are treated as one continuum of space for random files. They need not be contiguous in the DCT table and any file may cross a boundary (even a two cylinder file). Private random files are allocated in the same way.

## PROCESSOR MANAGEMENT

CP-V processors are considered shared processors when they are written in such a way that they are pure procedure and are described as such when they are added to the system. (User-associated data areas are initialized at first entry.) A shared processor has the following special characteristics:

1. Its name is known to TEL and it may be called by name.
2. It has dedicated residency on swap storage established at system initialization or via DRSP.
3. A single copy is shared by all requesting users.

## MEMORY LAYOUT

The system makes full use of address mapping hardware, access protection, and write locks in allocating available physical core pages to users. Physical core pages are allocated to users at their request. Use of the map obviates the need for program relocation or physical moves. Full protection is provided for one user from another. All programs and the monitor itself are divided into procedure and data. The procedure area is protected by write-locks or access codes, or both, against inadvertent stores.

The central features of the use of write-locks to protect master mode programs are as follows:

1. The monitor operates with a key of 01 and may store in

   a. Its own data area (LOCK = 01).

   b. Any batch, on-line or shared processor code (LOCK = 00).

   It may not store in its own procedure (LOCK = 11).

2. Keys of 10 and 11 are never used, nor is the lock of 10.

3. Write-locks are initialized only once at system start-up and are not changed thereafter except when running under control of Executive Delta where they are used to enable data breakpoints.

4. On the Xerox 560, write keys are four bits long and apply to the IOP memory writes as well as CPU write operations. To take advantage of this feature, the Xerox 560 I/O system always uses a key of 1000 which does not match any of the locks. This means that no I/O operation can accidentally overwrite the monitor or its data since the IOPs can only write into memory with locks of zero (the user area). Also on the Xerox 560, certain monitor buffers which are part of the monitor data area (usually with lock 01) are grouped together and the pages containing those buffers are set to a lock of 0000. Except for this difference, the rest of the locks are exactly the same as for the Sigma computers.

The access code on virtual memory pages controls references made by slave mode programs (user programs and shared processors). This code is retained in the JIT of each user and is loaded into the hardware access protect registers (which are part of the virtual mapping hardware) when the user gains control. Write access to JIT and other job context areas is given to TEL, CCI, LOGON, and any installation-defined command processors.

The layout of virtual memory that applies to user programs and ordinarily shared processors is shown in Figure 2. Allocation of the available area depends on the type of user that is running and the attributes of the load module to be executed. Allocation Type II is used when a core library or debugger is associated or when the load module to be executed has been built by Link. In all other cases, allocation of the available area is as shown in Type I for batch users, ghost jobs, and on-line users executing in the extended memory mode.

Core addresses shown are those appropriate for a typical system but more (or less) core may be established for the resident monitor at SYSGEN time depending on installation needs. More (or less) area may also be desirable for the library area and for the job context area to accommodate more buffers. These bounds may also be adjusted at SYSGEN time. The boundary at which the one-pass loader (Link) places the user program is also adjustable.

Virtual pages not currently allocated to the user are mapped into a resident monitor page that is write-locked, (the access code is set to no access). Thus, slave mode programs are denied access through the access code, and attempts to store at these virtual addresses by a master mode program are protected by write locks.

Typical layouts of physical memory are shown in Figure 3 for Sigma systems and in Figure 4 for Xerox 560 systems. Although these are similar to the actual layout, they should not be assumed to be exact.

## SYSTEM INTEGRITY

The monitor has a number of routines that have been included to guarantee system integrity. The objectives of these routines are, in order of importance, (1) to provide the highest possible security for user files even in the event of total system failure, (2) to provide automatic high-speed recovery in the event of a machine or software failure, and (3) to record sufficient information to isolate errors and failures caused by either hardware or software.

The major features of the CP-V system integrity routines are as follows:

1. Detection of malfunctions by hardware examination and software checks wherever the checks have been shown to enhance hardware error detection. Recovery from these malfunctions is through retries, operator assistance, etc.

2. Logging of all malfunctions, including recovered errors and permanent failures.

3. Protection from hardware failures.

4. Use of on-line exercisers to provide for repair or adjustment of peripherals without taking the CPU down.

5. File backup and recovery facilities to minimize the probability of losing user files, and in case of file failure, to facilitate complete recovery of the file system with a minimum of loss.

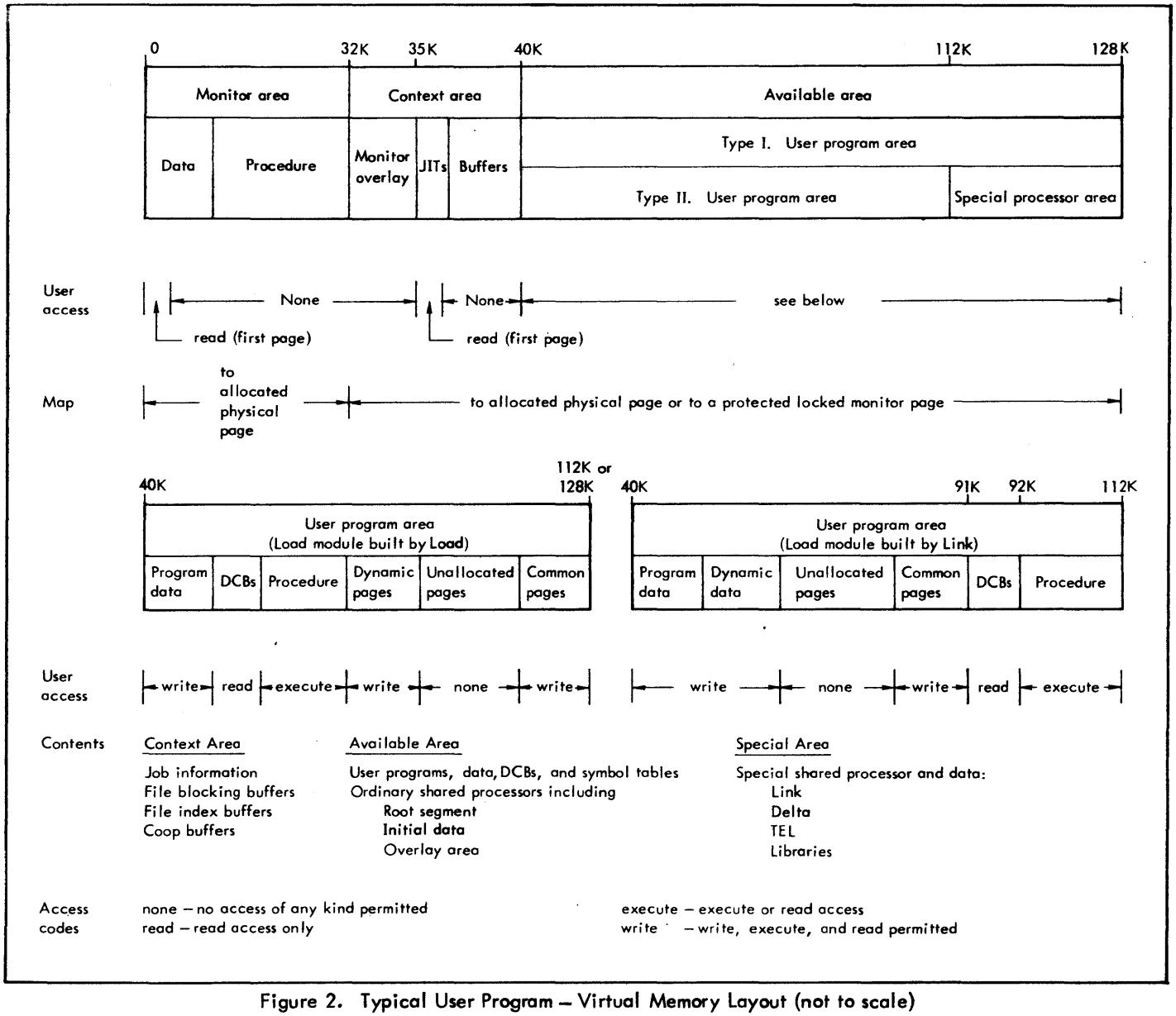



Figure 2. Typical User Program – Virtual Memory Layout (not to scale)

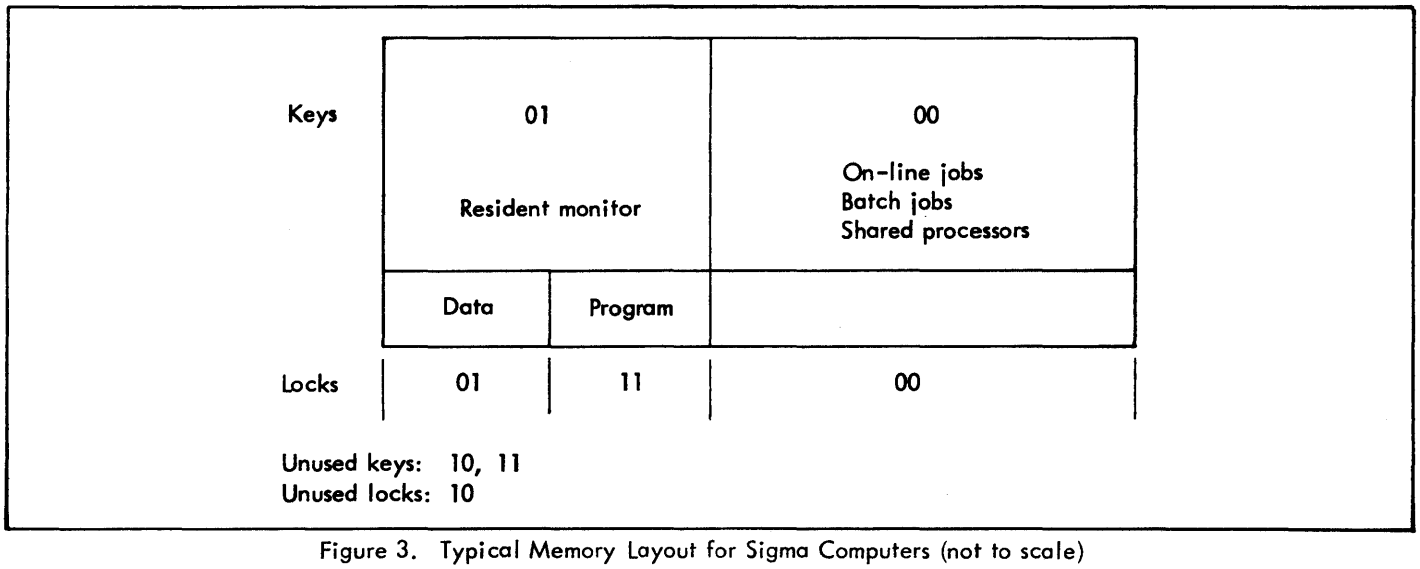



Figure 3. Typical Memory Layout for Sigma Computers (not to scale)

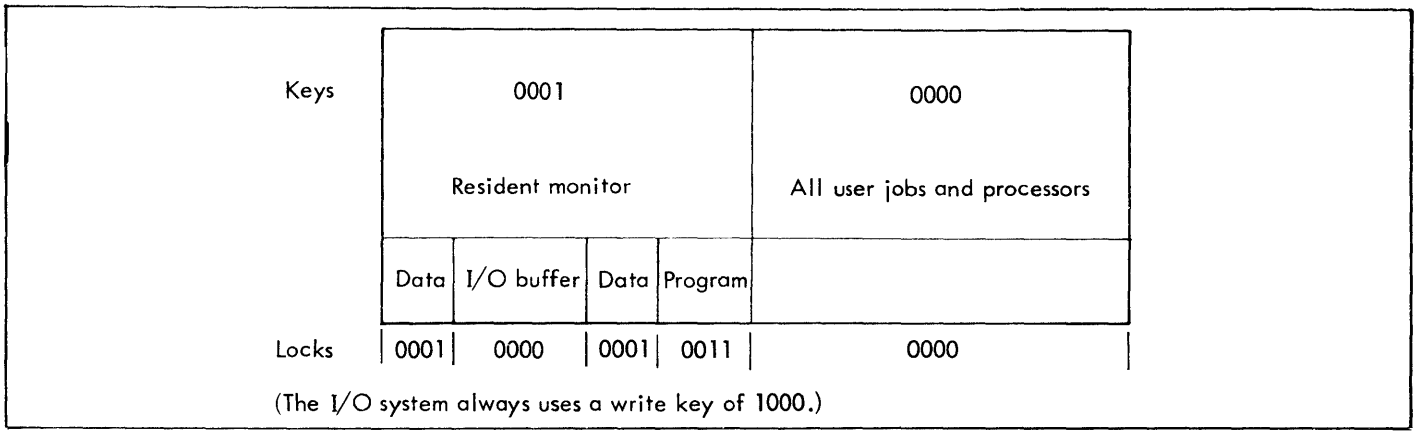



Figure 4. Typical Memory Layout for the Xerox 560 (not to scale)

6. Automatic recovery following a system failure with reasonable speed consistent with file security and the recording of information for later analysis.

7. Facilities to provide for analysis of system crashes. Information includes simple classification of failures as well as full information for both customer engineers and system programmers.

8. For the Xerox 560 – on-line interface for remote assistance.

## ERROR DETECTION AND RECOVERY

An effective operating system must be able to detect and, whenever possible, to correct errors. It must also be capable of restarting the system if necessary. CP-V uses a combination of hardware and software checks to efficiently meet these goals.

Hardware error protection features include memory protection against accidental overwriting of monitor and user programs, power fail-safe interrupts that ensure automatic restart in the event of power failure, memory parity checking, I/O read and write verification, and a watchdog timer to avoid instruction hangups. Detected errors are reported, logged, and if possible, recovered directly. Catastrophic failures cause an automatic system recovery if at all possible. Those failures which can be isolated to a single user cause only that user to be aborted. Some hardware errors, such as loss of a memory power supply, lead to system shutdown.

Software consistency checks, some of which are performed optionally on the setting of a console sense switch, check the integrity of the software at many critical locations in the system. These checks detect problems before they are allowed to go beyond a recoverable point. When an inconsistency that is catastrophic to the system is detected, the current users are logged off and all open files are closed. The system is then automatically rebooted for the fastest possible restart.

## ERROR AND FAILURE LOGGING

Malfunction messages are maintained in a special file by system integrity routines. Messages are placed in this file whenever malfunctions are detected by the various parts of the system. Hardware malfunctions that are recorded include such things as tape errors, card reader errors, memory parity errors, and illegal instructions. Software malfunctions that are recorded include the failure of software checks on RAD or disk addresses contained in index blocks and improper linkage of linked file blocks. In addition, a software recovery from a seek failure is recorded in this file (as a 757F code).

The error messages generated throughout the system (reporting both hardware and software errors) are placed initially in in-core buffers and then are transferred to a special file (actually a linked list of granules). This transfer is initiated whenever an error count threshold, or time limit is reached. This special file is then transferred to an ordered keyed file (ERRFILE) by the standard system ghost processor ERR:FIL which is automatically awakened by the system.

## ERROR LOG LISTING

This keyed file (ERRFILE) may be listed and sorted by the processor ELLA which allows the Customer Engineer to display and search the error file for patterns of errors to aid in preventive maintenance for the system.

## ON-LINE DIAGNOSTICS AND EXERCISERS

On-line diagnostics and exercisers may be called when there is a specific failure detected by the hardware or software, or when a failure is projected through analysis of the error log by the Customer Engineer. These programs may also be called by the Customer Engineer when needed for the test or adjustment of the card reader, card punch, line printer, magnetic tape, or other devices.

## REMOTE DIAGNOSTIC ASSISTANCE

On the Xerox 560, on-line diagnostics and certain on-line debugging processors (ANLZ, Delta, and ELLA) may be utilized via the Remote Assist Station (RAS) interface.

After control is obtained from the local operator, customer engineers and/or diagnostic programmers at remote locations may access the system via this interface without interfering with the on-line COC users and without using any of the normal communication equipment. By evaluating the system under normal operating conditions, many software errors and hardware malfunctions may be detected and eliminated expeditiously with a minimum of computer down time.

## FILE MAINTENANCE

CP-V provides a variety of processors designed to maintain a reliable backup of the file data base. These processors are summarized in the CP-V/SM Reference Manual, 90 16 74, and are described in detail in the CP-V/OPS Reference Manual, 90 16 75. The processors provide the ability to save and restore large volumes of files very quickly, to save and restore entire private and public disk devices at device speed, to handle user initiated backup of files, to restore the allocation tables for public disks after a system crash, to restore the allocation tables for a private disk pack after a crash which affected the pack, and to restore granule account information in the :USERS file.

## AUTOMATIC RECOVERY AFTER SYSTEM FAILURE

The CP-V monitor performs consistency checks on the results of hardware operations, checks intermediate results of operating system software functions, performs checks and balances at appropriate interfaces between the operating system's modules, and monitors itself for unexpected trap conditions caused by the hardware or operating system software. A software check code is assigned to each type of failure that the monitor may detect.

Some of these software check failures result in a momentary delay in service to all but the current user for whom the operating system is performing a service. In such case, the current user's job step is aborted, core is dumped to a file for later analysis and display, and normal operating then continues. The remaining software check failures are handled by the system's recovery routine.

The recovery routine performs the following functions:

1. Displays cause of failure.
2. Takes a full core dump for later analysis.
3. Closes all open files with default options.
4. Packages or releases all partial symbiont files.
5. Closes common TP journal if transaction processing is being used.
6. Saves in-core transaction processing files.
7. Packages error log.
8. Informs users of interruption.
9. Saves time, data, error log pointers, accounting information, symbiont file directory, public disk granule usage map, and executive communication.
10. Restarts system and restores items saved above.

When functions cannot be performed, they are noted on the operator's console. If the function is considered minor, recovery continues. If it is connected with file operations, the file identification is noted and recovery proceeds.

The recovery routine described above occurs automatically with a minimum delay (a few seconds) in system availability. Operator initiation of this recovery function is also allowed, providing for the event that the system fails by not responing to any operator key-in or user service request.

When the recovery routine executes, it is independent of all monitor services and functions and requires only that a small recovery driver be intact in memory. This driver reads the main recovery module into memory from the system swap device, overlaying the pure procedure portion of CP-V. Certain monitor system tables are also required intact for successful recovery. These tables are verified before proceeding. If the recovery process cannot be completed, the operator is instructed to initialize the system from the master system tape and restore files and backup tapes.

## CRASH ANALYSIS

In the event of a recovery or single user abort, one of the recovery functions is to dump the contents of core memory into a special file in the :SYS account. This information is saved for later analysis by a system programmer using a special debugging program, ANLZ.

The ANLZ program may be called by the operator or system programmers to run as a privileged ghost, on-line, or batch job. The ANLZ program is also called automatically as a privileged ghost job by the recovery routine as one of the first jobs following a recovery or the first job following a single user abort. In any mode, ANLZ is command driven (except in the ghost mode following a recovery). It responds to commands that selectively display monitor tables, examine memory, and compare the dump with the running monitor. (Reference: Chapter 4.)

# 3. BOOTSTRAP AND PATCHING OPERATIONS

## SYSTEM TAPE FORMAT

A CP-V system tape contains the following elements:

1. Bootstrap loader.
2. Root for an absolute monitor.
3. General information record concerning this system tape.
4. Other monitor segments (XDELTA, ALLOCAT, GHOST1, FIX).
5. Monitor overlay segments.
6. RECOVER.
7. Tape label information.
8. Files for all system load modules and other needed files.
9. Patches and GENMD commands.

The general arrangement of the information on a master system tape is shown in Figure 5.

## PATCH DECK STRUCTURE

Patch decks have the following structure:

1. The following two types of patches:

   a. Delta format patches for the monitor root and its overlays.

   b. Symbol definition patches.

   The monitor root patches can appear anywhere within the patch deck. The overlay patches must be in the same order as the system tape structure. Symbol definitions must precede the patches in which the symbol is used.

2. Boot-time reconfiguration and partitioning commands. These are optional, but if they are used, they must precede the first overlay patch.

3. A card that contains an asterisk in column one. This card terminates the monitor patches and boot-time reconfiguration and partitioning commands.

4. The following two types of patches (which may appear in any order):

   a. A GENDCB command to assign the account, a password, serial number, and type of tape drive for the boot tape.

   b. A group of GENMD commands and GENMD patches to the processors contained on the tape.

5. A !EOD command (the final command of the patch deck).

In addition, there are two types of cards that may appear anywhere within the patch deck (including the GENMD portion). These two types are the conditional patch control command and the comment card.

No patch, command, or comment may contain more than 72 characters of information.

When the patch deck is read, it is retained by the system in a file called PATCH in the :SYS account. This file can be examined using the PCL processor. It may also be assigned to M:PATCH and DEFed onto the PO tape.

The function and format of Delta format patches, symbol definition patches, reconfiguration and partitioning commands, GENDCB commands, GENMD commands, GENMD patches, conditional patch control commands, and comment cards are described in the paragraphs that follow.

### DELTA FORMAT PATCHES

Delta format patches are used to patch various segments of the monitor. The format of a Delta format patch is:

[segname]/loc/value[(old value)]/comment

where

segname is the name of the segment to be patched. The current segnames and the order in which they must be patched are shown in Figure 6.

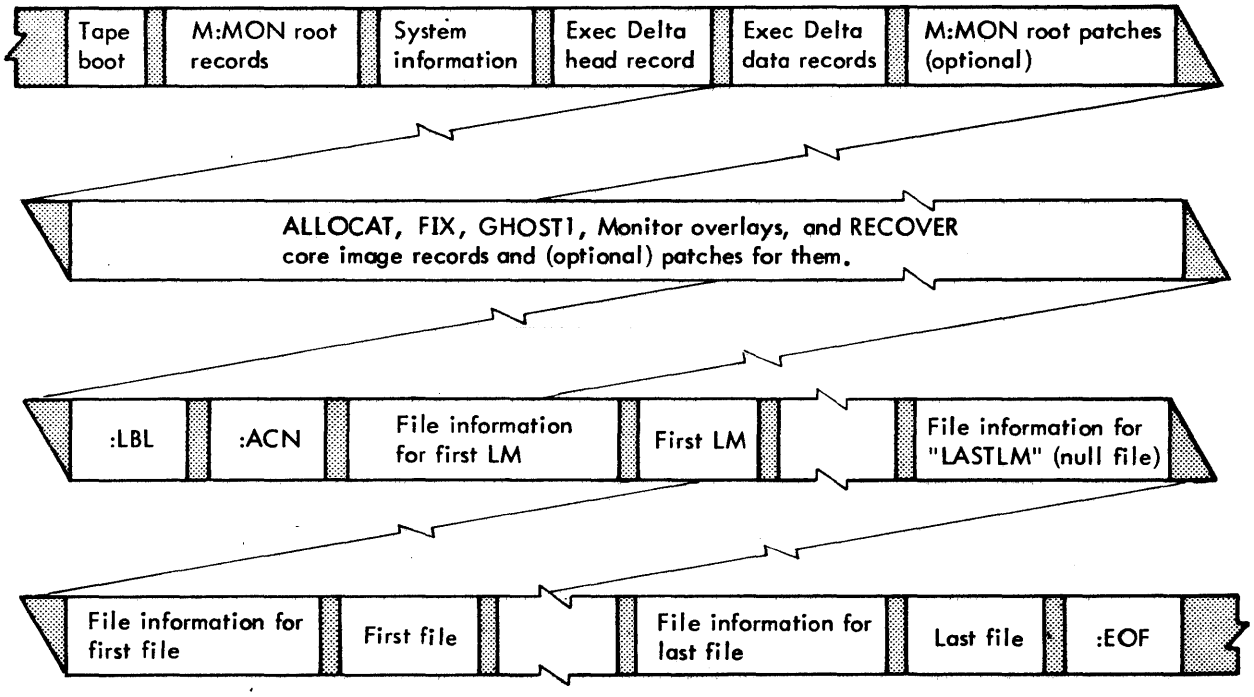
If a segname is present, the loc field must represent a location in the corresponding segment or the loc field (and value field) must be null. The latter type of patch would have the format.

segname// (the third slash is unnecessary)

and must be the first patch with its particular segname. (An example of this form of patch is given in the 'Conditional Patch Control Commands' section below.)

Example:

```
OPEN// START THE OPEN PATCHES
OPEN/OPNH+.52/B PATCH/
/PATCH/LW,13 TABLES+.74/
```




Note:

| | |
|---|---|
| Record sizes | The tape bootstrap is 22 words long. Patch records are 20 words long. All other records are 512 words long. The figure indicates groups of such physical records. |
| Head | Head portion of load module. |
| Data | Protection type 0 portion of load module. |
| Patches | Patches are included on the tape where shown if they exist in the file assigned to the M:PATCH DCB when DEF creates the system tape. The first group of M:MON root patches follows the Exec Delta data records. Any others are placed among segment patches according to their order in the patch file. The last record of each group of patches on the tape is the first patch for the next set of segment patches. The second through the last patch for a segment follow the segment to which they will be applied. GENMD patches follow the last of any patches following the RECOVER patches. |

Figure 5. Format of Master System Tape

```
**************************
       C P - V
SYSTEM GENERATED ON:
   12:00 AUG 16,'74
VERSION NO. IS:     C00
**************************
 PATCH SEGMENT NAMES:
             (ROOT)
   ALLOCAT0  (DATA)
   ALLOCAT1  (PROC)
   FIX0      (DATA)
   FIX2      (DCBS)
   FIX1      (PROC)
   GHOST10   (DATA)
   GHOST12   (DCBS)
   GHOST11   (PROC)
   CLOSE     (DATA)
   DEBUG     (DATA)
   ENQ       (DATA)
   KEYIN     (DATA)
   LDLNK     (DATA)
   LTAPE     (DATA)
   MISOV     (DATA)
   MULOV     (DATA)
   OPEN      (DATA)
   OPENTP    (DATA)
   RMAOV     (DATA)
   RTNRRT    (DATA)
   STEPOVR   (DATA)
   TPOV1     (DATA)
   TPOV2     (DATA)
   UMOV      (DATA)
   RECOVER   (DATA)
**************************
```

Figure 6. Segment Patching Order

If no segname is present, any location between $10_{16}$ and $FFF0_{16}$ may be patched. Such patches may appear anywhere within the patch deck.

loc is a Delta format symbolic location, possibly with offsets.

value is the Delta format value to be inserted at loc.

old value is the Delta format value of the previous contents of loc.

Example:

```
/IORT+.F8/PSM,9 TSTACK(PSM,6 TSTACK)/ FIX SIDR #6646
```

If a patch command is in error (e.g., has an illegal character, an incorrect old value, a value occupying more than one word, or an invalid loc value), it will be typed on the OC device. The operator must determine what was wrong and correct the problem.

If the error is apparent from examination of the patch, it can be corrected and the boot process restarted. If desired, the system may be examined with Executive Delta, which is now in control and requesting commands at the operator's console. The patch in error may be corrected from the operator's console using Delta by entering the patch correction mode by keying ¬ (use right bracket ]) on the Xerox 560) and then the correct patch in the form given above. After receiving the correct patch, the system resumes reading patches.

## PATCH DECK SYMBOL TABLES

The Delta format symbolic values that are recognized in patches are assembled by the system tape definition processor, DEF, from the REF/DEF stacks of the patchable modules using these items:

1. All DSECT names.
2. All DEFs ending in a colon (the colon is removed in the patch deck symbol table).
3. The first UDEF after each CSECT unless a colon DEF intervened.
4. Patch segnames.

For M:MON only, all LDEFs are also included. The symbols obtained from M:MON and XDELTA are available to XDELTA at any time. Those from other modules are available only while that module is being patched. DEF lists the symbols that are included as the tables are created.

In addition, two special symbols are available during the patching process.

The first is the symbol @ whose value is equal to the next available location in the patch area of the monitor. That is, it is initially equal to the monitor symbol, MPATCH, and its value is incremented by one each time a patch is encountered whose loc field is equal to the current value of @. The use of the special symbol @ frees the user from having to allocate space in the PATCH area of the monitor since Executive Delta will automatically relocate the patch area.

Example:

The following two patch decks are equivalent:

```
/IORT+.F8/B @/          /IORT+.F8/B PATCH/
/IORT+.FE/B @+1/        /IORT+.FE/B PATCH+1/
/@/LI,3 12/             /PATCH/LI,3 12/
/@/CB,3 5/              /.+1/CB,3 5/
/@/BNE $+2/             /.+1/BNE $+2/
/@/B IORT+.F9/          /.+1/B IORT+.F9/
/@/LI,3 0/              /.+1/LI,3 0/
/@/B IORT+.F9/          /.+1/B IORT+.F9/
```

The second special symbol is @@ and is used when an even address in MPATCH is required. The only restriction on this special symbol is that @@ cannot be referenced while patching @ (e.g., /@/@@/). The results are unpredictable.

Example:

```
/MM+.64/LPSD,8 @@/

/@@/MM+.65+4**28/

/@/.17000000/
```

New symbols may be added to the symbol table by including symbol definition patches in the patch deck. Symbol definition patches must have the format

# symbol = value

where

symbol is any Delta format symbol. (The symbol can be no longer than eight characters.)

value is any evaluatable expression terminated by a blank.

Example:

```
#GRUNCH=.D87

/GRUNCH/B GRUNCH+.20/

/.+1/B @/

/@/LW,3 TABLES+3/

#JK=@

/@/CI,3 10/

/@/B GRUNCH+.50/

.

.

.

/55+.1E8/B JK/
```

In the above example, the patch at 55+.1E8 branches to the instruction CI,3 10.

## RECONFIGURATION AND PARTITIONING COMMANDS

These commands provide a means of reconfiguring the system and partitioning devices and/or controllers at boot-time. All of the commands begin with a colon (:) and must end with a period or a trailing blank by at least column 72. The commands may be specified in any order with the exception of :END which must appear last (if it is used).

If no reconfiguration and partitioning commands are specified, the system responds as if the :GO command had been specified.

Three of these commands (:TYPE, :PART, and :REMOVE) contain the following parameter as part of the command format:

value (sometimes referred to as $value_1$ and $value_2$)

The description of this parameter is quite detailed. To avoid repeating the description several times, it will be given here and references will be made back to this section in the command descriptions.

The format of value is dependent on the CPU being used.

For Sigma 6/7/9 systems, value must be in the format

ndd

where

n represents a controller address and is specified as a letter. See Table B-2 in Appendix B.

dd specifies the device number. See Table B-3 in Appendix B.

For Xerox 560 systems, value may take one of two formats. The first format is

ndd

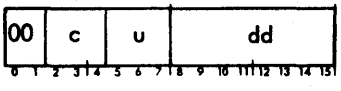
where

n represents a cluster number and a unit number.

See Table B-4 in Appendix B.

dd specifies the device number. See Table B-3 in Appendix B.

The second format consists of four hexadecimal digits which represent a hardware address in the format




where

c specifies the cluster number.

u specifies the unit number.

dd specifies the device number.

**:GO** This command specifies that the configuration specified on the system tape is to be used as is. The format of the command is

:GO

If :GO is specified, :TYPE and :REMOVE commands are not meaningful and the following message is output on the LL device:

| ALL :TYPE/:REMOVE COMMANDS IGNORED |
|---|

If no reconfiguration and partitioning commands are specified, the system responds as if the :GO command had been specified.

**:SAVE** This command specifies that all device addresses not changed by :TYPE commands are to remain as is, except according to restrictions listed in the following description. The format of the command is

:SAVE

When the :SAVE command is specified, the following message is output on the LL device:

```
**KEEP ALL DEV. ADDR. AS IS EXCEPT FOR :TYPE/
:REMOVE CHANGES
```

When the :SAVE command is used, only those device addresses which are different on the target machine from that of the SYSGENed system tape need be changed by the :TYPE command. All others remain as SYSGENed except when a :TYPE command redefines one or more device addresses for a specific device type. In this case, every device address of that device type must be defined by :TYPE commands whether or not the device address needs to be changed or the undefined ones will be removed from the system. The :REMOVE command may also be used to remove SYSGENed devices.

When :SAVE is not specified, all device addresses must be specified by :TYPE commands. Any SYSGENed devices for which addresses are not defined by :TYPE commands are removed from the system configuration (and cannot be returned to the system configuration without rebooting).

**:TYPE** The :TYPE command defines a device type, its model number, and its new device address or addessses. The format of the command for single access device definitions is

:TYPE device, value[, value]...

and the format for dual access device definitions is

:TYPE device, ($value_1$, $value_2$)[, ($value_1$, $value_2$)]...

where

device is a six character field. The first two characters specify the device type (e.g., CR) and the last four characters specify the device model number in hexadecimal.

value specifies the device address in the format described at the beginning of this section. The number of addresses depends upon the number of devices of that device type which are on the target machine or which need address changes (when :SAVE is used). For dual access devices, $value_1$ specifies the primary path address and $value_2$ specifies the alternate path address. When a device address change is required for a specific device type, all addresses must be specified even if no change is necessary, or those not specified for the device type will be removed from the system.

The model number is verified as a legitimate model number by searching the M:MODNUM table. (See the SYSCON chapter in the CP-V/SM Reference Manual, 90 16 74.) When found, its corresponding controller model number is obtained from the M:MODNUM table. The device/controller model numbers are then used to check if this combination is the same as that which was originally SYSGENed for the given device. If not the same, all similar device/controller model number combinations in M:MODNUM are used for this validation. As an example, if M:MODNUM contains the following entries:

| Device Model Number | Controller Model Number |
|---|---|
| 7120 | 7120 |
| 7120 | 7121 |
| 7121 | 7121 |
| 7121 | 7120 |

and the SYSGENed combination is

| | |
|---|---|
| 7120 | 7121 |

then the command :TYPE CR7121,.... will cause the following device/controller combinations to be checked with the indicated results:

| | |
|---|---|
| 7121/7120 | not valid |
| 7120/7120 | not valid |
| 7121/7121 | not valid |
| 7120/7121 | valid |

**:REMOVE** This command removes a device or controller from the system. The removed device or controller cannot be returned to the system without rebooting. (Although public disk packs are not partitionable, they can be removed from the system.) The format of the command is

:REMOVE {value / CONT, value}

where

value specifies the address of the device or controller to be removed in the format described at the beginning of this section.

CONT specifies that a controller is to be removed. When a controller is removed, all devices on that controller are also removed.

In the following example, four disk packs were SYSGENed and the target system is to have only two disk packs, one public and one private.

```
:TYPE DP7242,BF0,BF1,BF2

:REMOVE BF1
```

| SYSGENed Disk Packs | Result of :TYPE Command | Result of :REMOVE Command |
|---|---|---|
| AF0 - public | BF0 - public | BF0 - public |
| AF1 - public | BF1 - public | removed |
| AF2 - private | BF2 - private | BF2 - private |
| AF3 - private | removed | removed |

**:PART** The :PART command specifies a device or controller that is to be partitioned from the system. The device or controller is partitioned as if it had been partitioned by the SYSCON processer and can be returned to the system via SYSCON without re-booting the system. (Refer to the SYSCON processor description in the CP-V/SM Reference Manual, 90 16 74.) This is useful when a system is being booted and a partitionable device which was SYSGENed to be part of the system is currently unavailable. The format of the :PART command is

:PART {value / CONT, value}

where

value specifies the address of the device or controller to be removed in the format described at the beginning of this section.

CONT specifies that a controller is to be partitioned. When a controller is partitioned, all devices on that controller are also partitioned.

A device partition request causes all devices which have identical device addresses to be paritioned.

Example:

A system was SYSGENed to have four 9-track tape drives but two are down for maintenance when the system is booted.

```
:TYPE 9T7322,A80,A81,A82,A83

:PART A82

:PART A83
```

**:END** The :END command defines the end of the set of reconfiguration and partitioning commands. The command is optional because the occurrence of either the first nonroot patch or an asterisk (*) command would also indicate the end of reconfiguration and partitioning commands.

The format of the command is

```
:END
```

When the end of reconfiguration and partitioning commands is encountered, all :TYPE command definitions are processed first, then all :REMOVE requests, and finally all :PART requests.

## RECONFIGURATION AND PARTITIONING EXAMPLE

In the following example, a CP-V system was SYSGENed for four different hardware configurations. These configurations are referred to as the 560X, 7T, 7D, and 7E. A set of reconfiguration and partitioning commands was generated for each machine with the set of commands for each machine being bounded by a conditional patch control command. The four sets of reconfiguration and partitioning commands exist in the patch deck. The one set that is to be used for a particular boot is selected by a set of conditional patch control commands such as the following:

```
#560X = 0

#7T = 0

#7D = 1

#7E = 0
```

The above commands indicate that the 7D machine is to be booted. Figure 7 lists the entire set of devices that were SYSGENed for this example. Figure 8 lists the set of reconfiguraction and partitioning commands which were ignored because they were for machines not being booted. Figure 9 lists the set of reconfiguration and partitioning commands which were used in the boot process because the 7D was selected. Figure 10 lists the set of devices for the 7D configuration. This information is listed on the line printer during a boot, but not necessarily in the order shown in the Figures.

## RECONFIGURATION AND PARTITIONING MESSAGES

Table 5 lists the messages that may be output when reconfiguration and partitioning commands are being processed.

When an error is encountered, the error message is preceded by a message containing a dollar sign ($) beneath the character position in the command at which the error was found Processing of the command in error is discontinued.

```
*****************************************************************************
              D E V I C E   R E S O U R C E   C O N F I G U R A T I O N
DEV-TYP : DEV-ID : DEV-ADDR : DCT-I : PUB/PRIV : TYPE : RTOT : GENERAL INFORMATION
-----------------------------------------------------------------------------

 TY7012    TYA01     0001      01                 TY            NOT-PARTITIONABLE
 CR7140    CRA03     0003      02                 CR            NO-PART-DEV   :SYM
 CP7160    CPA04     0004      03                 CP            SYMBIONT
 LP7445    LPA02     0002      04                 LP            NO-PART-DEV   :SYM
 LP7445    LPA0F     000F      05                               SYMBIONT
 DC7212    DCBF0     01F0      06                 DC            NOT-PARTITIONABLE
 DC7212    DCBF1     01F1      07                               NOT-PARTITIONABLE
 DC7232    DCCF0     02F0      08                               NOT-PARTITIONABLE
 DC7232    DCCF1     02F1      09                               NOT-PARTITIONABLE
 9T7322    9TA80     0080      0A                 9T     06     NO-PART-CONT
 9T7322    9TA81     0081      0B                               NO-PART-CONT
 9T7323    9TA82     0082      0C                               NO-PART-CONT
 9T7323    9TA83     0083      0D                               NO-PART-CONT
 9T7323    9TA84     0084      0E                               NO-PART-CONT
 9T7323    9TA85     0085      0F                               NO-PART-CONT
 DP7242    DPD80     0380      10       PUB       DP     02     NOT-PARTITIONABLE
 DP7242    DPD81     0381      11       PUB                      NOT-PARTITIONABLE
 DP7242    DPD82     0382      12       PRIV
 DP7242    DPD83     0383      13       PRIV                     NO-PART-DEV
 DP7271    DPAE0     00E0      14       PUB                      NOT-PARTITIONABLE
 DP7271    DPAE1     00E1      15       PUB                      NOT-PARTITIONABLE
 DP7271    DPAE2     00E2      16       PUB                      NOT-PARTITIONABLE
 DP7271    DPAE3     00E3      17       PUB                      NOT-PARTITIONABLE
 DP7271    DPAE4     00E4      18       PUB                      NOT-PARTITIONABLE
 DP7271    DPAE5     00E5      19       PUB                      NOT-PARTITIONABLE
 DP7271    DPAE6     00E6      1A       PUB                      NOT-PARTITIONABLE
 DP7271    DPAE7     00E7      1B       PUB                      NOT-PARTITIONABLE
 DP7271    DPAF0     00F0      1C       PUB                      NOT-PARTITIONABLE
 DP7271    DPAF1     00F1      1D       PUB                      NOT-PARTITIONABLE
 RBFFFF    RBA16     0016      1E                 RB            NO-PART-DEV
 XP1200    XPC0D     020D      1F                 XP            NO-PART-CONT  :SYM
 ME7611    MEA10     0010      20                 ME            NOT-PARTITIONABLE
 ME7611    MEA11     0011      21                               NOT-PARTITIONABLE
*****************************************************************************
*
```

Figure 7. Device Resource Configuration from SYSGEN

```
.............................#7E
.............................:SAVE
.............................:TYPE RBFFFF,A14
.............................:TYPE DCBF0
.............................:REMOVE A84
.............................:REMOVE A85
.............................:TYPE DP7271,D90,D91
.............................:TYPE DP7242,D80,D81
.............................:END
#
.............................#7T
.............................:SAVE
.............................:REMOVE CONT,D80
.............................:END
#
.............................#560X
.............................:GO
.............................:END
#
```

Figure 8. Reconfiguration and Partitioning Commands that were Ignored

NOTE: When a device-type (DEV-TYP) is a pooled device (i.e., dual access), the information for the alternate device is the same as that for the primary device as listed above, except, the DEV-TYP column contains "DUAL" and the DEV-ADDR column contains the alternate device

```
#7D
:SAVE
**KEEP ALL DEV.ADDR.AS IS EXCEPT FOR :TYPE/:REMOVE CHANGES
:REMOVE A0F
:REMOVE A16
:TYPE 9T7322,A80,A81
:TYPE DC7212,BF0
:REMOVE CONT,AF0
:REMOVE CF1
:REMOVE CONT,AE0
:TYPE DP7242,D80
:TYPE ME7611,A05
:END
```

Figure 9. Reconfiguration and Partitioning Commands that were Used

```
*****************************************************************************
          D E V I C E   R E S O U R C E   C O N F I G U R A T I O N
DEV.TYP : DEV.ID : DEV.ADDR : DCT.I : PUB/PRIV : TYPE : RTOT : GENERAL INFORMATION
-----------------------------------------------------------------------------

 TY7012    TYA01     0001      01                 TY            NOT.PARTITIONABLE
 CR7140    CRA03     0003      02                 CR            NO.PART.DEV   :SYM
 CP7160    CPA04     0004      03                 CP            SYMBIONT
 LP7445    LPA02     0002      04                 LP            NO.PART.DEV   :SYM
 DC7212    DCBF0     01F0      06                 DC            NOT.PARTITIONABLE
 DC7232    DCCF0     02F0      08                               NOT.PARTITIONABLE
 9T7322    9TA80     0080      0A                 9T     02     NO.PART.CONT
 9T7322    9TA81     0081      0B                               NO.PART.CONT
 DP7242    DPD80     0380      10       PUB       DP     00     NOT.PARTITIONABLE
 XP1200    XPC0D     020D      1F                 XP            NO.PART.CONT  :SYM
 ME7611    MEA05     0005      20                 ME            NOT.PARTITIONABLE

*****************************************************************************
```

Figure 10. Device Resource Configuration for the Booted System

Table 5. Reconfiguration and Partitioning Messages

| Message | Description |
|---|---|
| ALL :TYPE/:REMOVE COMMANDS IGNORED | A :GO command has been specified. :TYPE and :REMOVE commands are not meaningful. |
| **device, value CANNOT BE ADDED TO SYSTEM | As the result of the :TYPE command, the SYSGENed system and target machine device/controller model number definitions are not equivalent. This message is preceded by a message containing a dollar sign ($) under the device type and also under the first device address for single access or the alternate device address for dual access devices.<br>device - device type and model number.<br>value - device address (in the format ndd described at the beginning of this section).<br>This message can also appear when there are more :TYPE definitions for the device type than allowed for in the SYSGENed system. |
| CANNOT PARTITION, CONT. ndd ALREADY PARTITIONED | The controller specified on a :PART command has already been partitioned. |

Table 5. Reconfiguration and Partitioning Messages (cont.)

| Message | Description |
|---|---|
| CANNOT PARTITION, CONT. ndd NON-PARTITIONABLE | The controller specified on a :PART command is not partitionable. (it is a controller for a Teletype, a RAD, or a COC, or it was defined at SYSGEN to be a non-partitionable controller.) |
| CANNOT PARTITION, CONT. ndd NOT PRESENT | The controller specified on a :PART command either does not exist or was removed in the reconfiguration process. |
| CANNOT PARTITION, CONT. ndd NOT PRIVATE PACK | A disk pack controller was specified on a :PART command and one or more of its associated disk pack spindles is public. Public disk pack spindles cannot be paritioned. |
| CANNOT PARTITION, DEV. ndd ALREADY PARTITIONED | The device specified on a :PART command has already been partitioned. |
| CANNOT PARTITION, DEV. ndd NON-PARTITIONABLE | The device specified on a :PART command is not partitionable. (It is either a Teletype, a RAD, or a COC, or it was defined at SYSGEN to be a non-partitionable device.) |
| CANNOT PARTITION, DEV. ndd NOT PRESENT | The device specified on a :PART command either doesn't exist or was removed in the reconfiguration process. |
| CANNOT PARTITION, DEV. ndd NOT PRIVATE PACK | Public disk pack spindles cannot be partitioned. |
| CANNOT REMOVE, CONT. ndd NOT PRESENT | The controller specified on a :REMOVE command either does not exist or was previously removed in the reconfiguration process. |
| CANNOT REMOVE, DEV. ndd NOT PRESENT | The device specified on a :REMOVE command either does not exist or was previously removed in the reconfiguration process. |
| CONT. ndd PARTITIONED | The specified controller has been successfully partitioned. |
| CONTINUATION ILLEGAL | Continuation commands (i.e., commands containing a semi-colon) are not allowed. |
| DEV. ndd PARTITIONED | The specified device has been successfully partitioned. |
| DUAL ACCESS DEFINED ILLEGAL $ndd_1$, $ndd_2$ | On a :TYPE command, the primary address and the alternate address on a dual access device are equivalent. |
| DUAL/SINGLE ACCESS MIXTURE | A :TYPE command specifies both single access and dual access device addresses; or the device type is for a single access device and the address is for a dual access device, (or vice versa). |
| INVALID TERMINATOR | A bad or unknown terminator terminates a field or option. Valid control command terminators are NEW LINE, period, carriage return, trailing blank, and end of control command image. |
| **KEEP ALL DEV. ADDR. AS IS EXCEPT FOR :TYPE/:REMOVE CHANGES | A :SAVE command has been encountered. |

Table 5. Reconfiguration and Partitioning Messages (cont.)

| Message | Description |
|---|---|
| NO RECONFIGURATION PERFORMED DUAL | The :GO command was specified, or no : commands were specified, or a :END command was specified by itself. |
| ACCESS DEFINITION CONFLICTS<br>(ndd$_1$[, ndd$_2$]), (ndd$_3$[, ndd$_4$]) | A device address conflict has occurred as the result of :TYPE commands. Either a single access device address is the same as a primary or alternate address on a dual access device, or the primary address is the same as the alternate address on a dual access device. The ndds indicate the addresses involved. This message will appear twice for each conflict encountered. |
| ***NO SPACE LEFT FOR CONFIG. INFO | Too many :TYPE,:REMOVE, and :PART command definitions have been encountered. The total size of the internal buffer which retains reconfiguration and partitioning command is 512 words. Each :PART and :REMOVE command requires one word, each :TYPE command for singie access controllers requires two words, and each :TYPE command fur dual access requires three words. Additionally, the buffer contains one control word. The buffer is needed to retain all control command information until every command has been processed. Actual processing of the commands takes place when the :END command, the first nonroot patch, or an asterisk command is encountered. |
| **PACK yyndd PARTITIONED, DIAL ndd NOT AVAILABLE | The device specified on a :PART command is a disk pack spindle. |
| **TAPE yyndd PARTITIONED, DIAL ndd NOT AVAILABLE | The device specified on a :PART command is a tape drive. |
| UNKNOWN COMMAND, FIELD, OR VALUE | An unknown command, an invalid name, or a value field which contains too many characters, is not hexadecimal, or is not in the correct format for the particular machine was encountered. This message also appears for each reconfiguration and partitioning command encountered after reconfiguration and partitioning processing has ended. It also appears when a :GO, :REMOVE, or :TYPE command is encountered after a :GO command has been processed and when a :SAVE command is encountered after a previous :SAVE command was processed. |

### :GENDCB COMMAND

This command defines the system DCB associated with tape input during PASS0. This command is required only if the files are on a different tape than the boot tape or if they occupy more than one reel. If the command is not present in the patch deck, PASS0 reads the account and serial number from the tape and performs an automatic premount of the tape. No operator intervention is required.

The format of the GENDCB command is:

:GENDCB (M:BI, account[, password];

:, (INSN, value[, value]...), device)

where

M:BI specifies that tape input is to be through the M:BI DCB. No other DCB is valid for this command.

account specifies an account identifier (up to eight eight alphanumeric characters) associated with the labeled tape to be read during PASS0.

password is the password associated with the labeled tape to be read during PASS0. The password (if any) must correspond to that specified when the tape was created, and may be up to eight alphanumeric characters in length.

INSN, value, ... specifies the serial number(s) (up to four alphanumeric characters in length) of the tape(s) to be read by PASS0. No more than three reels may be specified. The first reel specified must contain the first file to be read, and may be different from the reel used to boot the monitor.

device specifies a tape-type device code (e.g., 9T, 7T).

Example:

```
:GENDCB (M:BI,ACCT1,PASS1,;

:(INSN,001,002),9T)
```

Any number of GENDCB commands may appear in the patch deck. Only the last will be applied. If it is defective, files will be copied from the boot tape.

Any errors in the command are indicated by the message

```
***GENDCB ERROR
```

on the OC and LL devices.

## GENMD COMMANDS

The GENMD commands are used in conjunction with the GENMD patches described below. The three GENMD commands are GENMD, LIST, and DELETE. .

**GENMD** This command indicates which file is to be patched next. A GENMD command must precede the set of patches for each file to be patched. Any number of sets of patches to the same file may be present, provided each is preceded by a GENMD command. The format of the command is

```
GENMD filename
```

**LIST** This command lists the patches currently in the file being patched and has the format:

```
LIST
```

**DELETE** When a file is patched, a record is kept of the list of patches to the file within the file itself. The DELETE command removes this list of patches from the file (but does not remove the effect of the patches on the file). The command may be used to prevent files from growing too large if they are not restored when applying a new patch deck. The format of the command is

```
DELETE
```

## GENMD PATCHES

GENMD patches are used to modify nonresident elements of the system.

GENMD patches have the format:

$$\left\{\begin{matrix}\text{:GENMD [,segname]} \\ \text{[segname],}\end{matrix}\right\} \text{loc, value[,value]...[. comment]}$$

where

segname specifies the overlay segment name to be patched. If not present, the most recently specified segname is assumed. If not present and no segname was specified previously, the root segname is assumed.

loc specifies the location to be patched and has the format[name] [±hex value]. The hexadecimal value is added to or subtracted from the absolute address of name. A maximum of eight characters may be used for the hexadecimal value. The name need not be defined in any particular overlay since all the stacks are searched. If more than one overlay defines the same name, the first is used.

value specifies the value to be inserted at loc. If more than one value is specified, they will be inserted at successive locations. Each value must have the format

```
hex value[±name[±name]...]
```

The absolute address of the names are added to or subtracted from the hexadecimal value. A maximum of eight characters may be used for the hexadecimal value. The name need not be defined in any particular overlay since all the stacks are searched. If more than one overlay defines the same name, the first is used.

A GENMD command may be continued by terminating the first line with a semicolon (;). The semicolon must not divide a name or a hexadecimal string and is not permitted where a blank is required. The continuation line must begin with a colon (:) if the continued line began with :GENMD. Otherwise, the continuation line begins with the next character of the command.

## GENMD ERROR MESSAGES

Table 6 lists the error messages that may be output when GENMD commands and patches are being processed.

## CONDITIONAL PATCH CONTROL COMMANDS

A conditional patch control command specifies whether the patches that follow are to be used as patches or are to be effectively ignored. The conditional patch control command controls the SKIP flag. When the SKIP flag is set, all subsequent patches are effectively ignored until the SKIP flag is reset. The conditional patch control command can appear any number of times and anywhere within the patch deck (including the GENMD portion). The command has the format:

```
#[value]
```

where value is any well-formed, but not necessarily evaluatable, expression terminated by a blank. The value expression may contain an undefined symbol.

Table 6. GENMD Error Messages

| Message | Description |
|---|---|
| BAD LMN - 0000 | The file is not a load module. |
| BAD LMN - xxxx | An error occurred when accessing the load module. The code and subcode are indicated by xxxx. |
| BAD SEG | A segname is not in the TREE. |
| DLM AT xx | The delimiter in column xx is not what it should be. |
| **nn GENMD ERRORS DETECTED | This message is output on the OC and LL devices at the conclusion of the GENMD patching process and indicates how many errors occurred. |
| HEX AT xx | The hexadecimal number ending in column xx is null, too large, or not hexadecimal. |
| LOC AT xx | The location ending in column xx or whose value ends in column xx is not contained in the segment. |
| NAME AT xx | The name ending in column xx is null or is not in the load module's stacks. |
| NO FILE NAMED | A 'GENMD filename' command has not yet been encountered or has no filename on it. |
| TOO BIG | Not enough core is available to read the REF/DEF stack. It may be possible to do the patch if all names are converted to absolute hexadecimal values, since the stacks are read only if a name is used. |

If value contains an undefined symbol, is negative, or is zero, the SKIP flag is set. While the SKIP flag is set, only the segname field of a patch is examined to determine when the current segment's patches end. If value is absent or greater than zero, the SKIP flag is reset and normal patching resumes. The special symbol ELSE may be used to toggle the setting of the SKIP flag.

The SKIP flag is also changed when a Delta format patch that does not have a loc and value field is encountered (i.e., segname//). In this case, it is set if the segname is undefined and it is reset otherwise.

Examples:

1. The following patches will be included only if the system was generated for a large Sigma 9 or a large Xerox 560.

```
#:BIG
/SWAPPER+.C5/B @/
/@/L1,5 0/
/@/SLS,7 L1/
/@/B SWAPPER+.C6/
#
```

2. The following patches will be included only if the ENQ/DEQ feature was included in the system:

```
ENQ//
ENQ/ENQO+.266/B @(CW,13 ENQP+.1F4)/
/@/LB,15 ENQP+.1F4/
/@/CB,15  13/
/@/B ENQO+.256/
OPEN//
```

3. The symbol BPS will be set to one if the system was generated for a pack swapper with greater than 128K. Otherwise, it will be set to zero.

```
#:BIG*DPSIO
#BPS=1
#ELSE
#BPS=0
#
```

### COMMENT CARDS

Comment cards may appear anywhere within the patch deck. In the portion of the deck that contains Delta format patches and symbol definition patches, the comment card must contain a 'less than' character (<) in column one. In the GENDCB and GENMD portion of the deck, comment cards must contain one of the following in column one:

```
<
*
.
```

### PATCH FILE CREATION

All patches read during the startup of the system (except GENDCB commands) are copied to the file PATCH in the system account. Those that were read while the skip flag was set appear with the word SKIP in columns 77-80. The resulting file may be used as input to DEF to create a system tape with the complete, current patch deck on it.

## SEQUENCE OF OPERATIONS

The master system tape is loaded into the machine by use of the standard load procedure described in the CP-V/OPS Reference Manual, 90 16 75. The hardware bootstrap loads and enters the tape boot at the beginning of the system tape. This tape boot, in turn, loads the monitor root and the following functions are then performed.

If the system was generated with the BIG option on the :MON card and is not being booted on a Sigma 9 or Xerox 560, the following message is output to the operator's console and the bootstrap operation is terminated.

```
SYSTEM REQUIRES SIG9 OR X560
```

The operator's console (OC) device address is validated. If the actual OC device address is different than that of the SYSGENed address, the system will halt (wait). The operator should enter the appropriate OC address into register 0.

To enter the OC address on a Sigma machine:

1. Put the machine in IDLE.
2. Set the SELECT ADDRESS switches on the control panel to 0.
3. Enter the appropriate device address into the SELECTed ADDRESS.
4. Set the COMPUTE switch to the RUN position.

To enter the OC address on the Xerox 560:

1. Enter CONTROL P.
2. Enter 0/ (which displays the contents of register 0).
3. Enter the new device address followed by the letter M.
4. Enter an X. (This will cause the 560 to resume processing.)

After the OC address has been validated, the following message is output to the operator:

```
ENTER ANY OF:
I = TTY I/O
P = LP OUTPUT
F = TAPE FILES
S = :SYS FILES
T = TAPE PATCHES
C = CARD PATCHES
D = XDELTA
```

The operator must respond within 10 seconds by typing one or more of the characters above followed by new line or by entering new line alone. If new line alone or nothing is entered, T is assumed by default. If any characters other than those listed above are entered, they are ignored.

The letters have the following meanings:

- **I** specifies that the operator wants to read and respond to the normal OC messages during the boot. Otherwise, default responses are assumed up to the date/time request (see below) and normal output is suppressed. (Error messages will still be output.)
- **P** causes output to the LL device to occur. Otherwise, the printer is not used.
- **F** causes PASS0's tape copy operation to occur. Otherwise, a boot-under-the-files occurs.
- **S** causes the files to be copied from the POtape into :SYS without destroying the entire file system.
- **T and C** define that the patch deck(s) are to come from tape or cards respectively. Either, neither, or both may be specified. If both are specified, cards will be read first for root patches and last for overlay patches and GENMD commands. Card patches meant to repatch tape root patches should therefore be placed after a nonroot patch. Patches of the format **segname//** should be used in both patch decks to prevent the switching of devices from splitting up a logical patch.
- **D** causes Executive Delta to be retained after the boot for debugging purposes.
- **N** is meaningful only by itself and means "none of the above".

The message

```
NEW FILE SYSTEM
```

indicates that F was specified. If F was specified, the old file system is not destroyed until the entire patch deck has been read. The bootstrap operation may be halted at any time during this interval by triggering a console interrupt. The message

| INITIALIZATION HALTED – RESTART O.K. |
|---|

indicates a successful halt.

If I was specified and if the system includes the real-time option, the system then issues the following message.

| RESET RESDF YYY,XXXXX |
|---|

This message allows the operator to override the SYSGEN-defined values for the size of the RESDF area (dedicated real-time memory pages) and its starting address. The operator should respond:

[yyy][,xxxxx] ⏎

where

yyy is the optional decimal number of pages to be in the RESDF area; a value of 0 through 999 may be used.

xxxxx is the optional hexadecimal word address of the first page to be in the RESDF area. Any page address representing a value greater than or equal to $10,000_{16}$ (64K) may be used.

If either or both optional parameter(s) are not specified, the SYSGEN-defined default(s) will be used.

CP-V will then request the date

| DATE (MM/DD/YY) = |
|---|

and the operator should enter the date (e.g., 2/5/74).

Then CP-V will request the time

| TIME (HH:MM) = |
|---|

and the operator should type the time, which is represented by a 24-hour clock (e.g., 6:05 PM is typed as 18:05). The time of day is typed at the left margin of the console once every minute after the system has been initialized. The form of this type-out is

| hh:mm |
|---|

If the system is being loaded on a machine for which it was not SYSGENed, one of the following messages will be displayed on the OC device and the bootstrap operation will be terminated.

| SYSTEM NOT SYSGENED FOR SIGMA 6 |
|---|

| SYSTEM NOT SYSGENED FOR SIGMA 9 |
|---|

| SYSTEM NOT SYSGENED FOR XEROX 560 |
|---|

If the system and target machines match and if I was specified the following message is displayed:

| C/LL/DC ASSIGN OK (YES/NO) |
|---|

If the operator's response is YES or ⏎, it is assumed that the device addresses for the control device, listing log, and system device are not to be changed from those established when the monitor was defined. If the response is NO, then the following messages will be output to redefine these device addresses.

| CRndd ⇒ CR |
|---|

| LPndd ⇒ LP |
|---|

| DCndd ⇒ DC |
|---|

where each ndd is the current device identification and as many DC messages are output as there are swap devices.

In response to each of these messages the operator must type two or three characters. If two characters are typed, they must be 'SA' and indicate no change for this device. If three characters are typed, they must be the channel and device designation codes (ndd) defining the address of the indicated device (see Appendix B, Tables B-2 and B-3).

If the DC or swapper assignment is incorrect, one of the following five messages will be displayed. Three of the messages request a new swapper device address.

| NO RESPONSE FOR SWAPPER yyndd<br>yyndd => yy |
|---|

(The device address is unrecognizable by the hardware.)

| DISC PACK BAD yyndd<br>yyndd => yy |
|---|

(The disk pack spindle is not initialized or cannot be accessed due to I/O transmission errors.)

| SWAPPER NOT mmmm<br>DCndd = DC |
|---|

(The RAD or disk pack indicated by the device model number mmmm was expected as the swap device.)

| SWAPPER WRITE PROTECTED yyndd |
|---|

(The swapper device is write protected. If a write protect violation is encountered during the reading and patching of the monitor overlays, the above message is output and the system stops. However, if this occurs during swapper initialization, then a new swapper device is requested as for the above messages.)

| PSA TRACK FLAWED |
|---|

(The swapper disk pack contains flaws. The boot process terminates.)

Before completing any of the above responses with a (RAR) or (NL), the operator may cancel the response by striking the (EOM) key. Following this, or if a completed response is in error, the message

| ?? |
|---|

will be output and the key-in request will be repeated.

If no characters are typed within 10 seconds, a ⓝⓛ response is assumed.

After all necessary responses have been received, the boot subroutine reads the system information record from tape and writes it on the LL and OC devices if P and I are specified, respectively.

The following sense switch information is then listed on the OC device if I was specified.

```
SET SENSE SWITCHES AND TYPE N/L
SSW1 =>CHECKWRITE DISK WRITES
SSW2 =>NO AUTOMATIC LOGON/LOGOFF
SSW3 =>OPERATOR RECOVERY ON DISK BOOT
SSW4 =>SYSTEM SECURITY CHECKING
```

The system will continue when a NEW LINE or any other character is entered.

Next, the reconfiguration and partitioning commands (if any exist) are read and processed. A summary of the system's device will be output on the LL device (even if no :TYPE commands are encountered). Permanently down devices are not listed.

Next, the monitor patches are read and processed for the patching of the overlays, ALLOCAT, GHOST1, and RECOVER. (If the RECNFIG boot-time processor needs to be patched, XDELTA performs the patching as it does for the monitor root. However, these patches must precede the reconfiguration and partitioning commands in the patch deck.)

After the nonroot patches have begun, reconfiguration and partitioning commands are illegal. If any such commands appear in the deck, the following message is displayed on the OC device (and also on the LL device if P was specified) and the bootstrap continues.

```
':' COMMAND NOT IN PATCH DECK PROPERLY
```

This message is displayed only one time, even if additional reconfiguration and partitioning commands are encountered. It then copies the overlays, etc., to the swapping device, communicating the sizes and disk addresses to the resident root of the absolute monitor. Control then passes to another boot subroutine at WRTROOT. This second boot subroutine causes the monitor root to be copied to the disk, preceded by a disk bootstrap. At this point, the resident monitor is operational but the system has not yet been established on the resident swapping device. The GHOST1 processor performs this function.

If P was specified, GHOST1 determines whether any devices or controllers are partitioned. If none are partitioned, the following message is displayed on the LL device:

```
*** NOTHING PARTITIONED
```

However, if devices and/or controllers are partitioned, the following message is displayed on the LL device:

```
***** ITEMS PARTITIONED *****
```

followed by messages identifying each device or controller which is partitioned. The messages have the following formats:

```
DEV yyndd PARTITIONED
```

(for devices)

```
CONT yyndd PARTITIONED
```

(for controllers)

When all partitioned items have been identified, the following message concludes the list:

```
** END OF PARTITIONED ITEMS **
```

When P was not specified or when GHOST1 has completed the above listing, GHOST1 starts the symbiont ghost, Fix ghost, ERR:FIL ghost, and fill ghost, and then exits.

## BOOTING FROM DISK

Once the operating system has been bootstrapped from tape, it may thereafter be brought into core from the disk by means of the load procedure described in the CP-V/OPS Reference Manual, 90 16 75.

The hardware boot routine loads and transfers control to the disk boot which then loads the monitor root into core. The system requests the date and time and then asks

```
DO YOU WANT DELTA (Y/N)?
```

to determine whether the system debugger's memory should be released.

The following message is then output to the operator's console:

```
DO YOU WANT HGP RECONSTRUCTION (Y/N)?
```

A response of Y causes an HGP reconstruction of the public file system to be performed. If no response is received within one minute, N is assumed.

Partitioning information is displayed as described previously, and the system ghost jobs (Fill, ERR:FIL, and Fix) are started. Normal operation may then be resumed.

### BOOTSTRAP I/O ERROR RECOVERY

I/O error recovery during bootstrap is provided for the card reader, line printer, magnetic tape, and disk. However, error recovery is not possible until the tape boot and monitor root have been read from tape. The following error messages may appear on the OC device:

| |
|---|
| xx INOPERATIVE |
| xx ERROR. TIO value TDV value |
| xx MANUAL MODE |
| CHECKWRITE ERROR |

where

xx is MT, CR, DC, or LP.

value indicates the TIO or TDV results.

When either of the first two messages above occurs, the wait state is entered. To continue, the operator must place the CPU into IDLE, STEP, and then RUN state. The I/O will then be retried. If the third message above occurs, I/O will continue when the condition is corrected. When an error occurs for a magnetic tape or disk operation, the operation is retried ten times before an error message is output. If the fourth message above occurs, the wait state is entered. To continue, the operator must place the CPU into IDLE, STEP, and then RUN state. This message will appear if the checkwrite on disk fails. The checkwrite will be executed only if hardware sense switch 1 is set.

## PASS0 PROCESSOR

The PASS0 processor performs various system initialization functions and is entered automatically whenever a CP-V tape is booted.

PASS0 reads a tape specified by the user (via the GENDCB command) which contains the nonresident elements of the system (i.e., CCI, processors, libraries, etc.). (This is normally the labeled portion of the tape used to bootstrap the absolute monitor.) PASS0 allows the user to modify these elements via the GENMD portion of the deck.

### PASS0 MESSAGES

The messages in Table 7 may be output by the PASS0 program on the LL device. PASS0 continues its normal operation.

Table 7. PASS0 Messages

| Message | Description |
|---|---|
| ***CANNOT BOOT LMN | A load module cannot be read from the bootstrap tape because core is not large enough. PASS0 outputs the filename in error and continues to the next file, thus ignoring the file in error. |
| I/O ERR/ABN nn,xxxING FILE ffffffff ON dddd | An I/O error or abnormal condition has occurred on tape or disk.<br>nn is the error or abnormal code.<br>xxx is READ, WRITE, OPEN, or CLOS.<br>ffffffff is the current filename.<br>dddd is TAPE or DISC.<br>PASS0 continues after this message. |

# 4. MONITOR DUMP ANALYSIS PROGRAM

## INTRODUCTION

The monitor dump analysis program ANLZ (Analyse) is designed to aid in the debugging of CP-V crash dumps. ANLZ operates in the ghost, on-line, and batch modes. It accepts as input any tape or disk dump produced by the recovery procedure and any tape dump produced by executive Delta. If a tape is input, the ANLZ user must supply the tape type in response to the message

```
ENTER TAPE TYPE: 7T, 9T, BT, ETC...
```

Tape input results in the creation of a disk file (CP5DUMP); subsequent tape inputs replace the contents of this file.

### GHOST MODE

ANLZ is called automatically by the recovery procedure, and functions as a ghost job to interpret and summarize critical monitor tables and to dump the monitor's dynamic data area. When ANLZ is initiated after a system crash, it neither looks for nor accepts any commands, operating entirely on default options. It assumes an INPUT command option of LAST; if unable to open the last MONDMP file, it then assumes an INPUT command of TAPE. (Refer to the description of the INPUT command in the following text.) When Analyze is run in this manner, the output is an abbreviated form of the output produced by the ALL display command.

ANLZ is also automatically initiated after a single user abort. In this case, it functions just as though it had been initiated as a ghost job via an operator key-in. (This is described below.)

ANLZ may be called as a ghost job by the operator to examine the tape produced during an irrecoverable crash. The operator key-in used for this purpose is

GJOB ANLZ

ANLZ then asks the operator for a command:

```
ANLZ: ENTER COMMAND, N/L SAYS TO DO ALL
```

The operator may respond with one of the following commands:

NO - just exit.

TA - read a recovery-built tape.

ME - run interactively from the operator's console.

CP - read the CP5DUMP file.

1-7 - read the indicated MONDMP file.

? - list the ANLZ commands on the line printer.

N/L (new line alone) - do default ghost run.

In the interactive ghost mode, a key-in of

INT, id

will cause termination of the current ANLZ operation and a prompt for input. (id specifies the ANLZ user's number.)

### BATCH AND ON-LINE MODES

Any batch or on-line user may call ANLZ by specifying the name of the program. For on-line users, this program name is entered in response to a TEL prompt for commands, as follows:

!ANLZ ↵

Any user, in batch or in on-line mode, must have the proper privilege level (80 or better) to examine the monitor. If he does not, ANLZ outputs the following message

```
xx PRIVILEGE LEVEL NOT HIGH ENOUGH
```

where xx is the user's current privilege level. (Response messages are output on the line printer for a batch user.)

When accessed on-line, as an interactive ghost, or as a batch job, ANLZ is completely command-driven. It responds to commands that selectively display monitor tables, examine memory, and compare the dump with the running monitor.

An on-line user may terminate a display by depressing the BREAK key.

## COMMANDS

When ANLZ is first entered, it responds

```
ANALYZE HERE
```

and, if in on-line mode, it requests entry of an input command with the prompt character

<

All commands, options, and output are identical for batch, interactive ghost, and on-line modes.

### INPUT COMMAND

**INPUT** The INPUT command directs ANLZ to input from a particular disk or tape file, or to open a file. The format of the command is

IN[PUT]option

where option may be any one of the options shown in Table 8.

After reading a tape or disk file as directed by the INPUT command, ANLZ informs the user of the size of the file with the following message:

| THE LAST PHYSICAL PAGE IN THE FILE IS xx |
|---|

If in on-line mode, it then prompts (<) for the next command.

Table 8. INPUT Command Options

| Option | Meaning |
|---|---|
| TA[PE] | Directs ANLZ to read a tape created by the recovery process and to write it into the file CP5DUMP which is then used for input. |
| CP[5DUMP] | Directs ANLZ to open the CP5DUMP for input. |
| LA[ST] | Directs ANLZ to open the last file formed by the recovery procedure for input. (ANLZ must look at the running monitor to obtain this information.) |
| number | Directs ANLZ to open a crash file formed by recovery. Recovery file names are of the form<br><br>MONDMP(number)<br><br>where number is the number of the dump file (1 for the first dump since a "cold" start, 2 for the second, and so on). |

### DISPLAY COMMANDS

Three display commands may be used to output information from crash dumps. They are

DISPLAY

RUN

ALL

**DISPLAY** The DISPLAY command outputs information existing at the time of the crash. The format of the command is

DI[SPLAY]option

where option specifies the information to be displayed (Table 9).

**RUN** the RUN command outputs various linked lists of the monitor by running through the list and displaying each entry. The format of the RUN command is

RU[N] option

where option specifies the list to be printed (Table 10).

**ALL** The ALL command performs all of the functions of the dispaly commands described above and the functions of ANLZ (except dumps) when it is initiated by the automatic recovery procedure. The format of the command is

AL[L]

A numerically and alphanumerically sorted monitor map is output at the end of the ALL display.

### INTERACTIVE MONITOR DISPLAY COMMANDS

Commands in this group allow the user to examine either the dump or the running monitor. Both the monitor and user JIT and physical core may be examined. The commands are

loc

$loc_1, loc_2$

Line feed (or carriage return)

↑

*

MONITOR

loc = value

**loc** The loc command outputs the contents of the specified location. The format of the command is

loc

where loc is one hexadecimal value (1 to 8 hexadecimal digits) or two hexadecimal values separated by an operator indicating addition (+), subtraction (-), multiplication (*), or division (%). Note that loc values do not require a preceding delimiter character ".".

Table 9. DISPLAY Command Options

| Option | Meaning |
|---|---|
| AJ[ITS] | Displays JIT, AJIT, and context area of all incore users at the time of the crash. |
| AT[ABLES] | Displays the incore portion of ALLYCAT's tables. |
| AV[R] | Displays the tape and disk tables. |
| CI[TS][,index]... | Displays the requested entries of the Channel Information Table. Up to 20 entry indexes may be specified. If no index is specified, the entire Channel Information Table is displayed. |
| CO[C][,index]... | Displays the requested entries of the COC table. Up to 20 entry indexes may be specified. If no index is specified, the entire COC table is displayed. |
| CU[N] | Displays the current user's JIT, AJIT, and context area. |
| DC[T][,index]... | Displays the requested entries of the Device Control Table. Up to 20 entry indexes may be specified. If no index is specified, the entire Device Control Table is displayed. |
| EL[OG] | Displays and validates the incore error log buffers. |
| FM[,index]... | Displays the requested entries of the file management read-ahead tables. Up to 20 entry indexes may be specified. If no index is specified, the entire set of file management read-ahead tables are displayed. |
| FQ | Displays the Free I/O Queueing tables. |
| IO[,chan]... | Displays the device on the requested I/O channel. Up to 20 channels may be specified. If no channel is specified, the devices for all channels are displayed. |
| IQ[,index]... | Displays the requested entries of the I/O queueing tables. Up to 20 entry indexes may be specified. If no entry index is specified, the entire set of I/O queueing tables are displayed. |
| JIT[,id][,loc1,loc2] | Displays the contents of the JIT for the user specified by id. Loc1 and loc2 specify that only a portion of the JIT page is desired and represent a relative offset into the page in hexadecimal. If an id of 0 is given or if no id is specified, the monitor's JIT is displayed. |
| MR[,loc1,loc2] | Displays the requested portion of the monitor's root. The displacements (loc1 and loc2) must be absolute hexadecimal addresses. If no displacements are given, only the monitor's data area will be dumped. |
| OJ[IT] | Displays all of the out of core JITs at the time of the crash. |
| OS | Displays the user outswap tables (if outswap is in progress). |
| PA[RTITIONS][,index]... | Displays the requested entries of the partition tables. Up to 20 entry indexes may be specified. If no index is specified, the entire partition table is displayed. |
| PF[ILE] | Displays the patch file that was created last. |
| PM | Displays the contents of the page matrix identifying the owners of all pages. This option assumes that page identifying routines (such as RUN USERS, RUN PROCS) have been run previously. |
| PN | Displays the processor inswap tables (if inswap is in progress). |

Table 9. DISPLAY Command Options (cont.)

| Option | Meaning |
|---|---|
| PP, pageno[, loc1, loc2] | Displays the contents of the indicated physical page. Loc1 and loc2 are relative page offsets expressed in decimal (0-512). If they are specified, only the portion of the page in the indicated range is displayed. |
| RA[T] | Displays the resource allocation tables. |
| RB[T] | Displays the remote batch tables (if there are any). |
| RE[GISTERS] | Displays the software check code, software check message, and the first two register blocks at the time of the crash. |
| RC[XT] | Displays the area of memory occupied by the recovery routines. |
| RQ | Displays the resource subqueue lists. |
| ST[ABLE] | Displays the output symbiont tables. |
| SW[APPER] | Displays the contents of the swap/swap scheduling tables. |
| SY[MBIONT] | Displays the contents of the RBBAT recovery file. |
| TP | Displays the transaction processing tables (if there are any). |
| TR[APS] | Displays the contents of the trap and interrupt locations. |
| TS[TACK][, id] | Dumps the temp stack of the user indicated by id. If no id is specified, the monitor's temp stack is dumped. |
| US[ER][,id]... | Displays the user tables of the specified users. Up to 20 users may be specified. |
| VP, pageno[, loc1, loc2] | Displays the contents of the specified virtual page. Loc1 and loc2 are relative page offsets expressed in decimal (0-512). If they are specified, only the portion of the page in the indicated range is displayed. |
| WHY | Displays the software check code and the software check message. |

Table 10. RUN Command Options

| Option | Meaning |
|---|---|
| MO[NITOR] | Specifies monitor pages. |
| PR[OCESSOR][, {S / name}] | Specifies processor pages or specific processor. The default is S, indicating all processor pages. |
| RT | Specifies real-time page chains. |
| ST[ATE][, {S / q#}] | Specifies state queues. The number of a specific state queue may be specified (q#), or S indicates all. The default is S. |
| US[ER][, {S / id[, id]...}] | Specifies user pages for all users (S), or for particular users (id). Up to 20 users may be specified. The default is S. |
| XD[ELTA] | Specifies XDELTA's page chains. |

**$loc_1$,$loc_2$** This command outputs the contents of the memory locations between $loc_1$ and $loc_2$. The format of the command is

$loc_1, loc_2$

where $loc_i$ is a hexadecimal number or an expression indicating a sum or difference of two hexadecimal numbers.

Two levels of $loc_i$ commands may be joined by the +, -, *, and % (division) operators. For example, the following are permissible:

$loc + loc_1, loc_2$

$loc - loc_1, loc_2$

$loc_1, loc_2 + loc$

$loc_1, loc_2 - loc$

$loc_1 + loc_2, loc_3 - loc_4$

$loc_1 * loc_2, loc_3 \% loc_4$

The resultant dump suppresses identical lines and an * is inserted next to the line number following the identical line encountered. An EBCDIC translation is included to the right of the dump.

**LINE FEED** The line feed (or carriage return) character may be used in conjunction with loc and $loc_1$, $loc_2$ commands to dump the contents of the next location.

↑ This command may be used in conjunction with the loc and $loc_1$, $loc_2$ commands to dump the last location. The format of the command is

↑

* This command may be used in conjunction with the loc and $loc_1$, $loc_2$ commands to dump the location whose address is contained in the location specified by loc. The format of the command is

*

**MONITOR** The MONITOR command turns the monitor display mode on and off (as does any explicit command). When the display mode is on, the current monitor is displayed. When the display mode is off, the dump is displayed. The format of the command is

MO[NITOR] [DI[SPLAY]]

where DISPLAY turns the monitor display mode on. Omission of DISPLAY turns the monitor display mode off.

**loc = value** This command places the specified value into the specified location (loc) of the running monitor. (The display mode must be on.) The format of the command is

loc = value

where

loc is the specified location.

value is the specified value.

## MAP COMMANDS

These commands turn the map mode on and off. They work only with interactive commands and apply only to a particular user. The two map commands are

MAP

UNMAP

**MAP** The MAP command loads the map of the specified user if his JIT is in core. The format of the command is

MA[P], id

where id is the user identification assigned by the system. Dump output following a MAP command is assumed to be virtual addressed.

**UNMAP** The UNMAP command turns the mapping mode of operation off. The format of the command is

UN[MAP]

Dump output following an UNMAP command is assumed to be physical addressed.

## SEARCH COMMANDS

Commands in this group allow core to be searched. The commands are

COMPARE

SMASK

SEARCH

**COMPARE** The COMPARE command compares dump locations between $loc_1$ and $loc_2$ with the running monitor, and outputs locations with nonequal contents. The format of the command is

CO[MPARE],$loc_1$,$loc_2$

**SMASK** The SMASK command sets the mask to the specified value. The format of the command is

SM[ASK],value

where value is a hexadecimal mask.

**SEARCH** The SEARCH command searches for and outputs all words between locations $loc_1$ and $loc_2$ that contain the specified value under the mask. The format of the command is

SE[ARCH],value,$loc_1$,$loc_2$

where

value is a hexadecimal value.

$loc_1$ is the beginning location and may be a hexadecimal number or an expression indicating a sum or difference of two hexadecimal numbers.

$loc_2$ is the ending location and may be a hexadecimal number or an expression indicating a sum or difference of two hexadecimal numbers.

## OUTPUT COMMANDS

Commands in this group direct or format the output of ANLZ. Four output commands are provided:

ROWS

LP

UC

PRINT

**ROWS** The ROWS command establishes the width of dump output. The format of the command is

ROWS value

where value is a number between 1 and 12. ROWS 1 would cause all hexadecimal dumps to be one word wide; ROWS 8 would cause the dumps to be eight words wide. (Platen width may need to be extended at ROWS = 8.)

**LP** The LP command directs output from ANLZ to the line printer. The format of the command is

LP [rows]

where rows indicates the dump width in number of words.

**UC** The UC command directs output from ANLZ to the on-line terminal. The format of the command is

UC [rows]

where rows indicates the dump width in number of words.

**PRINT** The PRINT command closes the output symbiont file to allow output to the line printer without requiring a return to TEL. The format of the command is

PR[INT]

## DEBUG COMMANDS

Commands in this group permit the use of Delta to facilitate monitor debugging. The three debug commands are

BF

DELTA

NODELTA

**BF** The BF command specifies the name of the boot file that represents the monitor being examined by ANLZ. This enables the debugger Delta to read in the required symbol tables. If the BF command is not specified, the file M:MON in :SYS is the boot file that is assumed by default.

The form of the command is

BF fid

where fid is the file identification and is in the form

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

**DELTA** The Delta command associates the debugger Delta with ANLZ and gives control to Delta. If the BF command has been issued, the Delta command ;S loads the global symbol table of the monitor root from the specified boot file. The Delta command name ;S loads the local symbol table of the module named. If the BF command was not executed, the file M:MON in :SYS is used to obtain the monitor symbol tables and the Delta commands apply to the running monitor being examined, not to the monitor in the boot file. The Delta command ;G is used to exit from Delta and to return control to ANLZ.

The form of the DELTA command is

DE[LTA]

**NODELTA** The NODELTA command disassociates the debugger Delta from ANLZ. The form of the command is

NO[DELTA]

### MISCELLANEOUS COMMANDS

**SYMBOLS** This command creates an alphanumerically sorted monitor map by reading, sorting, and formatting the monitor's REF/DEF stack in the file MONSTK.:SYS.

The form of the command is

SY[MBOLS] [fid]

where fid is used to select symbols from a file and has the format

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

MONSTK.:SYS is the default.

**IS** This command reads the sorted symbol table that was saved the last time ANLZ ran as a ghost job. The command produces no output. When the IS command is used, the SYMBOLS command is unnecessary. The format of the IS command is:

IS

**SYMBOL/** The symbol/ command displays the contents of a monitor location. The format of the command is

symbol/

where symbol specifies the name of a location in the monitor.

Note: The symbol table must have been retrieved by use of the SYMBOLS or IS command prior to use of this command.

**DUMP** This command causes a specified range of addresses to be dumped. The command's format is

DUMP $loc_1$, $loc_2$

Dump output following a MAP command is assumed to be virtual addressed; after an UNMAP command, physical addressed.

**CLOSE** This command causes the input dump file to be closed. The format is

CL[OSE]

A user should close a file prior to entering the monitor display mode.

**HELP** This command lists all ANLZ commands and options, and gives a brief description of the purpose of each. The form of the HELP command is

HE[LP]

### EXIT COMMANDS

**END** The END command causes an exit from ANLZ. The format of the command is

EN[D]

## OUTPUT

The output produced by ANLZ consists of displays of formatted monitor and user tables and the contents of registers existing at the time of the crash. The time and date information in the output page headings refer to the time at which the crash occurred.

Some of the output tables are chain type displays. That is, they are formed by starting at the head of a chained list and outputting that list until the tail of the chain is reached. If the tail and the last page in the chain do not agree, the following message is output:

| TAIL ERROR |
|---|

If the count differs from the number of pages in the chain, the following message is output:

| COUNT ERROR |
|---|

Table 11 lists all of the ANLZ displays in order of appearance in the ANLZ dump. The left-hand column specifies the heading that appears at the top of each display. The right-hand column describes the contents of the display.

Table 11. Displays

| Heading | Contents |
|---|---|
| REGISTERS: | The contents of the registers at the time the dump was taken. |
| TRAPS/INTERRUPTS: | The output for trap and interrupt locations. The trap and interrupt locations are those used by the associated XPSD instructions and are listed in Table 12. |
| PAGE IN WHICH TRAP OCCURRED: | The core page in which the trap occurred, if a trap was the cause of the recovery. |

Table 11. Displays (cont.)

| Heading | Contents |
|---|---|
| USER TABLES: | The user tables. This display includes the tables associated with each user that has a page chain. The meaning and source of items in this display are defined in Table 13. |
| ADDITIONAL USER TABLES: | The remainder of the User Tables display above. The meaning and source of items in this display are defined in Table 14. |
| USER STATE CHAINS: | The user state chains which indicate the state of each user in the system. |
| RESOURCE WAIT QUEUES: | The queues of users waiting for resources. The queues are listed and defined in Table 15. |
| SWAP TABLES: | The swap tables. The meaning of each location in the table is defined in Table 16. |
| PARTITION TABLES: | The partition tables. Table 17 defines the headings in this display. |
| PROCESSOR TABLES: | The processor tables. Table 18 defines the headings in the display. |
| MONITOR (FREE) PAGE CHAIN: | The monitor free page chain. The swapper page chain is formatted in the same manner as this display. Usually there is no page chain data output. |
| USER PAGE CHAINS: | The user page chain display. This display indicates which pages and how many pages were being used by the various users resident in core. |
| PROCESSOR PAGE CHAINS: | The processor page chain display. This display indicates which pages and how many pages were being used by the various processors resident in core. |
| READ AHEAD TABLES: | The read-ahead tables. |
| REAL TIME PAGES: | The real-time page chain. |
| XDELTA/HANDLER PAGE CHAINS: | XDELTA's page chain. |
| PHYSICAL MEMORY ALLOCATION: | The actual physical memory allocation on a page-by-page basis. This display is a composite picture of the monitor free page chain, user page chain, and processor page chain displays, plus the resident monitor and its JIT, plus any unallocated pages. |
| ALLYCAT TABLES: | The ALLYCAT buffer adjustment tables. The headings used in this display are defined in Table 19. |
| UNALLOCATED PAGES: | The contents of any unallocated pages. |
| I/O CHANNEL DEVICE STATES: | The I/O channel and device states. The display is separated into tables pertaining to each logical channel. For each channel, ANLZ prints the channel information table (CIT), the device control tables (DCT) for devices on the channel, and the user I/O request queues on those devices. Table 20 defines the headings used in the display. |
| FREE QUEUE ENTRIES: | The free queues entries which are used to contain user I/O requests for I/O devices defined in the I/O Channel Device States display above. |
| CHANNEL INFORMATION TABLE: | The channel information tables (CIT). |
| DEVICE CONTROL TABLES: | The device control tables (DCT). Table 21 defines the meaning of the headings used in this display. |

Table 11. Displays (cont.)

| Heading | Contents |
|---|---|
| IOQ TABLES: | The IOQ tables. Table 22 defines the meaning for the headings used in the IOQ tables display. |
| COC TABLES: | The COC tables. This display includes the line table values for those lines having an associated user (determined by a non-zero value in LB:UN). Table 23 defines the headings used in the COC tables display. |
| RESOURCE ALLOCATION TABLES: | The resource allocation tables. |
| AVR TABLES: | The AVR tables. Table 24 defines the headings used in the AVR tables display. |
| IN CORE ERROR LOG DATA: | The contents of the incore error log buffers. |
| OUTPUT SYMBIONT TABLES: | The output symbiont tables. The headings used in this display are defined in Table 25. |
| *** ASSIGNED CPOOLS:<br>:<br>:<br>*** AND THEIR SPOOLS: | The contents of the assigned CPOOLs and corresponding SPOOLs. |
| MONITOR JIT: | The monitor JIT contents and the monitor TSTACK contents. TSTACK headings are defined in Table 26. |
| CURRENT USER: | The current user's JIT. |
| CONTENTS OF TSTACK: | The current user's TSTACK. TSTACK headings are defined in Table 26. |
| ADDITIONAL JIT FOR USER# nn: | The current user's AJIT (additional JIT). |
| CONTEXT AREA FOR USER# nn: | The current user's context area. |
| *** PHYSICAL PAGE# nn: | The current user's physical pages. |
| MONITOR ROOT: | The monitor root. |
| RBBAT RECOVERY FILE: | The RBBAT recovery file, which includes ghost communication buffers, the RBBAT environment, the RBBAT static data, and the RBBAT dynamic data. (Usually there is no dynamic data output.) |
| USER IDENTIFICATION: | The user identification. This display is a composite of all JITs in the MONDMP file. |
| PATCH FILE: | The patch file built by GHOST1 at system boot time. |
| INSWAP USER: | The current inswap and outswap users' core (if any). This figure has the same format as the Incore Users display. |
| INCORE USERS: | The current incore users' core. |
| CONTROL SECTION MAP: | A map of the monitor modules' start addresses. |
| SYMBOL MAP: | The symbol map. |
| TABLE OF CONTENTS: | The Table of Contents for the ANLZ dump. |

Table 12. Trap and Interrupt Locations for XPSD Instructions

| Location of XPSD | Meaning | Name of Handler |
|---|---|---|
| X'40' | Nonallowed operation trap | NOPPSD |
| X'41' | Unimplemented instruction trap | UNIMP |
| X'42' | Stack overflow trap | STKOVF |
| X'43' | Fixed-point arithmetic overflow | FIXOV |
| X'44' | Floating-point fault | FLTFLT |
| X'45' | Decimal arithmetic fault | DECFLT |
| X'46' | Watchdog timer runout | CSE$ERR |
| X'47' | Multiprocessing usage | IPT47 |
| X'48' | CAL1 instruction | CAL1PSD |
| X'49' | CAL2 instruction | CAL2PSD |
| X'4A' | CAL3 instruction | CAL3PSD |
| X'4B' | CAL4 instruction | CAL4PSD |
| X'4C' | Hardware error trap | CSE$ERR |
| X'4D' | Instruction exception trap | CSE$ERR |
| X'4E' | XDELTA entry | LEE20 |
| X'4F' | JIT pointer | - |
| X'50' | Power on | PONPSD |
| X'51' | Power off | POFPSD |
| X'54' | CLOCK3 counter | - |
| X'55' | CLOCK4 counter | - |
| X'56' | Parity error | PERPSD |
| X'58' | Counter 1 zero | CLK1PSD |
| X'59' | Counter 2 zero | CLK2PSD |
| X'5A' | Counter 3 zero | CLK3PSD |
| X'5B' | Counter 4 zero | CLK4PSD |
| X'5C' | Input/output interrupt | IOPSD |
| X'5D' | Control panel | OCPSD |
| X'60' | COC input interrupt | COCIN1 |
| X'61' | COC output interrupt | COCOUT1 |

Table 13. User Table Headings

| Heading | Source | Meaning |
|---|---|---|
| USER | - | Internal user number. |
| ST | UB:US | User's state. |
| BL | UB:BL | Link to previous user in same state. |
| FL | UB:FL | Link to next user in same state. |
| FLG | UH:FLG | User's flags. |
| FLG2 | UH:FLG2 | Exit control bits, miscellaneous control flags. |
| JIT | UB:JIT | Physical page address of user's JIT. |
| SWPI | UB:SWAPI | Swap table index. |
| HJIT | UH:JIT | Track/sector address on the swapping RAD of user's JIT. |
| AJIT | UH:AJIT | Track/sector address of user's additional JIT. |
| PCT | UB:PCT | User's page count. |
| ACP | UB:ACP | Number of associated command processor. |
| APR | UB:APR | Number of associated processor's root. |
| APO | UB:APO | Number of associated processor's overlay. |
| ASP | UB:ASP | Number of associated special processor. |
| DB | UB:DB | Number of associated debugger. |
| OV | UB:OV | Number of associated overlay. |
| MF | UB:MF | Number of I/O events outstanding. |

Table 14. Additional User Table Headings

| Heading | Meaning |
|---|---|
| USER | User number. |
| MISC | Either time left for user to remain asleep or resource wait queue forward link. |
| UH:DL | DO-list address. |
| CYL | User's procedure cylinder number if disk pack swapper. |
| PRI | User's current priority. |
| PRIB | User's priority base value. |
| NECB | Number of ECBs to be posted for this user. |
| UH:NL | Pointer to head of ECBs to be posted. |

Table 15. Resource Wait Queues

| Name | Description |
|---|---|
| R:SYMF | Users queued for symbiont file space. |
| R:SYMD | Users queued for symbiont disk granule. |
| R:OCR | Users queued for OPEN/CLOSE. |
| R:DPA | Users queued for swapper granule. |
| R:QFAC | Users queued for ALLOCAT. |
| R:NQW | Users queued for ENQ. |

Table 16. Swap Table Terms

| Location | Meaning |
|---|---|
| S:SIR | Swap in requests posted. |
| S:HIR | High priority requests posted. |
| S:SIP | Swap-in progress flag. |
| #SWAP$DEV | Interrupt bypass count. |
| S:CUN | Current user number. |
| S:ISUN | In-swap user number. |

Table 16. Swap Table Terms (cont.)

| Location | Meaning |
|---|---|
| S:CUIS | Count of users in system. |
| S:IDLF | Idle flag. |
| SB:OSN | Number of out-swap users. |
| SB:OSUL | Out-swap user list. |
| S:BECL | Beginning and end command list for each outswap user. |
| SB:NP | Number of in-swap processors. |
| SB:PNL | In-swap processor numbers. |
| SB:FPN | Number of freed processors. |
| SB:FPL | List of freed processors. |
| M:SWAPD | Address of swap device. |
| MB:SDI | DCT index. |
| MB:SFC | Swap function code. |
| MB:#RTRY | Retry count. |
| M:CLBGN | Beginning of current command list. |
| MH:CLEND | End of current command list. |

Table 17. Partition Tables Headings

| Heading | Source | Meaning |
|---|---|---|
| # | Calculated | Index to partition tables. |
| ACCOUNT | PLD:ACT | Current running account. |
| USR | PLB:USR | Number of users in partition. |
| FLG | PLH:FLG | Partition control flags. |
| QN | PLH:QN | Quantum time of partition. |
| TOL | PLH:TOL | Total jobs run in this partition. |
| CUR | PLH:CUR | Current jobs selected in this partition. |
| TL | PLH:TL | Lower time limit. |
| TU | PLH:TU | Upper time limit. |
| SID | PLH:SID | System ID. |

Table 18. Processor Table Headings

| Heading | Source | Meaning |
|---|---|---|
| p# | - | Processor index number. |
| P:NAME | P:NAME | Processor name. |
| HPP | PB:HPP | Head of processor's physical page chain. |
| TPP | PB:TPP | Tail of processor's physical page chain. |
| PSZ | PB:PSZ | Processor's procedure size in pages. |
| DSZ | PB:DSZ | Processor's initial data size in pages. |
| DCBSZ | PB:DCBSZ | Size in pages of DCB area. |
| PDA | PH:PDA | Disk address of procedure. |
| DDA | PH:DDA | Disk address of data and DCBs. |
| UC | PB:UC | Use count on processor. |
| LNK | PB:LNK | First overlay number for this processor. |
| PVA | PB:PVA | Virtual page address of the processor's procedure. |
| HVA | PB:HVA | First page available to the processor. |
| PC# | PB:PC# | Procedure cylinder number. |
| DC# | PB:DC# | Data cylinder number. |
| SA | P:TCB | Starting address and flags. |
| TCB | P:TCB | TCB address. |

Table 19. ALLYCAT Headings

| Heading | Meaning |
|---|---|
| TOP | Top index into buffer. |
| BOTTOM | Bottom index into buffer. |
| WORDCNT | Number of disk addresses in buffer. |
| TEMPBOT | Set if ALLYCAT changing buffer. |
| BUFLAGS | Bit 0 = HGP empty, Bit 1 = buffer just filled, Bit 2 = buffer just emptied. |
| ADJSTCNT | Number of entries manipulated by ALLYCAT; may be either positive or negative. |
| GRANULES AVAIL | Total number of granules/cylinders remaining in system (in hexadecimal notation). |

Table 20. I/O Table Headings

| Heading | Meaning |
|---|---|
| CIT3-5 | Channel Information Tables 3-5 |
| DEVICE | yyndd for this device |
| ADDR | Hardware address |
| CX | Channel index |
| OIDTS | From DCT3 - Set bits indicate:<br>O output<br>I input<br>D down<br>T timed out<br>S SIO reject |
| BPWXKCSB | From DCT5 - Sets bits indicate:<br>B Device busy<br>P Clean-up pending<br>W Wait until done<br>X Data transfer<br>K Wait for key-in<br>C Control task<br>S SIO while manual<br>B BIN mode |
| QX | I/O queue index |
| AIO | Last AIO status |
| TDV | Last TDV status |

Table 21. Device Control Table Headings

| Heading | Meaning |
|---|---|
| # | DCT number. |
| DEV | Active I/O address. |
| PRI | Primary I/O address. |
| ALT | Alternate I/O address. |
| CIT # | Channel (CIT) index. |

Table 21. Device Control Table Headings (cont.)

| Heading | Meaning |
|---|---|
| IO FLG | I/O legality:<br>11 = in and out<br>10 = out only<br>01 = in only |
| DEV TYP | Type mnemonic. |
| DEV FLGS | State of device. |
| IO Q# | IOQ index. |
| CDW ADRS | Command doubleword address (WA resolution). |
| PRE HAND | Handler preprocessor word address. |
| POST HAND | Handler postprocessor word address. |
| ACT CNTR | Device activity counter. |
| IO INT DEADLINE | Value to match against I/O clock. |
| AIO INT STAT | AIO status word. |
| TDV STATUS | TDV status doubleword. |
| CHAN FLNK | Link to next entry. |
| PRE-EMPT | Real-time pre-empt flag. |
| IS | 7446 table. |
| HAND CODES | Handler function flags (first 8 bits contain retry function code; the second 8 bits contain the follow on code). |
| TIME INCR | Time-out increments. |
| SIO CC | SIO condition codes. |
| TDV CC | TDV condition codes. |
| TIO STATUS | TIO status. |
| DISC FLAG | Disk flag. |
| HGP DISP | Heading Granule Pool (HGP) displacement if disk. |
| RMA FLGS | Partitioning flags. |
| SIO COUNTER | Number of SIOs done to this device. |

Table 22. IOQ Table Headings

| Heading | Meaning |
|---|---|
| # | IOQ table number. |
| BAK | Back link to next entry. |
| FWD | Forward link to next entry. |
| DCT# | DCT index. |
| MNE | TEXT name of device from SYSGEN. |
| STAT | Software status. |
| FCN | Original function code (IOQ4). |
| CODS | Current function code (IOQ5). |
| DCBAD | DCB word address (if any). |
| BUF | Buffer word address if bit 0 and 1 reset; CDW word address if bit 1 set (swapper); CDW word address if bit 0 set (other). |

Table 22. IOQ Table Headings (cont.)

| Heading | Meaning |
|---|---|
| TIM | Number of timeout increments. |
| CDW | Number of commands used if IOQ8 bit 0 or 1 set. |
| NRA | Original number of recovery tries. |
| NRT | Remaining number of recovery tries. |
| RAD AD | Disk address. |
| E A ADR | End action word address. |
| E A INFO | One word to return to end action receiver. |
| PRIO | Priority of this event. |
| USER | User number of I/O requester. |

Table 23. COC Line Table Headings

| Heading | Source | Meaning |
|---|---|---|
| LINE | Calculated | Line number. |
| USER | LB:UN | Associated internal user. |
| TYPE | COCTERM | Terminal type. |
| EOMTIME | EOMTIME | End of message time for a read. |
| BUFCNT | BUFCNT | Number of buffers in use for line. |
| CPOS | CPOS | Current carriage position. |
| RSZ | RSZ | Record size requested by user while read is pending. |
| MODE BYTES | MODE-MODE4 | Terminal mode indicators. |
| TL | TL | Pointer to tab buffer. |
| II | COCII | Input insertion pointer for line. |
| IR | COCIR | Input removal pointer for line. |
| ARSZ | ARSZ | Accumulated record size while read is pending. |
| CPI | CPI | Initial carriage position for a read. |
| OI | COCOI | Output insertion pointer for line. |
| OR | COCOR | Output removal pointer for line. |
| OC | COCOC | Count of characters pending output. |

Table 24. AVR Table Headings

| Heading | Meaning |
|---|---|
| SER# | Serial number of tape or pack. |
| PUB | Set if public. |
| POS | Set if positioned. |
| AVR | Set if AVRed. |
| SCR | Set if scratch tape. |
| HLD | Set if held. |
| PTL | Set if positioned to label. |
| UPL | Set if user positioned label. |
| OPN | Set if open. |
| NOU | Number of users. |
| TPOS | Tape mark count. |
| USER | User number. |
| SOLICIT | Index to special AVR tables. |
| INI | Set if volume initialized. |
| VER | Set if volume verified. |
| MTD | Set if mounted. |
| PRIM | Set if primary volume of private set of volumes. |
| HGPDISP | Displacement from HGP. |

Table 25. Symbiont Table Headings

| Heading | Meaning |
|---|---|
| # | Index number of table. |
| SQUE | Symbiont queue chain. |
| SNDDX | DCT index of symbiont device. |
| TYPE | TEXT name of symbiont device from SYSGEN. |

Table 25. Symbiont Table Headings (cont.)

| Heading | Meaning |
|---|---|
| SSTAT | Symbiont Status:<br>0 = input symbiont<br>1 = output symbiont |
| SSIG | Symbiont signal character (e.g., L, Q, etc.). |
| SRET | Symbiont return when activated from chain. |
| SCNTXT | Context block doubleword address displacement. |
| SYMX | Symbiont index:<br>1 = input<br>2 = output |
| TYP | Device type. |
| LNK | Remote chain. |
| FLAG | Remote flags. |
| SUSP | Suspend bit for IRBT. |
| QUE | IOQ index for IRBT. |
| SQHD | Symbiont queue chain head. |
| SQTL | Symbiont queue chain tail. |

Table 26. TSTACK Headings

| Heading | Meaning |
|---|---|
| ADDRS | Virtual address of displayed contents. |
| STACK OFFSET | Index into stack. |
| CONTENTS | Contents of stack. |
| RELATIVE LOC | Address that stack contents point to, in symbol plus displacement form. If the stack cell contains a relative location, the instruction at that location will be displayed if it is an address modifying instruction (e.g., B, BAL, LPSD). |
| INSTRUCTION | Symbolic instruction at the address contained in the stack position. |

## ANLZ MESSAGES

Table 27 contains the messages that are output by ANLZ. Most of these messages identify error conditions. Others merely supply information.

## ANLZ COMMAND SUMMARY

Table 28 summarizes ANLZ commands. The left-hand column contains the command format, the right-hand column contains the command description.

Table 27. ANLZ Messages

| Message | Description |
|---|---|
| ANLZ HERE | The ANLZ program has begun operation. |
| ANLZ: ENTER COMMAND, N/L SAYS TO DO ALL | This message issued to operator after GJOB ANLZ key-in. Operator may respond with one of the following:<br><br>NO = just exit<br>TA = read recovery-built tape<br>HE = run interactively from console<br>CP = read CP5DMP file<br>0-7 = read indicated MONDMP file<br>N/L = do default ghost run |
| ANLZ GHOST FINISHED | The ANLZ ghost has completed processing the core image file. |
| ANLZ USING MONDMPn | ANLZ has been commanded to read a MONDMP file. The value specified for n indicates the number of the MONDMP file. |
| BAD COMMAND | The command was unrecognizable. |
| CANNOT OPEN FILE name | The file specified by the INPUT command cannot be opened. |
| CAN'T GET THE BUFFER | The user was not allowed enough core in his account to read in the monitor symbol stack. |
| COUNT ERROR<br>TAIL ERROR | The tail and last page in a chain do not agree. |
| ENTER TAPE TYPE: 7T, 9T, BT, ETC. . . | The user must supply the tape type if tape input is to be used. |
| ERR/ABN CODE = xxxx**dcb | An I/O error or abnormal condition occurred during an INPUT operation.<br><br>xxxx is the error or abnormal code.<br><br>dcb is the address of the DCB associated with it. |
| LOC1 > LOC2 | The first location entered for a $loc_1, loc_2$ (or similar) command was greater than the second location. |
| xx PRIVILEGE LEVEL NOT HIGH ENOUGH | The user privilege level was not high enough for the requested operation. |
| SORRY, NO PAGE xx | The page containing the location specified by the user was not found in the input file. |
| THE LAST PHYSICAL PAGE IN THE FILE IS xx | The size of the file read from tape by the INPUT command is specified by the last physical page in the file. |

Table 28. ANLZ Command Summary

| Command | Description |
|---|---|
| ↑ | Dumps the last location and is used in connection with loc and $loc_1$, $loc_2$. |
| * | Dumps the indirect location and is used in conjunction with loc and $loc_1$, $loc_2$. |
| AL[L] | Performs the functions of the INPUT, DISPLAY, and RUN commands and of ANLZ (except dumps) when initiated by the automatic recovery procedure. A numerically and alphanumerically sorted monitor map is output at the end of the ALL display. |
| BF fid | Specifies the name of the boot file that represents the monitor being examined by ANLZ. The file M:MON in :SYS is assumed by default. |
| CL[OSE] | Causes input dump file to be closed. |
| CO[MPARE],$loc_1$,$loc_2$ | Compares the dump (locations $loc_1$ through $loc_2$) with the running monitor and outputs the locations with nonequal contents. |
| DE[LTA] | Associates the debugger Delta with ANLZ. |
| DI[SPLAY] option | Outputs information existing at the time of the crash. The options are<br>AJ[ITS] –JIT, AJIT, and context area of all incore users.<br>AT[ABLES] –incore portion of ALLYCAT's tables.<br>AV[R] –tape and disk tables.<br>CI[TS][, index]... –all or requested entries of Channel Information Tables.<br>CO[C][,index]... –all or requested entries of COC tables.<br>CU[N] –current user's JIT, AJIT, and context area.<br>DC[T][,index]... –all or requested entries of Device Control Tables.<br>EL[OG] –incore error log buffers.<br>FQ–Free I/O Queueing tables.<br>IO[, chan]... –devices on requested I/O channels.<br>IQ[, index]... –all or requested entries of I/O queueing tables.<br>JIT[, id][, loc1, loc2] –contents of the JIT for the user specified by id or for the monitor.<br>MR[, loc1, loc2] –monitor's root.<br>OJ[IT] –all of the out of core JITs. |

Table 28. ANLZ Command Summary (cont.)

| Command | Description |
|---|---|
| DI[SPLAY] option (cont.) | OS – user outswap tables.<br>PA[RTITIONS][, index]... –all or requested entries of partition tables.<br>PF[ILE] –patch file that was created last.<br>PM–page matrix identifying the owners of all pages.<br>PN–processor inswap tables.<br>PP, pageno[, loc1, loc2] –contents of the indicated physical page.<br>RA[T] –resource allocation tables.<br>RE[GISTERS] –software check code, software check message, and the first two register blocks.<br>RB[T] –remote batch tables.<br>RC[XT] –area of memory occupied by the recovery routines.<br>RQ –resource subqueue lists.<br>ST[ABLE] –output symbiont tables.<br>SW[APPER] –contents of the swap/swap scheduling tables.<br>SY[MBIONT] –RBBAT recovery file.<br>TP –transaction processing tables.<br>TR[APS] –contents of trap and interrupt locations.<br>TS[TACK][, id] –temp stack of the user specified by id or of the monitor.<br>US[ER][, id]... –user tables of the specified users.<br>VP, pageno[, loc1, loc2] –contents of the specified virtual page.<br>WHY –software check code and software check message. |
| DU[MP] $loc_1$,$loc_2$ | Dumps specified range of addresses. |
| EN[D] | Exits from ANLZ. |
| HE[LP] | Lists all ANLZ commands. |
| IN[PUT] option | Directs ANLZ to input from a particular disk or tape file or to open a file. The options are<br>LA[ST] – opens the last file formed by the recovery procedure. |

Table 28. ANLZ Command Summary (cont.)

| Command | Description |
|---|---|
| IN[PUT] option (cont.) | number – opens the numbered crash file formed by the recovery procedure.<br><br>TA[PE] – reads a labeled tape created by the recovery procedure.<br><br>CP[5DUMP] – opens the CP5DUMP file. |
| IS | Reads the sorted symbol table from a previous ANLZ run. |
| Line Feed (or carriage return) | Dumps the contents of the next location and is used in conjunction with loc and $loc_1$, $loc_2$. |
| loc | Outputs the contents of the specified location. |
| $loc_1$, $loc_2$ | Outputs the contents of memory locations between $loc_1$ and $loc_2$. |
| loc = value | Places the value in the specified location of the running monitor. |
| LP[rows] | Directs the output of ANLZ to the line printer, where rows is dump width in hexadecimal words. Default is full line. |
| MA[P], id | Loads the map of the specified user if his JIT is in core. |
| MO[NITOR][DI[SPLAY]] | Turns the monitor display mode on and off.<br><br>MONITOR turns the display mode off.<br><br>MONITOR DISPLAY turns the display mode on. |
| NO[DELTA] | Disassociates the debugger Delta from ANLZ. |
| PR[INT] | Closes the output symbiont file to allow output to the line printer without requiring a return to TEL. |
| RO[WS], value | Establishes width of dump output in number of words, where value may be 1 through 12. |
| RU[N] option | Outputs various linked lists of the monitor by running through the list and displaying each entry. The options are<br><br>MO[NITOR][, {S / pgno}] – monitor pages. S, the default, indicates all. A specific page may be requested.<br><br>PR[OCESSOR][, {S / name}] – processor pages. S, the default, indicates all. A particular processor may be specified.<br><br>RT – real-time page chains.<br><br>ST[ATE][, {S / q#}] – state queues. A particular queue number may be specified, or S, the default, indicates all. |

Table 28. ANLZ Command Summary (cont.)

| Command | Description |
|---|---|
| RU[N] option (cont.) | US[ER][,{S / id}] – user pages for a particular user (id), or for all users (S). S is the default.<br><br>XD[ELTA] – XDELTA's page chains. |
| SE[ARCH], value,$loc_1$,$loc_2$ | Searches for and outputs all words between $loc_1$ and $loc_2$ that contain the value under the mask. |
| SM[ASK], value | Sets the mask to the specified value. |
| symbol/ | Displays the contents of the monitor location specified by symbol. |
| SY[MBOLS][fid] | Creates a numerically sorted monitor map, using the fid specified or MONSTK. :SYS. |
| UC[rows] | Directs the output of ANLZ to the on-line terminal, where rows is dump width in hexadecimal words. Default is full line. |
| UN[MAP] | Turns off the mapping mode of operation. |

# 5. ERROR MESSAGE FILE

## INTRODUCTION

The error messages for the CP-V monitor and several CP-V processors are contained in an error message file, called ERRMSG. This file is initially created either through punched card or on-line terminal input and is maintained through use of the Edit processor. This chapter describes the structure of the ERRMSG file and the techniques required to create and modify the file.

Codes for detected error conditions are recorded in the job information table (JIT). The error code is placed in J:ABC (high-order byte) and the subcode is placed in ERO (right-justified). When CCI (batch jobs) or TEL (on-line jobs) is entered, a message is printed to correspond to the code and subcode. This message is obtained from the error message file (ERRMSG) via a keyed read using a key constructed from the group code, error code, and subcode. If either the file or the record corresponding to the code is missing, the error code itself will be printed. Otherwise, the message and the error code will be printed.

## FORMAT OF ERROR MESSAGE FILE

Each record in the error message file contains the EBCDIC text of one error message. The key of each record is one word long and has the form

The first byte always contains 03, which is the count of bytes in the key. The second byte is the group code, the third is the error code, and the fourth is the error subcode.

Group codes presently assigned are

| | | | |
|---|---|---|---|
| 0 | Monitor | 5 | CCI |
| 1 | PCL | 6 | DRSP |
| 2 | Loader | 7 | Batch |
| 3 | TEL | 8 | Analyze |
| 4 | Runner | | |

Messages in the file with group codes other than zero are not handled by the monitor itself. Error codes currently assigned within the monitor group are

| | |
|---|---|
| 0 – 7F | I/O error and abnormal codes |
| 80 – 9F | COBOL error codes |
| A0 – BF | Other Monitor codes |
| C0 – FF | Unused |

The meaning of the assigned codes are defined in CP-V/TS Reference Manual, 90 09 07, CP-V/BP Reference Manual, 90 17 64, and in the ANS COBOL/LN Reference Manual, 90 15 00.

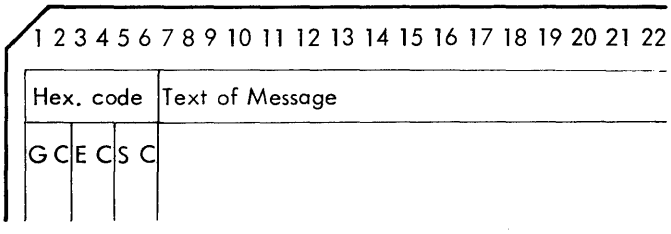
## CREATING ERROR MESSAGE FILE

The ERRMSG file is initially entered into the system either through a card reader or an on-line terminal at the central site. The procedures for each type of input are described below.

### CARD READER INPUT

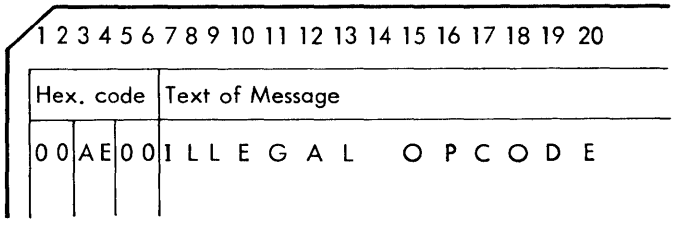
Card input of the error message file is handled by the Error Message File Writer (ERRMWR). This program reads cards, interprets the first six columns as a hexadecimal number, converts this number into a three-byte key, and writes the card image exclusive of trailing blanks as a keyed record in the ERRMSG file in the account under which ERRMWR is executed. This account should be :SYS for the system error message file.

The card format is

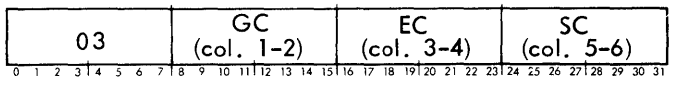



Example:

Assume that the message ILLEGAL OPCODE is to be placed in the error message file for the monitor error code AE. The group code and subcode in this case are both zero. Thus, the card for this message would be punched as follows:




Keys generated by the ERRMWR program have the form




During conversion of the key, leading blanks are treated as zeros. Nonhexadecimal letters result in output of a warning message and cause the card to be ignored. The card image is scanned from right to left to determine the rightmost non-blank character, and the count of characters is adjusted so that trailing blanks are not written. A new line character X'15' is appended to the message.

**Warning:** If an installation modifies the text of TEL error messages, it should be noted that TEL stores dynamic information in the error message buffer and some of

The message may be continued in column 1 of the following card by appending a continuation character (;) at the end of the message in the first card. Only two cards per message are allowed.

A card containing an asterisk in column 1 is a control card and is used to set the format of the record written in the file. If column 2 of the control card contains a 0, the message key is appended to the front of the message text and is included in the record. If column 2 of the control card contains a 1, the key is not included in the record text (this is the default condition). Control cards can be placed anywhere within the data deck except between continuation cards.

## TERMINAL INPUT

Creating or modifying the error message file can be accomplished from the terminal by using Edit or ERRMWR.

Example 1: Using Edit

```
!BUILD MSG (RET)
    1.000 00AB00 THAT'S NO DEBUGGER!(RET)
    2.000 00AB01 THAT'S NO OP CODE(RET)
    3.000(RET)
!SET M:E1 DC/MSG (RET)
!ERRMWR (RET)
```

Example 2: Using ERRMWR

```
!SET M:E1 UC(RET)
!ERRMWR(RET)
>00AC01 DON'T ISSUE CAL3 OR CAL4(RET)
>(RET)
!
```

# 6. SYSTEM ERROR LOG FILE

## INTRODUCTION

All hardware malfunctions and some software problems occurring during system operation, whether recovered or not, are recorded in a special disk storage file. This file is periodically copied into a standard file (ERRFILE) by a ghost program (ERR:FIL) which is initiated automatically for that purpose.

ERRFILE may be listed and summarized by the Error Log Listing processor that is described in this chapter. ERRFILE is also available for on-line preventive maintenance of the system and for diagnosis and prediction of hardware malfunctions.

## ERR:FIL PROGRAM

ERR:FIL copies the special file created by ERRLOG onto a normal keyed file (ERRFILE) in the :SYS account that is more readily available to diagnostic programs.

ERR:FIL is a ghost job that is awakened by ERRLOG whenever five errors have been recorded. ERR:FIL may also be awakened by a program with diagnostic privilege by using the initiate job CAL (CAL1,6 FPT) or by an operator key-in of GJOB ERR:FIL.

## ERROR LOG LISTING PROCESSOR

The Error Log Listing processor (ELLA) provides an efficient tool for listing and sorting the error log file, ERRFILE, which is automatically generated and updated by the CP-V system. (ERRFILE is described in Appendix E.) ELLA output furnishes a meaningful and comprehensive diagnostic evaluation of the system and its peripherals, aiding in the early detection of product failures and thus increasing the reliability, maintainability, and availability of the system.

The set of ELLA commands allows the user to first specify the kinds of errors in which he is interested, and then request a listing of those kinds. Four types of listings are available:

- A chronological listing of error log entries.
- A sorted listing of error log entries.
- A summary of error log entries by category.
- A summary of error log entries in graphic form.

### STARTING EXECUTION

ELLA may be run as an on-line, batch, or ghost job. Normal operating procedures are observed in each of these modes. Batch and on-line operations are illustrated in Examples 1 and 2. These first two examples are intended only for ELLA users who are not familiar with CP-V.

The use of ELLA is restricted to authorized system users whose accounts have a diagnostic privilege level (A0 or higher). If the user has insufficient privilege, ELLA will abort with the message

| INSUFFICIENT PRIVILEGE LEVEL ABORT |
|---|

Note: Initiating ELLA as a ghost job enables the operator to issue ELLA commands from the operator's console. However, judgement should be exercised when initiating ELLA in this fashion since ELLA commands will be intermixed with normal operator console material.

### INPUT/OUTPUT ASSIGNMENTS

ELLA input and output is divided into three separate functions:

- Error log input.
- User command input.
- Listing output.

Error log input is always taken from the system error log file, ERRFILE. Without user intervention, the remaining two functions assume default assignments depending on the mode in which ELLA is run. The default assignments are listed in Tables 29, 30, and 31. (They are based upon the assumption that the SI and LO operational labels were given the standard assignments during SYSGEN.) The assignment of the output listing function may be altered by the user during ELLA execution through use of the ELLA SET command. The tables specify the ELLA SET command formats that are required to make the reassignments. The SET command is described in detail below.

**SET** The SET command reassigns the listing and message output device assignment during execution of ELLA. (It changes the device assignment in the M:LO DCB.) The format of the command is

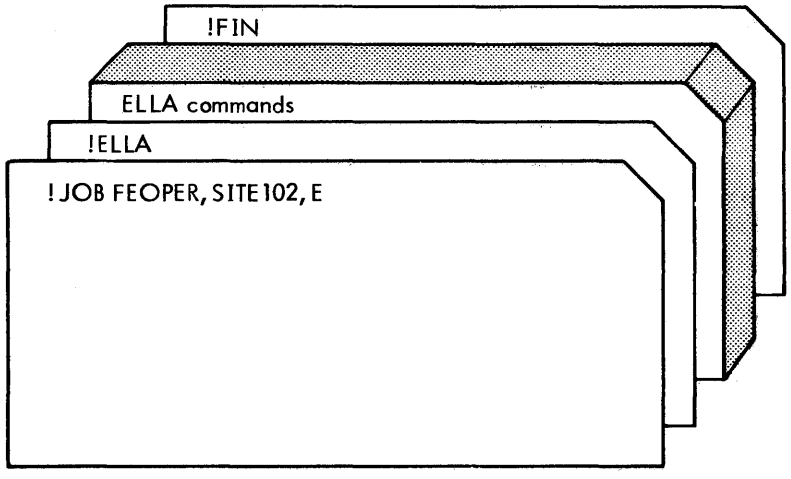
$$\text{SET, LIST, } \left\{ \begin{matrix} \text{LP} \\ \text{KP} \end{matrix} \right\}$$

where

LP specifies line printer.

KP specifies operator's console for the ghost and batch modes and on-line terminal for the on-line mode.

Example 1. Batch Operation of ELLA




For batch operation of ELLA, control commands and ELLA commands are punched on cards and the cards are submitted to the site operator.

In this example, the account number (FEOPER) and account name (SITE102) were chosen because they had been reserved for diagnostic activity at that particular site. In order to run ELLA in the batch mode, the account was authorized a privilege level of A0. (The privilege level is not specified on the JOB card because it is automatically associated with the account.) The execution priority E was specified to given the job a high execution priority. (The privilege level determines the types of things that a job is allowed to do; the execution priority is a determining factor in how quickly a job will be selected for execution.)

Example 2. On-Line Operation of ELLA

```
XEROX CP-V AT YOUR SERVICE
ON AT 13:48 JUL 08, '74
LOGON PLEASE:  FEOPER,SITE102,RSD (RET)

!ELLA (RET)
13:49 JUL 08, '74
ELLA 7080D6-A00
* .
  .
  .
*END (RET)
!PRINT (RET)
  .
  .
  .
!OFF (RET)
```

In this example, the user logged onto the system after receiving the CP-V salutation and log-on request. The account number and name used are the same as in the previous example. (The account was authorized for both batch and on-line operations.) The account has a password associated with it which is to be used for security reasons during on-line operation; i.e., if the password is kept confidential, it prevents unauthorized on-line use of this special diagnostic account. The password is entered following the name and account. Here, the password RSD was entered.

After the log-on, CP-V prompted for input with an exclamation point. The user entered

ELLA ⏎

to request the Error Log Listing program and ELLA responded with its salutation and prompted for input with an asterisk. The user then entered ELLA commands, finishing with the END command which returned control to the system. The system then prompted with an exclamation point. (Actually, control was returned to a system command processor called TEL which is described in detail in the CP-V/TS Reference Manual, 90 09 07.)

ELLA can output its listing on the user's terminal or on the line printer. If printer output is selected, the system holds the output on a disk file until either the PRINT or OFF command is entered. In the example, the user executed the PRINT command which caused the system to produce the printer output. The user then proceeded to perform other tasks, eventually ending the on-line session with the OFF command which logged the user off the system.

## INPUT/OUTPUT CHARACTERISTICS

Whenever ELLA listing output is assigned to the line printer, the output contains two additional types of information: user commands received and diagnostic messages. ELLA user commands are listed on the printer to present a complete record of the user listing session. They are preceded by one asterisk. Diagnostic messages (due to abnormal conditions or operational errors) are preceded by two asterisks.

Whenever the command input function is assigned to the operator's console (ghost initiation of ELLA) or the user's terminal (on-line initiation), diagnostic messages are printed on that input device (preceded by two asterisks) as well as on the line printer.

Table 29. ELLA On-Line I/O Functions

| Function | Associated DCB | Default Assignment | Possible Assignments | ELLA SET Reassignment Command | Comments |
|---|---|---|---|---|---|
| Source error log | M:BI | ERRFILE | ERRFILE | (none) | Data base from which ELLA reads source records for printing. |
| Command input | M:SI | User's terminal | User's terminal | (none) | Device from which ELLA reads commands (and to which it prints diagnostic messages). |
| List output | M:LO | User's terminal | User's terminal | SET, LIST, KP | Device to which ELLA lists error log data. |
| | | | Line printer | SET, LIST, LP | Device to which ELLA lists error log data, commands received, and diagnostic messages. |

Table 30. ELLA Batch I/O Functions

| Function | Associated DCB | Default Assignment | Possible Assignments | ELLA SET Reassignment Command | Comments |
|---|---|---|---|---|---|
| Source error log | M:BI | ERRFILE | ERRFILE | (none) | Data base from which ELLA reads source records for printing. |
| Command input | M:SI | Card reader | Card reader | (none) | Device from which ELLA reads commands. |
| List output | M:LO | Line printer | Line printer | SET, LIST, LP | Device to which ELLA lists error log data, commands received, and diagnostic messages. |
| | | | Operator's console | SET, LIST, KP | Device to which ELLA lists error log data and diagnostic messages. (Using the operator's console for lengthy messages is not recommended.) |

Table 31. ELLA Ghost I/O Functions

| Function | Associated DCB | Default Assignment | Possible Assignments | ELLA SET Reassignment Commands | Comments |
|---|---|---|---|---|---|
| Source error log | M:BI | ERRFILE | ERRFILE | (none) | Data base from which ELLA reads source records for printing. |
| Command input | M:SI | Operator's console | Operator's console | (none) | Device from which ELLA reads commands (and to which it prints diagnostic messages.) |
| List output | M:LO | Line printer | Line printer | SET, LIST, LP | Device to which ELLA lists error log data, commands received, and diagnostic messages. |
| | | | Operator's console | SET, LIST, KP | Device to which ELLA lists error log data and diagnostic messages. (Using the operator's console for lengthy output is not recommended.) |

## INTERRUPTING ELLA EXECUTION

On-line ELLA execution may be interrupted at any time by use of the BREAK key on the user's terminal. This causes ELLA to terminate its current activity and to prompt for a new command.

When ELLA is initiated as a ghost job or a batch job, execution may be interrupted through use of the operator INT key-in. The effect upon a ghost job is similar to that of the BREAK function on-line. The effect upon a batch job is to cause the next command to be read from the card reader.

## ELLA COMMANDS

ELLA accepts three types of commands: boundary commands, task commands, and the device assignment command (SET, described previously). Boundary commands establish or change the limits that are to be applied to all subsequent task commans; i.e., boundary commands allow the user to specify the types of errors in which he is interested. Task commands initiate the execution of a particular type of listing. The device assignment command is used to change the listing and message output device during execution of ELLA.

### TASK COMMANDS

Task commands are used to request the ELLA displays and to terminate ELLA. ELLA task commands are:

CLIS produces a chronological listing of qualified error log entries.

SLIS produces a sorted listing of qualified error log entries.

SUM produces a categorized summary of qualified error log entries.

DISP produces a summary of qualified error log entries in graphic form.

END terminates ELLA.

Note that error log entries are displayed only if they qualify. To qualify for inclusion in a display, an error log entry must pass all boundary tests in force at the time the display is generated. If no boundary commands have been entered, all error log entries qualify. Those error log entries which fail to pass one or more of the boundary tests are ignored. (Boundary commands are described following the task commands.)

**CLIS** The CLIS command requests a chronological listing of the error entries in the order in which they appear in the error file.

The format of the CLIS command is

C[LIS]

An example of a CLIS listing is given in Example 3. Table 32 lists the error log entry headings printed by ELLA and notes the manner in which all values are printed.

Example 3. Use of the CLIS Command

In this example, the user chose to initiate ELLA on-line. The user did not desire a lengthy listing at his terminal. Therefore he reassigned the listing function to the line printer using the SET command.

```
!ELLA (RET)
*SET,LIST,LP (RET)
*CLIS (RET)
*END (RET)
!PRINT (RET)
```

After the CLIS command was issued, ELLA produced the chronological listing and then prompted for another command. The user desired no further listings, so he terminated ELLA with the END command. He then issued the system PRINT command which caused the listing to be output to the printer. The output that was sent to the line printer is shown below:

```
*CLIS


C H R O N O L O G I C A L  L I S T I N G
- - - - - - - - - - - - - - - - - - - - -


FROM   00/00/00   00:00:00:000
TO     12/31/99   23:59:59:999


*** SYSTEM IDENTIFICATION ***
                CORE                                    -OPTIONS--   TIME
TIME            (K)     SITE I.D.   SYSTEM     CPU     SYMB RT RB   RES
11:36:00:000    00128   PRT101      CP-V C00   S67      Y   N   N    02


*** CONFIGURATION ***
                        I/O    ADRS   DCT
TIME            MDL     PRIM   ALTN   INDEX
11:36:00:000    7012    0001   0001    01
                7140    0003   0003    02
                7160    0004   0004    03
                7445    0002   0002    04
                7212    01F0   01F0    05
11:36:00:000    7322    0080   0080    06
                7322    0081   0081    07
                7271    00E0   00E0    08
                7271    00E1   00E1    09
                7271    00E2   00E2    0A
11:36:00:000    7271    00E3   00E3    0B
                7611    0010   0010    0C


*** TIME STAMP ***  DATE=07/10/74  TIME=12:00:00:004


*** SIO FAILURE ***
                        I/O    ---SIO-   ---TDV-   SUBC   TDV CUR   REM
TIME            MDL     ADRS   STAT CC   STAT CC   STAT   COMM DA   BYTES   MFI
12:36:30:782    7323    0083   2000 6    1000 6    00     0011B7    0001    00
12:37:29:518    7323    0083   2000 6    1000 6    00     0011B7    0001    00
12:40:10:398    7323    0083   2000 6    1000 6    00     0011B7    0001    00
```

```
*** TIME STAMP ***   DATE=07/10/74   TIME=13:00:00:005
*** TIME STAMP ***   DATE=07/10/74   TIME=14:00:00:003


***SYMBIONT INCONSISTENCY***
                DCT     REL.    SYMB.
TIME            INDEX   SECT.   DCT
14:03:13:648    09      0110    02


*** TIME STAMP ***   DATE=07/10/74   TIME=15:00:00:004
*** TIME STAMP ***   DATE=07/10/74   TIME=16:00:00:006
```

Note that the CLIS command is listed in the line printer listing and that the existing time boundaries are printed after the title. If other boundaries were in force, they too would have appeared. Certain values, such as core size and recovery count, are printed in decimal for convenience. Other fields, such as the OPTIONS field, contain flags. The true condition is represented by the letter Y, the false condition by the letter N. In this example, the system has symbiont capability but does not have remote processing and real-time facilities.

Table 32. Error Log Entry Headings

| Heading | Description |
|---|---|
| ACCOUNT<br>eeeeeeee | The account (eeeeeeee) in which the faulty file resides. |
| ---AIO-<br>STAT CC<br>xxxx x | A hexadecimal number (xxxx) representing the AIO device and operational status bytes (STAT) and a hexadecimal value (x) representing the condition code (CC) returned as the result of the AIO instruction. |
| CL<br>xx | A hexadecimal value (xx) representing the cluster protion of the unit address. |
| CONTRLR<br>f | A flag (f) indicating whether or not the controller is partitioned in addition to the device. Y means the controller is partitioned; N means it is not partitioned. |
| CORE<br>(K)<br>dddd | Core size in decimal thousands (dddd). |
| COUNT<br>dddd | The number of entries (in decimal) that duplicate the previous entry. |
| CPU<br>ddd | CPU type (ddd). |
| ----CUR COMM DW--<br>1 2<br>xxxxxxxx xxxxxxxx | Two hexadecimal numbers (xxxxxxxx) representing the command doubleword currently being processed for a device. |
| DATE<br>mm/dd/yy | The month (mm), day (dd), and year (yy) that the error log entry occurred. |
| DCT<br>INDEX<br>xx | A hexadecimal value (xx) indicating the order in which the device is configured into the system at SYSGEN. The index value for the first device is 1. |

Table 32. Error Log Entry Headings (cont.)

| Heading | Description |
|---|---|
| ENTRIES<br>LOST<br>dddd | A decimal value (dddd) representing the number of error log records lost when logging became temporarily impossible for any reason. |
| ENTRY<br>COUNT<br>dddd | A decimal value (dddd) representing the number of entries in the enqueue table belonging to the specified user at the time the error log entry was made. |
| ERROR<br>CODE<br>xxxx | A hexadecimal value (xxxx) giving the error type code for the failure. See Appendix B, "Monitor Error Messages" in the CP-V/BP Reference Manual, 90 17 64, for error code definitions. |
| ERRLOG---<br>CALL ADRS<br>xxxxxxxx | A hexadecimal value (xxxxxxxx) representing the caller's address to which the error logging routine will return when logging is completed. This is used in isolating software faults. |
| FILE NAME | The name of the file in which a fault has been detected. |
| ---HIO-<br>STAT CC<br>xxxx x | A hexadecimal value (xxxx) representing the status (STAT) and a hexadecimal value (x) representing the condition codes (CC) returned in response to an HIO instruction. |
| --INDEX--<br>BAD ENTRY<br>xxxxxxxx | The hexadecimal offset (xxxxxxxx) into a 64-word block in ERRFILE that locates the first word of the incorrect entry. |
| I/O<br>ADRS<br>xxxx | A hexadecimal value (xxxx) representing the physical I/O address. |
| I/O<br>COUNT<br>ddddddddd | A decimal value (ddddddddd) representing the number of SIO instructions executed for a device. This value is reset at system boot time and is not reset at recovery (i.e., it is reset for system start-up types 1, 2, and 3; see START TYPE in this table). |
| I/O ADRS<br>PRIM ALTN<br>xxxx xxxx | A hexadecimal value (xxxx) representing the primary I/O address (PRIM) by which a device can be referenced, and another hexadecimal value (xxxx) representing the alternate address (ALTN) for dual access devices. |
| --I/O--<br>STAT CC<br>xxxx x | A hexadecimal value (xxxx) representing the status (STAT) and a hexadecimal value (x) representing the condition codes (CC) returned in response to an I/O instruction. |
| LOCATIONS<br>xxxxxxxx<br>xxxxxxxx<br>:<br>: | One to fourteen hexadecimal values indicating the addresses of the first fourteen (or less) memory locations exhibiting parity errors. |
| --MEMORY STATUS--<br>1 2<br>xxxxxxxx xxxxxxxx | Two hexadecimal values (xxxxxxxx) representing the status returned in response to an LMS instruction. |
| ---MEMORY STATUS WORDS----<br>1 2 3<br>xxxxxxxx xxxxxxxx xxxxxxxx | Three hexadecimal values (xxxxxxxx) representing status returned in response to an LMS instruction. |
| MDL<br>dddd | A decimal number (dddd) that uniquely identifies peripheral devices by the Xerox model number (defined at SYSGEN). |

Table 32. Error Log Entry Headings (cont.)

| Heading | Description |
|---|---|
| MFI<br>xx | A hexadecimal value (xx) representing the current state of the memory fault indicators returned by the hardware in response to an RD instruction. All memory fault indicators will be reset. (Sigma 6 and 7 only.) |
| MODE<br>d | A decimal value (d) encoding the mode in which the file was opened where: 1 - IN; 2 - OUT; 4 - INOUT; 8 - OUTIN. |
| --OPTIONS--<br>SYMB RT RB<br>f f f | Indicates whether or not the following facilities are available in the system: symbiont routines (SYMB), real-time processing (RT), and remote processing (RB). The flag (f) is equal to Y (present) or N (absent). |
| ORG<br>d | A single decimal digit that indicates the file organization where: 1 - consecutive; 2 - keyed; 3 - random. |
| PAR<br>ERRS<br>xxxx | A hexadecimal value (xxxx) representing the number of memory locations exhibiting parity errors after a memory scan. |
| --POLL-<br>STAT CC<br>xxxx x | A hexadecimal value (xxxx) representing the processor fault status (STAT) and a hexadecimal value (x) representing the condition codes (CC) returned by the hardware in response to a POLP or POLR instruction. |
| POLR<br>RESULTS<br>xxxx | A hexadecimal value (xxxx) representing the processor fault status as returned by the hardware in response to a POLR instruction. |
| ------PSDW-------<br>1 2<br>xxxxxxxx xxxxxxxx | Two hexadecimal numbers (xxxxxxxx) representing the contents of the program status doubleword. |
| RB:FLAGS<br>xxxxxxxx | A word containing bits which define the current state of processing for the failing remote station. The meanings of the bits are defined in Table 32-1 |
| REAL<br>ADRS<br>xxxxxxxx | A hexadecimal value (xxxxxxxx) representing the actual memory address. In an unmapped system, this is the same as the IA field of the PSD. |
| RECOV<br>COUNT<br>dd | A decimal value (dd) which is set to zero at system initialization and incremented by one for every system recovery. |
| REL.<br>SECT.<br>xxxx | A hexadecimal value (xxxx) representing the relative sector at which the inconsistency was detected |
| RELATIVE-<br>SECT.ADRS<br>xxxx | A hexadecimal value (xxxx) representing the relative sector number at which the inconsistency was detected. A relative sector is 256 words long with each sector on a given device being numbered from zero through device end. CP-V maintains file pointers by relative sector number to expedite addressing different devices. |

Table 32. Error Log Entry Headings (cont.)

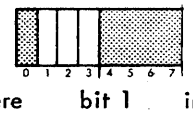
| Heading | Description |
|---|---|
| REM<br>BYTES<br>xxxx | A hexadecimal value (xxxx) representing the remaining byte count as returned in response to a TDV instruction. |
| -RETRY-<br>REQ REM<br>dd dd | A two digit decimal number (dd) representing the maximum number of retries (REQ) after which a device error is returned to requester (value obtained from requester's DCB), and another two-digit value (dd) representing retry request minus the number of entries attempted (REM). The range is between retry request and 0. A 0 value indicates the operation was terminated due to retry count rundown. |
| RP1 RP2 RP3 RP4<br>xx xx xx xx | RP1 through RP4 have unique meanings for each type of remote terminal. See Tables E2 through E7 of Appendix E. |
| SCREECH<br>CODE<br>xx | The hexadecimal code (xx) used by CP-V to identify the system failure that has occurred. See Appendix C. |
| SEEK ADRS<br>xxxxxxxx | A hexadecimal value (xxxxxxxx) representing the physical disk address last used to access this device. |
| ---------SENSE INFORMATION---------<br>xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx | A hexadecimal value (xxxxxxxx) representing the diagnostic information returned from the device as a result of sending a "sense" order to the device. The value has a 4-word maximum, depending on the device. |
| ---SIO-<br>STAT CC<br>xxxx x | A hexadecimal value (xxxx) representing the status (STAT) returned in response to an SIO instruction, and another hexadecimal value (x) representing the condition codes (CC) returned. |
| SITE I.D.<br>eeeeeeee | An EBCDIC value (eeeeeeee) identifying the site (specified at SYSGEN). |
| START<br>TYPE<br>xx | A hexadecimal value (xx) indicating the degree of initialization:<br>1 - PO boot (initial)<br>2 - PO boot under the files<br>3 - System device boot (no recovery)<br>4 - System recovery<br>5 - Operator recovery |
| SUB<br>CODE<br>xx | A hexadecimal code (xx) that differentiates several similar CP-V software check codes. See Appendix C. |
| SUBC<br>STAT<br>xx | A hexadecimal value (xx) representing the status (STAT) of the I/O subchannel received as a result of a TDV instruction. The first byte of the second word of the status received from the device is:<br>  <br>0 1 2 3 4 5 6 7<br>where bit 1 indicates bus check fault if set to one.<br>bit 2 indicates control check fault if set to one.<br>bit 3 indicates memory interface error if set to one. (Xerox 560 only). |

Table 32. Error Log Entry Headings (cont.)

| Heading | Description |
|---|---|
| SUBTYPE<br>xx | A hexadecimal value (xx) indicating the type of copy error that occurred. Type 01 indicates read error; i.e., the ghost ERR:FIL received an error indication when reading the original error file. Type 02 indicates read error end, meaning that subsequent error log entries were correctly read from the original error file. Type 03 indicates a length error; i.e., the original error file record length was incorrect. Type 05 indicates incorrect time; i.e., the time of the following entry is either out of range or goes backward. Type 06 indicates illegal entry type; i.e., the type code of the following entry was found to be illegal by the ghost ERR:FIL. |
| SYMB.<br>DCT<br>xx | A hexadecimal value (xx) representing the order in which the symbiont device is configured into the system; i.e., the DCT index of the symbiont device. |
| SYSTEM<br>CP-V eee | Displays the operating system name (CP-V) and three EBCDIC characters (eee) representing the system version specified at SYSGEN. |
| TDV CUR<br>COMM DA<br>xxxxxx | A hexadecimal value (xxxxxx) representing the current command doubleword address returned in response to a TDV instruction. The address is in doubleword form. Therefore it should be multiplied by two to obtain the absolute word location. |
| ---TDV-<br>STAT CC<br>xxxx xx | A hexadecimal value (xxxx) representing the status (STAT), and a hexadecimal value (x) representing the condition codes (CC) returned in response to a TDV instruction. |
| TIME<br>hh:mm:ss:nnn | The time the error occurred, in hours (hh), minutes (mm), seconds (ss), and milliseconds (nnn). |
| TIME LAST<br>DUPLICATE<br>hh:mm:ss:nnn | The time in hours, minutes, seconds, and milliseconds at which the last duplicate of the preceding entry occurred. |
| TIME LAST<br>LOST ENTRY<br>hh:mm:ss:nnn | The time of occurrance when the last entry was lost in hours, minutes, seconds and milliseconds. |
| TIME<br>RES<br>dd | A decimal value (dd) in milliseconds representing the resolution of the time field of all error log entries; e.g., if the time resolution is 2, then the time value for all error log entries is accurate to two milliseconds. |
| ---TIO-<br>STAT CC<br>xxxx x | A hexadecimal value (xxxx) representing the status (STAT) and a hexadecimal value (x) representing the condition codes (CC) returned in response to a TIO instruction. |
| --TRAPPED--<br>INSTRUCT CC<br>xxxxxxxx x | A hexadecimal value (xxxxxxxx) representing the contents of the location pointed to by the trapped instruction's address (INSTRUCT) in the PSD, and another hexadecimal value (x) representing the trap condition codes (CC). |
| ---------TRAPPED-----<br>INSTRUCT CC EFF.ADRS<br>xxxxxxxx x xxxxxxxx | A hexadecimal value (xxxxxxxx) representing the contents of the location pointed to by the trapped instruction's address (INSTRUCT) in the PSD; a hexadecimal value (x) representing the trap condition codes (CC); and another hexadecimal value (xxxxxxxx) representing the final address (EFF.ADRS) computed for the trapped instruction. |

Table 32. Error Log Entry Headings (cont.)

| Heading | Description |
|---|---|
| UN<br>xx | A hexadecimal value (xx) representing the unit portion of the Xerox 560 unit address. |
| UNIT<br>NAME<br>eeee | A two-to-four EBCDIC character mnemonic name identifying one of the following: CPU; MI (Memory Interface); PI (Processor Interface); MIOP (Multiplexed Input Output Processor); RMP (Rotating Memory Processor); CT (Communication Terminator); SU (System Unit). |
| USER<br>I.D.<br>xxxx | A hexadecimal value (xxxx) which is a unique number assigned by the system to the particular job or session. |
| USER<br>NO.<br>xx | A hexadecimal value (xx) representing the index into internal system tables used to access user-specific information. |
| VOLUME<br>SERIAL<br>eeeeee | Four to six EBCDIC characters (eeeeee) that a user has supplied to identify a tape or private disk pack. |
| WORKSTATION<br>NAME<br>eeeeeeee | A one to eight EBCDIC character name which defines the identity and characteristics of a remote station to the system. A workstation name is not necessarily associated with one fixed physical terminal. The workstation name is specified when the remote terminal logs on. |
| message | An operator message of up to 72 alphabetical characters. |

Table 32-1. RB:FLAGS Structure

| Bit | Name | 7670 | 2780 | IRBT | Meaning |
|---|---|---|---|---|---|
| 0 | BPBIT | x | x | | Block protect toggle (ACK0/ACK1). |
| 1 | IGBIT | x | x | x | Cards after !FIN were ignored. |
| 2 | MORBIT | x | | | Waiting for next portion of deck. |
| 3 | HUBIT | x | x | x | Line hung up. |
| 4 | PUNBIT | x | | | Punching is allowed. |
| 5 | DCBIT | x | x | x | WSN specified at SYSGEN. |
| 6 | HASPBIT | x | x | x | IRBT line. |
| 7 | SLVBIT | | | x | This system is slave. |
| 8 | ALBIT | x | x | x | RBLOG key-in done. |
| 9 | XP1BIT | | | x | X1 specified in Super. |
| 10 | 2780BIT | x | x | x | 2780 line (may be changed to IRBT at logon). |
| 11 | IBMBIT | | | x | N3 specified in Super. |
| 12 | DIALBIT | x | x | x | DIAL specified at SYSGEN. |

Table 32-1. RB:FLAGS Structure (cont.)

| Bit | Name | 7670 | 2780 | IRBT | Meaning |
|---|---|---|---|---|---|
| 13 | EDISBIT | x | x | x | ERROR MAX on line. |
| 14 | OFFBIT | x | x | x | Do not connect line (RBX) - Set except at logon for IRBT. |
| 15 | RBXBIT | x | x | x | Disconnect line now. |
| 16 | DUPBIT | x | x | x | 1 – full-duplex; 0 – half-duplex. |
| 17 | DISCBIT | x | x | x | Disconnect when output done. |
| 18 | LOFBIT | x | x | x | RBDISC sent (temporary setting). |
| 19 | SYSBIT | x | x | x | :SYS jobs legal. |
| 20 | HALBIT | x | x | x | HOLD all flag set. |
| 21 | CLKBIT | | | x | Wait before ACK0-idle. |
| 22 | ACTBIT | x | x | x | Line logged on. |
| 23 | CRTBIT | x | | | RBBAT disables RBSSS. |
| 24 | XP2BIT | | | x | X2 specified in Super. |
| 25 | OADBIT | x | x | x | Set OFFBIT after disconnect. |
| 26 | FIABIT | x | x | x | RBCC altered the stream status. |
| 27 | SSSBIT | x | | | Inputting with output suspended. |
| 28 | LIPBIT | x | x | x | Logging on. |
| 29 | FINBIT | x | | | FIN has been read. |
| 30 | EMBIT | x | x | | 1 – NOEM specified; 0 – EM specified. |
| 31 | OBBIT | | | x | Old BCB was read. |
| 31 | FRBIT | x | | | Initial read of file. |

**SLIS** A sorted listing is requested with the SLIS command. The command has the form

SL[IS]

As in the chronological listing, the sorted listing includes all qualified error log entries. In this listing, however, entries are ordered by their type, and if they are peripheral class errors, by their model number and I/O address also.

Error records are first categorized by ELLA as system, peripheral or secondary records, (see Table 33). System records and their associated secondaries are listed first. Except for the system ID record and configuration record (which are printed in front of all other records), system records are listed in ascending type code order as given in Table 33. Secondary records are printed following their associated primary records.

Peripheral class records are sorted in three phases. They are first separated by model number and printed in ascending model number order. All peripheral records with the same model number are then separated and listed in ascending device address order. Finally, all the records containing both the same device address and the same model number are printed in ascending type code order. Any secondary records associated with peripheral class entries are printed following the associated peripheral records.

Any secondary record that appears in the error file that cannot be linked with a primary record through the above rules of association will be printed after all peripheral records and their associated secondaries under the heading

>>>UNASSOCIATED SECONDARIES<<<

Each time a record is listed that has a different device address than that of the preceding record listed, a Model Number/Address heading is produced under the heading

>>>MODEL NO. :xxxx I/O ADDRESS; xxxx

where MODEL NO. is the 4-digit Xerox model number designation of the device and I/O ADDRESS is the 4-digit (hexadecimal) I/O address of the device.

An example of a sorted listing is given in Example 4.

Table 33. Error Log Entry Types

| Name | Type Code | Description |
|---|---|---|
| System Class | | |
| COPY ERROR | 10 | Recorded as a result of error conditions in the error logging mechanism. The particular malfunction is identified in the subtype field (see SUBTYPE in Table 32). If the record subtype is 03, 05, or 06, the record is followed by the 64-word buffer in which the error occurred. |
| PARITY ERROR | 17 | Recorded when program execution is interrupted to location X'56' (MFI) on Sigma 6 or 7 or is trapped to location X'4C' (parity trap) on Sigma 9 or Xerox 560. |
| SYSTEM STARTUP | 18 | Recorded when the system is booted and at each recovery. |
| WATCHDOG TIMER | 19 | Recorded when program execution traps to location X'46' due to a watchdog timer run-out condition. |
| FILE INCONSISTENCY | 1A | Recorded when the operating system cannot access a file in the file management system. The code displayed is described in Appendix B of the CP-V/BP Reference Manual, 90 17 64. |
| SYMBIONT INCONSISTENCY | 1B | Recorded when the operating system cannot access a symbiont file in the symbiont file management system. |
| INSTRUCTION EXCEPTION | 1D | Recorded when program execution traps to location X'4D' on Sigma 9 or Xerox 560 due to an instruction exception condition. |
| LOST ENTRY | 1E | Recorded when error log buffering constraints, timing considerations, and error detection rates force error logging to be temporarily suspended or otherwise impossible. |

Table 33. Error Log Entry Types (cont.)

| Name | Type Code | Description |
|---|---|---|
| System Class (cont.) | | |
| POWER ON | 20 | Recorded when the hardware power monitor forces program execution to trap to location X'51' as a result of detecting a restoration of power condition. This normally occurs as a result of a power outage of 500 milliseconds or more in duration. |
| CONFIGURATION | 21 | Peripheral device configuration data recorded when ERRFILE is entered. |
| SYSTEM IDENTIFICATION | 22 | System information recorded when ERRFILE is entered. |
| TIME STAMP | 23 | The date and time which is recorded when ERRFILE is entered in the system and every hour on the hour. |
| BAD GRANULE RELEASE | 24 | Recorded when either a bad disk address has been detected or when the granule to be released is already free (dual allocation). |
| REMOTE PROCESSING ERROR | 26 | Recorded when an error is detected in the transmission of data to or from a remote processing workstation. |
| OPERATOR MESSAGE | 27 | A message entered by the operator through use of the ERSEND key-in. |
| PROCESSOR FAULT INTERRUPT | 30 | Recorded when there is a processor fault interrupt (location X'56') on the Xerox 560. |
| MEMORY FAULT INTERRUPT | 31 | Recorded when there is a memory fault interrupt (location X'57') on Sigma 9 or Xerox 560. |
| PROCESSOR CONFIGURATION | 41 | Processor configuration from Configuration Control Panel (Xerox 560 only) recorded when the system is booted. |
| ENQUEUE TABLE OVERFLOW | 50 | Recorded to log specific information after the operating system has detected an enqueue table overflow condition. |
| UNKNOWN TYPE = xx | xx | An unknown type code xx has been encountered by ELLA in an error log entry. |
| Peripheral Class | | |
| SIO FAILURE | 11 | Recorded when the condition codes returned by the SIO instruction are such that either CC1 or CC2 are true. Conditions that indicate IOP busy, or lack of operator action such as "device manual" may not be considered an error condition (and in such case will not be recorded). |
| DEVICE TIMEOUT | 12 | Recorded when the time-out value specified by DCT11 has been exceeded. |

Table 33. Error Log Entry Types (cont.)

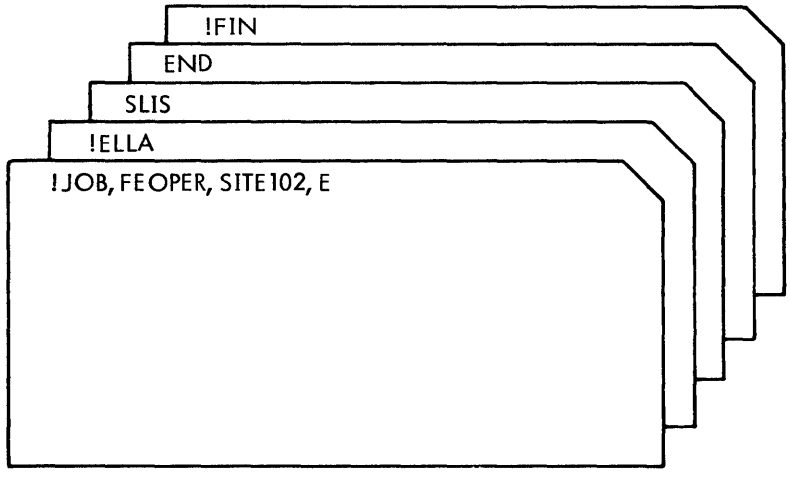
| Name | Type Code | Description |
|---|---|---|
| Peripheral Class (cont.) | | |
| UNEXP. INTERRUPT | 13 | Recorded when no match can be found between the I/O address returned in the status register by the AIO instruction and any DCT1 I/O address of a device known to be busy. AIO CC = 11xx will not be logged. |
| DEVICE ERROR | 15 | Recorded when an I/O request is not successful upon one of the specified number of retries. (It may or may not have eventually been successful.) |

Table 33. Error Log Entry Types (cont.)

| Name | Type Code | Description |
|---|---|---|
| Peripheral Class (cont.) | | |
| PARTITIONED RESOURCE | 51 | Recorded when a resource has been partitioned from the system. |
| RETURNED RESOURCE | 52 | Recorded when a previously partitioned resource has been returned to the system. |
| Secondary Data Class | | |
| DEVICE ERROR SECONDARY | 16 | Recorded when nonzero sense data is available following a device error. |
| DUPLICATE ENTRIES | 1F | Recorded when the error logging mechanism detects identical consecutive errors. This prevents the error log from becoming saturated with redundant information. |
| SECONDARY POLL RECORD | 32 | Recorded for each nonzero poll status received by the processor polling routines. |
| MEMORY PARITY SECONDARY | 42 | Recorded for each memory unit that has recorded an error as determined by the memory polling routines (i.e., bits 22-31 of status word zero are nonzero) for Xerox 560. |
| MEMORY PARITY SECONDARY | 43 | Recorded for each memory unit that has recorded an error as determined by the memory polling routines (i.e., bits 22-31 of status word zero are nonzero) for Sigma 9. |
| MEMORY PARITY SECONDARY | 44 | Recorded to log specific information obtained by scanning memory to attempt to isolate locations which cannot sustain correct parity. |

Example 4. Use of the SLIS Command

For this example, the following batch job deck was submitted.




In the resultant sorted listing below, all related entries are grouped together. This facilitates the scanning of the error log that is necessary in order to determine the common characteristics of related failures that have occurred over a period of time.

Note the OPERATOR MESSAGE entry in the listing. Such messages can be entered into the error log at the operator's console by means of the ERSEND key-in. (See the CP-V/OPS Reference Manual, 90 16 75.)

```
*SLIS


S O R T E D   L I S T I N G
- - - - - -   - - - - - - -


FROM  07/10/74  12:00:00:000
TO    12/31/99  23:59:59:999


*** FILE INCONSISTENCY ***
                        DCT     RELATIVE                ERROR
TIME            ACCOUNT INDEX   SECT ADRS  MODE  ORG   CODE              ------------FILE NAME------------
16:03:13:648    771731  09      0110       01    02    757F               RTRTEXT:V113
16:42:32:197    771731  09      0110       01    02    757F               RTRTEXT:V113


*** TIME STAMP ***   DATE=07/10/74  TIME=12:00:00:004
*** TIME STAMP ***   DATE=07/10/74  TIME=13:00:00:004
*** TIME STAMP ***   DATE=07/10/74  TIME=14:00:00:004
*** TIME STAMP ***   DATE=07/10/74  TIME=15:00:00:004
*** TIME STAMP ***   DATE=07/10/74  TIME=16:00:00:006


***OPERATOR MESSAGE***   TIME = 14:22:03:782
9TA81 CAPSTAN DRIVE NOISY (JDR)


,,, MODEL NO:7160 I/O ADDRESS:0004 ...

*** SIO FAILURE ***
                ---SIO-  ---TDV-  SUBC  TDV CUR  REM
TIME            STAT CC  STAT CC  STAT  COMM DA  BYTES  MFI
07:02:28:922    2A42 6   2042 6   00    001179   004C   00
15:05:21:166    2A42 6   2042 6   00    001179   004C   00


*** DEVICE ERROR ***
                ---AIO-  ---TIO-  ---TDV-  TDV CUR  REM           I/O       ----CUR COMM DW--  -RETRY-  VOLUME SUBC
TIME            STAT CC  STAT CC  STAT CC  COMM DA  BYTES  MFI   COUNT       1         2      REQ REM  SERIAL STAT SEEK ADRS
15:09:20:862    0048 6   1842 0   2042 2   00118B   0000   00 0000000984  09008C70  2E000078  03  03          00   0000000B



,,, MODEL NO:7271 I/O ADDRESS:00E0 ...

*** DEVICE ERROR ***
                ---AIO-  ---TIO-  ---TDV-  TDV CUR  REM           I/O       ----CUR COMM DW--  -RETRY-  VOLUME SUBC
TIME            STAT CC  STAT CC  STAT CC  COMM DA  BYTES  MFI   COUNT       1         2      REQ REM  SERIAL STAT SEEK ADRS
12:56:21:278    0458 6   1842 0   0442 2   0011C7   0000   00 0000089148  02031800  1E000800  03  03          00   00F90A00
15:42:55:694    0458 6   1842 0   0442 2   000A7E   0000   00 0000138032  02075800  1E000570  08  08          00   00E70404
15:42:55:906    0458 6   1842 0   0442 2   000A7E   0000   00 0000138036  02075800  1E000570  08  07          00   00E70404

... DEVICE ERROR SECONDARY
                I/O
TIME            ADRS  ---------SENSE INFORMATION---------
12:56:21:284    00E0  00F90A02 0305F500 00CA0000 00000000
15:42:55:700    00E0  00E70500 0119FA02 80CA0000 00000000
15:42:55:910    00E0  00E70500 0119FA0E 03E20000 00000000
```

**SUM** The SUM command requests a summary of the contents of the error file which lists the total number (in decimal) of qualified error log entries for each error type. The command has the form

SU[M]

In addition to error totals, the summary contains an I/O activity count for each device that has errors recorded. IO ACTIVITY is the count of all SIOs issued to a given device for the time period covered by the summary.

Unlike other listings produced by ELLA, logical device addresses are used in the summary rather than physical addresses. For example, a device CRA03 would appear as A03 and a device DCBF0 would appear as BF0.

Example 5 provides an example of the SUM command.

Example 5. Use of the SUM Command

```
!ELLA
*SUM

E R R O R   S U M M A R Y
- - - - -   - - - - - - -


FROM   07/10/74   00:00:00:000
TO     12/31/99   23:59:59:999


SYSTEM ERRORS

TYPE                      ERRORS
SYSTEM STARTUP                 1
CONFIGURATION                  6
SYSTEM I.D.                    1
TIME STAMP                    16
FILE INCONSISTENCY             2


DEVICE ERRORS

        IO      SIO    UNEXP    DEV      DEV        IO
MDL     ADRS    FAIL   INTRPT   ERROR    TIMEOUT    ACTIVITY
7140     A03       0        0       13         0    0000014798
7160     A04       0        0        6         0    0000004906
7322     A80       0        0       78         0    0000227574
7323     A83       3        0        3         0    0000013295
7323     A84       0        0        4         0    0000021926
7323     A85       0        0        6         0    **********
7271     AE0       0        0        3         0    0000161268
7232     BF0       0        0        2         0    0000248749


TOTAL ERRORS:  00144
```

**DISP** The DISP command requests a graphical display of error log entries. The DISP command has the form

DI[SP] [, interval]

where interval specifies the time interval, in minutes, to be used for the graph. The interval specified may range from 1 to 60. The default interval is ten minutes.

The graph produced by the DISP command is a bar graph. Each line begins with the end time of the interval, followed by the 2-digit error type code of each error recorded during the interval. If the number of errors for a given interval exceeds 30, then only the first 30 error type codes are printed, and FF is printed at the end of the line.

Only qualified error log entries are included. Time Stamp, Configuration, and ID entries are always excluded.

The first and last lines in the graph are the first and last intervals within the TIME boundary that contains qualified error log entries. The actual time period scanned is printed at the beginning of the listing.

An example of a graphic display is given in Example 6.

Example 6. Use of DISP Command

The user in Example 5 continues as follows:

```
*DISP,45 ®

G R A P H I C  D I S P L A Y
- - - - - - -  - - - - - - -


FROM  00/00/00  00:00:00:000
TO    12/31/99  23:59:59:999


TIME ERROR
-----0----------10----------20----------30--
03:56 15
04:41 1A1A
05:26
06:11
06:56
07:41 15161515161615111516
08:26 1B111121515111515
09:11 15
09:56
10:41
11:26
12:11 2715

END OF FILE
```

The distribution of errors over the scanned time period is more readily apparent in this display than in the other forms of error listings. This display is used to check for patterns and trends in error occurrences. The digits that form this bar graph are in pairs. (e.g., the line 1B111121515111515 contains eight digit pairs). Each digit pair represents one error and the two digits are the type code of the error.

**END** The END command terminates ELLA and exits to the monitor. The format of the command is

E[ND]

## BOUNDARY COMMANDS

The boundary commands are used to select specified portions of the error file for display. In order for an error record to be accepted for display, it must satisfy each boundary. There are four boundaries:

- Time
- Model number
- Device address
- Error type code

An error log entry will be listed by a subsequent task command if it was recorded within the time limits specified by the TIME command and if it has one of the error type codes specified by the TYPE command. If the entry is a peripheral class entry (see Table 33), it must also have a model number field and an address field which agrees with one of the model numbers and one of the device addresses specified by the MOD and DEV commands respectively.

It is not necessary, however, to use any of the boundary commands. If a boundary command is not used or a boundary has been reset, all error log entries are considered to have met the conditions of display for that boundary.

Boundary commands, if judiciously used, can be especially helpful in minimizing ELLA output when the output listing function has been assigned to a slow speed device such as an on-line terminal.

**RSET** The RSET command resets all boundary parameters to their default values. (The default values are given in the subsequent boundary command descriptions.) The RSET command has the form

R[SET]

**TIME** The TIME command sets both date and time boundaries. Error log entries are displayed only if they occurred between the begin date and time and the end date and time. The TIME command has the form

TI[ME][,begin][-end]

where begin and end have the form

[month/day/year][,hour:minute]

or

[hour:minute][,month/day/year]

where

month = 1-12

day = 1-31

year = 01-99

hour = 00-23 (24 hour clock)

minute = 00-59

If the TIME command is not used (or if time and date are reset by the RSET command) ELLA establishes the following beginning and ending times:

begin = 00/00/00, 00:00

end = 12/31/99, one millisecond before midnight. (The time is recorded internally in millisecond increments.)

If only one group (i.e., 'begin' or 'end') is entered under the TIME command, the current state of the other group remains in affect.

It is not necessary for both fields within a group to be entered. If time is the only field entered in a group, then the date for that group is the current day by default. Time by default is a bit more complex. If the date field is the only field entered for 'begin', then 00:00 is the time by default. If the date field is the only field entered for 'end' then 1 millisecond before midnight is the time by default.

Examples of the TIME command are given in Example 7.

Example 7. TIME Command Usage

The following series demonstrates TIME command usage. Assume all of the TIME entries have been entered consecutively at the console.

*TIME, 4/25/73-5/27/73 ↵

The time limits have been set by the entry above as follows: starting time is 00:00 on 4/25/73, and ending time in one millisecond before midnight on 5/27/73. The only error log entries that will be displayed by subsequent task commands are those that lie between these two time points.

*TIME, 2:00- 8:00 ↵

The limits have now changed so that the starting time is 2:00 AM on the current day (i.e., the day on which the ELLA run is being made), and ending time is 6:00 PM on the current day. (When no date is entered, the current date is implied.)

*TIME, 18:00-10:00 ↵

This entry is illegal and will produce a diagnostic message because the starting time is later than the ending time. The limits 2:00 and 18:00 from the previous entry are still in effect.

*TIME 00:00 ↵

Here the starting time has been changed to 00:00 on the current day. Since no ending time has been entered, the previous ending time of 18:00 remains in effect.

*TIME, 1/1/74-12:00 ↵

This sets the starting limit to 00:00 on 1 January 1974, and the ending limit to noon on the current day.

*TIME, -13:00 ↵

The previously entered starting limits (1/1/74) remain in effect because no starting parameter is entered here. The ending limit is changed to 13:00 for the current day.

*RSET ↵

ELLA time defaults are reestablished. The default is the entire time span of the error log.

*TIME, 12:00, 10/15/73-10/16/73, 12:00 ↵

Finally, the starting time is set to noon on 10/15/73 and the ending limit to noon on 10/16/73. Note that the order of time and date entry is immaterial.

**TYPE** The TYPE command allows the user to select error log entries for display by specifying an error record type code (see Table 33). The TYPE command has the form

$$TY[PE],\left\{\begin{matrix}0\\type_1[,\ldots type_5]\end{matrix}\right\}$$

where

type is a hexadecimal error type code.

0 specifies that the default (all types) is to be reestablished.

If error log entry types have been specified via the TYPE command, error log entries are displayed only if they have a type code equal to one of the types specified. Up to five types may be specified for display at one time.

If the TYPE command is not used, records of all types are displayed (including any records that may have illegal type codes). Displaying all types is the default condition. Having once used the TYPE command, the default condition may be reestablished by entering TYPE,0 or by using the RSET command.

Each time the TYPE command is used, the previoualy specified types are replaced with the newly entered types.

**DEV** The DEV command selects error log entries for display by specifying up to five I/O addresses. The DEV command has the form

$$DE[V],\left\{\begin{matrix}0\\address_1[,\ldots address_5]\end{matrix}\right\}$$

where

address is a 1 to 4-digit hexadecimal physical I/O address. (Leading zeros in the address need not be specified.)

0 specifies that the default (all devices) is to be reestablished.

Up to five physical I/O addresses may be specified. Each time the DEV command is used, the previously specified addresses are replaced with the newly entered addresses.

If this command is not used, records are displayed without regard to their associated device address. This is the default condition. Having once used the DEV command, the default condition may be reestablished by entering DEV,0 or by using the RSET command.

When particular device addresses have been specified through use of the DEV command, error log entries classified as system records (see Table 33) are not displayed, and a peripheral class entry is displayed only if the device address field in that entry is equal to one of the addresses specified by the DEV command.

**MOD** The MOD command selects error log entries for display by specifying up to five model numbers. When particular model numbers have been specified through use of the MOD command, error log entries classified as system records (see Table 33) are not displayed, and a peripheral class entry is displayed only if the model number associated with that record is equal to one of the model numbers specified by the MOD command. The MOD command has the form

$$M[OD],\left\{\begin{matrix}0\\model_1[,\ldots model_5]\end{matrix}\right\}$$

where

model is a 4-digit model number (e.g., 7446, 7271).

0 specifies that the default (all models) is to be reestablished.

Each time the MOD command is used, the previously specified model numbers are replaced with the newly entered model numbers.

If this command is not used, records are displayed regardless of their associated model number. This is the default condition. Having once used the MOD command, the default condition may be reestablished by entering MOD,0 or by using the RSET command.

Examples 8 through 12 demonstrate the use of the MOD, DEV, and TYPE commands for selecting specific portions of ERRFILE for display. The examples are consecutive portions of one continuous on-line session. In these examples, the user has chosen to display everything at the terminal. This means that the user will be able to see the output immediately, but the user must make judicious use of this rather slow output device.

Example 8. Use of the MOD, DEV, and TYPE Commands

Assume that the user had already requested and received a summary of the error file (which would be a logical first step). The user then proceeded to display the operator messages present in the error file.

```
*TYPE,27 (RET)
*CLIS (RET)


C H R O N O L O G I C A L  L I S T I N G
- - - - - - - - - - - - - - - - - - - - -


TYPE =27
FROM  00/00/00  00:00:00:000
TO    12/31/99  23:59:59:999


*** OPERATOR MESSAGE ***     TIME  =  12:33:00:079
9TA31 CAPTSTAN NOISY (JBR)
```

Note that the TYPE parameter is listed at the beginning of the display because TYPE is no longer set to the default.

Because only one type of error was requested, the terminal is a practical display device. If more than one type of error is requested, a slightly different procedure can be used as shown in the next example.

Example 9. Use of the MOD, DEV, and TYPE Commands

The user from the previous example next desired to examine some system failures. Note that the new TYPEs entered in this example replace the old TYPE entered in the previous example.

When more than one type of error is requested, a sorted listing often reduces the time required for output. This is due to the fact that ELLA only prints headings when a new type of entry is to be listed and SLIS groups all related entries together.

```
*TYPE,18,1B (RET)
*SLIS (RET)


S O R T E D  L I S T I N G
- - - - - -  - - - - - - -


TYPE =18 1B
FROM  00/00/00  00:00:00:000
TO    12/31/99  23:59:59:999


*** SYSTEM STARTUP ***
                          START  RECOV  SCREECH  SUB-
TIME          DATE        TYPE   COUNT  CODE     CODE
09:32:00:000  07/10/74    04     01     19       00
12:05:00:000  07/10/74    01     01     00       00
17:21:00:000  07/10/74    05     02     00       00


***SYMBIONT INCONSISTENCY***
              DCT    REL.   SYMB.
TIME          INDEX  SECT.  DCT
11:29:08:406  09     00A0   02
16:03:13:648  09     0110   02
```

Example 10. Use of the MOD, DEV, and TYPE Commands

The user proceeded as follows:

```
*TYPE,17 (RET)
*DEV,1F0 (RET)
*CLIS (RET)
```

No output was produced because TYPE 17 and DEV X'1F0' are mutually exclusive. Type 17 (Parity Error) is a system class error while device X'1F0' implies that peripheral class errors are desired. Entering either MOD or DEV values precludes the display of any system errors (only peripheral class errors will have model or device address information, and all four boundaries tests – MOD, DEV, TYPE, and TIME – must be passed for an error log entry to be displayed).

Example 11. Use of the MOD, DEV, and TYPE Commands

The user next decided to examine SIO failures on several devices.

```
*TYPE,11 (RET)
*DEV,4,81,82,83 (RET)
*CLIS (RET)


C H R O N O L O G I C A L   L I S T I N G
- - - - - - - - - - - - - - - - - - - - - -


TYPE =11
DEV  =0004 0081 0082 0083
FROM  00/00/00  00:00:00:000
TO    12/31/99  23:59:59:999


*** SIO FAILURE ***
                        I/O    ---SIO-   ---TDV-   SUBC   TDV CUR   REM
TIME            MDL     ADRS   STAT CC   STAT CC   STAT   COMM DA   BYTES   MFI
12:36:30:782    7323    0083   2000 6    1000 6    00     0011B7    0001    00
12:37:29:518    7323    0083   2000 6    1000 6    00     0011B7    0001    00
12:40:10:398    7323    0083   2000 6    1000 6    00     0011B7    0001    00
**BREAK
*
```

After examining several of the failures on device X'0083', the user realized that no new information would be gained by listing the remaining errors. Therefore, the user interrupted the listing process by activating the BREAK key at the terminal.

Example 12. Use of the MOD, DEV, and TYPE Commands

Finally, the user summarized, in graphic form, all Model 7322 and 7323 failures that occurred on the current day.

```
*DEV,0
*TYPE,0
*MOD,7322,7323
*TIME,00:00
*DISP


G R A P H I C  D I S P L A Y
- - - - - - -  - - - - - - -


MODL =7322 7323
FROM   02/09/74   00:00:00:000
TO     12/31/99   23:59:59:999


TIME ERROR
-----0----------------10----------------20----------------30--
09:11 11
09:21
09:31
09:41
09:51
10:01
10:11
10:21
10:31
10:41
10:51
11:01
11:11
11:21
11:31
11:41 15161515161611151516 16
11:51 1315161516
12:01
12:11
12:21
12:31 1516

END OF FILE
```

**DSPL** The DSPL command displays the current state of those ELLA parameters that are alterable by the boundary commands. The date and time boundaries are always listed by this command. Each of the remaining boundaries will also be listed unless its current state is its default state.

The DSPL output is printed both on the output listing device and the command input device. In batch operation, DSPL output is only directed to the output listing device since the input device is the card reader.

The format of the command is

DS[PL]

An example of the command is given in Example 13.

## PREDEFINED TASKS

This section contains a set of predefined tasks that should be useful to the person who needs periodic error log reports, but has no need for a more precise knowledge of the ELLA processor's command structure. These tasks could be maintained as job decks (as illustrated here), or the commands might be entered into a file to facilitate on-line submission to the batch stream (see the TEL BATCH command in the CP-V/TS Reference Manual, 90 09 07). The account from which these jobs are run must have a diagnostic privilege level (A0 or higher). The tasks are listed in Examples 14 through 16.

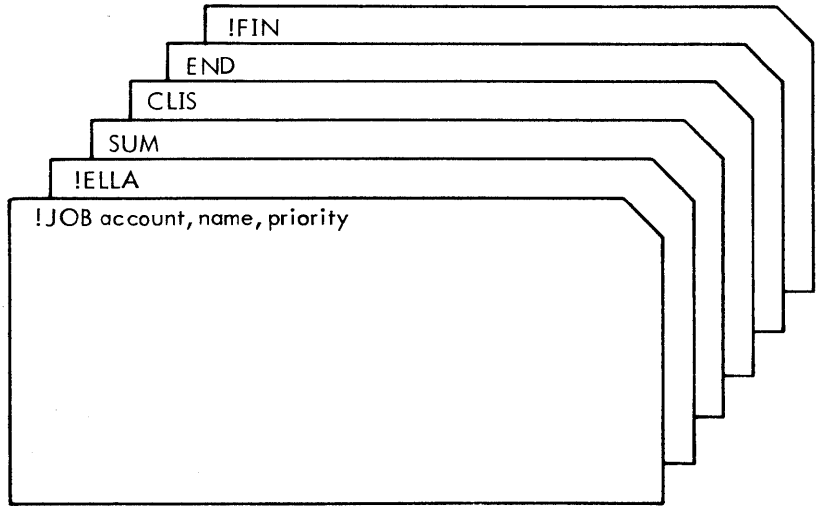
Example 13. Parameter Display

The on-line user may check the current state of the ELLA boundaries conveniently with the DSPL command. If the listing device has been assigned to a line printer, the boundary information will be displayed on both the line printer and the user's terminal.

```
*RSET (RET)
*SET,LIST,LP (RET)
*DSPL (RET)
FROM  00/00/00    00:00:00:000
TO    12/31/99    23:59:59:999
*TYPE,11,12,15 (RET)
*DEV,E1 (RET)
*DSPL (RET)
TYPE =11 12 15
DEV=00E1
FROM  00/00/00    00:00:00:000
TO    12/31/99    23:59:59:999
```

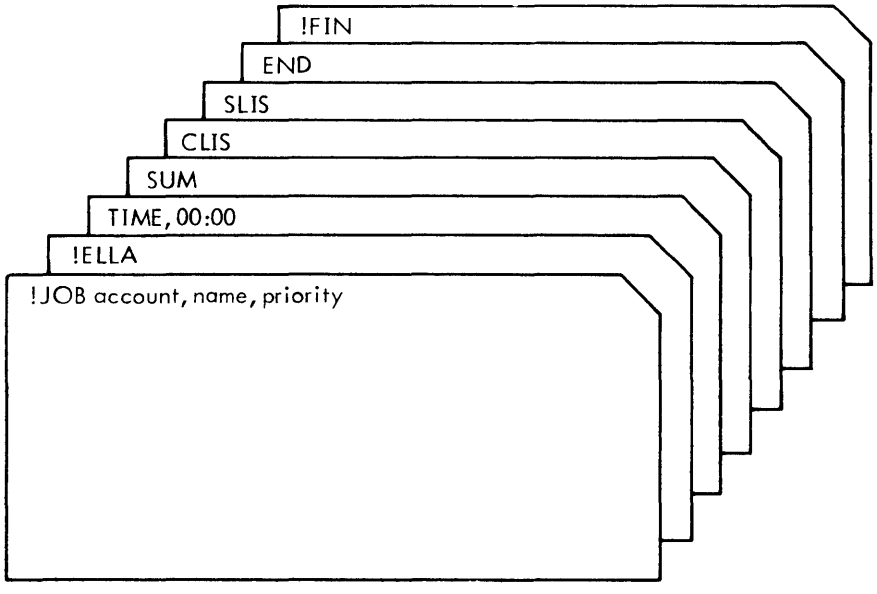
Example 14. Listing the Entire Error File

The following deck obtains an error summary and a chronological listing of the entire contents of the error file.



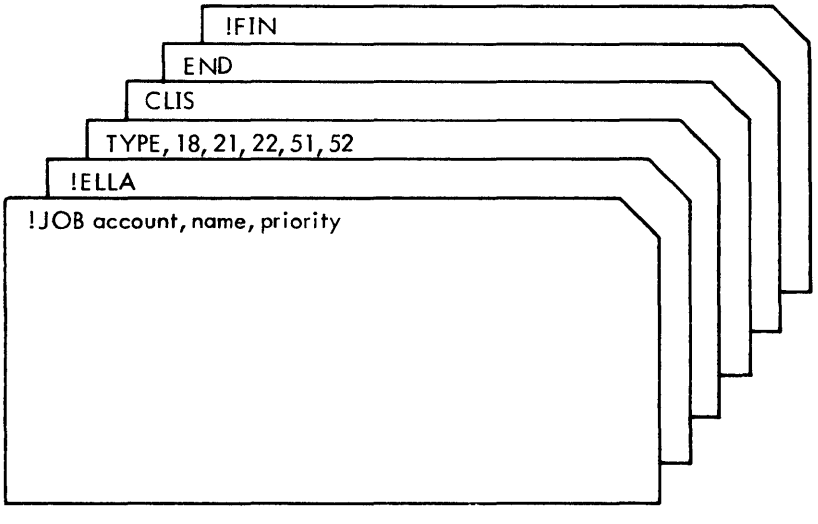
Example 15. Listing Errors for the Current Day

The following job deck obtains an error summary, a chronological listing, and a sorted listing of the errors recorded by the system on the current day. If error log reports are to be obtained daily, it is recommended that this job be run at the end of the processing day.




Example 16. Listing Start-Ups, Configuration, and Device Partitioning Activity

The following job deck obtains all the configuration data together with system start-up, partitioned resource, and returned resource entries in chronological order.



## ELLA MESSAGES

Messages output by the ELLA processor are listed in Table 34.

## ELLA COMMAND SUMMARY

ELLA commands are summarized in Table 35. The left-hand side lists the command formats. The right-hand side describes the function of the command.

Table 34. ELLA Messages

| Message | Meaning |
|---|---|
| ABNORMAL ERROR CODE = xx<br>SUBCODE = xx | An abnormal condition was detected in issuing a system CAL. The abnormal code and subcode are described in the CP-V/BP Reference Manual, 90 17 64, (and the CP-V/TS Reference Manual, 90 09 07). See the system analyst. |
| BREAK | The BREAK key was depressed. ELLA stops processing and waits for a new command. |
| ELLA 708006-A00 | This heading is output when ELLA is first loaded. |
| ERRFILE IS BUSY, WILL TRY AGAIN | ELLA tried to access the error log file and found it busy. |
| **ERRLOG NON-EXISTENT | The ERRFILE file does not exist. See the system analyst. |
| ERROR IN KEY FORMAT<br>(YEAR/DATE NOT IN PACK DECIMAL) | An ERRFILE entry had an erroneous key. See the system analyst. |
| ERROR IN SYSTEM TIME | The time in the error log file was not logical. See the system analyst. |
| ERROR OCCURRED: CODE = xx<br>SUBCODE = xx | An error was detected in issuing a system CAL. The error code and subcode are described in the CP-V/BP Reference Manual, 90 17 64, (and the CP-V/TS Reference Manual, 90 09 07). See the system analyst. |
| ERROR: TIME .GT.<br>99:59:59:999 | The time in an error log entry was greater than 99 hours, 59 minutes, 59 seconds, or 999 milliseconds. See the system analyst. |
| ERROR: TOO MANY CHARACTERS OR LINES | ELLA tried to output more than 132 characters to the line printer. See the system analyst. |
| FILE ASSIGNMENT IS NOT IMPLEMENTED YET | ELLA tried to use the default assignment to ERRFILE and was unsuccessful. Assign M:BI to ERRFILE. |
| INSUFFICIENT PRIVILEGE LEVEL ABORT | ELLA requires an A0 or higher privilege level. |
| INVALID REQUEST | The command entered was invalid. |
| NON-REASSIGNABLE | Once the operator's console is assigned as the control device, it cannot be reassigned. |
| NOTHING IN ERRFILE | ERRFILE does not contain any records. |
| **OVERFLOW OF SORT OR MOD/IO TABLES | ELLA will only support 50 unique I/O addresses. Use the boundary commands to restrict the number of I/O addresses. |
| UNABLE TO LOAD SEGMENT = nn | ELLA tried to load overlay number nn and an error was detected. See the system analyst. |

Table 35. ELLA Command Summary

| Format | Description |
|---|---|
| C[LIS] | Requests a chronological listing of the error entries in the order in which they appear in ERRFILE. |
| DE[V], $\left\{\begin{matrix}0\\ address_1[, \ldots, address_5]\end{matrix}\right\}$ | Selects error log entries for display by specifying up to five physical device I/O addresses or (if 0 is specified) specifies that error log entries for all devices are to be displayed. |
| DI[SP] [, interval] | Requests a graphical display of error log entries. |
| DS[PL] | Displays the current state of the four types of boundaries. |
| E[ND] | Terminates ELLA and exits to the monitor. |
| M[OD], $\left\{\begin{matrix}0\\ model_1[, \ldots, model_5]\end{matrix}\right\}$ | Selects error log entries for display by specifying up to five model numbers or (if 0 is specified) specifies that error log entries for all models are to be displayed. |
| R[SET] | Resets all boundary parameters to their default values. |
| SET, LIST, $\left\{\begin{matrix}LP\\ KP\end{matrix}\right\}$ | Reassigns the listing and message output device assignment during execution of ELLA. LP specifies line printer. KP specifies operator's console for the ghost and batch modes and on-line terminal for the on-line mode. |
| SL[IS] | Requests a sorted listing of the error log entries. |
| SU[M] | Requests a summary of the contents of the error file which lists the total number (in decimal) of error log entries for each error type. |
| TI[ME][, begin][-end] | Sets both the date and time boundaries where begin and end have the form<br>[month/day/year][, hour:minute]<br>or<br>[hour:minute][, month/day/year] |
| TY[PE], $\left\{\begin{matrix}0\\ type_1[, \ldots type_5]\end{matrix}\right\}$ | Selects error log entries for display through the specification of error record type codes (see Table 33) or (if 0 is specified) specifies that all types are to be displayed. |

## HARDWARE-ERROR DIAGNOSTIC CALS

The following three CALs are intended for use by the monitor in performing diagnostic functions relating to the hardware-error log and must be issued by a program from the :SYS account. They provide the following services: reading from the hardware-error log, writing to the hardware-error log, and initiation of diagnostic ghost jobs.

These three services are all invoked by a CAL1, 6 fpt instruction; the addressed FPT contains a code and a parameter. The FPT codes and the functions performed are as follows:

| FPT Code | Function |
|---|---|
| 0 | Read Error Log |
| 1 | Write Error Log |
| 6 | Initiate Ghost Job |

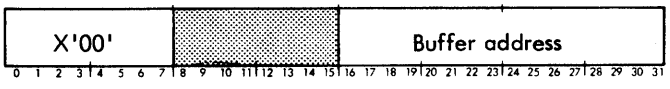
The status of the requested operation is reported via condition-code settings summarized below. (Not all of the status indicated are appropriate to, or reported by, all three CALs.)

| CC1 | CC2 | CC3 | CC4 | Status |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Normal return. |
| 1 | 0 | 0 | 0 | Request denied: insufficient privilege, not in :SYS account, or buffer is not a data page. |
| 0 | 1 | 0 | 0 | Error during operation (Read or Write), or job unknown (Initiate). |
| 0 | 0 | 1 | 0 | Last buffer. |
| 0 | 0 | 0 | 1 | Error log does not yet exist (Read). |

In each case, the calling program must be of privilege level C0 or greater; otherwise CC1 is set to 1 and no action is taken.

### READ ERROR LOG

The format of the FPT for a read-error-log request is

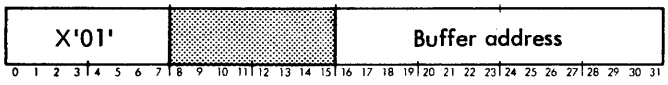



A variable number of words up to a maximum of 256, depending upon the contents of the error log, is read to the area addressed by the FPT. This is a 'destructive' read, returning error-log granules to the monitor's available pool as they are exhausted.

The error-log file is not protected against simultaneous use; thus only one program in the entire system should read this file.

### WRITE ERROR LOG

The format of the FPT for a write-error-log request is

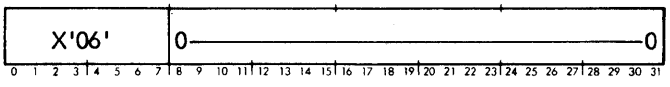



The second byte of the data record addressed by the FPT must specify the number of words to be written, up to a maximum of 253. The first byte of the record should contain a type code.

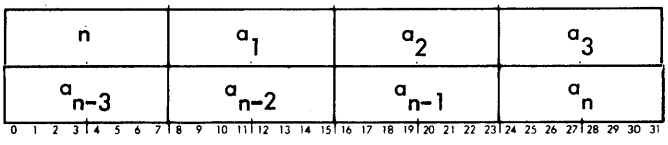
### INITIATE GHOST JOB

The format of the three-word FPT for an initiate-job request is

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.

# 7. SHARED PROCESSOR FACILITIES

## INTRODUCTION

This chapter describes the shared processor facilities of CP-V. These facilities permit the sharing of the code for compilers, assemblers, command language processors, debuggers, libraries, and other programs among all simultaneous users.

Shared processors are not limited to programs provided by Xerox. The facilities may be effectively used whenever a program has a high probability of common usage. Service bureaus, for example, may use the mechanism for proprietary packages. Corporate installations may use the mechanism for programs with a high use frequency.

Most programs may be established as shared processors by naming them at SYSGEN time. This causes the file copy of the program from the :SYS account to be written on the swapping disk during system initialization. The program is then available through high-speed swapping I/O.

The file copy of the program is retained for recovery purposes and may be copied to another account and run as an unshared program under Delta for development and debugging purposes. If the load module in the :SYS account is replaced, the shared copy of the program on the swapping disk is updated to the newer version in the event of a system recovery.

To qualify as a shared processor, a program must meet certain requirements. These requirements are outlined in the remainder of this chapter. The most stringent requirement relates to the single overlay level that is described in the section below titled "Overlay Restrictions".

To avoid confusion, the use of processor names which resemble monitor mnemonics is discouraged. However, if such names are used, the following rules must be followed:

1. If the first three characters of the processor name are JOB, BIN, BCD, EOD, or FIN, then usage of that processor in the batch mode requires that at least one blank appear between the ! and the processor name on the control command which calls the processor.
2. The names of monitor control commands (listed in the CP-V/BP Reference Manual, 90 17 64) are reserved words and must not be used as processor names.

### PUBLIC PROGRAMS

A program whose load module is in the :SYS account is a public program in the sense that it may be called either by a control card containing the ! symbol and the program name, or by an entry of the program name in response to a TEL prompt (!) for commands. Each user of a public program has his own copy of the program.

### SHARED PROGRAMS

Shared programs are called in the same manner as public programs. However, each user of a shared program has his own copy of only the data and DCB portion of that program; the procedure portion is shared by all users associated with the shared program.

There are four distinct kinds of shared programs:

1. Ordinary shared processors.
2. Special shared processors.
3. Shared debuggers.
4. Public libraries.

All shared processors must be built by the batch loader. Ordinary shared processors occupy the same virtual memory as user programs and may not be associated with them.

Special shared processors, shared debuggers and public libraries occupy (and are overlayed in) the special processor area. Figure 11 shows the virtual memory allocation for shared programs that are biased within the special processor area. Shared debuggers may be associated only with user programs; they may not be associated with any other shared processors. Public libraries may be associated with user programs or ordinary shared processors; a public library may not be assocated with a special shared processor. Note

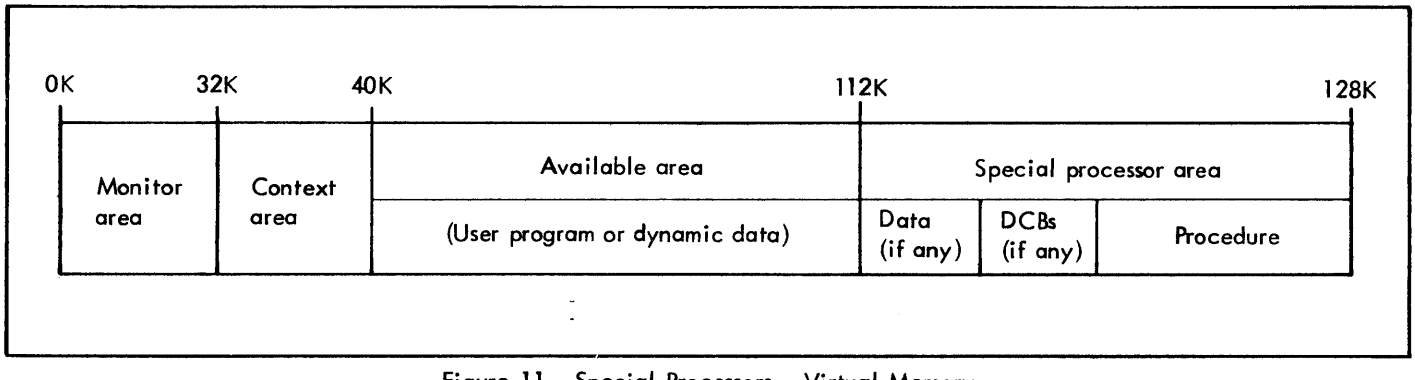



Figure 11. Special Processors – Virtual Memory

that both a shared debugger and a core library may be concurrently associated with a user program. This is possible because the procedure portion of the debugger and the library may be overlayed in the special processor area.

### LOG-ON CONNECTION

Commonly used programs, such as BASIC, may be called automatically by LOGON. The name of the program to be called, which may be either a shared or public program from any accessible account, is established in the user's log-on record by Super. LOGON calls the named program for the user following a successful log-on.

## SHARED PROCESSOR PROGRAMMING

The programming of shared processors may require certain information about the CP-V monitor. This information is outlined below.

### FIXED MONITOR LOCATIONS

For certain purposes, such as the choice of an effective core allocation technique, it is desirable for processors and other programs to be able to identify the monitor in operation, certain critical locations of the monitor, and the location of job information table (JIT). This is accomplished by having locations 2A, 2B, and 4F common to all Xerox monitors. Figure 12 illustrates the contents of these locations.

Location 2A contains a flag that differentiates between an initial boot (nonzero) and a recovery boot (zero).

Location 2B contains three items:

1. Monitor – This field contains the code number of the monitor. The codes are as follows:

| Code | Monitor |
|---|---|
| 0 | None or indeterminate |
| 1 | BCM |
| 2 | RBM |
| 3 | RBM-2 |
| 4 | BPM |
| 5 | BTM/BPM |
| 6 | UTS |
| 7 | CP-V |
| 8 | CP-R |
| 9-F | Reserved for future use |

2. Version – This is the version code of the monitor and is coded to correspond to the common designation for versions. The alphabetic count of the version designation is the high-order part of the code and the version number is the low-order part. For example, A00 is coded X'10' and D02 is coded X'42'.

3. Parameters – The bits in this field are used to indicate suboptions of the monitor. They are meaningful only in relation to a particular monitor. However, the following assignments have been made for BPM, BTM, and CP-V.

| Bit(s) | Meaning |
|---|---|
| 31 set | Symbiont routines included. |
| 30 set | Remote processing routines included. |
| 29 set | Real-time routines included. |
| 28 set | Unused. |
| 27 set | Reserved for Data Management System. |
| 26 set | Reserved. |
| 24 reset; 25 set | Computer is Sigma 6 or 7. |
| 24 set; 25 reset | Computer is Sigma 9. |
| 24 set; 25 set | Computer is Xerox 560. |

Location 4F contains the virtual JIT address right-justified.

### JOB INFORMATION TABLE (JIT)

For each active job, the system maintains an in-core record (job information table) that allows the job to be scheduled and swapped. This job information table (JIT) is the first page of each job, both in core and on the swapping disk, and contains accounting information, memory map, swap storage, addresses, and other information for the job that

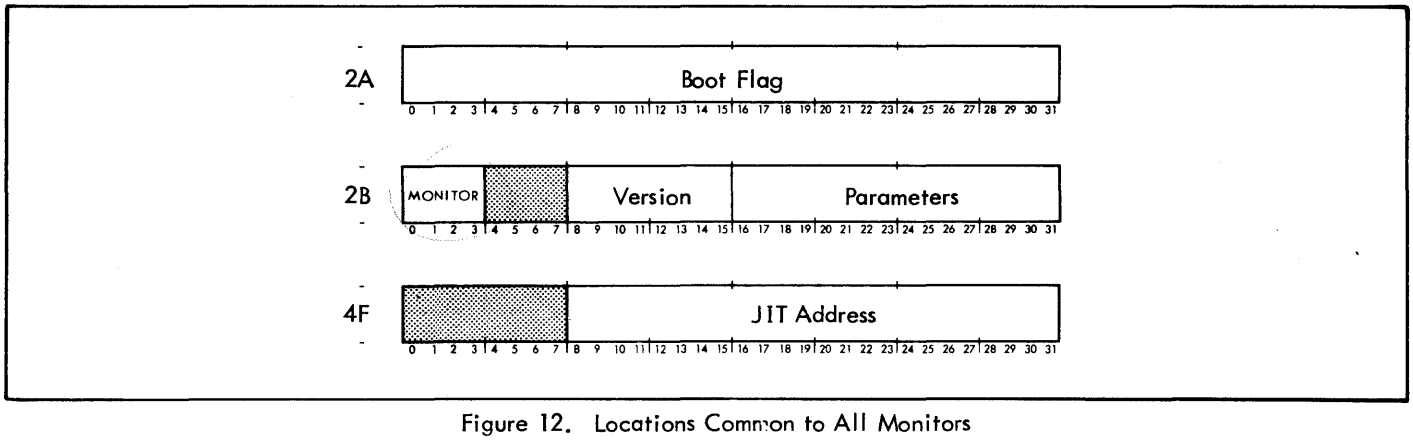



Figure 12. Locations Common to All Monitors

may be of use to a processor.[t] In order to reference these values, the processor should REF the required symbol and then specify that :J0, the JIT definition package, be loaded along with the processor. The entire JIT is available on a read-only basis to all programs including processors. Contents that are particularly useful to processors are given in Table 36. The complete contents are described in the CP-V Data Base Technical Manual, 90 19 95.

Table 36. Partial Contents of JIT

| Location | Size | Contents |
|---|---|---|
| J:JIT | | |
| (bit 0) | 1 bit | Set if the job is on-line and reset if the job is batch. |
| (bit 1) | 1 bit | Set if the job is a ghost job. For example, the meaning of bits 0 and 1 is as follows:<br>00 batch job<br>01 ghost job<br>10 on-line |
| (bit 2) | 1 bit | Set if user is a non-COC on-line user. (Bit 0 also set.) |
| (bits 16-31) | halfword | Job identification number that is guaranteed to be unique to each currently executing job. |
| JB:LPP | byte | Number of printable lines per page (COC). |
| JB:LC | byte | Current print line number (COC). |
| JOPT | word | Peripheral usage flags set by TEL (see section titled "TEL Scan"). |
| J:CCBUF | 20 words | Image of the command line received by TEL. |
| J:USER | 2 words | On doubleword boundary for any use by installation. |
| M:UC | 22 words | Console I/O DCB. |

[t] With respect to accounting, only shared processors are processors, i.e., time spent compiling a COBOL program is accounted under "user time" while time spent in FORTRAN, PCL, etc., is considered "processor time".

## MEMORY CONTROL

No special memory restrictions apply to programs operating as shared processors. In CP-V, as in any other time-shared or multiprogrammed system, prudent use of memory can substantially improve system throughput. Requests for all available memory should be avoided. A request for enough memory to cover typical processing should be made initially, then a request for additional memory should be made during processing if the need arises. Memory should be returned to the system at major changes of control, but the frequent acquisition and release of memory will increase system overhead out of proportion to the gain.

## OVERLAY RESTRICTIONS

Any processor intended for shared use may be created and debugged as an ordinary program. It may be coded in assembly language and debugged under Delta or created in FORTRAN and debugged with FDP. To qualify for inclusion as a shared processor, it must be coded within the following restrictions:

1. Shared processors are allowed only one level of overlay. There is no restriction on the number of overlays but only one of them can be associated at a time.

2. Data cannot be included in overlays; it must be in the processor root.

3. Overlay names are restricted to seven characters or less.

4. All parts of an overlay disappear from core when another overlay is called. (Portions of a previously used overlay are not available when a shorter overlay is invoked.)

5. Shared processors written in FORTRAN must be preceded by some Meta-Symbol code that associates the library and links to the FORTRAN code.

6. The root must be greater than one page in length.

When an overlayed shared processor is requested, the processor root and its first overlay are loaded. Assembled data and DCBs are loaded when the root is loaded. Whenever overlays are not required, memory usage can be held down by declaring an overlay length of zero and issuing a CAL to associate that overlay.

Overlays are declared and associated in the same way as they are for batch programs (CP-V/BP Reference Manual, 90 17 64). TREE command cards and M:SEGLD remain the same. CSECT 2 and 3 are converted to CSECT 1 by CP-V loaders.

Shared debuggers (Delta is the only current example) must have only one page of context and no overlays. They reside in the special virtual area of high memory that is currently fixed in virtual (not physical) size in the highest 16K of virtual storage. They may be any physical size less than 16K including their context page.

## DATA CONTROL BLOCKS

Most processor I/O operations are performed through standard monitor DCBs. For example, source input is normally read by

M:READ M:SI[options]

The standard DCBs are

M:BI

M:CI

M:EI

M:SI

M:C

M:BO

M:CO

M:DO

M:EO

M:LO

M:SO

M:PO

M:AL

M:LL

M:OC

M:SL

M:GO

The default assignment of monitor DCBs is the operational label of the same name (M:DO is assigned to DO, etc.). The assignment of operational labels to devices is shown in Appendix A. The default assignments for batch operations differ from those of on-line operations. This is done so that a program that writes through LO and reads through SI will automatically use the line printer and card reader for batch operations and the terminal for on-line operations. The logical functions associated with the operational labels are described in the CP-V/BP Reference Manual, 90 17 64.

Details concerning input buffers, error handling, and so on are specified as parameters in a read or write call. Parameters associated with files and devices are specified by the ASSIGN (batch) or SET (on-line) control command.

A processor may construct its own DCBs by means of the M:DCB procedure. However, processors are not required to construct DCBs. DCBs not constructed by a processor will be constructed by the loader. Standard DCBs constructed by the loader occupy 51 words and are connected to a device either by the loader or by an on-line user by means of special terminal commands. The M:DCB procedure must be used if optional parameters such as read or write accounts exceed the allocation of the standard DCBs (Table 37).

DCBs are also provided in library form and may be explicitly called during a load. The sizes of these DCBs are shown in Table 37.

Processors may use nonstandard DCBs, if necessary. Nonstandard DCBs are constructed by the loader if not constructed by the processor. They must be explicitly connected to a device either by an M:OPEN call in the processor or by a SET command issued by an on-line user since no default assignment via operational labels is provided.

It is common practice for a processor to obtain source input through M:SI, to print a source listing through M:LO, and to print diagnostic output through M:DO. However, processor I/O operations are complicated by the fact that an on-line user can connect SI, LO, and DO either to different devices or to the same device (the on-line default assignment for SI, LO, and DO is the terminal). In particular an on-line user may connect two or more of these standard operational labels to the same device. For this reason, processors must take precautions to avoid duplications in printed output. This means that processors must know at all times whether they were called in batch or in on-line mode and what specific device connections have been made for standard DCBs.

Processors may examine DCBs directly to determine when the DCBs are connected to the same device. Fields within a DCB may be referenced relative to the name of the DCB. Fields that may be useful to processors are as follows:

| Field | Use |
|---|---|
| FCD | Bit 10 of word 0 of a DCB. This is the file-closed flag. A 1 means the associated file is open; a 0 means the file is closed. |

Table 37. Standard DCBs

| Name | Device | Name | Account | Pass-word | Expiration Date | Read Accounts | Write Accounts | Execute Accounts | Execute Vehicle | INSNS | OUTSNS | Synonymous Name | Key Buffer | Total Words |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Loader Built DCBs | 22 | 4 | 3 | 3 | 3 | 0 | 0 | | | 4 | 4 | 0 | 8 | 51 |
| 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 | 17 | 17 | 17 | 4 | 4 | | 9 | 8 | 116 |
| 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 | 17 | 4 | 4 | | 9 | 8 | 116 |
| M:SL | 22 | 4 | 3 | 3 | 3 | | | | | | | | 8 | 43 |
| M:AL | 22 | 4 | 3 | 3 | 3 | | | | | | | | 8 | 43 |

| Field | Use |
|---|---|
| TYPE | Bits 18-23 of word 1 of a DCB. These bits specify a code for the type of device connected to the DCB (printer, terminal, card reader, etc.). |
| DEV | Bits 24-31 of word 1 of a DCB. These bits specify an index to the monitor device table. |

Under CP-V, all device assignments are direct. This means that DEV always contains a direct device assignment. A complete layout and description of DCBs is contained in the CP-V/BP Reference Manual, 90 17 64.

The same effect can be obtained by the CORRES device CAL, but the CAL is much slower than the direct comparison. The direct comparison of the combined TYPE-DEV fields is meaningful only if the DCB has been opened. This means that processors must explicitly open DCBs for which device assignments will be tested.

## FILE IDENTIFICATION

All on-line processors use a common format and common character set for constructing file identifiers (fid). The standard format is

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

where name, account, and password consist of character strings with maximum lengths of 11, 8, and 8, respectively (name has a maximum of 13 characters for CCI, Edit, and PCL and a maximum of 10 characters for Link and Load). Any of the following characters may be used:

A-z a-z 0-9 ␣ $ * % : # @ -

Lowercase alphabetical characters are not available on all terminals (e.g., Teletype Models 33 and 35). If lowercase letters are sent to these terminals, they are printed in upper case.

Account and password are optional. If account is omitted, the log-on account is the default account. If password is omitted, no password is required to access the file.

### TEL SCAN

A processor call entered through a terminal via TEL has the form

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

where

lm is the name of the processor and is a file identification (fid). Account :SYS is assumed.

sp specifies a source program and may be either a file identification (fid) or a terminal identification (ME).

ON indicates that ROM output is to be on a new file.

OVER indicates that ROM output is to be over an existing file.

rom specifies that the relocatable object module produced by the processor is to be directed to a specified file (fid). If no file is specified, output is directed to a special file that may be subsequently referenced by a dollar sign.

list specifies that a file (fid), a line printer (LP), or the terminal (ME) should be used for listing. If list is not specified, no listing output is produced.

These specifications are implicit ASSIGN and SET commands for the DCBs M:SI, M:GO, and M:LO. A processor call causes the specified processor to be executed with M:SI DCB input from the file sp. Processor output through M:GO DCB is placed in the file specified by "rom" and listing output (M:LO DCB) is directed to the file or device specified by "list". Processor calls are interpreted by TEL.

Parts of a processor call may be enclosed in parentheses. TEL does not do anything to these parts of a processor call. However, the processor may examine these and other parts of the command line that is in its JIT buffer (J:CCBUF).

Processors may reside in storage in three forms:

1. System swap storage contains absolute shared copies of frequently-used processors. These copies can be located and loaded quickly. The absolute shared processor file is created during system initialization and contains reentrant processors that are shared among all concurrent users.

2. The :SYS account may also contain copies of processors in load module form. Processors in this form cannot be loaded as quickly as absolute processors, but the :SYS account may be useful during processor construction, debugging, and extension. Public programs in the :SYS account may be called by entering their names in TEL commands or on control cards.

3. A user may store his own processors or his copies of system processors in his own files (account). A processor stored in a user's file area is identified by its file name and may be called by the RUN command in batch or START command in on-line operations.

When TEL encounters a processor call, it issues an exit CAL specifying the requested processor. The monitor routine STEP checks to see if this user has any processor restrictions. If the user is not restricted from using the requested processor, STEP checks to see if the processor is a shared processor. If it is shared, STEP checks to see if the processor is in core. If it isn't in core, STEP loads it into core. If the processor is not shared, STEP searches the :SYS account and loads the processor from there. If the processor cannot be found, an error message is sent to the terminal. Before control passes to the processor, TEL checks the parameters of the processor call for correct syntax and for existence of the "sp" file and a "rom" or "list".

TEL sets and resets bits in JIT to correspond to the commands LIST, DONT LIST, etc., and to the initial occurrence of assignments in the command string. One JIT word (JOPT) contains a bit for each option that can be specified for a processor. The options and their corresponding bit assignments are as follows:

| Identifier | Bit | Set | Reset |
|---|---|---|---|
| LO | 31 | LIST | DONT LIST |
| BO | 30 | Unused | Unused |
| GO | 24 | OUTPUT | DONT OUTPUT |
| DO | 23 | COMMENT | DONT COMMENT |

The underlined values are default values. If a SET command is issued for the corresponding DCB, or the list output or binary output fields are specified in a TEL command, the corresponding bits are set. Each processor must assign meaning to the bits and interpret them. Unassigned bits are available for future use. Checks of these bits should be made on each write command since TEL allows on-line users to interrupt the processor and turn on or off the LO, GO, and DO devices.

Each processor should establish conventions to maintain orderly output when two or more DCBs are connected to the same device. The usual convention is that if diagnostic

output has been written via M:LO, and M:LO and M:DO are connected to the same device, then the diagnostic output should not be written via M:DO. The following example illustrates some of the special cases that processors should consider:

1. M:SI, M:DO, M:LO connected to the same device (the input line should not appear three times).

2. M:DO connected to a device that is different from SI and LO (the diagnostic comment should probably be printed beneath the line in error).

3. M:SI and M:DO connected to a Teletype (processors may or may not want to type a line in error).

Processors may read each input image via the M:SI DCB. The last record of the sp will cause an end-of-data abnormal condition (see the CP-V/BP Reference Manual, 90 17 04 for a description of abnormal conditions). To obtain control of an error or abnormal condition, a processor must issue the M:SETDCB command and/or include error and abnormal exits in its read and write CALs. Since source input may come from a Teletype (sp = ME), processors must be able to handle Teletype input. The problems associated with Teletype I/O are discussed in the section on terminal I/O.

## CCI SCAN

On transferring control to a user's program or to a processor, the monitor communicates the TCB address via general register 0. Processors may fetch the card image of the command that called them by reading through a DCB connected to the C device.

When running in batch mode, the processor must read the C device once to clear the control command. The command is transferred to the user's buffer to allow the user's program to examine parameters.

## TERMINAL I/O

An on-line user may direct output to his Teletype at any time during execution of a processor. Similarly, portions of the input to a processor may come from a Teletype. In general, Teletype I/O is the same as other I/O in its use of M:READ and M:WRITE operators and the standard abnormal and error situations. However, Teletype I/O has some features that are significantly different from those for other devices. Some of the differences require special attention by processors, but the interface is designed in such a way that processors will not have to know whether or not I/O operations are via Teletype, providing they observe certain conventions. On terminal I/O, like all I/O, the user should note that byte displacements in the DCB remain in effect until replaced, once they have been given. The special problems associated with Teletype I/O are outlined in the following paragraphs.

### END CHARACTERS

On input from a Teletype, each record read is terminated by an end character (CR, FF, LF, RS, US, FS, GS). The end character, if any, is included in the actual record size (ARS) count reported in the DCB (bits 0-14 or word 4). Each processor must interpret the different end characters. Processors do not have to know that input is via Teletype, provided they treat these characters as terminators and use ARS to determine the actual record received.

Source files for all processors, including those in batch operations, may have been prepared on-line. Since records prepared on-line are variable length, it may no longer be assumed that input records are 80-byte card images.

All characters received from terminals, no matter of what type, are translated to the standard EBCDIC character set. The hexadecimal codes for EBCDIC characters are listed in Appendix H.

### WRITE OUTPUT

The length of each output line is specified by the SIZE parameter in the M:WRITE procedure call. It is terminated only by the character zero. That is, the user may terminate a message with a zero character if he wishes and the COC routines will compute the proper message length. Carriage return or new line characters do not terminate a message.

### CARRIAGE RETURN

A new line or carriage return sequence, as appropriate to the type of terminal, is appended to the character string supplied by each write under the following circumstances:

1. The DCB is not M:UC.

2. The suppress space option is not specified.

Thus, under ordinary circumstances, carriage return characters will be supplied when output consists of one line per write and the DCB is connected to a terminal. By using the suppress space option or by writing through M:UC, the program may supply carriage returns exactly to requirements—either none or several for each write CAL.

### PARITY ERRORS AND LOST DATA

When an M:READ CAL specifies a terminal, any character received with a parity error is replaced by SUB (USASCII code 1A) and the lost data abnormal code (07) is returned to the user if an abnormal address exists. If there is no abnormal address, control proceeds to the CAL plus 1. The line is returned to the user's buffer and the program may expect to encounter the SUB code as it scans.

In designing a response to messages that contain parity error characters, two facts are important:

1. The user has already been informed of the error by the COC routines that echo the exact bits received on the line followed by the # character.

2. If the received image is sent back to the terminal together with an error message, the # character will be printed when SUB codes appear.

In the absence of special considerations unique to the processor, it is recommended that lines received with lost data be sent back to the terminal together with the comment "EH?". This procedure is helpful as an aid in diagnosing faulty terminals and communication lines.

## END-OF-FILE

If the user types the character pair ESC F, an end-of-file abnormal code will be returned to the program reading the terminal at the abnormal address (if there is one). An input line that contains all characters received prior to the end-of-file sequence will also be transmitted to the user's buffer. This line is always terminated with a carriage return which is also sent to the user's terminal. If no abnormal address is specified, the line appears as an ordinary input line. If both bad data and end-of-file occur in the same input, then the bad data is reported.

## OTHER ABNORMAL CONDITIONS

If unknown operations are requested of the COC routines (e.g., write end-of-file), the abnormal code for beginning-of-tape will be returned. If there is no abnormal address, the operation will be ignored.

## FORMAT CONTROL

COC routine action for the various formatting CALs is specified in the CP-V/TS Reference Manual, 90 09 07. It is briefly reviewed below.

It is sometimes necessary to print a line with special spacing or without a carriage return. Processors can obtain vertical carriage control by means of two parameters (SPACE and VFC), both of which can be set by the DEVICE CAL. The SPACE and VFC parameters have the following interpretations for Teletypes.

| Parameter | Meaning |
|---|---|
| SPACE | If this parameter is set and VFC is not on, the number of spaces indicated minus 1 is inserted before each write. Counts of 0 and 1 result in single spacing. |
| VFC | If this flag is set, the COC routines simulate the printer's vertical format control as specified in the first character of the text lines written. The simulation is limited to one of the following cases: |

| Hex. Code | Action |
|---|---|
| C1-CF | COC inserts 1-15 spaces before printing. |
| F1 | COC skips to top-of-page by skipping six lines and printing the heading information followed by the print line. |
| 60,E0 | COC does not insert CRLF after the print line (suppress space). |

For page control, COC routines count the number of lines transmitted to and received from the user's terminal. New page headings are printed for every read or write when the line count exceeds the maximum specified in JIT (via the PLATEN command). New page headings are also printed if the user program issues a PAGE device CAL or if the terminal user types the FF character L$^{c}$ (CONTROL L).

Information in the page heading may be specified by the user by means of the HEADER and COUNT device CALs. Heading information is taken from the DCB through which the read or write was given. Thus, if a write call is issued to a Teletype through more than one DCB, the heading printed depends upon the DCB through which the top line of the page was written. The automatic page heading occupies one line and contains current time, date, user name and account number, user identification and line number, page number, and possibly an administrative message. Headings specified in the DCB of the read or write are produced after the automatic heading with position, text, and page number as specified in the CP-V/BP Reference Manual, 90 17 64. The page count in this heading is that carried in the DCB and is reset with each COUNT device CAL. The page count for the automatic heading is carried in JIT and may be reset via the TEL PAGE command. The automatic heading is suppressed if the page length is less than eleven lines. Headings are also not printed if the automatic page heading is turned off via the TEL PLATEN command.

Tab characters are replaced with an appropriate number of blanks in input lines. Tabs are not required in output lines. However, if a highly formatted output line is sent to the Teletype, the operation will be more efficient — and more satisfactory for the on-line user. Tabs are activated by inserting a tab character (X'05') in the output stream. Tabs may be sent directly to the terminal or simulated by the software as requested by the terminal user who may turn simulation on and off using the sequence ( ) T. When simulated by the software, each tab character in the output stream causes insertion of spaces to move the carrier to the right of the next higher position specified in the DCB.

Simulated tab stops can be set by a processor with the TAB device CAL or by an on-line user (for the M:UC DCB) with the TABS command. Tabs must be specified in ascending order beginning with tab stop position 1. Note that this is different from the line printer tabbing, where the tabs need not be in ascending sequence. Tab stops can be set at any time for any DCB. During output operations, tabs are expanded as specified by the DCB through which the write is issued or, if not specified there, as specified in the M:UC DCB. Tabs typed by an on-line user are simulated at the user's console according to the tab settings in the M:UC DCB.

If the backspace character is typed at the terminal, the character is passed to the reading program. No special action is taken by the COC routines other than that necessary to record current carrier position (which for backspace depends on terminal type). Terminals that have a physical backspace may, at the user's option, use a "backspace-edit" mode for intra-line editing. (Reference: CP-V/TS Reference Manual, 90 09 07.)

A program can request control when the user presses the BREAK key by means of the M:INT procedure. Whenever the user presses the BREAK key, the program environment at the time of the break is recorded in the user's pushdown stack in his TCB. Execution can be returned to the location following the interrupted instruction by execution of the M:TRTN procedure. A program can return break control to TEL by executing the M:INT procedure with a break routine address of zero. The break routine address is checked by the monitor to guarantee that the address lies within the memory allocated to the user. Even if a processor has obtained break control, an on-line user can return execution control to TEL by pressing the ⓒ ⓒ, ⓒ Y, or $Y^C$ keys.

As a safety measure to protect the user against faulty programming in break control routines, the number of times the BREAK key is pressed by a user without intervening characters is recorded. When the count reaches four, control is sent to TEL as if $Y^C$ had been pressed. Thus, the user at the terminal will never find himself locked out. The count of four allows processors (e.g., FDP) to make special interpretations on two and three breaks in a row.

### FILE EXTENSION

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

File extension takes effect at the time CP-V opens system output DCBs. The output DCBs that are affected by file extension are those that are currently assigned to files, although normally assigned to devices. They include: M:LO, LL, DO, PO, BO, SL, SO, CO, AL, EO, and GO.

File extension is discontinued when a file is reassigned with a SET or ASSIGN command or when a file is opened with an OPEN procedure call that specifies an explicit file name. In these cases, a new file is created. Extension of the GO file is terminated following a LINK or RUN command.

### SHARED FILE USE

Shared processors must ensure that temporary files used during operation are distinct for each instance of execution. A common technique for accomplishing this is to append the current users ID, from the right half of the first word of JIT, to the filename when it is created and used. This ID is guaranteed by the system to be unique for all concurrently running batch or on-line programs. A discussion of shared files is contained in CP-V/BP Reference Manual, 90 17 64.

## COMMAND PROCESSOR PROGRAMMING

A command processor is a shared processor which interfaces between the user and that which the user wants to access – the monitor, a processor, or another program. Four command processors are supplied with CP-V. They are LOGON, TEL, CCI, and EASY. CP-V will also support installation-specific command processors. Information about the programming of command processors is outlined below.

Generally, command processors have the same restrictions as listed for shared processors previously. In addition:

1. A command processor may not have any overlay structure.

2. A command processor which resides in the special processor area (above X'1C000') may not have any dynamic data and must be biased at X'1C400'.

3. A command processor must intercept all exits, errors, and aborts from user programs and must clean up correctly. (Special CALs for command processors are listed below.)

4. Command processors should not be given special JIT access. (The special CALs for command processor interface eliminate the need for it.)

5. When programs error or abort, control will be given to the command processor with the following restrictions:

   If the command processor resides in the user program area (X'A000' to X'1C000') or the user program is loaded in the extended mode (X'A000'-X'1FFFF'), the exiting user program will be completely disassociated before associating the command processor, eliminating the possibility of continuation of the job step.

   If the command processor resides in the special area (X'1C400' to X'1FFFF'), has no dynamic data or DCBs, uses only M:UC, M:OC, and M:XX, control will pass to the command processor with the user intact, allowing analysis of the exit and continuation of the current job step.

Command processors may be entered into the system during PASS2 of SYSGEN by using the T, B, G, and C flags of the :SPROCS command. They may also be added to the system, replaced, or deleted from the system via the DRSP processor.

The following capabilities are available to command processors:

1. Interpretive Exit – An interpretive exit is a natural exit CAL (M:EXIT) performed by a command processor with the following register setup required.

   R6, R7 R8 — Contain the TEXTC name of the requested load module or shared processor. A maximum of seven bytes is allowed for a shared processor. If R6 is zero and the command processor is special shared, (biased at X'1C400'), the program is reentered at the point of interruption.

   R13, R14 — Contain the account (in TEXT format) in which the load module resides. :SYS is specified for shared processors.

   R10, R11 — Contain the password in TEXT format. If there is no password, zero should be used.

   R0, R1 — Contain either FDP or DELTA in TEXTC format or a zero. If one of the two debuggers is specified, the interpretive exit is to be taken with the debugger associated.

   The system job step processor, STEP, interprets such an exit as a call on the specified program. It also loads the TEXTC name of the command processor that issued the interpretive exit into R4 and R5. Before a command processor issues an interpretive exit, it must have closed all its DCBs and, in general, have cleaned up.

   The job step processor arbitrarily removes the command processor from the user's virtual map. This means that all data and DCBs are gone.

2. BREAK and CONTROL Y Control – If the terminal user depresses the BREAK key during operation of a processor or user program and that program did not request BREAK control, the program is aborted and the command processor is loaded and entered with bit 30 of J:TELFLGS in the JIT set. If the interrupted program has requested BREAK control, the program's BREAK routine is entered.

   If the terminal user depresses CONTROL Y during the execution of a processor or user program and the command processor is not special shared, the program is aborted and the command processor is loaded and entered. If the command processor is special shared and has no data and no DCBs, the user program is left as is and the command processor is entered. This gives the command processor the opportunity to continue the interrupted program.

   If the terminal user depresses CONTROL Y while a command processor is in control, the event is ignored and the current operation is continued where it was interrupted.

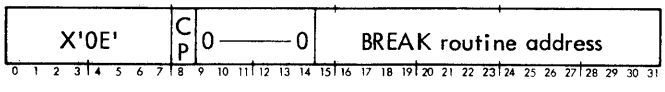
   If the terminal user depresses BREAK while a command processor is in control and BREAK control has not been requested or BREAK control has been reset via the M:INT CAL, the BREAK event is ignored and the command processor is continued where it was interrupted. If a command processor has requested BREAK control, it is interrupted at its BREAK control address.

   The format of the BREAK control CAL is:

   ```
   CAL1,8    FPT
   ```

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

Word 0




If the CP bit is set, the BREAK control routine of the interrupted program is reestablished. This allows a user to depress CONTROL Y while in a program with BREAK control, enter his special shared command processor which remembers the old BREAK control address, and then establish BREAK control for the command processor. If the user wishes to continue, the command processor may set the CP bit and execute the BREAK control CAL before exiting back to the user's program. The BREAK routine address in this case should be the one that was active when the command processor was first entered as a result of CONTROL Y.

3. Exit, Error, Abort CAL, and I/O Abort Control – If any exit or abort condition occurs during execution of a program, the program is aborted and the command processor is loaded and entered. Error conditions are described in four fields of the JIT as follows:

- J:ABC is the address of the word in the JIT that contains the abort code in byte 0 (see Appendix B of the CP-V/TS Reference Manual, 90 09 07).

- ERO is the word offset into the JIT of the word that contains the abort subcode in byte 3.

- J:RNST is the address of the word in the JIT that contains the current run status. Status settings are:

  All zeros – means the job is executing normally.

  Bit 1 – if set, the job is to be errored because of an M:ERR call to the monitor.

  Bit 2 – if set, the job is to be aborted because of an M:XXX call to the monitor.

  Bit 3 – if set, the job is to be errored because of an E key-in by the operator.

  Bit 4 – if set, the job is to be aborted because of an X key-in or a line disconnect.

  Bit 5 – is reserved for future use.

  Bit 6 – if set, the job is to be aborted because a limit has been exceeded (e.g., maximum pages out).

  Bit 7 – if set, the job is to be aborted because of an error (most likely I/O) as specified in J:ABC and ERO.

  Bit 8 – if set, the job is to be aborted because of an illegal trap.

- J:ASSIGN contains the address of the word in the JIT, the rightmost nine bits of which indicate which limit was exceeded. This field is set in conjunction with bit 6 in the RNST field of the JIT. The bits, if set, mean:

  Bit 23 – the maximum disk allocation limit exceeded.

  Bit 24 – the maximum time limit exceeded.

  Bit 25 – the maximum scratch tape limit exceeded.

  Bit 26 – the maximum temporary disk space limit exceeded.

  Bit 27 – the maximum permanent disk space limit exceeded.

  Bit 28 – the maximum diagnostic pages output limit exceeded.

  Bit 29 – the maximum user pages output limit exceeded.

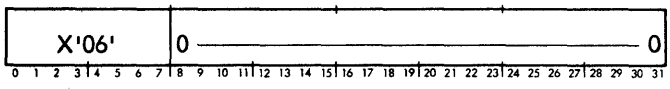
  Bit 30 – the maximum processor pages output limit exceeded.

  Bit 31 – the maximum punch output limit exceeded.

4. CAL Control of JIT Error Condition – This CAL allows control of JIT error conditions without special JIT access. The form of the CAL is:

```
CAL1,4    fpt
```

where fpt points to the word shown below.




The monitor (the ALTCP portion) verifies that the program issuing the request is a command processor through use of UH:FLG. It then sets J:ABC, ERO, byte 0 of J:RNST, and bit 30 of J:TELFLGS to zero. (Bit 30 of J:TELFLGS indicates whether or not the BREAK key has been depressed.) If the program issuing the CAL is not a command processor, control is returned to the user program with CC1 set.

5. Registers – Upon entry to a shared processor from a command processor, the registers must contain the following:

  R0 – the TCB address of the user program.

  R4, R5 – the name of the calling command processor in TEXTC format.

  R6, R7, R8 – the name of the called processor in TEXTC format.

  R10, R11 – the password in TEXT format (zero if none).

  R13, R14 – the account of the called processor in TEXT format.

6. CAL Control of Terminal Modes – Control of terminal modes is provided by a variation of the Change Terminal Type CAL (see the CP-V/TS Reference Manual, 90 09 07).

## PUBLIC LIBRARIES

The system may have several shared public libraries. Each library is a unit tailored to the requirements of the installation. The user associates a public library with his program by specifying the library name (Pi where i = 0-9, J0, or J1) in a LINK or RUN command. The rule governing library units are as follows:

1. Link loads the user data immediately above the area reserved for the library data. Load reserves an entire page for library data.

2. No initialization is provided for this temporary library data either by the loader or by the system. There must be an initialization program if initialization is required.

3. Each library unit must separate data (CSECT0) and program (CSECT1) information into separate assemblies so that separate ROMs will be produced for each.

4. All code must be under CSECTs with protection type 0 for variable data or 1 for procedure and constant data. No DSECT section may be used.

5. The library must be self-contained (i.e., there can be no unsatisfied references). This must be true for the data portion itself and the total library. For example, a FORTRAN I/O library must search the DCB chain rather than make a direct reference to the DCB itself.

### CP-V PUBLIC LIBRARIES

Five public libraries are available to the system programmer. One library (:P1) includes the most commonly required routines from the Extended FORTRAN IV library (about 65 routines). Another (:P0) includes :P1 plus the FORTRAN Debug Package (FDP). A third (:P4) includes :P1 plus the FORTRAN real-time features. The fourth library (J0) contains the JIT definition. Most executing users need only the first library; users who are debugging need the second; real-time users need the third. The fifth library (J1) contains the monitor (M:MON) definitions and is useful only to programs which interface directly with monitor tables and routines.

The entire Extended FORTRAN IV library consists of 252 routines (ROMs) totaling more then ten thousand instructions and over 800 data words.

The package includes more than 350 DEFs. These routines are described in Extended FORTRAN IV Library Technical Manual, 90 15 24, and Sigma 5/7 Mathematical Routines Technical Manual, 90 09 06.

Public library :P1 contains single and double precision trigonometric functions, exponential and logarithmic functions, standard set-up routines, initialization and termination routines, and input/output conversion and transmission routines. Fewer than 1000 words of storage are required for temporary storage by each user of the library. Over 5100 words of library code are shared among all concurrent users.

FDP users require public library :P0 which consists of nearly 1400 words of temporary storage per user; over 10,000 words of code are shared among the concurrent users.

Real-time users require public library :P4 which consists of fewer than 1000 words of temporary storage per user; over 5300 words of code are shared among the concurrent users.

The remaining routines (approximately 190) of the complete FORTRAN library are organized in two ways:

1. They are organized in the :BLIB file as card-image ROM decks that are used by the Link loader to satisfy library references.

2. They are organized in the :LIB/:DIC files as 22 library load modules.

This organization permits rapid loading by the batch loader (Load). Load uses the file :DIC, which consists of a record keyed by each DEF in :LIB and the group number as its value to find the LM names necessary to satisfy references.

Real-time versions of :BLIB, :LIB, and :DIC must be maintained in the real-time system account (e.g., :SYSRT).

One essential monitor subroutine must be added to the standard released library, S:OVRL. It is normally added during the System Generation process but must be remembered whenever a new library is being installed.

The size and description of routines in :LIB are given in Table 38.

### CREATING PUBLIC LIBRARIES

Users may add their own public libraries to meet specific requirements. The necessary procedures are given below.

The procedure for creating public libraries consists of several steps. The desired data and program elements are loaded, and the dictionary for the library (DEFs) is filed for loader use. Next, the procedure is filed so that SYSMAK can place it on swap storage during system initialization. In the process, the program SYMCON is used to retain only those DEFs required in the final linking process, thus saving loader stack search time. Figure 13 illustrates the process of creating a public library.

### LOADING PUBLIC LIBRARIES

Default loading for Link includes the basic FORTRAN public library (:P1) and a search of the system (ROM) library if there are unsatisfied references. This is the same as if the user had specified (:P1) in a RUN or LINK command. If the user has not explicitly asked for :P1 and no reference to 9INITIAL is found, the procedure for :P1 is not associated with the user program execution although the 900 data words remain committed because of the single pass loader operation. Figure 14 is a generalized flow of the LINK process relative to libraries.

Since the batch loader operates in two passes, it makes an explicit association of :P0 and :P1 to a program in absence of other instructions. This process is illustrated in Fibure 15.

Real-time users must specify public library :P4 and the real-time version of the system library. This means that the Link processor requires specification of P4 and inclusion of file :BLIB in the real-time system account (e.g., :SYSRT) as a library file identification. It also means that the batch loader requires specification of :P4 and the real-time system account in the UNSAT list.

Table 38. Routines in :LIB Library File

| Group | Size | Description |
|---|---|---|
| 1 | 96 | Complex double precision mathematical routine drivers. |
| 2 | 72 | Complex mathematical routine drivers. |
| 3 | 92 | Double precision mathematical routine drivers. |
| 4 | 86 | Single precision mathematical routine drivers. |
| 5 | 277 | External revisions of compiler intrinsic functions. |
| 6 | 618 | Complex double precision mathematical routines. |
| 7 | 538 | Complex single precision mathematical routines. |
| 8 | 74 | Double precision mathematical routines. |
| 9 | 104 | Miscellaneous integer functions. |

Table 38. Routines in :LIB Library File (cont.)

| Group | Size | Description |
|---|---|---|
| 10 | 76 | Miscellaneous real functions. |
| 11 | 78 | Logical functions. |
| 12 | 18 | Conversion routines. |
| 13 | 362 | DSINH, DTANH, DASIN, DTAN. |
| 14 | 308 | Miscellaneous nonnumeric functions. |
| 15 | 20 | Overflow and divide check. |
| 16 | 508 | Nonstandard and asynchronous I/O. |
| 17 | 750 | Input and INPUTL. |
| 18 | 160 | Random access. |
| 19 | 514 | Disk buffer. |
| 20 | 102 | Keyed file I/O. |
| 21 | 836 | Namelist I/O. |
| 22 | 938 | Defined file I/O. |

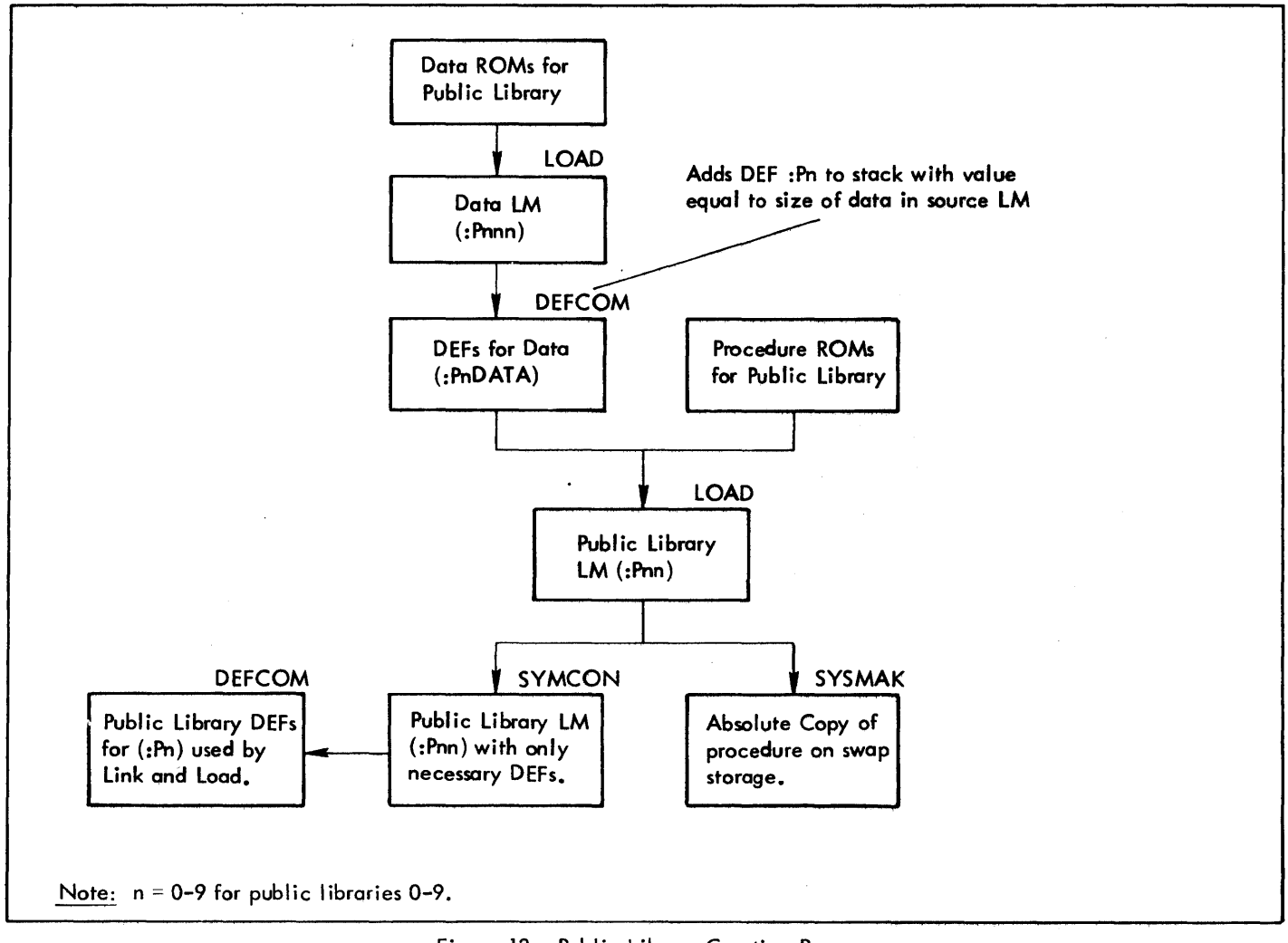



Figure 13. Public Library Creation Process

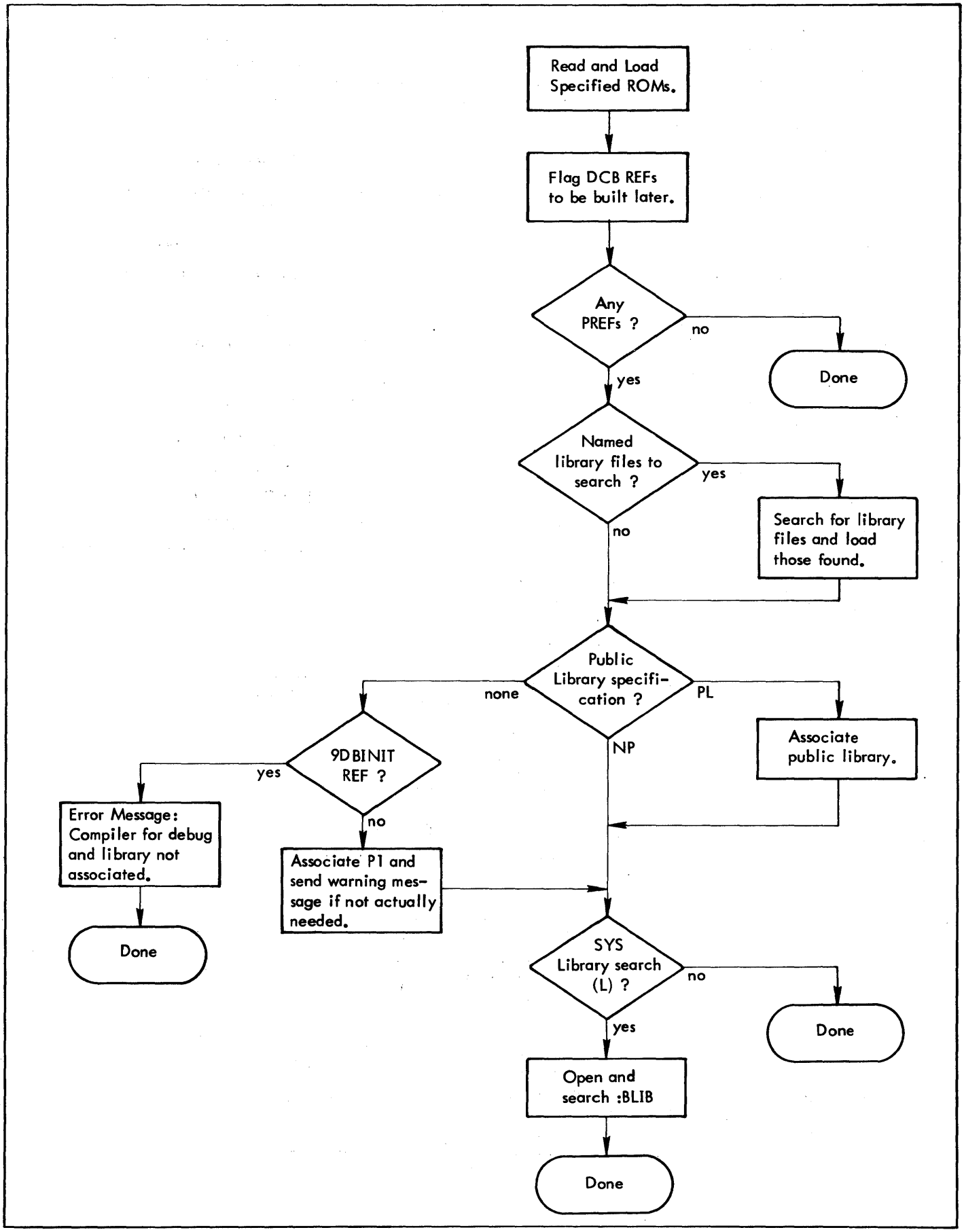
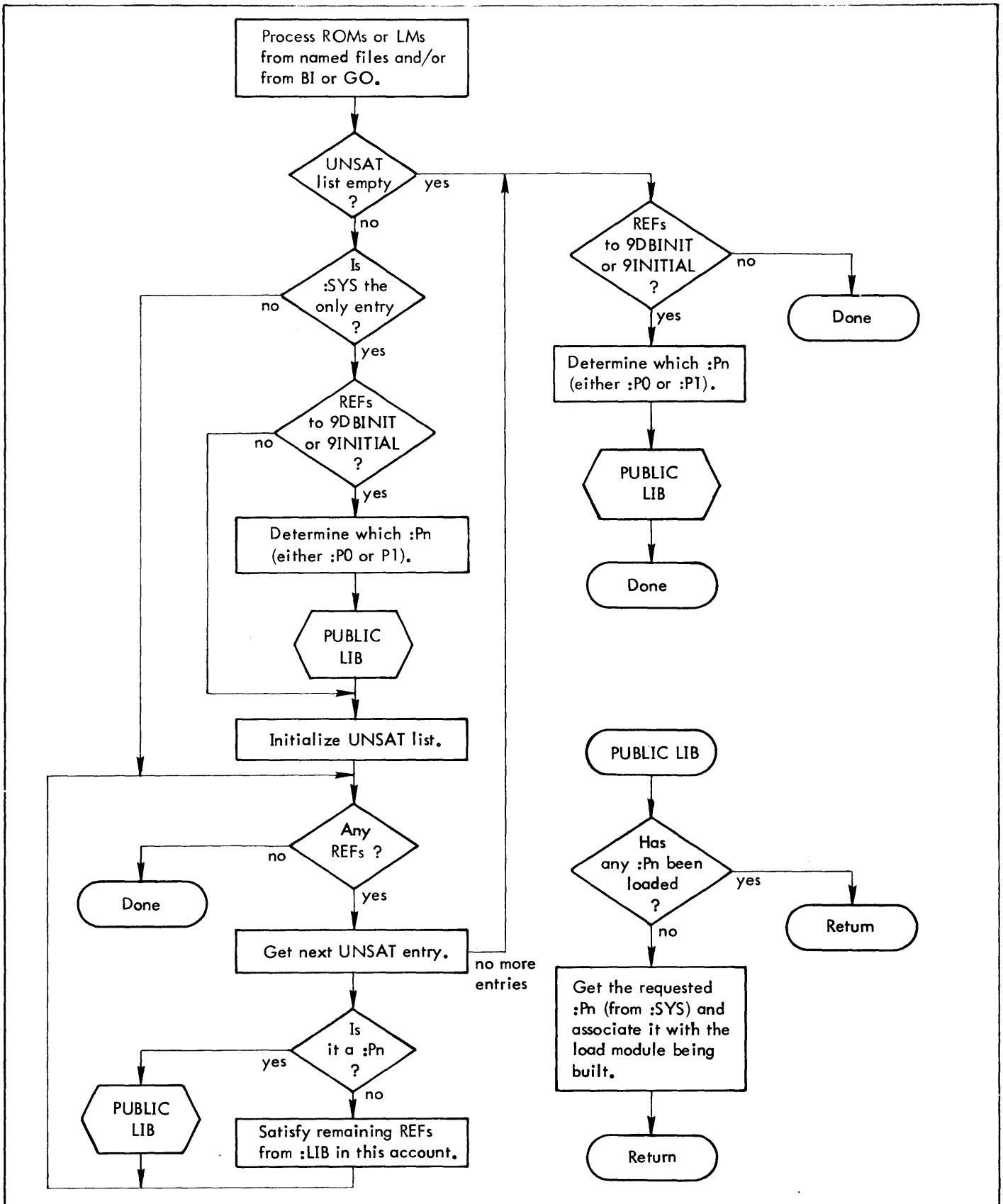



Figure 14. Generalized Library Load Process (Link)




Note: If NOSYSLIB is present on the !LOAD card, the UNSAT list is empty or consists of those sources (accounts and/or :Pn) mentioned under the UNSAT option. If NOSYSLIB is not present, the UNSAT list consists of the above plus the :SYS account (which occurs last).

Figure 15. Generalized Library Load Process (Load)

## SHARED PROCESSOR MAINTENANCE

Development and check out of CP-V systems is simplified through use of DRSP (Dynamic Replacement of Shared Processors). DRSP allows replacement, creation, or deletion of shared processors while the system is operational. The extra processor space in the shared processor tables must be allocated during system generation (PASS2). Processors that are normally invoked following a recovery cycle (ALLOCAT, GHOST1, RECOVERY, and XDELTA) are not dynamically replaceable. DRSP must be run as a shared processor in order to maintain integrity of the monitor's processor tables.

Note: XDELTA (Executive Delta) is an additional debugging aid that is optionally retained at system initialization. XDELTA is described in the Delta chapter of the CP-V/TS Reference Manual, 90 09 07.

DRSP can be run either as an on-line or a batch processor. Input can be either from the command device or from a terminal. DRSP is called on-line by entering the name of the processor as a TEL command.

Example:

```
!DRSP ↵

DRSP HERE

≥
```

The DCBs used by DRSP which may be assigned by the user are

1. M:SI for command language input.
2. M:LL for terminal output.
3. M:SL for listing of input commands during a batch run and diagnostic message output.

### DRSP COMMANDS

The seven DRSP commands are

ENTER
REPLACE
DELETE
LIST
LISTALL
?
END

In the DRSP command descriptions, the term "proname' refers to the name of a processor as found in the shared processor tables. The file specified by proname must be in load module format.

All of the above commands except "?" can be followed by comments, which will be printed as part of the command line during a batch run of DRSP. To add comments, terminate the command with a blank character followed by a period. All characters entered after the period are treated as comments. The comments are terminated by ↵ or end-of-card. Comments cannot be continued to the next record.

**ENTER** The ENTER command is used to enter a new shared processor into the system.

The format of the command is

E[NTER] proname [{FROM / WITH} fid] [,option] [,option] [,option]

where the options are as follows:

[J][S][D][P][M][T][B][G][C] specifies one or more flags to be associated with the processor. The flags indicate the following:

- J processor is allowed to alter the JIT.
- S special shared processor.
- D processor is a debugger.
- P public library.
- M processor allowed maximum memory during execution.
- T command processor accessible by terminal users.
- B command processor accessible by batch users.
- G command processor accessible by ghost users.
- C command processor accessibly by terminal, batch, and ghost users.

If D or P is specified, S is redundant and is assumed. If the C flag is used, the specific flags (T, B, G) are redundant and should not be used. Various combinations of the above are possible up to a maximum of six characters; e.g., a processor that is allowed to alter the JIT and has maximum memory available for execution would be flagged JM. The flag combination PD or usage of the P flag when the processor name is other than :Pnn results in an error message.

PERM specifies that the processor is to be available to users even after a system crash. The processor will be present both in the system account (:SYS) and on swap disk. "Empty" slots must be available in the disk copy of the processor tables. If this option is not used, the new processor version will reside only on swap disk and will be lost in the event of a crash. The version of the processor that will be restored is the version in the system account at the time of the crash.

W specifies that if the proname cannot be entered into the processor table because there are no name slots free, DRSP is to wait until there is a slot available. If this wait option is not specified, the command terminates without entering the new processor.

**REPLACE** The REPLACE command is used to replace an existing shared processor. If this command is used, the previous version of the processor is lost. However, current users continue to use the old copy until they are disassociated from the processor.

The format of the command is

R[EPLACE] proname [{FROM / WITH}] fid [,option][,option][,option]

where the options are as follows:

[J][S][D][P][M][T][B][G][C] specifies flags to be associated with the processor. The option is the same as for the ENTER command.

PERM specifies that the new version of the processor is to be available to users even after a system crash. This version of the processor will be present both in the system account (:SYS) and on swap disk. "Empty" slots must be available in the disk copy of the processor tables. If this option is not used, the new processor version will reside only on swap disk and will be overwritten in event of a crash by the processor version in the system account.

W specifies that if the proname cannot be entered into the processor table because there are no name slots free, DRSP is to wait until there is a slot available. If this wait option is not specified, the command terminates without replacing the old processor.

**DELETE** The DELETE command prevents further user association with a processor. Users associated with the processor when this command is issued will continue to use the processor until they disassociate.

The format of the command is

D[ELETE] proname [, PERM]

where PERM specifies that no new users will ever be associated with this processor (even after a system crash).

**LIST** The LIST command lists the processor name, the name associated with each entry in the processor name table, and the amount of disk space occupied by the processor.

The format of the command is

L[IST] [{proname / #xx[-yy]}]

where

proname specifies an explicit processor name. (The proname M:DUMLM appears many times in the processor tables. If selected, all these entries will be listed.)

xx-[yy] specifies the name table index or a range of name table indexes to be listed.

Initial use of the LIST command with no proname or index specified will provide a list of each processor table entry and its corresponding table index.

**LISTALL** The LISTALL command lists each shared processor name and its entries in the following tables:

| | |
|---|---|
| PB:HPP | Head of the physical page chain. |
| PB:TPP | Tail of the physical page chain. |
| PB:DSZ | Number of data pages. |
| PB:DCBSZ | Number of DCB pages. |
| PH:PDA | Disk address of first procedure page. |
| PH:DDA | Disk address of first page of data and DCBs. |
| PB:UC | Number of users in core using the processor. |
| PB:LNK | Processor number of next overlay. |
| PB:PVA | Virtual page number of first procedure page. |
| PB:HVA | Virtual page number of first unused page. |
| P:SA | Processor flags and start address. |

The format of the command is

LISTALL [{proname / #xx[-yy]}]

where proname and xx[-yy] are as defined in the LIST command.

? The question mark command requests a detailed error message when an error has been noted by DRSP. The command is applicable only for the on-line mode. Its function is described in detail in the section "DRSP Error Messages". The format of the command is

?

**END** The END command terminates DRSP. The format of the command is

END

## DRSP LIMITATIONS AND RESTRICTIONS

The following lists DRSP limitations and restrictions:

1. Only users with a privilege level of C0 or greater are allowed to use the ENTER, REPLACE and DELETE commands. The LIST command requires a privilege level of 80 or greater.

2. There must be sufficient space in the swap disk processor/overlay area to hold the new or replacing entry. This extra space is allocated by SYSGEN PASS2 via a :SPROCS control card.

3. Replaced or entered items must be accessible load modules.

4. Only one level of overlay is permitted in a processor.

5. A processor overlay must be PROCEDURE only.

6. ALLOCAT, GHOST1, RECOVER, XDELTA, M:DUMLM may not be processed with DRSP commands.

7. Overlays for processors cannot be replaced or entered individually.

8. GETs of programs saved with an associated processor most likely will not work if the processor has been changed between SAVE and GET.

9. When replacing the FILL processor a modified procedure is required: Following REPLACE FILL WITH N.A.P., OPTION 1 thru 3, the user has to abort the FILL ghost. This is done via a message to the operator to key in X, id, where id is the SYSID of the FILL ghost which appears when the message 'REQUEST FILL, NO FILL, OR INSTANT SQUIRREL (F, N, S)' is output on operator's console. This will ensure that the FILL copy in the user swap disk area is destroyed and the replaced version of FILL is brought in the next time FILL wakes up.

## DRSP ERROR MESSAGES

The error message structure of DRSP is designed to give a user detailed information when so desired without burdening him with long typeouts when the error is obvious. When running on-line, DRSP will respond to commands in error by typing

EH@ n

where n is the character position at which an error was first detected. If the user requires more information, he responds with a question mark (?). DRSP responds with a detailed error message (see Table 39). If the error is obvious, the user may retype the command (or proceed to the next command). For errors that occur after command syntax is completed, this message changes to

EH

since command character position is meaningless.

In batch mode, the detailed error messages are printed without the interrogative sequence described above.

In addition to error messages, certain other messages are given for information purposes only (see Table 40). No response is expected.

Except where noted, the error condition truncates execution of the requested command.

## DRSP COMMAND SUMMARY

Table 41 contains a summary of commands for the DRSP processor. The left-hand column specifies the format and the right-hand column defines the function.

Table 39. DRSP Error Messages

| Message | Meaning |
|---|---|
| BREAK 50<br>BREAK 51<br>BREAK 52<br>BREAK 53 | User hit BREAK during DRSP execution. The number defines the point at which the DRSP processor exited, as described in the UTS Reliability and Maintainability Technical Manual, 90 19 90. |
| CANNOT OPEN THE FID | DRSP cannot access the load module defined by the fid. |
| CAN'T OPEN M:BO (PERM) | I/O error detected while trying to open the output file in :SYS. The processor is entered/replaced on non-"PERM" basis. |

Table 39. DRSP Error Messages (cont.)

| Message | Meaning |
|---|---|
| DON'T USE COMMAND ON TEL/CCI | ENTER or DELETE commands must not specify the proname 'TEL' or 'CCI'. |
| DRSP I/O ERROR IN READING COMMAND | Error detected in reading DRSP command. |
| DRSP I/O ERR/ABN (CLOSE) | Error or abnormal condition detected at CLOSE of output file. The processor is entered/replaced on non-"PERM" basis. |
| DRSP M:BO ERROR (PERM) | I/O error detected while writing or closing the output file in :SYS. The processor is entered/replaced on non-"PERM" basis. |
| DRSP M:EI ERROR (PERM) | I/O error detected while reading file fid. The processor is entered/replaced on non-"PERM" basis. |
| DRSP M:EI ERROR (WRITESWAP) | I/O error detected while reading fid for writing on the swap disk. |
| DRSP NOT FOUND IN PROCESSOR TABLES | DRSP must be run as a shared processor in order to maintain integrity over the monitor's processor tables. |
| DRSP PROGRAM ERROR (SHOULDN'T HAPPEN) | DRSP detected contradictory conditions during processing. Requires system programmer intervention. |
| ERR MSG NOT FOUND. KEY = xxxxxx | No error message corresponds to the error code xxxxxx generated. Please report this system error. |
| FID IS NOT A LOAD MODULE | Error or abnormal return executed while trying to read the TREE record of the load module specified by fid. |
| FILE STORAGE LIMIT IN SYSTEM ACCOUNT | When writing the load module into the :SYS account for the PERM option, the file space for that account is exceeded. |
| ILLEGAL COMMAND | Command entered is not defined in DRSP. |
| ILLEGAL COMMAND OPTION | An optional parameter typed in the command is not recognized. |
| ILLEGAL INDEX RANGE | Index specified in LIST/LISTALL command not within legal range of processor name table. |
| ILLEGAL LMN (LOAD BIAS CHECK) | Illegal load bias detected when processor written to swap disk. |
| ILLEGAL PRONAME, NOT :PNN FORMAT | A processor flagged as a public library must conform to the name format :Pnn. |
| ILLEGAL PROTECTION TYPE FOR PUBLIC LIBRARY | The load module for a public library must be root only and procedure only. |
| INCORRECT FID | The fid specified exceeds the field maximum for name (15 characters) or account (8 characters) or password (8 characters). |
| INSUFFICIENT MEMORY TO READ MAX RECORD OF FID | DRSP has failed to acquire enough memory to read the largest record of the load module specified as fid. |
| INSUFFICIENT MEMORY TO READ TREE | Memory space available to user is not sufficient to process the load module specified in the ENTER or REPLACE commands. |
| INSUFFICIENT PRIVILEGE FOR DRSP USAGE | The user must have a privilege level of 80 or greater to execute any DRSP commands. |

Table 39. DRSP Error Messages (cont.)

| Message | Meaning |
|---|---|
| INSUFFICIENT PRIVILEGE LEVEL TO PROCESS THIS COMMAND | The user does not have sufficient privilege of C0 to process ENTER, REPLACE, and DELETE commands. |
| INSUFFICIENT SPACE ON SWAP RAD | The disk space allotted for new or replaced load modules is too small for the load module specified. |
| INSUFFICIENT VIRTUAL MEMORY TO EXECUTE DRSP | There are not enough virtual pages to allow DRSP to access the monitor. |
| NO ERRORS | No errors were encountered during command execution. |
| NO PRONAME SLOTS AVAILABLE | The number of extra processor name table entries is exhausted. |
| NO SUCH PROCESSOR | The proname entered cannot be found in the processor tables. |
| ONLY ONE LEVEL OF OVERLAYS FOR SHARED PROCESSORS | When analyzing the load module TREE record, more than one level of processor overlay was indicated. |
| ONLY PROCEDURE IS ALLOWED IN A PROCESSOR OVERLAY | DRSP checks a load module specified as an overlay for procedure only. |
| OVLY LINK EXCEEDS TABLE LIMIT | A system error to be reported. |
| PROCESSOR OVERLAY SLOTS EXHAUSTED | There are not enough empty processor overlay locations in the name table to fill the load module requirement. This check on the name table occurs during the write to the swap disk. |
| PROCESSOR/OVERLAY ALREADY EXISTS | User tried to ENTER a processor or overlay name that exists in the table. |
| PRONAME IS ILLEGAL | Some routine cannot be entered or replaced with DRSP (e.g., XDELTA, RECOVER, GHOST1, ALLOCAT, M:DUMLM). |
| PRONAME REQUIRED | A program must be specified with the ENTER, REPLACE, and DELETE commands. |
| RAD OVERFLOW | Disk space allotted for the shared processors is exhausted. |
| READ ERROR READING FID (COPY) | I/O error detected while trying to read the processor for the copy into the system account. |
| SWAP I/O ERROR (QUEUE) | I/O error detected while writing processor to the swap disk. |
| WRITE ERROR WRITING FID (COPY) | I/O error detected while trying to write the processor into the system account. The processor is entered/replaced on non-"PERM" basis. |
| WRITE RAD FILE I/O ERRORS | I/O error detected while writing the processor to the swap disk. |

Table 40. DRSP Information Messages

| Message | Meaning |
|---|---|
| DRSP HERE | Routine title typed when user first enters DRSP. |
| DRSP INHIBIT SET | Another user is manipulating the shared processor tables and prevents any other user executing the ENTER, REPLACE, and DELETE commands. However, the LIST and LISTALL commands can be executed at any time. |

Table 40. DRSP Information Messages (cont.)

| Message | Meaning |
|---|---|
| fid NEEDS xxxx GRANULES | If DRSP cannot find sufficient disk space in any available slot, it feeds back to the user the number of granules required to enter/replace the new load module. |
| proname REPLACED IN RAD SLOT #x | While exercising the "PERM" option, the proname in slot #x has been replaced by the proname specified in the current command. |
| PRONAME FOUND ON RAD | The proname already exists in the disk version of the processor tables when DRSP tries to execute the ENTER, PERM option. The "PERM" function is completed for the new copy. |
| PRONAME NOT FOUND ON RAD | The proname cannot be found in the disk version of the processor tables when DRSP tries to execute the REPLACE, PERM option. The "PERM" function is completed for the new copy. |
| USERS ASSOCIATED | DRSP attempts to replace TEL or CCI but finds there are users associated. The message is repeated periodically as long as users remain associated. |

Table 41. DRSP Command Summary

| Command | Description |
|---|---|
| D[ELETE] proname ,[PERM] | Prevents further user association with a processor. |
| END | Exits normally from DRSP. |
| E[NTER] proname [{FROM / WITH} fid] [, option][, option][, option] | Enters a new shared processor into the system. |
| L[IST] {proname / #xx[-yy]} | Lists the processor name, the name table index, and the amount of disk space occupied by the processor. |
| LISTALL {proname / #xx[-yy]} | Lists each shared processor name and its entries in certain tables. |
| R[EPLACE] proname [{FROM / WITH} fid] [, option][, option][, option] | Replaces an existing shared processor with a new shared processor. |
| ? | Requests a detailed error message when an error has been noted by DRSP. |

# 8. ON-LINE PERIPHERAL DIAGNOSTIC FACILITIES

## INTRODUCTION

This chapter describes the system facilities that are designed for use by Xerox in the development of peripheral hardware diagnostic programs. The system procedures and the Diagnostic DCB described in this chapter should never be used in any user-written programs. Their description is included in this manual only for completeness of documentation. Any program that uses them may seriously affect the operation and integrity of the system.

The facilities described in this chapter are used in the following types of Xerox processors:

- Functional tests for peripheral devices that isolate hardware problems to the lowest possible level.
- Exercisers that verify that the peripherals are operating correctly.
- Preventive maintenance tests that reduce the amount of time that peripherals are down for repair.

These tests and exercisers may be run at an on-line terminal while the CP-V system is in normal operation.

The facilities described in this chapter include one assembler directive, the special Diagnostic DCB (DDCB), and eight system procedures. The assembler directive allows the user to specify that a control section is to begin at a page boundary. The Diagnostic DCB is a data area that allows the user to issue his own I/O commands.

These eight procedures reside in SYSTEM DIAG along with two other system procedures – M:DPART and D:DRET. (M:DPART and M:DRET are described in the SYSCON chapter in the CP-V/SM Reference Manual, 90 16 74, because they are used by SYSCON.) The e
perform the following functions:

| Procedure | Function |
|---|---|
| M:DDCB | Generates a diag block. |
| M:DOPEN | Opens the devic Diagnostic DCB f |
| M:DCLOSE | Terminates and in with the Diagnostic DCB. |
| M:BLIST | Converts the user's virtual command list into a physical command list and stores the result in the Diagnostic DCB. |
| M:SIO | Initiates the user's I/O. The commands for the I/O are stored in the Diagnostic DCB. |
| M:LOCK | Either locks the user in core or resumes normal swapping for the user. |
| M:MAP | Converts a specified virtual address to a physical address or a specified physical address to a virtual address. |
| M:DMOD# | Obtains the controller model number, the device model number, and the type mnemonic associated with a given device address, and availability information (i.e., device busy, dev. partitioned, controler pant., and DIAG keyin has been performed). |

## RESTRICTIONS

For both security and system performance reasons, there are certain restrictions on the use of the facilities described in this chapter. These restrictions are:

1. The system manager must give approval before the system will process some of the CALs. (Note that M:DDCB does not generate a CAL.) This approval is transmitted to the monitor via the operator key-in.

   !DIAG id

   where id is the diagnostic user's id and identifies the user as the current diagnostic user. This is reset by the monitor between job steps.

2. The M:MAP procedure requires a privilege of A0 or higher. The user is aborted if his privilege level is insufficient.

3. The M:LOCK procedure requires a privilege level of A0 or higher and the user must have been specified as the current diagnostic user via the DIAG keyin or a privilege level of C0 or higher. If one of these conditions is not met, the user is aborted.

4. The M:DOPEN, M:DCLOSE, M:BLIST and M:SIO procedures require a privilege level of A0 or higher and the user must have been specified as the current diagnostic user via the DIAG keyin. If the conditions are not met, the user aborted.

## PSECT DIRECTIVE

The PSECT directive specifies that the control section which follows is to begin on a page boundary. This directive allows diagnosticians to ensure that such things as the Diagnostic DCB and buffers do not cross page boundaries. The PSECT directive is described in detail in the Meta-Symbol/LN, OPS Reference Manual, 90 09 52.

# SYSTEM PROCEDURES

Monitor procedures enable the user's symbolic Meta-Symbol program to request a variety of monitor functions. The on-line diagnostic procedures described in this chapter have the same general format as those described in the CP-V/BP Reference Manual, 90 17 64.

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

```
SYSTEM DIAG
```

This directive defines all of the monitor procedures. The Sigma 6 and 7 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 privilege 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 DIAG and the SYSTEM SIG7 or SYSTEM SIG9 directives should be used. The SYSTEM BPM directive should also be used if any of the procedures described in the CP-V/BP Reference Manual, 90 17 64, are used in the program.

## CREATE DIAGNOSTIC DATA CONTROL BLOCK

**M:DDCB** The diagnostic data control block procedure generates a data area in the user's program that is accessible by the user. This data area must be given a label, the first two characters of which are F: (e.g., F:DIAG).

The Diagnostic DCB (hereafter referred to as the DDCB) must be used when the diagnostician is going to perform his own I/O through use of the diagnostic procedures described in this chapter. In addition to containing standard types of DCB information, the DDCB contains the user's I/O command list. The DDCB format is described in detail at the end of the chapter. Because the DDCB has its own format, the only CALs that may be issued to the DDCB are the diagnostic CALs.

The M:DDCB procedure call is of the form

```
label M:DDCB (DEVICE,name), (CLIST,n)[,(option)]...
```

where

label is a label that begins with the two characters F: and must previously have been declared a dummy section via a directive of the form

```
label   DSECT   1
```

DEVICE,name specifies the device that is to be associated with the DDCB. Name may be specified in one of the following forms:

1. A device type in quotes (e.g., 'CR', 'LP').
2. An operational label in quotes (e.g., 'LO', 'EO').
3. The physical address of the device expressed in hexadecimal (e.g., X'0080', X'0202').

CLIST,n specifies that n words are to be reserved for the user's command list. The maximum value that can be specified for n is 24.

The options are:

SN[,{n / 'serial number'}] specifies one of the following:

1. The number of words (n) to be reserved for serial numbers. The serial numbers will be inserted into the DDCB when the DDCB is opened (M:DOPEN). The maximum value that can be specified for n is 12.
2. The serial number of the volume to be used for input or output. There may be from one to twelve serial numbers of from one to four alphanumeric characters each.

If the SN option is not specified in M:DDCB, then it cannot be specified in M:DOPEN.

ABN,address specifies the symbolic address of a user's routine that is to be used to analyze any abnormal conditions resulting from insufficient or conflicting information. This address remains in the DDCB until it is overridden by an ABN specification in a DOPEN CAL.

The CLIST and SN options produce variable-length parameters which follow the fixed-length parameters in the DDCB. Each variable length parameter entry is preceded by a control word of the following form:

Byte 0 is the code number (X'07' for SN; X'12' for CLIST).

Byte 1 is the code for entry position (X'00' means more parameter entries to follow; X'01' means last parameter entry).

Byte 2 is, for the SN option, the number of significant data words in the parameter entry when serial numbers are specified. Otherwise it is zero.

Byte 3 is the total number of words reserved for the entry, not including the control word (i.e., maximum entry length).

Special Note:

After generating the DDCB, Meta-Symbol resumes assembly in the control or dummy section that was in effect when the M:DDCB procedure reference line was encountered. In order to prevent the statements following the M:DDCB procedure reference line from being assembled in the same section as the DDCB, one of the following is recommended:

1. The control section directive preceding an M:DDCB reference line should be a CSECT, and the DSECT associated with an M:DDCB should precede the CSECT.

2. The statement immediately following an M:DDCB procedure reference line should be either a CSECT or a USECT referencing a prior CSECT.

## OPEN DIAGNOSTIC DATA CONTROL BLOCK

**M:DOPEN** The monitor Diagnostic OPEN routine opens the device specified in the DDCB for diagnostic purposes. The DDCB will not be opened if the information in the DDCB is inaccurate, insufficient, or contradictory. If the M:DOPEN is made with no options specified, the existing parameters in the DDCB are used. If the DDCB is already open when the DOPEN routine is called, an abnormal condition is signaled. If the DDCB is not open when the DOPEN routine is called, the DDCB is reinitialized according to the parameters specified in the M:DOPEN procedure call.

Symbiont devices will only be opened if they have been locked, suspended, or partitioned. Nonsymbiont devices and devices opened with a device address specified (as opposed to device type or an operational label) must be partitioned.* Partitioning is accomplished by using SYSCON or as a result of a previous DCLOSE CAL with the PART option specified.

*and not busy or allocated to an active user.

The M:DOPEN procedure call is of the form

M:DOPEN [*]dcb name, (DEVICE, [*]name),
(STATUS, [*]address)[, (option)]...

where

[*]dcb name specifies the name of the DDCB.

DEVICE, [*]name specifies the device that is to be associated with the DDCB. Name may be specified in one of the following forms:

1. A device type in quotes (e.g., 'CR', 'LP').

2. An operational label in quotes (e.g., 'LO', 'EO').

3. The physical address of the device expressed in hexadecimal (e.g., X'0080', X'0202').

STATUS, [*]address specifies the address of the user's data area where the I/O status is to be stored. The status that is returned is in the same format as for the Error Log (see Appendix E).

The options are:

SN, 'serial number'[, 'serial number']... specifies the serial number(s) of the volume(s) that are to be used for input or output. The serial number may be from one to four alphanumeric characters. A request for the volume(s) will be sent to the operator's console* which the operator responds to with an AVR sequence (e.g., MOUNT key-in).

*when opening to a device type or operational label

NOERR specifies that records of errors from this device are to be suppressed from the Error Log. However, the user has the option of writing records to the Error Log himself, with the Write Error Log CAL.

ABN, address specifies the symbolic address of a user's routine that is to be used to analyze any abnormal conditions resulting from insufficient or conflicting information. If an X'09' abnormal code occurs on the open, this open abnormal address is set into the DDCB and return is to this address. If an address is not present, the user is aborted.

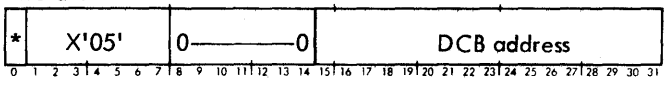
CHAN specifies that the controller is to be reserved for use by this diagnostic program. A controller may be reserved only if it is partitioned.

Calls generated by the M:DOPEN procedure have the form:

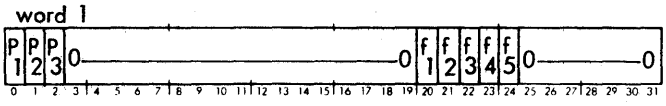
CAL1,6 fpt

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

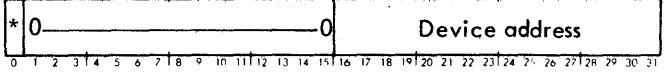
word 0



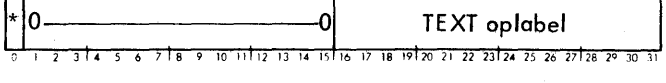

word 1




word 2 - device code (P1)

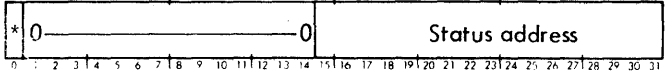



device code (P1) alternate form

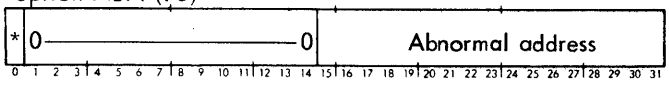



where TEXT oplabel is an operational label in TEXT format.

word 3 - STATUS (P2)



option ABN (P3)




Flags $f_1$ through $f_5$ in word 1 of the FPT have the significance indicated below (when $f_i = 1$).

| Flag | Significance |
|---|---|
| $f_1$ | NOERR was specified. Error records are to be suppressed from the Error Log for this device. |
| $f_2$ | CHAN was specified. The controller is to be reserved. |
| $f_3$ | SN was specified. Serial numbers are present in the FPT (in the format described below). |
| $f_4$ | An operational label was specified. Word 2 of the FPT has the alternate form. |
| $f_5$ | Reserved for future use. |

The format for the SN variable length parameter is identical to that in the DDCB. The variable length parameter entry is preceded by a control word of the form:

Byte 0 – Code number (X'07') identifying the variable-length parameter.

Byte 1 – Code for entry position (X'00' means more parameter entries to follow; X'01' means last parameter entry).

Byte 2 – Number of significant data words in the parameter entry (if SN).

Byte 3 – Total number of words reserved for the entry, not including the control word (i.e., maximum entry length).

If the user does not have at least A0 privilege, the return is to CAL+1 with CC1 set.

## CLOSE DIAGNOSTIC DATA CONTROL BLOCK

**M:DCLOSE** The Diagnostic CLOSE routine terminates and inhibits I/O through the ~~F.DIAG~~ DDCB. I/O cannot be performed through the DDCB until it is opened again. M:DCLOSE allows the user to specify whether or not the device is down (partitioned).

The M:DCLOSE procedure call is of the form

M:DCLOSE [*]dcb name [, (PART / RETURN / SAME)]

where

[*]dcb name specifies the name of the DDCB.

PART specifies that the device associated with the DDCB is to be partitioned from the system resources.

RETURN specifies that the device associated with the DDCB is to be returned to the system resources.

SAME specifies that the device associated with the DDCB is to remain in the same status (partitioned or not partitioned). The default is SAME.

The Diagnostic CLOSE routine reports the status of the device to the operator with the following message:

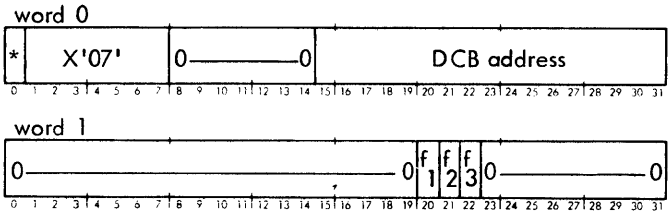
yyndd {PARTITIONED / RETURNED}

where yyndd identifies the device.

Calls generated by the M:DCLOSE procedure have the form

CAL1,6 fpt

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




where

$f_1$ specifies the PART option when set.

$f_2$ specifies the RETURN option when set.

$f_3$ specifies the SAME option when set.

If the user does not have at least A0 privilege, the return is to CAL+1 with CC1 set.

## BUILD COMMAND LIST

**M:BLIST** The monitor BLIST routine converts the user's virtual command list into a physical command list and stores the results in the DDCB. The routine validates that no command crosses a page boundary, that the flags in the command are correct, and that the number of I/O command double-words is less than or equal to 12.

The user's virtual command list must adhere to certain restrictions.

- The list must use virtual rather than physical addresses.
- No input/output command doubleword (IOCD) is allowed to perform I/O across a page boundary or specify a byte count greater than one page (X'800' bytes).
- The number of IOCDs must not be greater than 12.
- The IOCD flags must be set as follows:
  - IUE must always be set.
  - IZC must always be reset.
  - The last IOCD must always have ICE set.
  - Command and data chained IOCDs must have ICE reset.
  - The HTE flag must be consistently set or reset throughout all IOCDs that are data chained together.
- I/O commands which do not cause a transfer of data (e.g., skip file, rewind) must have a valid byte address and byte count. When such commands are used with the ICE flag, the I/O completion interrupt occurs immediately. Hence, the user must handle any desired I/O wait activity independently of any I/O end action.

The user may optionally request that the I/O be started. If this request is made, the monitor will not return control to the user until either the request to start I/O has been rejected, the I/O is complete, or the I/O has timed-out. The AIO, TDV, and TIO status and condition codes are returned in the user area specified by the STATUS parameter of M:DOPEN and in the exact format as for Error Log (see Appendix E)

The M:BLIST

M:BLIS

where

[*]dcb

ADR, [*]
cor

The options

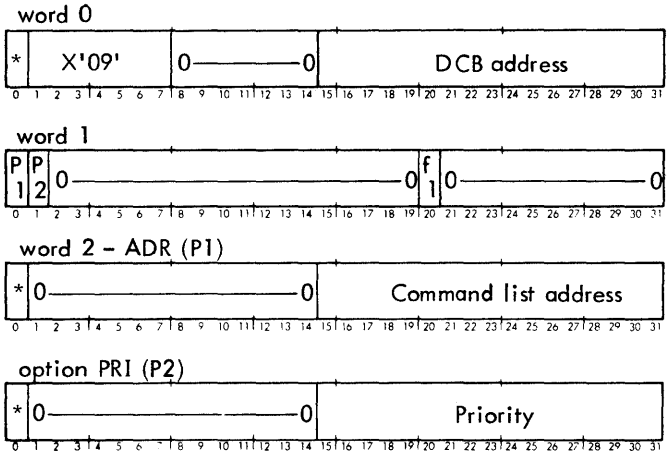
PRI,[*]p
que
X'F
low
hig
the
Th

SIO

Calls generated by the M:BLIST procedure have the form:

```
CAL1,6   fpt
```

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




where $f_1$ is set to one if SIO was specified. Otherwise, it is set to zero.

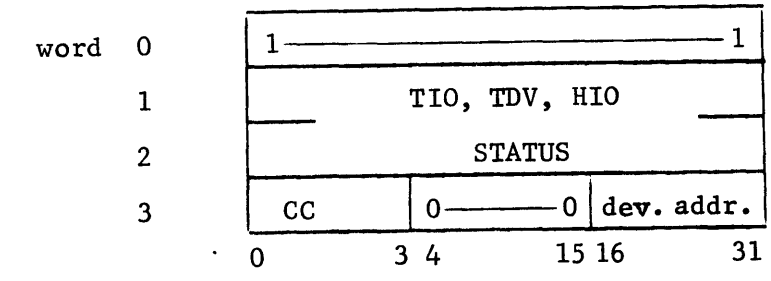
If incorrect or conflicting information exists, the abnormal address specified in the DDCB will be used if it has been specified. If the user does not have at least A0 privilege, the return is to CAL+1 with CC1 set.

## START I/O

**M:SIO** The start I/O procedure call initiates the diagnostic I/O specified in the di

the monitor will not return c

If a TIO, TDV, or HIO request is made, the appropriate request is executed and the status is returned in the STATUS area within the user in the format:




where CC = instructions condition codes
dev. addr. = device address from DDCB

The M:BLIST procedure call has the form:

```
M:BLIST [*]dcb name, ( (ADR,[*]address)[,(option)]... )
                     ( ( {TIO} ) )
                     ( ( {TDV} ) )
                     ( ( {HIO} ) )
```

SIO

Page 112, Column 1

SIO....

TIMEOUT,[*]value specifies the length of time allowed before an I/O timeout occurs. The value is decimal and represents the number of 4.8 second intervals prior to I/O timeout (e.g., value of 2 = 9.6 seconds before timeout). For spindles or disk packs, the maximum value used is one even if a longer value is requested. The maximum value accepted for all other devices is 63 and the minimum is 1. A value greater than 63 will be forced to 63 and a value less than 1 will be forced to one. The default is one.

NOTE: Minimum timeout for an I/O is X * 4.8 seconds to a maximum timeout of (X+1)*4.8 seconds. Therefore, a timeout value of 2(X) may timeout within a range of 9.8 seconds to 14.4 seconds depending on when the timeout count was initiated.

Specific I/O instruction requests (other than SIO) include:

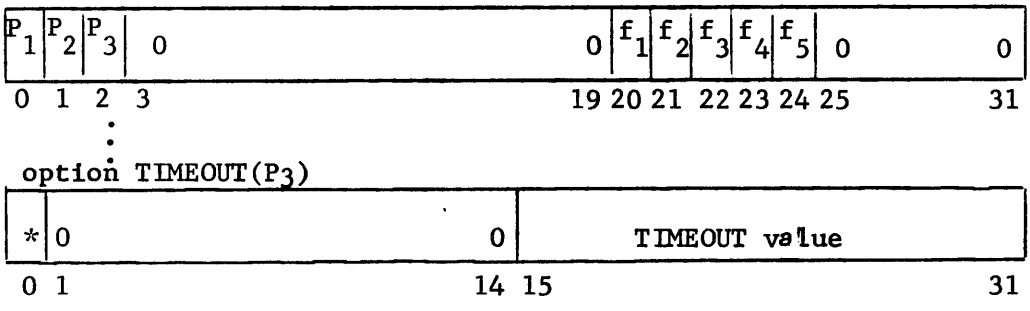
TIO perform a TIO to the device identified by the DDCB.

TDV perform a TDV to the device identified by the DDCB.

HIO perform an HIO to the device identified by the DDCB.

Page 112, Column 2

word 1




option TIMEOUT($P_3$)




where $f_1$ is set to one if SIO..... Otherwise, it is set to zero. $f_2$ is always set to zero when M:BLIST is not for TIO, TDV, or HIO. However, $f_2$ is always set to one when a TIO, TDV, or HIO is requested. $f_3$ is set to one for a TIO request, $f_4$ is set to one for a TDV request, and $f_5$ is set to one for an HIO request. Otherwise, $f_3$, $f_4$, and $f_5$ are set to zero.

## LOCK IN CORE

**M:LOCK** The LOCK routine either locks the user in core or resumes normal swapping for the user. This lock in core reduces the user's chances of being swapped but does not ensure that the user will not be swapped. The user may ascertain whether a swap has occurred since the BLIST CAL by comparing J:NRS (the swap count) in the JIT with the SWAPCT field in the DDCB. (SWAPCT contains the swap count at the time of the BLIST CAL.) The user has not been swapped if the two values are equal. (The external reference J:NRS is satisfied by loading with :J0 from the :SYS account.)

The M:LOCK procedure call is of the form

M:LOCK $\begin{pmatrix}\text{YES}\\ \text{NO}\end{pmatrix}$

where

YES specifies that the user is to be locked in core.

NO specifies that normal system swapping is to resume for the user.

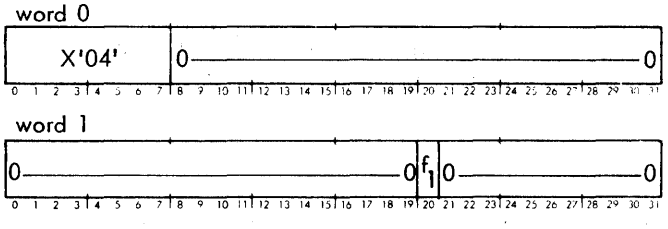
Once a user is locked in core, his size may not change. Therefore, the following services ~~are not allowed~~ may result in size modification, thus, resulting in the user being swapped.

1. Memory management CALs.
2. M:SEGLOAD, M:LINK, and M:LDTRC procedure calls.
3. Associate and disassociate processor CALs.
4. Get page CALs.

Calls generated by the M:LOCK procedure have the form

CAL1, 6 fpt

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




where $f_1$ in word 1 specifies that LOCK in core has been requested ($f_1 = 1$) or that the LOCK is to be released ($f_1 = 0$).

If the user's privilege level is not at least A0, the return is to CAL+1 with CC1 set.

## CONVERT ADDRESS

**M:MAP** The M:MAP procedure converts a specified virtual address to a physical address or a specified physical address to a virtual address. The converted address is stored in general register 8. The M:MAP procedure call has the form

M:MAP $\begin{pmatrix}\text{VTP}\\ \text{PTV}\end{pmatrix}$, (ADR, [*]address)

where

VTP specifies virtual to physical address conversion.

PTV specifies physical to virtual address conversion.

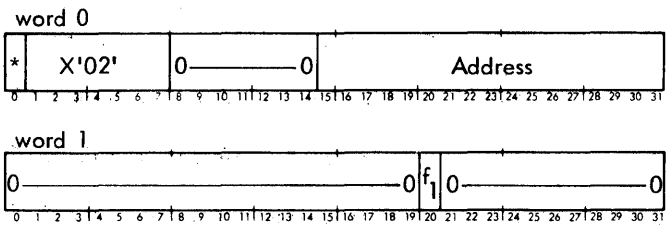
ADR, [*]address specifies the location of the address to be converted.

If the user has been swapped in between issuing a BLIST CAL and issuing a MAP CAL, the address returned from the MAP CAL is invalid. The user has not been swapped if J:NRS in the JIT is equal to SWAPCT in the DDCB. The user may reduce the chances of being swapped through the use of M:LOCK.

Calls generated by the M:MAP procedure have the form

CAL1, 6 fpt

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




where $f_1$ indicates virtual to physical address conversion ($f_1 = 0$) or physical to virtual address conversion ($f_1 = 1$).

If the user's privilege level is not at least A0, the return is to CAL+1 with CC1 set.

## OBTAIN MODEL NUMBERS AND TYPE MNEMONICS

**M:DMOD#** * ~~The DMOD# routine obtains the controller model number, the device model number, and the type mnemonic associated with a given device address.~~ The format of the procedure call is:

M:DMOD# [*]device address

where device address has the form ndd in which n specifies the IOP unit address (the number associated with the IOP letter; see Table B-2 in Appendix B) and dd specifies the device number (see Table B-3 in Appendix B).

**M:DMOD#** The M:DMOD# procedure obtains the controller model number, the device model number, the type mnemonic associated with a given device address, and specific information concerning the device (i.e., device availability, device partitioned, controller partitioned, and

Example:

M:DMOD# X'20F'

The procedure verifies that such an address exists. If no such device address exists, CC1 is set to one. However, if the device address is valid, CC1 is set to zero and the following general registers are set:

R8 is the device model number in hexadecimal (e.g., X'00007122').

R9 is the controller number in hexadecimal (e.g., X'00007120').

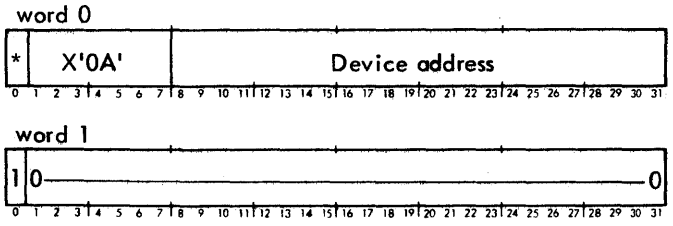
R10 is the type mnemonic in EBCDIC (e.g., X'0000C3D9' for CR), and special information flags formatted as follows: *

In either case, the return is to CAL+1.

Calls generated by the M:DMOD# procedure have the form

CAL1,6 fpt

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




## ABNORMAL CODES AND MESSAGES

The codes and messages for abnormal conditions that can occur when using the on-line diagnostics facilities are listed in Table 42. The abnormal code (bits 0-7) and subcode (bits 8-14) are returned in user's register SR3 and the address of the procedure plus one word (CAL+1) is returned in user's register SR1. (The messages reside in the system error message file, ERRMSG.)

## DDCB

The format for the DDCB is given in Figure 16. Following each format, the parameter fields of the DDCB are described in alphabetical order by their mnemonic. All referenced addresses have word resolution.

Table 42. On-Line Diagnostics Abnormal Messages

| Abnormal Code | Subcode | Meaning of Code |
|---|---|---|
| 09 | 00 | A diagnostic close is attempting to return a nonpartitioned device. |
| 09 | 01 | The device referenced in the DDCB is a nonexistent device. |
| 09 | 02 | The device referenced in the DDCB is currently in use. |
| 09 | 03 | The device referenced in the DDCB is currently in use by a symbiont. |
| 09 | 04 | The DDCB does not contain a command list. |
| 09 | 05 | The command list was invalidated by a swap. |
| 09 | 06 | There are more than 12 I/O command doublewords (IOCDs). |
| 09 | 07 | The I/O command list is invalid due to either invalid flags, an invalid TIC address, an invalid user-specified command list address, or insufficient room in the DDCB for the command list. |
| 09 | 08 | An error was found during the BLIST CAL. Either an invalid page was found during physical-to-virtual or virtual-to-physical address conversion, the status address is in error, or the byte count is illegal in the IOCD. |
| 09 | 09 | A buffer crosses a page boundary. |
| 09 | 0A | 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 | The amount of available core is not sufficient to allow the diagnostic program to lock itself in core. |

* bit 0 = 1 device is currently busy with another user.
bit 1 = 1 device is partitioned.
bit 2 = 1 device's controller is partitioned.
bit 3 = 1 DIAG keyin has been made by the operator

Table 42. On-Line Diagnostics Abnormal Messages (cont.)

| Abnormal Code | Subcode | Meaning of Code |
|---|---|---|
| 09 | 0C | The requested controller is not partitioned. |
| 09 | 0D | The device specifically requested on an open is not partitioned. |
| 09 | 0E | A MAP CAL error occurred due to an invalid page number during a physical-to-virtual or virtual-to-physical address conversion. |
| 09 | 0F | Monitor buffer space (MPOOL) is unavailable for processing the command list. |
| 09 | 10 | Hand coded TIO, TDV, HIO type FPT does not have $f_3$, $f_4$, or $f_5$ set to one when $f_2$ is one. |

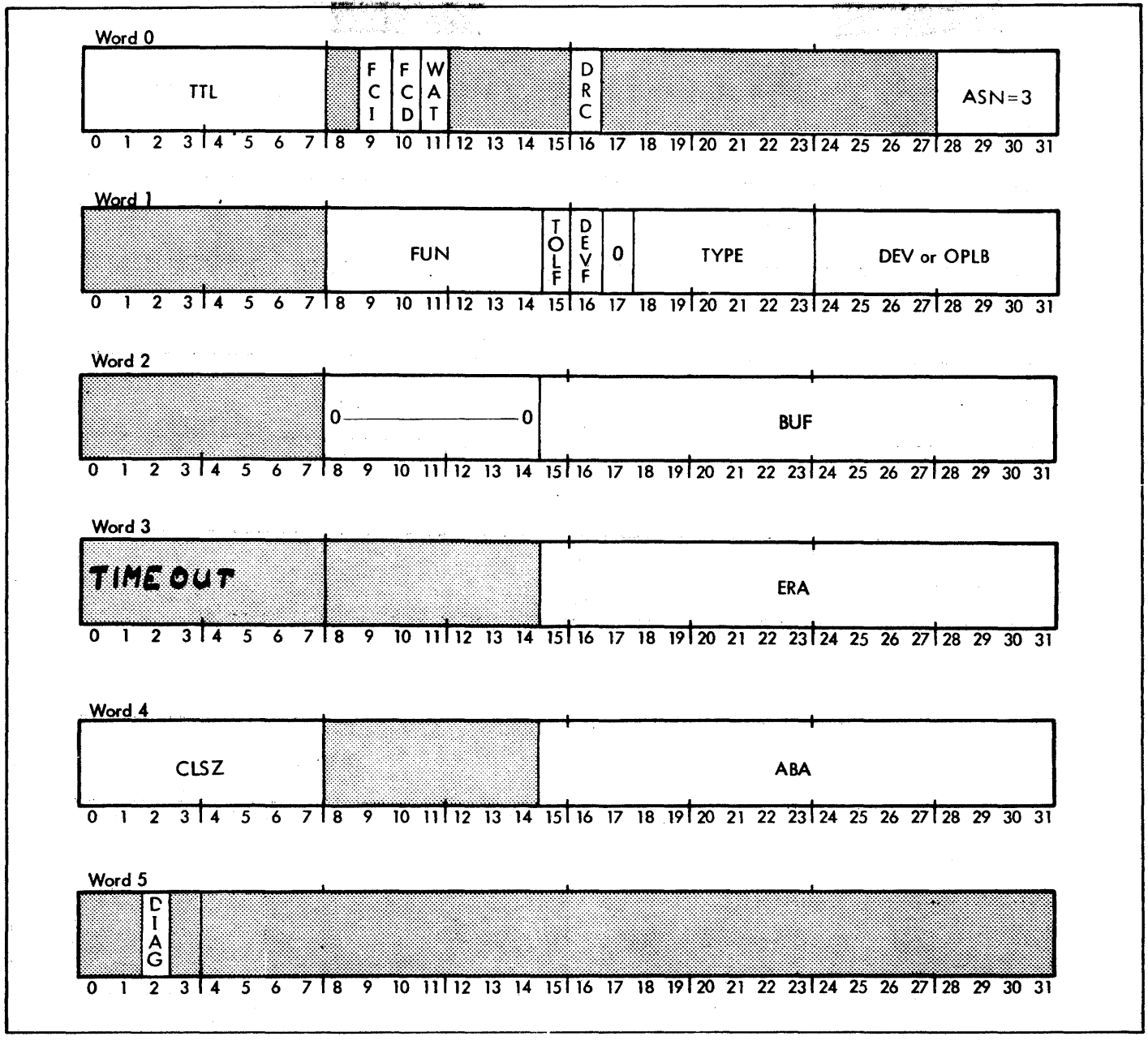



Figure 16. Format of the DDCB

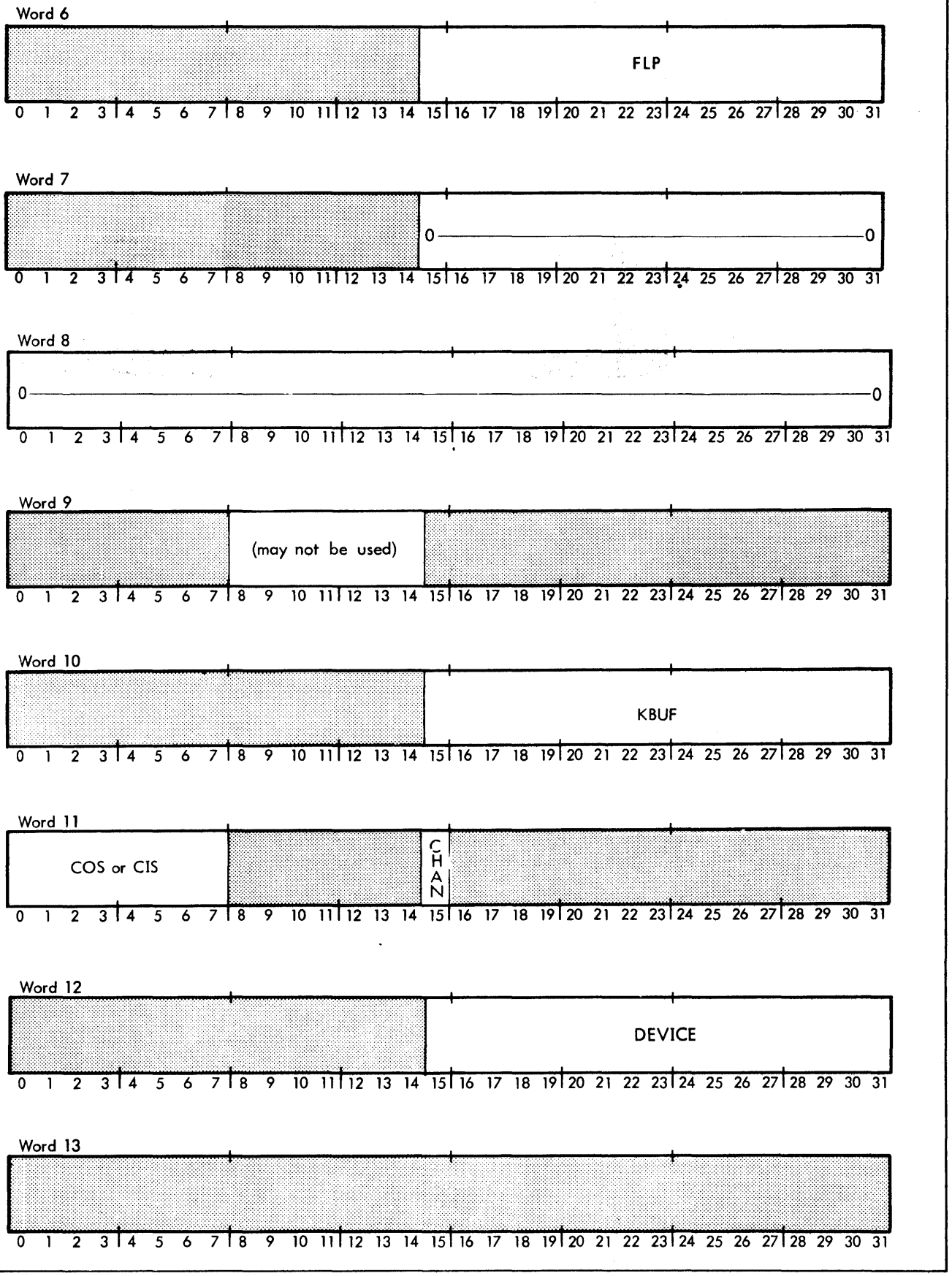



Figure 16. Format of the DDCB (cont.)

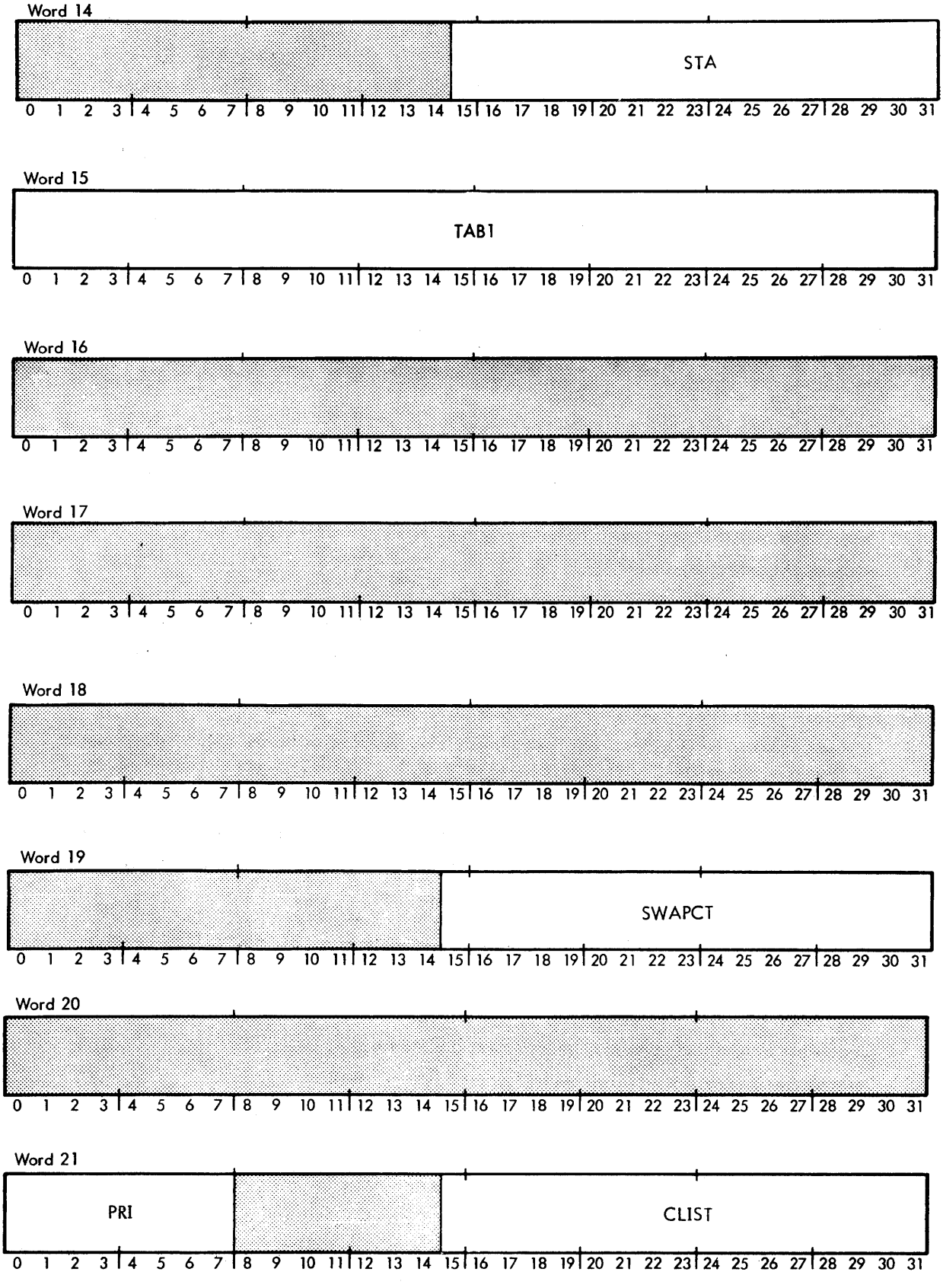



Figure 16. Format of the DDCB (cont.)

| FIELD | DESCRIPTION | WORD |
|---|---|---|
| ABA | Contains the address of the user's routine that will handle abnormal conditions resulting from insufficient or conflicting information. | 4 |
| 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 |
| BUF | Contains the address of a monitor MPOOL buffer used when processing the user's command list. | 2 |
| CHAN | Is the controller reservation flag (0 = no, 1 = yes). | 11 |
| CIS | Contains the relative position of the serial number (in the SN list) of the magnetic tape reel used for current file input. | 11 |
| CLIST | Contains the virtual address of the physical command list in the DDCB. | 21 |
| CLSZ | Contains the number of words in the physical command list in the DDCB. | 4 |
| COS | Contains the relative position of the serial number (in the SN list) of the magnetic tape reel used for current file output. | 11 |
| DEV | Contains the DCT index of the device assigned to the DCB. DEV is only meaningful if DEVF is set to one. | 1 |
| DEVF | Indicates whether the DDCB is assigned to a device or an operational label (0 = operational label, 1 = device). | 1 |
| DEVICE | Contains the EBCDIC name specified on the DEVICE option in the M:DOPEN call. | 12 |
| DIAG | Indicates diagnostic device DDCB. | 5 |
| DRC | Is the format control flag and indicates whether (DRC = 0) or not (DRC = 1) the monitor is to do special formatting of records on read or write operations. DRC is always set to 1 in a DDCB. | 0 |
| ERA | Contains the address of the user's routine that will handle error conditions resulting from insufficient or conflicting infomation. | 3 |
| FCD | Indicates whether the DDCB is opened or closed (0 = closed, 1 = opened). | 0 |
| FCI | Indicates whether the DDCB has ever been closed. The flag is set when the DDCB is first closed, and then is never reset (0 = DDCB has never been closed; 1 = DDCB has been previously opened and closed). | 0 |
| FLP | Contains the address of the variable length parameters in the DDCB (called the file list-pointer). | 6 |
| FUN | Contains the device mode function (0 =null, 1=IN, 2=OUT, 3=IN and OUT, 4=INOUT, 8=OUTIN). | 1 |
| KBUF | Contains the virtual address of the user's command list. | 10 |
| OPLB | Contains the OPLB table index of the operational label assigned to the DDCB. OPLB is only meaningful if DEVF equals 0. | 1 |
| PRI | Specifies priority of I/O request. | 21 |
| STA | Contains address of user data area used to return I/O status. | 14 |
| SWAPCT | Contains user's swap count at the time a diagnostic CAL is issued. | 19 |
| TAB1 | Contains the physical doubleword address of the command list in the DDCB. | 15 |
| TIMEOUT | contains I/O timeout value from M:BLIST cal, | 3 |
| TOLF | If 1, bits 16-31 of DDCB are TEXT OPLABEL. If 0, DEVF is meaningful. | 1 |
| TTL | Specifies the length of the DDCB in words. | 0 |
| TYPE | Contains the device type code assigned to the DDCB. This field is set whether the DDCB is assigned directly to a device or indirectly through an operational label. | 1 |
| WAT | Is the wait flag and indicates whether (WAT = 1) or not (WAT = 0) WAIT was specified for the I/O. WAT is always set to 1 in a DDCB. | 0 |

# 9. REAL-TIME PROCEDURES

Real-time processing involves reacting to external events (including clock pulses) within microseconds. Selected external events are allowed to interrupt the real-time user's program so that they can be processed at the time they occur. After an interrupt has been processed, control may then return to the interrupted program or may be directed elsewhere.

In CP-V real-time processing, there are three distinct types of interrupts:

1. Real, hardware interrupts.

2. Multiple clock interval interrupts derived through software from a single hardware clock interrupt.

3. User written pseudo-interrupts that are triggered by software rather than by hardware. This type of interrupt is quite useful for interprogram communication and synchronization. Pseudo-interrupts use interrupt addresses X'1000' through X'7FFF'.

Note: Any interrupt connected by real-time procedures must have a hardware priority below that of the I/O interrupt. Note also that the swapper performs I/O at a software priority of X'10'. (This would be a consideration when specifying a priority to be associated with certain real-time I/O requests; e.g., M:IOEX.)

CP-V real-time provides services that allow a user program to connect to and control interrupts, to request interruption at specified clock intervals, and to lock itself into core so that it will not be swapped out until it is ready to be swapped out.

The following terms appear in the discussion of the real-time services:

Disarmed

When an interrupt is in the disarmed state, no signal to that interrupt is admitted; that is, no record is retained of the existence of the signal, nor is any program interrupt caused by it at any time.

Armed

When an interrupt is in the armed state, it can accept and remember an interrupt signal. The receipt of such a signal advances the interrupt to the waiting state.

Waiting

When an interrupt in the armed state receives an interrupt signal, it advances to the waiting state, and remains in the waiting state until it is allowed to advance to the active state.

Enabled

When an interrupt is in the enabled state, it is allowed to move to the active state when the interrupt signal is received provided that it is also in the armed state. If the interrupt is already in the waiting state, it moves to the active state when it becomes enabled, provided that no higher priority interrupt is currently active.

Disabled

An interrupt can undergo all state changes except that of moving from the waiting to the active state when it is in the disabled state.

Active

When an interrupt meets all of the conditions necessary to permit it to move from the waiting state to the active state, it is permitted to do so by being acknowledged by the computer, which then executes the contents of the assigned interrupt location as the next instruction.

Cleared

When an interrupt is changed from the active state to the cleared state, the interrupt states are reset so that the interrupt can be recognized again and the priority is reset to that of the job that was running when the interrupt occurred.

Interrupt Control Blocks (ICBs)

Areas of memory set aside for use by the monitor interrupt processing routines. ICBs are established at SYSGEN.

Interrupt Label

The two-character name of an interrupt. Interrupt labels are defined at SYSGEN.

## INTERRUPT CONNECTION AND CONTROL SERVICES

CP-V real-time provides services that connect interrupts to mapped programs, control the state of interrupts (e.g., trigger, arm, enable, disable), clear interrupts either at time of occurrence or upon completion of processing, and disconnect interrupts that are no longer required. Most of these services are provided through procedures which, except where noted, reside in

SYSTEM RTPROCS

and require real-time privilege (E0 or higher).

## CONNECT INTERRUPT TO GHOST FILE

**M:GJOBCON** The GJOBCON routine associates an interrupt with a load module such that if the interrupt occurs, the designated load module will be put into execution as a ghost job. If the ghost job is already active as the result of a previous interrupt, the interrupt will be ignored. An interrupt occurring while the ghost is asleep (M:WAIT) causes a wake-up event.

The M:GJOBCON procedure call has the form

```
M:GJOBCON (INT, [*]{interrupt}), ──────────┐
                   {'intlbl'  }            │
└──(LMN, 'load module')[, (ACN, 'account')]──┐
└──, (PRIO, [*]priority)]
```

where

INT, [*]interrupt specifies an interrupt address.

INT, [*]'intlbl' specifies an interrupt label. If indirect addressing is used, the label must be in EBCDIC format, right-justified in the word, and preceded by blanks.

LMN, 'load module' specifies the name of the load module to be placed in execution when the interrupt occurs. This will be the name of the resulting ghost job. The name must be seven characters or less in length.

ACN, 'account' specifies the account of the load module and consequent running account for the ghost job. The default is the :SYS account.

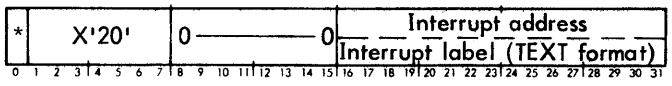
PRIO, [*]priority specifies the execution priority for the ghost job. The default is as follows: if 'intlbl' was specified (via the INT keyword), the default is the SYSGEN-defined execution priority associated within the interrupt label; if an interrupt address was specified (via the INT keyword) and the interrupt is a real interrupt, the default is n-X'4F' where n is the value of the interrupt address (e.g., programs attached to interrupt level X'60' would have a default execution priority of X'11'); if an interrupt address was specified (via the INT keyword) and the interrupt is a pseudo interrupt, the default is the SYSGEN-defined default execution priority for ghost jobs.

Calls generated by the M:GJOBCON procedure have the form

CAL1,5 fpt

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

word 0



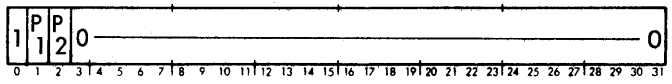
| | | | |
|---|---|---|---|
| * | X'20' | 0 ——— 0 | Interrupt address / Interrupt label (TEXT format) |

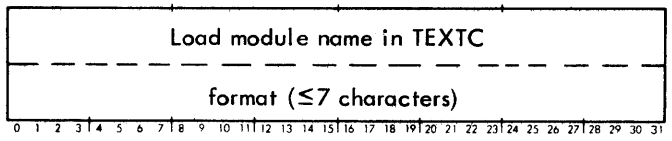
word 1

| | | | |
|---|---|---|---|
| 1 | P1 | P2 | 0 ——— 0 |

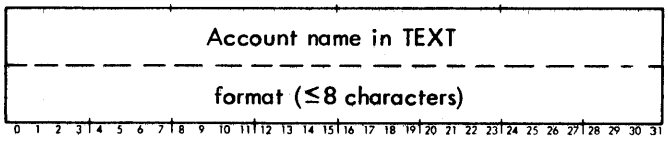
words 2 and 3

| |
|---|
| Load module name in TEXTC |
| format (≤7 characters) |





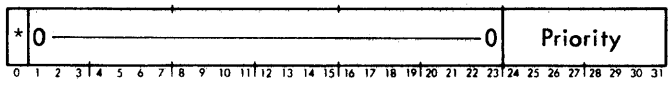
words 4 and 5 (P1)



| |
|---|
| Account name in TEXT |
| format (≤8 characters) |

word 6 (P2)

| | | |
|---|---|---|
| * | 0 ——— 0 | Priority |



Condition code settings resulting from an M:GJOBCON CAL are:

CC1 - set if the user does not have real-time privilege.

CC2 - set if no interrupt control blocks are available.

CC3 - set if the interrupt specified is already connected.

CC4 - set if the ghost already exists. This M:GJOBCON procedure call is ignored.

## CONNECT USER PROGRAM TO INTERRUPT

**M:CONNECT** Any mapped user program with real-time privilege may use this service to establish a connection to an interrupt such that the user program will be entered at the specified address when the interrupt occurs. Interrupts connected in this way report events to the CP-V execution scheduler and therefore permit the entered program to use all monitor services. The connected interrupt will be armed and enabled or disabled as specified by the user.

The M:CONNECT procedure call has the form

M:CONNECT (INT,[*]{interrupt / 'intlbl'}),
(ENTRY,[*]address)[,CLEAR][,MASTER]
[,DISABLE][,PRIO,[*]priority]

where

INT,[*]interrupt specifies an interrupt address.

INT,[*]'intlbl' specifies an interrupt label. If indirect addressing is used, the label must be in EBCDIC format, right-justified in the word, and preceded by blanks.

ENTRY,[*]address specifies the address at which entry is to be made into the user program.

PRIO,[*]priority specifies the execution priority for this interrupt. The default is as follows: if 'intlbl' was specified (via the INT keyword), the default is the SYSGEN-defined execution priority associated with the interrupt label; if an interrupt address was specified (via the INT keyword) and the interrupt is a real interrupt, the default is n-X'4F' where n is the value of the interrupt address; if an interrupt address was specified (via the INT keyword) and the interrupt is a pseudo interrupt, the default is the SYSGEN-defined default execution priority for either on-line, batch, or ghost jobs, depending on the mode in which the job is being executed.

CLEAR specifies that the interrupt is to be cleared immediately upon occurrence and reported to the scheduler. The default is to leave the interrupt active.

MASTER specifies that the user is to be given control in the master mode. The default is the slave mode.

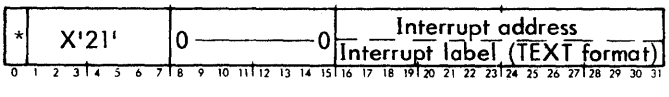
DISABLE specifies that the interrupt is to be connected, armed, and disabled. The default is to arm and enable.

Calls generated by the M:CONNECT procedure have the form

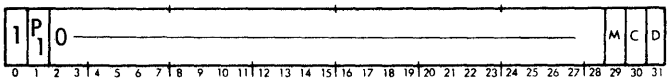
CAL1,5 fpt

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

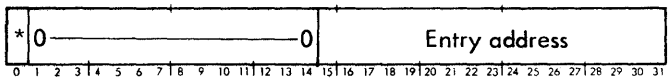
word 0



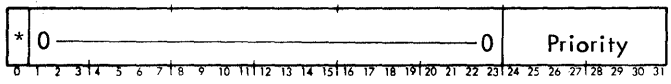

word 1




word 2




word 3 (P1)




where

M=1 specifies MASTER mode.

C=1 specifies CLEAR.

D=1 specifies DISABLE.

condition code settings resulting from an M:CONNECT CAL are:

CC1 - set if the user does not have real-time privilege.

CC2 - set if no interrupt control blocks are available. (Interrupt control blocks are established at SYSGEN.)

CC3 - set if the interrupt specified is already connected.

The environment existing for the real-time program at the time of the interrupt occurrence is saved in the user's TCB before entering the specified interrupt routine. The TCB is identical to the one shown in Figure 5 except that the last word contains the interrupt location rather than a trap location.

## DISCONNECT USER PROGRAM OR GHOST JOB FROM INTERRUPT

**M:DISCONNECT** The DISCONNECT routine releases the specified interrupt if it is associated with the current user. If honored, the M:DISCONNECT procedure disarms the specified interrupt and releases the associated interrupt control block.

The M:DISCONNECT procedure call has the form

M:DISCONNECT (INT,[*]{interrupt / 'intlbl'})

where

INT,[*]interrupt specifies the interrupt address.

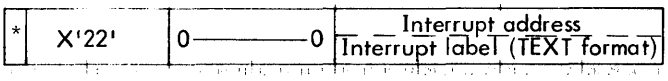
INT,[*]'intlbl' specifies the interrupt label. If indirect addressing is used, the label must be in

EBCDIC format, right-justified in the word, and preceded by blanks.

Calls generated by the M:DISCONNECT procedure have the form

CAL1,5 fpt

where fpt points to the FPT shown below.




Condition code settings resulting from an M:DISCONNECT CAL are:

CC1 - set if the user does not have real-time privilege.

CC2 - set if the interrupt specified is not associated with the current user.

CC3 - set if the specified interrupt is currently active.

## CONTROL AN INTERRUPT

**M:INTCON** This service permits a program with real-time privilege to control the states of interrupts. Interrupts may be armed, disarmed, enabled, disabled, or triggered. If the designated interrupt is a pseudo-interrupt, the action specified does not affect any real hardware interrupt but is instead recorded in the associated interrupt control block. The use of this service does not require that the user issuing the M:INTCON request be connected to the designated interrupt, thus permitting inter-user interrupts.

The M:INTCON procedure call has the form

$$\text{M:INTCON}\ (\text{INT},[*]\{\text{interrupt}\atop\text{'intlbl'}\},\{\text{ARM,ENABLE}\atop\text{ARM,DISABLE}\atop\text{DISARM}\atop\text{ENABLE}\atop\text{DISABLE}\atop\text{TRIGGER}\})$$

where

INT,[*]interrupt specifies the interrupt address.

INT,[*]'intlbl' specifies an interrupt label. If indirect addressing is used, the label must be in EBCDIC format, right-justified in the word, and preceded by blanks.

ARM,ENABLE specifies that the interrupt is to be armed and enabled.

ARM,DISABLE specifies that the interrupt is to be armed and disabled.

DISARM specifies that the interrupt is to be disarmed.

ENABLE specifies that the interrupt is to be enabled.

DISABLE specified that the interrupt is to be disabled.

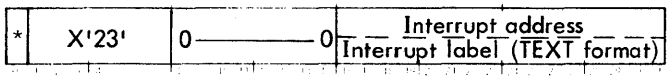
TRIGGER specifies that the interrupt is to be triggered.

Calls generated by the M:INTCON procedure have the form

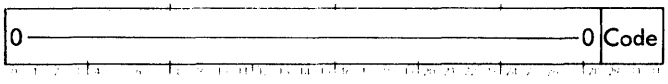
CAL1,5 fpt

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

word 0




word 1




where the 3-bit code has the following meanings:

001 - DISARM

010 - ARM and ENABLE

011 - ARM and DISABLE

100 - ENABLE

101 - DISABLE

111 - TRIGGER

condition code settings resulting from an M:INTCON CAL are:

CC1 - set if the user does not have real-time privilege.

CC2 - set if the designated interrupt is not centrally connected (i.e., no M:CONNECT or M:GJOBCON has been performed on the interrupt). The requested operation is not performed in this case.

## GENERAL INTERRUPT INHIBIT

**M:INHIBIT** This service permits a program with real-time privilege to prevent itself from being interrupted by any higher priority real-time task. Note that this is a software (not hardware) inhibit and applies to both real and pseudo interrupts.

The M:INHIBIT procedure call has the form:

M:INHIBIT {ON / [OFF]}

where

ON specifies that the program is not to be interrupted.

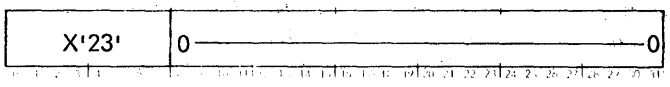
OFF specifies that the program may be interrupted and is the default.

Calls generated by the M:INHIBIT procedure have the form

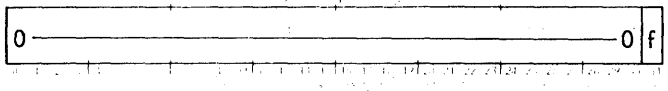
CAL1,5 fpt

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

word 0




word 1




where f specifies OFF if 0; or ON if 1.

Condition code settings resulting from an M:INHIBIT CAL are:

CC1 - set if user does not have real-time privilege.

## RETURN FROM INTERRUPT PROCESSING

**M:INTRTN** This service allows a mapped, scheduled program entered as the result of a centrally connected interrupt or elapsed clock interval to return to the point of interruption. The actual return is to the environment that existed for this program or user when the interrupt occurred even if this user was not in control when the interrupt occurred. The environment that is restored was saved in the user's TCB at the time of interrupt entry.

The M:INTRTN procedure call has the form

M:INTRTN [{LEAVE / ARM[,ENABLE] / ARM,DISABLE / DISARM}]

where

LEAVE specifies that the interrupt is to be left in its current state. LEAVE is the default for this procedure.

ARM[,ENABLE] specifies that the interrupt is to be left armed and enabled. (It is not necessary to specify ENABLE.)

ARM,DISABLE specifies that the interrupt is to be left armed and disabled. (It is necessary to specify DISABLE.)

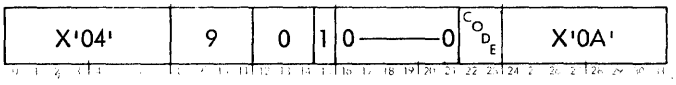
DISARM specifies that the interrupt is to be left disarmed.

None of the above options are recognized when exiting a clock-processing routine (see M:CLOCK below).

Calls generated by the M:INTRTN procedure have the form:

CAL1,9 X'0A'

where the CAL instruction is as follows:




where the 2-bit code has the following meanings:

00 - LEAVE

01 - DISARM

10 - ARM and ENABLE

11 - ARM and DISABLE

When an error condition occurs, the user is aborted with an error code of either A301 or B802 (see Appendix B of the CP-V/BP Reference Manual, 90 17 64).

## QUEUE FOR INTERRUPT

**M:QFI** This service permits the user to suspend execution while awaiting interrupts or elapsed clock intervals assigned a priority higher than the current execution priority. If there are no interrupts connected for this user that satisfy this condition, the user is aborted with a code of B8 and a subcode of 01.

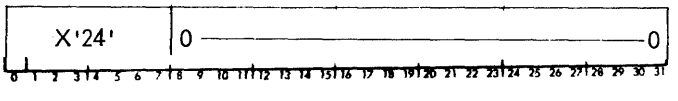
The M:QFI procedure call has the form

M:QFI

Calls generated by the M:QFI procedure have the form

CAL1,5 fpt

where fpt points to the FPT shown below.



### OBTAIN INTERRUPT STATUS

**M:INTSTAT** The service permits any user to query the status of any real or pseudo interrupt location. The format of the M:INTSTAT procedure call is

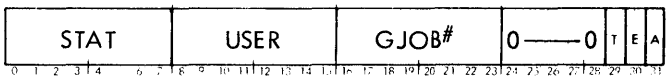
M:INTSTAT (INT, [*]{interrupt / 'intlbl'})

where

INT, [*]interrupt specifies an interrupt address.

INT, [*]'intlbl' specifies an interrupt label. If indirect addressing is used, the label must be in EBCDIC format, right-justified in the word, and preceded by blanks.

The following word of information is returned to the user in general register 8:




where

STAT indicates the status of the task associated with the interrupt location:

| STAT | Meaning |
|---|---|
| X'80' | Task is active. |
| X'40' | Task is asleep or queued for interrupt. |
| X'20' | Task is waiting for I/O completion. |
| X'10' | Task is blocked and waiting for a resource. |
| X'01' | Specified interrupt is not currently associated with any user (i.e., inactive). |

USER is the user number of the user program which issued the M:CONNECT or M:GJOBCON.

GJOB# is the user number of the ghost job (if it is active) which will be entered upon the occurrence of the interrupt. If the ghost job is not active, GJOB# contains zero.

T specifies that the interrupt has been triggered, if set to one.

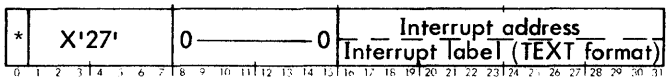
E specifies that the interrupt is enabled, if set to one.

A specifies that the interrupt is armed, if set to one.

Calls generated by the M:INTSTAT procedure have the form

CAL1,5 fpt

where fpt points to the FPT shown below.




Condition code settings resulting from an M:INTSTAT CAL are:

CC2 - set if the specified interrupt is not currently associated with any user (i.e., inactive). (The STAT field of general register 8 is set to 01.)

## LOCK IN CORE SERVICE

**M:HOLD** Many real-time applications require that a program be held in core while various forms of special I/O occur. Since the CP-V scheduler will swap users as conditions require in order to keep as many executable users in core as possible, it is necessary for those real-time programs which require extended core residency to identify themselves via the M:HOLD service.

The format of the M:HOLD procedure call is

M:HOLD {ON / OFF} [, PURGE]

where

ON specifies that swapping is to be prevented for this user (i.e., the user is to be locked in core).

OFF releases the hold.

PURGE specifies that the user's pages should not be released if the user exits (or aborts) while locked in core.

Condition code settings resulting from an M:HOLD call are:

CC1 - set if user does not have real-time privilege.

CC2 - set if there is not enough room left in core to hold the routine that allocates new disk space or the routine that communicates with the symbiont ghost. Any monitor service that invokes these routines must not be used in this case.

Restrictions:

1. The user must have real-time privilege.
2. All memory management services which increase this user's size and the M:LINK and M:LDTRC services will not be allowed once the user is held in core.

3. Monitor services may be further restricted if CC2 is set. All services requiring the allocation of new disk space or communication with the symbiont ghost are prohibited for the user.

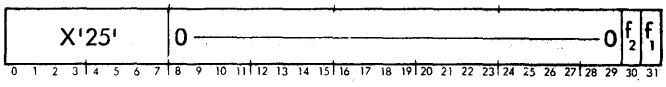
It is important to note that any program using M:HOLD should take exit control to cover abort conditions because if an abort or exit occurs while the user is locked in memory, the memory involved will not be released if PURGE was specified.

Calls generated by the M:HOLD procedure have the form

CAL1,5 fpt

where fpt points to the FPT shown below.

| | | | |
|---|---|---|---|
| X'25' | 0 ——— 0 | $f_2$ | $f_1$ |



where

$f_1$ specifies ON if set to 0 or OFF if set to 1.

$f_2$ specifies PURGE if set to 1.

## CLOCK SERVICE

**M:CLOCK** This service permits a user with a privilege level of 80 or higher to request entry at a specified address when a specified time interval has elapsed. The format of the M:CLOCK procedure call is:

```
M:CLOCK (ENTRY, [*]address), ──────────────┐
└── {CANCEL                                  │
    {(INTERVAL, [*]units)[, (PRIO, [*]priority)]──┐
└── [, ONESHOT][, MASTER]}
```

where

ENTRY, [*]address specifies the address at which the user is to be given control when the specified interval has expired. The environment existing for the user at the time of the interval expiration is saved in the user's TCB as described under M:CONNECT.

CANCEL causes any outstanding M:CLOCK requests for the specified entry address to be canceled.

INTERVAL, [*]units specifies the time interval in two-millisecond units.

PRIO, [*]priority permits users with real-time privilege to specify the software priority. This option is ignored if the user does not have real-time privilege.

ONESHOT cuases the M:CLOCK request to be automatically canceled after one occurrence. If ONESHOT is not specified, the interval timing is to be automatically repeated until CANCELed.

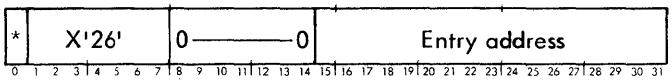
MASTER specifies that the user will be given control in the MASTER mode. (This is only honored if the user has real-time privilege.) The default is the SLAVE mode.

Calls generated by the M:CLOCK procedure have the form

CAL1,5 fpt

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

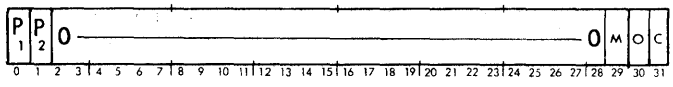
word 0



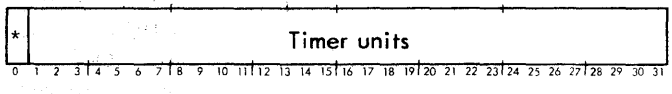
| | | | |
|---|---|---|---|
| * | X'26' | 0 ——— 0 | Entry address |

word 1

| | | | | | |
|---|---|---|---|---|---|
| $P_1$ | $P_2$ | 0 ——— 0 | M | O | C |



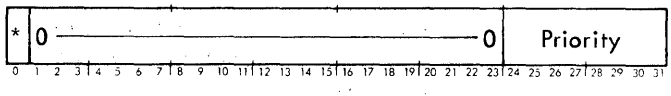
word 2 (P1)



| | |
|---|---|
| * | Timer units |

word 3 (P2)

| | | |
|---|---|---|
| * | 0 ——— 0 | Priority |



where

M specifies MASTER mode if set to 1.

O specifies ONESHOT if set to 1.

C specifies CANCEL if set to 1.

Condition code settings resulting from an M:CLOCK CAL are:

CC1 - set if no interrupt control blocks are available.

CC2 - set if no interrupt control block is associated with the user's entry address when CANCEL is specified.

CC3 - set if user does not have a privilege level of at least 80.

## DEVICE PREEMPTION SERVICES

### PREEMPT DEVICE

**M:STOPIO** Certain real-time applications require that there be direct control over the I/O associated with a particular device and that there be no contention for a particular device during certain critical processing periods. This includes the ability to request I/O end action off of the I/O interrupt associated with the I/O operation.

The real-time user may request that a specific device be preempted from use by any user other than a real-time user doing direct I/O to the device via the M:STOPIO service. The following types of devices may not be preempted.

Teletypes (i.e., Operator's Console)

COC Devices

Public RADs

Public Disk Packs

The format of the M:STOPIO procedure call is

M:STOPIO {(DCB, [*]dcb adr) / (DEV, [*]X'device adr')} [, (EA, [*]vadr)]

where

DCB, [*]dcb adr specifies that the device associated with this currently open device-type DCB is to be preempted from use by any other user. Only the user requesting the STOPIO may perform subsequent I/O to the device.

DEV, [*]X 'device adr' specifies which device is to be preempted from use by all but this user and is one of the following:

ndd - a 12-bit physical address as used by Sigma hardware.

cudd - a 14-bit physical address as used by Xerox 560 hardware (cluster/unit/device).

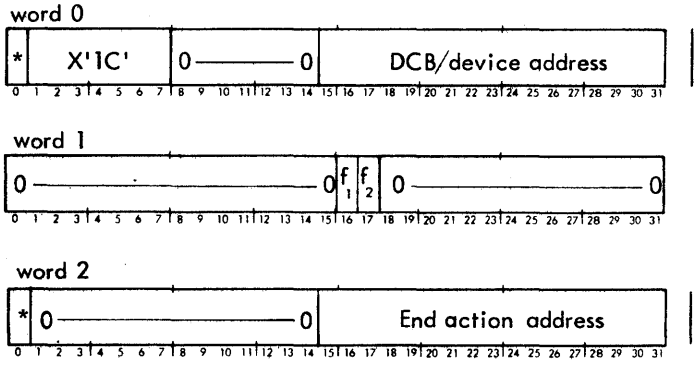
EA, [*]vadr is the virtual address of a routine that is to handle any I/O interrupts from the device being preempted. This address is converted to a physical address and stored in the DCT tables. Therefore the user, prior to issuing the M:STOPIO request, must have locked himself in core via the M:HOLD CAL. This routine is entered master mode, unmapped, via a BAL on register 11 with the I/O interrupt active (high). Register 1 contains the AIO status of the interrupting device; register 2 contains the right-justified address of the interrupting device; byte 0 of register 3 contains the condition codes as set by the AIO instruction; registers 4 and 5 contain the TIO status of the interrupting device with byte 0 of register 4 containing the condition codes as set by the TIO instruction; register 6 contains the physical address of the (user's) end-action-receiving routine; and register 7 contains the DCT index of the interrupting device. No monitor services may be requested by the receiving routine. All registers may be considered volatile except register 11 through which return to the monitor must be made.

The DCB form of the M:STOPIO procedure call should be used whenever the user depends upon the operator to mount removable volumes on private spindles or tape drives. The DEV form should be used whenever the user wants a symbiont-type device (e.g., LP, CR, CP, RBT). Use of the DEV form to preempt any other device type results in an abnormal return (see the condition code settings below).

Calls generated by the M:STOPIO procedure have the form

CAL1,5 fpt

where fpt points to word 0 of the following FPT.




where

$f_1$ specifies DEV if 0 or DCB if 1.

$f_2$ indicates that EA was not specified if 0 or that EA was specified if 1.

The return from the procedure call is to CAL+1 with the following possible condition code settings:

| 1 | 2 | 3 | 4 | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | device successfully preempted. |
| 0 | 0 | 0 | 1 | user doesn't have real-time privileges; or the physical EA address is greater than 128K (Xerox 560 only). |
| 0 | 0 | 1 | 0 | requested device is not preemptable (i.e., a public pack or RAD), is already preempted by another user, the specified DCB is not opened properly, or there was an illegal use of the DEV form. |
| 0 | 1 | 0 | 0 | unknown device address; request ignored. |
| 1 | 0 | 0 | 0 | requested device was associated with a suspended symbiont; request ignored. |

Should the application require that the multi-device controller (associated with the device to be preempted) also be preempted, the SYSCON processor should be used. This would imply that the application cannot tolerate any contention for either the particular device or the multi-device controller associated with that device (tape drive or private disk pack). In the case of a disk pack controller, the spindles associated with that controller must have been designated as private.

### RETURN PREEMPTED DEVICE

**M:STARTIO** Any preempted I/O device may be returned to the system via the M:STARTIO service.

The format of the M:STARTIO procedure call is

M:STARTIO {(DCB, [*]dcb adr) / (DEV, [*]X'device adr')}

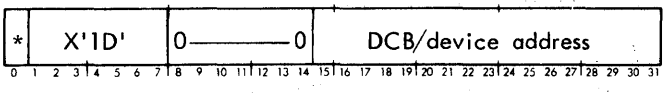
where dcb adr and device adr are as described under M:STOPIO.

Calls generated by the M:STARTIO procedure have the form

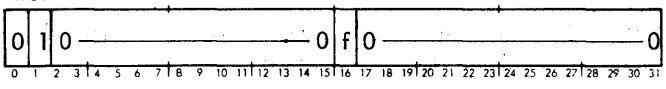
```
CAL1,5     fpt
```

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

word 0




word 1




where f specifies DEV if 0 or DCB if 1.

The return from the procedure call is to CAL+1 with the following possible condition code settings:

| 1 | 2 | 3 | 4 | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | device successfully returned. |
| 0 | 0 | 0 | 1 | user doesn't have real-time privileges. |
| 0 | 0 | 1 | 0 | device wasn't preempted by this user, or the specified DCB is not opened properly, or there was an illegal use of the DEV form. |
| 0 | 1 | 0 | 0 | unknown device address; request ignored. |
| 1 | 0 | 0 | 0 | device was busy; request ignored. |

## DIRECT I/O SERVICES

### IOEX SERVICES

The M:IOEX service is provided as one means of enabling the real-time user to exercise direct control over I/O operations without having to run in the master mode (see also the M:EXU service). The only requirements are that the device specified be preempted (either via the M:STOPIO service or the SYSCON processor), and that an end-action routine be provided (either via M:STOPIO or M:IOEX). The I/O functions that can be controlled via M:IOEX are:

SIO – Start input/output.

HIO – Halt input/output.

TIO – Test input/output.

TDV – Test device.

**M:IOEX (SIO)** The format of the M:IOEX(SIO) procedure call is

M:IOEX {(DCB, [*]dcb adr) / (DEV, [*]X'device adr')},
[(SIO, [*]clist[,REL])[,(EA, [*]vadr)]
[,(TO,[*]value)[,(PRI, [*]prio)]

where

DCB,[*]dcb adr specifies that the I/O function is to be performed for the device associated with the currently open DCB addressed.

DEV,[*]X'device adr' specifies the device for which the I/O function is to be performed and is one of the following:

ndd – a 12-bit physical address as used by Sigma hardware.

cudd – a 14-bit physical address as used by Xerox 560 hardware (cluster/unit/device).

SIO,[*]clist is the starting virtual address (double-word bound) of the I/O command list to be initiated. All buffer addresses within the command list itself must be physical addresses. The channel program must request a "Channel End Interrupt" (unless REL has been specified, see below); however, multiple interrupts per I/O requests are permitted (e.g., "Zero Byte Count Interrupt" and "Channel End Interrupt").

REL specifies that the channel is to be released after issuing the SIO. This would be used with command lists which do not result in data transfer operations (e.g., seek orders, rewind orders, head-positioning orders).

EA, [*]vadr is as described under M:STOPIO; the M:HOLD requirement applies to M:IOEX also.

TO, [*]value is the number of 4.8 second intervals allowed to elapse following the issuance of the SIO instruction before the EA address will be entered. In this case, the user's EA routine is entered master mode, unmapped, via a BAL on register 11 with registers 1 and 2 equal to zero. Register 7 contains the DCT index and register 6 contains the physical address of the (user's) end-action-receiving routine. Byte 0 of register 3 contains the condition codes as set by the SIO instruction; registers 4 and 5 contain the SIO status registers' information. No monitor services may be requested by the receiving routine. All registers may be considered volatile except register 11 through which return to the monitor must be made. A time-out value of zero

implies that no time-out facility is desired (default), however the user's EA address will always be entered should an SIO failure occur (in this case register 1 will be nonzero).

PRI, [*]prio is the priority at which to queue the request and is a value between 0 and X'FF'. The default is the value of the user's current execution priority.

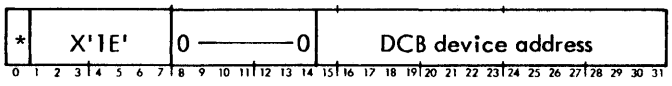
Table 43 summarizes the various possible register settings for end-action routines.

Calls generated by the M:IOEX (SIO) procedure have the form

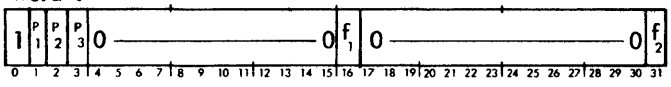
```
CAL1,5    fpt
```

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

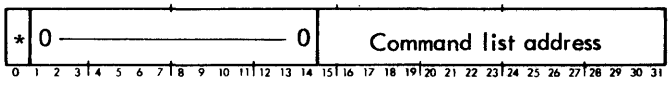
word 0



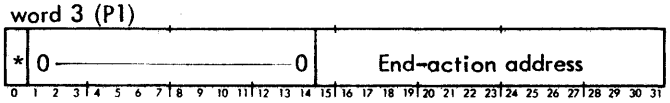

word 1



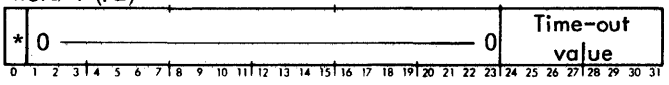

word 2



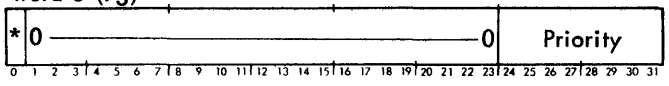

word 3 (P1)




word 4 (P2)




word 5 ($P_3$)




where

$f_1$ specifies DEV if 0 or DCB if 1.

$f_2$ is set to 1 if REL was specified on SIO.

The return from the procedure call is to CAL+1 with the following possible condition code settings:

| 1 | 2 | 3 | 4 | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | I/O request successfully queued. |
| 0 | 0 | 0 | 1 | user doesn't have real-time privileges. |
| 0 | 0 | 1 | 0 | specified device doesn't exist or the specified DCB is not opened properly. |
| 0 | 1 | 0 | 0 | EA was not specified and the DCT tables do not contain the address of an end-action receiver; or the physical address EA is greater than 128K (Xerox 560 only). |
| 1 | 0 | 0 | 0 | specified device is not preempted, or has been preempted by another user. |

Table 43. Register Settings for End-Action Routines

| Register | Contents when Routine Entered due to Interrupt | Contents when Routine Entered due to Timeout or SIO Failure |
|---|---|---|
| R0 | - | - |
| R1[t] | AIO status. | 0 = timeout; Nonzero = SIO failure. |
| R2[t] | Device address. | 0 |
| R3 | AIO condition codes (byte 0). | SIO condition codes (byte 0). |
| R4 and R5 | TIO status; byte 0 of R4 contains the condition codes from TIO. | SIO status (if timeout). R4 contains the doubleword address of the failing channel program (if SIO failure). |
| R6 | Physical address of EA routine. | Physical address of EA routine. |
| R7 | DCT index. | DCT index. |

[t]R1 and R2 indicate how the end-action routine was entered.

**M:IOEX (HIO/TIO/TDV)** The format of the M:IOEX (HIO/TIO/TDV) procedure call is

$$\text{M:IOEX}\ \left\{\begin{matrix}\text{(DCB,[*]dcb adr}\\ \text{(DEV,[*]X'device adr')}\end{matrix}\right\},\left\{\begin{matrix}\text{HIO}\\ \text{TIO}\\ \text{TDV}\end{matrix}\right\}$$

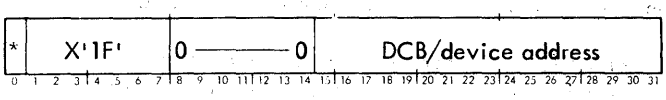
where dcb adr and device adr are as described under M:IOEX.

Calls generated by the M:IOEX (HIO/TIO/TDV) procedure have the form

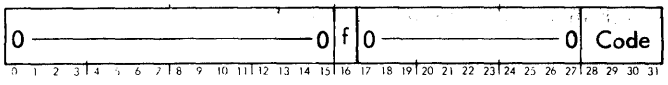
```
CAL1,5     fpt
```

where fpt points to word 0 of the following FPT.

word 0




word 1




where

code is:

0 if TIO

1 if TDV

2 if HIO

f specifies DEV if 0 or DCB if 1.

The return from the procedure call is to CAL+1 with the condition codes and registers set as if the user had issued the following instruction:

$$\left\{\begin{matrix}\text{HIO}\\ \text{TIO}\\ \text{TDV}\end{matrix}\right\},8 \qquad \text{X'device address'}$$

Since the condition codes cannot be used to communicate abnormal conditions for any of the above three services, any of the abnormal conditions indicated below will result in a program abort (code B9, subcode as indicated). Such aborts may be intercepted by the user via TRAP control (M:TRAP procedure call specifying CAL).

| Subcode | Meaning |
|---|---|
| 01 | User doesn't have real-time privilege. |
| 02 | Specified device doesn't exist, is not preempted by this user, or the specified DCB is not opened properly. |

## EXECUTE PRIVILEGED INSTRUCTION SERVICE

**M:EXU** The M:EXU service is provided as another way to enable the real-time user to execute I/O instructions and other privileged instructions without having to run in the master mode (see also the M:IOEX Service). The only requirement is that the instruction op code to be executed be one of the following:

| Op Code | Mneumonic |
|---|---|
| X'4C' | SIO |
| X'4D' | TIO |
| X'4E' | TDV |
| X'4F' | HIO |
| X'6C' | RD |
| X'6D' | WD |

The SIO execution service is intended primarily for interfacing to devices not known to the operating system (DCT tables) and which do not generate I/O interrupts (X'5C'). However, no validity checks are made and if the SIO will result in an I/O interrupt, it is assumed that the user will have provided an end-action receiver via the M:STOPIO service.

For a complete discussion of the M:EXU service, see the CP-V/BP Reference Manual, 90 17 64.

## ENTER MASTER MODE

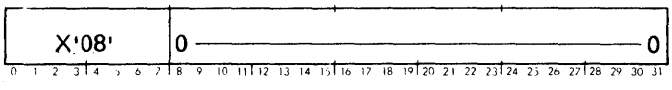
**M:MASTER** The M:MASTER procedure allows a user with sufficient privilege level (C0 or higher) to operate in the master mode (master-protected mode if running on a Sigma 9 or Xerox 560) with a write key of 1. (This procedure resides in SYSTEM BPM.) The format of the procedure call is

```
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. (This procedure resides in SYSTEM BPM.) The format of the procedure call is

```
M:SLAVE
```

Calls generated by the M:SLAVE procedure have the form

CAL1,5 fpt

### PSECT DIRECTIVE

The Meta-Symbol PSECT directive specifies that the control section which follows is to begin on a page boundary. The directive can be useful for controlling the placement of I/O buffers, data, and end-action-receiving routines which will be accessed unmapped.

### VIRTUAL/PHYSICAL ADDRESS CONVERSION

**M:MAP** The M:MAP procedure converts a specified virtual address to a physical address or a specified physical address to a virtual address. The converted address is stored in general register 8. The M:MAP procedure call has the form:

M:MAP{VTP / PTV}, (ADR, [*]address)

where

VTP specifies virtual to physical address conversion.

PTV specifies physical to virtual address conversion.

ADR, [*]address specifies the location of the address to be converted.

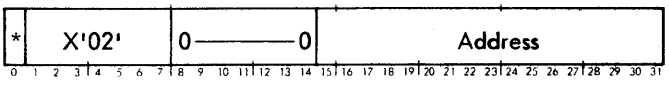
M:MAP should be used with M:HOLD since the address returned via M:MAP may not be valid if a swap occurs.

Calls generated by the M:MAP procedure have the form

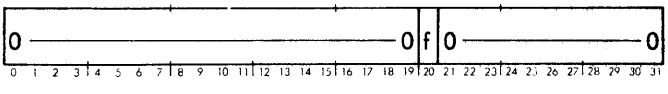
CAL1,6 fpt

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

word 0




word 1




where f indicates virtual to physical address conversion (f = 0) or physical to virtual address conversion (f = 1).

If the user's privilege level is not at least A0, the return is to CAL+1 with CC1 set.

## MISCELLANEOUS REAL-TIME SERVICES

The following is a set of services provided to master mode, mapped or unmapped real-time programs. These services are provided via Meta-Symbol procedure references that result in BAL linkages to monitor routines (hence the master mode requirement) as opposed to CAL1 linkages. The routines entry points are REFed as a result of the various procedure calls, therefore the program must be loaded with reference to the MONSTK or J1 files in order to satisfy these external references. All user registers are preserved by pushing them into TSTACK except as indicated for specific services.

### GET OR FREE PHYSICAL PAGE

**M:GPP** The M:GPP procedure acquires a physical page of memory. The procedure call has the format

M:GPP

On return from the procedure, general register 3 contains the physical page number of the newly allocated page of memory or the value zero if none was available.

In order for a mapped user to reference a physical page acquired by M:GPP, it is necessary to perform a Change Virtual Map (M:CVM) specifying the physical address of the page acquired by M:GPP and the virtual address into which this page is to be mapped.

**M:FPP** The M:FPP procedure releases a physical page of memory that was acquired by M:GPP. The procedure call has the format

M:FPP [*]page

where page specifies the physical page number of a page of memory which is to be returned to the system.

It is the user's responsibility to return any pages obtained via M:GPP since the system keeps no record of this transaction.

### INITIATE GHOST JOB

**M:GJOB** The M:GJOB procedure activates (or awakens) a program as a ghost job. The format of the procedure is

M:GJOB (LMN,loc)[,(ACN,loc)][,(PRI,[*]value)]

where

LMN,loc specifies the location containing the name of the program to be activated (or awakened) as a ghost job. The name must be in TEXTC format and must not be greater than 7 characters in length. If the name is less than four characters, a word of blanks (X'40's) must immediately follow the name.

ACN,loc specifies the location containing the name of the account in which the program exists. The account name must be in TEXT format, left-justified with trailing blanks to occupy two words. The default is the :SYS account.

PRI, value specifies the execution priority to be associated with the ghost job. The default priority will be that defined for ghost jobs at SYSGEN.

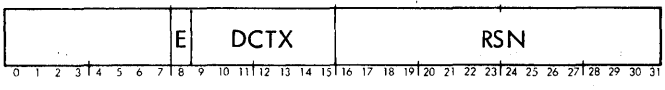
On return, CC1 is set if the named program was not found or was already active but not asleep. Otherwise, the condition codes are set to zero.

## GET AND RELEASE DISK GRANULE

**M:GDG** The M:GDG procedure dynamically acquires a disk granule. The procedure has the format

M:GDG

The starting disk address of the acquired granule is returned to the user in general register 8 in the following format




The disk address shown above is in standard format for disk addresses in CP-V, where:

E is the extension bit necessary to represent a 17-bit relative sector number on large capacity disk packs.

DCTX is the DCT index for the device.

RSN is the 16-bit relative sector number.

If no granule is available, register 8 is set to zero.

Note: Currently, E (in the disk address shown for M:GDG) is a single bit. In the future Xerox may add disk packs which require 18-bits in order to represent the highest relative sector number. Therefore it is recommended that users utilize the following procedures (referenced via the SYSTEM UTS directive of Meta-Symbol) when referencing disk address fields.

To load or store a DCT index, use

{LDCTX / STDCTX},r [*]loc[,index]

where

r is the register to be loaded (stored).

loc is a location, or optionally, a pointer to a location, containing (destined to contain) the disk address (see index below).

index is an index register containing a word displacement which, when added to the address given by loc, yields an effective address containing (destined to contain) the disk address.

To load or store a relative sector number:

{LSECTA / STSECTA},oddreg [*]loc[,index].

where

oddreg is the odd numbered register to be loaded or stored.

loc and index are as described under LDCTX/STDCTX.

Note that it is the user's responsibility to return any granules obtained via M:GDG since the system keeps no record of this transaction.

**M:RDG** The M:RDG procedure dynamically releases a granule acquired via M:GDG. The procedure has the format

M:RDG [*]disk address

where disk address is the starting address of the disk granule to be returned to the system. It must have the same format as described for M:GDG.

Under certain conditions, the monitor may not be able to accept a granule from the user at a particular time. In this case, register 8 will contain a zero indicating that the user must try again.

## REPORT USER EVENT

**M:RUE** The M:RUE procedure reports an event on a particular user (i.e., it simulates that the event took place for the user). The format of the procedure call is

M:RUE (UN,[*]user#),(EV,{event / *loc2})

where

user# is the number of the user for whom the event is to be reported. A user may determine his own user number (for purposes of communicating this to other programs) by referencing the monitor cell S:CUN (Current User Number) which is located in page 0 of the monitor and is therefore available to any user program loaded with MONSTK.

event is one of the following symbols signifying the event to be reported on the user:

| Symbol | Event | Resulting Action |
|---|---|---|
| E:CBK | BREAK | Control passes to the user at the address specified via an M:INT procedure call. |
| E:OFF | Log-off | The user is deleted from the system. |

| Symbol | Event | Resulting Action |
|---|---|---|
| E:ERR | Error | The user is errored and deleted from the system. |
| E:WU | Wake-up | The specified user is scheduled for execution and reentered at the instruction following the M:WAIT CAL1. |
| E:UQA | Unqueue for access | The specified user is scheduled for execution and reentered at the instruction following the CAL1 which caused him to be queued for access. |

loc2 is a word location containing the value associated with the event symbol (defined by the assembly SYSTEM).

Note: Care must be taken to ensure that the user for whom the event is being reported is in the appropriate state since an illegal current state/event combination will cause the sysfem to crash. E:CBK, E:OFF, and E:ERR may be safely reported on a user at any point in time.

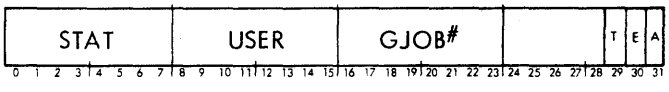
## CHECK INTERRUPT STATUS

**M:CHKINT** The M:CHKINT procedure checks the status of an interrupt. The format of the procedure call is:

M:CHKINT (INT,[*]int)

where int is the location of a word containing the address of the interrupt to be checked.

The following word of information will be returned to the user in general register 8:




where

A specifies, if set, that the interrupt is armed.

E specifies, if set, that the interrupt is enabled.

T specifies, if set, that the interrupt has been triggered and the interrupt processing routine has not yet finished.

STAT indicates the status of the task associated with the interrupt location as follows:

| STAT | Meaning |
|---|---|
| X'80' | Task is active. |
| X'40' | Task is asleep or queued for interrupt. |
| X'20' | Task is waiting for I/O completion. |
| X'10' | Task is blocked, waiting for a resource. |
| X'01' | Specified interrupt is not currently associated with any user (i.e., inactive). |

USER is the internal user table index for the user currently associated with this interrupt.

GJOB# is the user number of the ghost job (if it is active) which will be entered on the occurrence of the interrupt. If the ghost job is not active, GJOB# contains zero.

## I/O SERVICES

The following services result in BAL linkages to the monitor's I/O Supervisor module (IOQ). They are separated into three types:

1. I/O without a DCB where the user supplies the channel program (M:EXCP). This should be used only where no handler exists for a particular device or the user requires unusual control over the device.

2. I/O without a DCB while not requiring the user to build his own channel program (M:NEWQ).

3. I/O with parameters supplied in a pseudo DCB (M:QUE).

Special problems exist when applying these techniques to disk I/O. Unless the volume is being managed entirely by the user, the user must be aware of the physical location of the data on the disk volume (or volumes). A random file would be the most common way of allocating space on a public or private volume for use by both privileged and non-privileged users. A random file is allocated contiguously on a public or private volume when it is opened. By specifying the FPARAM option on the M:OPEN call to an existing file, the user requests the monitor to pass the file attribute (FIT table) parameters to a specified location (see the DCB discussion in Appendix A of the CP-V/BP Reference Manual, 90 17 64). FDA (First Disk Address) is returned in word one of the X'0C' – coded FIT entry. File size (in granules) is given in word one of the X'0D' – coded FIT entry.

Flawed tracks are automatically taken care of by the I/O system assuming that the requested byte count does not cause the transfer to cross a track boundary from a good track to a flawed track. If the user ensures that all tracks are good, the hardware will automatically handle the case in which a track boundary is crossed. However, the user must handle the cylinder overflow condition himself. (A new seek must be issued between accessing the last sector of one cylinder and the first sector of the next cylinder.)

### CALCULATING PHYSICAL ADDRESSES

All of the I/O procedure calls described below are available to the mapped or unmapped user. Several require that

physical addresses by passed. For all mapped users, the user may convert a virtual address to a physical address by using the M:MAP procedure call described previously.

In order to ensure that a mapped user is not swapped between the time that the physical address is calculated and the time the I/O is requested, the M:HOLD (Lock in Core) service should be performed.

Note, however, that a mapped, master mode program is assured of not being swapped as long as it does not request any monitor services via CAL1s.

## EXECUTE CHANNEL PROGRAM

**M:EXCP** The M:EXCP procedure causes the user's own channel program to be executed. The format of the procedure call is

```
M:EXCP (CPA,{DA(loc)}),(DCT, [*]index)
        {*loc  }
  [,(PRI, [*]priority)][,(EA,[*]loc
  [,[*]eai])][,(TOI,[*]value)]
```

where

DA(loc) specifies the physical doubleword address of the start of the channel program.

*loc specifies the word address of a word which contains the physical doubleword address of the start of the channel program. (The asterisk is required.)

DCT, [*]index is the DCT index of the device associated with the channel program.

PRI, [*]priority is the priority to be associated with the requested I/O operation. Priority requests range from 0 to X'FF' (highest to lowest). Priorities in the range of 0 to X'BF' are treated as real-time priority requests; X'C0' to X'FF' are treated as background priority requests. The only system I/O that operates at a real-time priority is swapping I/O (priority = X'10'). The default priority is X'FF'.

EA, [*]loc is the physical address of the user's end-action routine.

eai is a word of end-action information. This information is passed back to the user's end-action routine.

TOI, [*]value is a time-out value specified in five second increments. The default value is five seconds.

The user's end-action routine (if specified) is entered unmapped, via a BAL on register 11. All registers may be considered volatile (except register 11, through which return is made to the monitor). The following information is passed to the end-action routine:

| Register | Bit Fields | Contents |
|---|---|---|
| 7 | 24, 8 | -, DCT |
| 12 | 8, 8, 16 | TYC, -, RBC |
| 13 | 16, 16 | -, CCA |
| 14 | 32 | EAI |
| 15 | 13, 19 | -, BUF |

where

DCT is the DCT index.

TYC is the type of completion code returned by the device handler.

RBC is the remaining byte count.

CCA is the current IOP command address.

EAI is the end-action information specified in the procedure call.

BUF is the doubleword address of the start of the channel program command list specified by the M:EXCP call.

The end-action routine may obtain the complete TDV status by referencing the doubleword table DCT13 using the DCT index in register 7.

## CALL NEWQ

**M:NEWQ** The M:NEWQ procedure requests I/O to be performed without a DCB and without a user' built channel program. The format of the procedure call is

```
M:NEWQ [,{W}] (FC, [*]code),
        {NW}
  (BUF,{BA(loc)}),(SIZ, [*]value);
  {,(DA, ~~[*]disk address~~)}[,(PRI, [*]priority)];
  {,(DCT, [*]index)}
  [,(NRT, [*]value2)],[,(EA, [*]loc2[,[*]eai])]
```

*disk address

where

W/NW is the WAIT/NO-WAIT option. The unmapped user always does I/O with NO-WAIT. This implies that the unmapped user should always (except for unusual cases) specify an end-action address in order to ascertain when the I/O has completed. The mapped user will do I/O with WAIT unless otherwise specified by the procedure call.

FC, [*]code is the function code which defines (to the device handler) the type of I/O operation to be performed. See discussion of function codes below.

BUF,BA(loc) specifies the byte address of the user's buffer to be used in this I/O operation.

BUF, *loc specifies the word address of a word which contains the byte address of the user's buffer.

[*]value

SIZ, ~~value~~ is the byte count to be used in this I/O operation. Note: count for mapped progr. should not exceed 32K bytes (X'8000')

DA, ~~[*]disk address~~ specifies, for random-access-device operations only, the disk address to be used in this I/O operation. Disk addresses are of the format described under the discussion of the M:GDG procedure call.

*disk address

DCT, [*]index specifies for non-random-access-device operations only, the DCT index of the device to be used in this I/O operation.

PRI, [*]priority is the priority to be associated with the requested I/O operation. See the description of priority under the discussion of M:EXCP.

NRT, [*]value is the number of recovery tries to attempt before declaring an error.

EA,[*]loc is the physical address, or optionally a pointer to a location containing the physical address, of the user's end-action routine.

eai is a word of end-action information. See the end-action description under the discussion of the M:EXCP procedure call. The only difference is that BUF is the byte-address of the user's buffer as supplied by the M:NEWQ procedure call.

To assist the user in determining the correct function codes to be used with the M:NEWQ procedure calls, the following is a discussion describing the function codes of the existing device in the system.

Typewriter Handler. The typewriter handler accepts the following function codes:

0 - read with editing
1 - write
2 - write with device name
3 - read without editing
4 - read with editing and retry
5 - write new line character
6 - write with device name tabbed

RAD Handler. The RAD handler accepts the following function codes:

0 - seek-read
1 - seek-write
2 - sense
3 - seek-checkwrite
4 - seek-write, seek-checkwrite

Error recovery on the RAD generally amounts to redoing the same operation when an error has been detected. One exception is when a checkwrite is being performed for a write and an error is indicated. In this case, the write is done over, followed by another checkwrite. Checkwrites are performed for all writes if sense switch 1 is set on the operator's console. Special conditions checked for are write violation and illegal seek address.

9-Track Tape Handler. The 9-track tape handler accepts the following function codes:

0 - read
1 - write
2 - read reverse
3 - write tape mark
4 - backspace record
5 - forwardspace record
6 - backspace file
7 - forwardspace file
8 - rewind
9 - sense
10 - correctable read recovery
11 - noncorrectable read recovery
12 - write recovery
13 - correctable read reverse recovery
14 - noncorrectable read reverse recovery
15 - write tape mark recovery

7-Track Tape Handler. The 7-track tape handler accepts the following function codes:

0 - read packed
1 - write packed
2 - read reverse packed
3 - write tape mark
4 - backspace record
5 - forwardspace record
6 - backspace file
7 - forwardspace file
8 - rewind
9 - read binary
10 - write binary
11 - read reverse binary
12 - read decimal
13 - write decimal
14 - read reverse decimal
15 - read packed recovery
16 - write packed recovery
17 - write tape mark recovery
18 - read binary recovery
19 - write binary recovery

20 - read decimal recovery
21 - write decimal recovery
22 - final backspace record for reverse read
23 - final backspace record if unrecoverable error

Card Reader Handler. The card reader handler accepts the following function codes:

0 - read binary
2 - read automatic

Line Printer Handler. The line printer handler accepts the following function codes:

1 - write without format
3 - write with format

Paper Tape Handler (PTAP). The paper tape handler accepts the following function codes:

0 - read automatic
1 - write BCD
2 - read count
3 - write binary
4 - read direct
5 - write direct
6 - read BCD
7 - read binary

Card Punch Handlers. The card punch handlers accept the following function codes:

0 - punch BCD
1 - punch binary

Disk Pack Handler (DPAK). The disk pack handler uses the following function codes:

0 - seek-read
1 - seek-write
2 - sense
3 - seek-checkwrite
4 - read
5 - write
6 - checkwrite
7 - restore
8 - seek-read header
9 - read header

CALL QUE

**M:QUE** The M:QUE procedure requests that I/O be performed through parameters supplied in a specified DCB. At the time of the call, the specified DCB need only be 9 words in length but must contain valid information in the following fields: NRT, QBUF, BLK, and CDA. (See Appendix A of the CP-V/BP Reference Manual, 90 17 64.) The format of the M:QUE procedure call is

M:QUE [*]dcb,(FC,[*]code)[,(EA,[*]loc[,[*]eai])]

where

dcb specifies the DCB associated with the requested I/O operation.

code is an 8-bit code (described in Figure 17 below) which defines (to the device handler) the type of I/O operation to be performed. The code may be expressed as a decimal number or as a hexadecimal number in the format X'dd'.

loc and eai function exactly as described under the discussion of the M:EXCP procedure call. The user's end-action routine (if specified) will be entered unmapped via a BAL on register 11 after the TYC (type of completion code) and ARS (actual record size) have been entered into the DCB. The following information is passed to the end-action routine.

| Register | Bit Fields | Contents |
|---|---|---|
| 6 | 15, 17 | -, BUF |
| 7 | 24, 8 | -, DCT |
| 8 | 8, 7, 17 | FC, -, DCB |
| 14 | 32 | EAI |

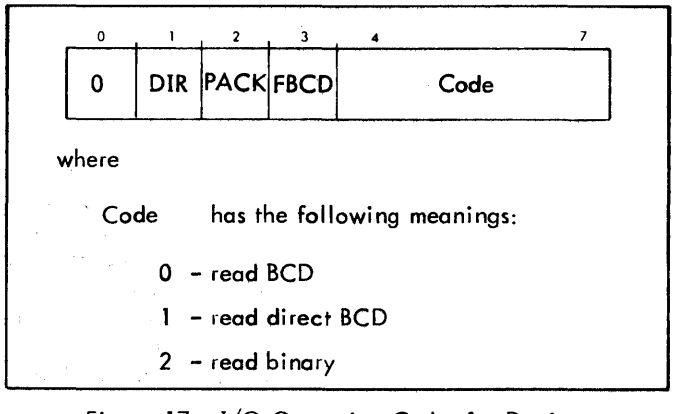
where

BUF is the word address of the user's buffer associated with this I/O request

DCT is the DCT index as specified in the CDA field of the DCB at the time of the M:QUE procedure call.

FC, DCB, and EAI are as specified in the M:QUE procedure call.

For the unmapped user, the I/O will be queued at a priority of X'FF'. For the mapped user, the I/O will be queued based upon the user's current execution priority.




where

Code has the following meanings:

0 - read BCD
1 - read direct BCD
2 - read binary

Figure 17. I/O Operation Codes for Device Handler (M:QUE)

3 - read direct binary

4 - write BCD

5 - write direct BCD

6 - write binary (write and format)

7 - write direct binary

A - skip record forward
B - skip record reverse
C - skip file forward
D - skip file reverse
E - rewind
F - write end-of-file

} bits 1-3 are ignored for these codes

FBCD specifies no FORTRAN conversions if 0 or FORTRAN conversions if 1.

DIR specifies forward direction if 0 or reverse direction if 1.

If the device is not 9T, 7T, or MT, only bits 5 through 7 are meaningful.

Figure 17. I/O Operation Codes for Device Handler (M:QUE) (cont.)

SEND CHARACTER TO TERMINAL

**M:COC** The M:COC procedure sends a character to a user terminal.

$$\text{M:COC} \begin{Bmatrix}(\text{UN}, [*]\text{user}^{\#}) \\ (\text{LN}, [*]\text{line}^{\#})\end{Bmatrix}, (\text{CHAR}, \begin{Bmatrix}\text{'character'} \\ *\text{loc}[, \text{ireg}]\end{Bmatrix})$$

where

user# is the user number of the user whose terminal is to receive the character.

line# is the line number of the terminal which is to receive the character.

'character' is the EBCDIC character to be sent to the specified terminal.

*loc[,ireg] specifies the address of a location which contains the character to be sent to the terminal (loc). (The asterisk is required but does not indicate indirectness.) The ireg field specifies an index register which contains the byte displacement which, when added to the address specified by loc, will yield the byte address of the character to be sent to the terminal. If ireg is absent, loc is assumed to contain the left-justified character to be sent to the terminal.

# DYNAMIC PHYSICAL PAGE ALLOCATION FOR REAL-TIME PROCESSING

## INTRODUCTION

Physical pages are made available for real-time processing in either of two ways:

- Dedication of physical core pages at boot-time. These pages are known as the Resident Foreground (RESDF) pages. SYSGEN parameters define the physical pages that are to be removed from the system and dedicated to real-time processing. These pages remain dedicated real-time pages until returned to the system via the Physical Page Stealer (PPS) Ghost.

- Dynamic acquisition and release of physical core pages during normal operations. These pages are known as the Dynamic Resident Foreground (DYNRESDF) pages. The operator can acquire or release DYNRESDF pages by communicating with the Physical Page Stealer (PPS) ghost job.

In both cases, foreground memory is allocated in 'memory segments'. A memory segment in this context is simply a set of contiguous physical pages. There is only one RESDF memory segment (i.e., that which may be allocated at boot-time). There may be several DYNRESDF memory segments, the maximum number of which is specified at SYSGEN time. All real-time memory segments must be allocated in the area between 64K and the end of physical core.

The operator, by communicating with the Physical Page Stealer ghost job, has control over the allocation of both RESDF and DYNRESDF pages. The operator also has the ability to reset the SYSGEN defined RESDF size and maximum DYNRESDF size thus affecting the system's maximum user size. Increases to RESDF size or to maximum DYNRESDF size cause a decrease of the maximum user size; decreases to RESDF size or the DYNRESDF size cause the maximum user size to be increased. By setting the maximum number of real-time pages that may be allocated to a minimum, the operator is able to allow very large jobs to be scheduled. Decreases to the maximum real-time page values may be effected at any time. Increases that would cause the maximum user size to be set to less than 186 pages are limited to times when there are no users on the system other than system ghosts; i.e., the system must be quiescent except for ALLOCAT, RBBAT, FILL and the PPS ghosts. Neither RESDF nor DYNRESDF maximum size may be increased to the point where the maximum user size is too small to allow the system ghosts to run.

## SYSGEN CONSIDERATIONS

The system parameters that define the pages to be allocated at boot-time, the maximum number of pages that may be dedicated for real-time use, and the maximum number of memory segments that may be allocated for real-time

processing may be specified via options, of the :FRGD command of PASS2. The format of these options is as follows:

(RESDF, size, address)

where

size specifies, in decimal, number of pages, the default size of the dedicated foreground memory area to be allocated at system initialization.

address specifies, in hexadecimal, the word address of the first page in the RESDF memory segment. This value must be equal to or greater than $10,000_{16}$.

Both size and address may be overridden by the operator at system initialization. Both parameters may be reset via communication with the Physical Page Stealer ghost job.

(DYNRESDF, pages, segments)

where

pages specifies in decimal the maximum number of pages that may be dynamically allocated for foreground use. These pages are not removed from the system until requested, but the maximum user size is reduced by the value specified. This value may be altered by the operator via the PPS ghost.

segments specifies in decimal the maximum number of dedicated real-time memory segments that may be allocated for foreground use. The default value is one.

## INITIALIZATION

When a real-time system is booted from a system tape and operator console interaction is requested, or when a real-time system is booted from the system RAD, the following message is output on the OC device:

```
RESET RESDF YYY,XXXXX?
```

This allows the operator to override the SYSGEN-defined values for the beginning of the RESDF area and/or the size of the RESDF area. The operator should respond as follows:

[yyy][,xxxxx] ㊡

where

yyy is the number of pages in decimal to be in the RESDF area. A value of 0 through 999 may be used.

xxxxx is the word address in hexadecimal of the first page in the RESDF area. A value greater than $10,000_{16}$ (64K) must be used.

Either value may be omitted, or a response of NEW LINE alone may be used to request the SYSGEN-defined default for the omitted value(s).

## THE PHYSICAL PAGE STEALER GHOST JOB (PPS)

The Physical Page Stealer ghost job is used for the management of all dedicated foreground memory. It is loaded for execution via the following keyin:

!GJOB PPS

PPS then asks the operator for a command:

```
PPS: ENTER COMMAND
```

The operator may respond with one of the following commands:

DI[SPLAY] Display memory segments currently allocated.

GE[T] yyy, xxxxx Get DYNRESDF pages.

FR[EE] yyy, xxxxx Free DYNRESDF pages.

DY[NRESDF] yyy Reset maximum number of DYNRESDF pages.

RE[SDF] [yyy][, xxxxx] Redefine the RESDF area.

EN[D] Exit ghost job.

where

yyy specifies in decimal the number of pages.

xxxxx specifies the word address in hexadecimal of the first page in the real-time memory segment. This value must be equal to or greater than $10,000_{16}$.

PPS will attempt to perform the requested function, type an error message if the function cannot be performed, and reprompt the operator to get the next command. The END command is used to terminate PPS processing.

If the format of the command is in error, such as missing parameters, bad delimiters, etc., PPS will type '??' and reprompt the operator to reenter the command.

The following message will be displayed if the number of pages specified is in error:

```
EXPRESS # OF PAGES IN DECIMAL 0-999
```

The following message will be displayed if the page address specified is in error:

| EXPRESS PG ADDR IN HEX 10,000-xxxxx |
|---|

where xxxxx is the word address of the last page of physical core.

Since the operator is the only one who is allowed to communicate with the PPS ghost, running PPS is not allowed from on-line or batch. If attempting to run PPS other than as a ghost job, the following message will be typed:

| MUST BE EXECUTED AS A GHOST JOB |
|---|

The PPS commands are described in detail in the following paragraphs.

**DISPLAY** The DISPLAY command is used to obtain information concerning allocated real-time pages and the current settings of system parameters that define the maximum real-time pages allocation.

The following information is output on the OC device:

MAX DYNRESDF = yyy

CURRENT DYNRESDF = yyy

DYNRESDF SEGMENT yyy xxxxx

RESDF SEGMENT yyy xxxxx

MAXIMUM USER CORE = yyy

where

yyy is the decimal number of pages.

xxxxx is the hexadecimal word address of the first page in the real-time memory segment.

The DYNRESDF SEGMENT message is repeated for each currently allocated DYNRESDF memory segment.

**GET** The GET command is used to allocate DYNRESDF pages. This command may be used at any time and has no effect on the maximum user size. The format of the command is

GE[T] yyy,xxxxx

where

yyy specifies in decimal the number of pages.

xxxxx specifies the word address in hexadecimal of the first page in the real-time memory segment. This value must be equal to or greater than $10,000_{16}$.

The PPS ghost first validates that it is valid to allocate DYNRESDF pages. If the maximum number of DYNRESDF segments has already been allocated, the following message is displayed:

| MAXIMUM DYNRESDF SEGMENTS ALLOCATED |
|---|

If the allocation of the DYNRESDF memory segment would cause the number of DYNRESDF pages to exceed the maximum allowed, the following message is displayed:

| EXCEEDS DYNRESDF |
|---|

The PPS ghost then validates that the pages specified are available. If the pages are currently being used by the monitor, (i.e., for transaction processing), the following message is typed:

| PAGES IN USE BY MONITOR |
|---|

If some or all of the pages specified are allocated as RESDF or DYNRESDF pages, the following message is typed:

| PAGES ARE REAL TIME PAGES |
|---|

The DISPLAY command should be used to determine the current allocation of real-time memory segments.

If the pages cannot be obtained for any other reason, the following message is typed:

| UNABLE TO OBTAIN PAGES |
|---|

Otherwise, the pages specified are removed from the system and the operator is prompted to enter the next command.

**FREE** The FREE command is used to return currently allocated DYNRESDF pages to the system. This command may be used at any time and has no effect on the maximum user size.

The format of the command is

FR[EE] yyy,xxxxx

where

yyy specifies the number of pages in decimal.

xxxxx specifies the word address in hexadecimal of the first page in the real-time memory segment. This value must be equal to or greater than $10,000_{16}$.

DYNRESDF memory segments cannot be partially released. That is, all pages within the memory segment must be released with one FREE command. If the pages specified are not totally contained in one memory segment, or the entire memory segment was not specified, the following message is displayed:

| NOT A DYNRESDF MEMORY SEGMENT |
|---|

The display command should be used to determine the currently allocated DYNRESDF segments.

If the segment specified is valid, the pages will be returned to the system and the operator will be prompted to enter the next command.

**DYNRESDF** The DYNRESDF command is used to redefine the maximum number of pages that may be removed from the system to be used as dynamic RESDF pages. No pages are obtained or released as a result of this command. This command alters the maximum user size.

The format of the command is

DY[NRESDF] yyy

where yyy specifies the number of pages in decimal.

The value specified is compared to the current setting of maximum number of DYNRESDF pages. If attempting to increase the maximum size, the system must have no users other than system ghosts. If other users are on the system, the following message is typed:

| SYSTEM ACTIVE |
|---|

The maximum user size will be decreased by an amount equal to the increase in maximum DYNRESDF pages. PPS checks to determine that the system ghosts would be able to tolerate the decrease in user size. If not, the following message is displayed:

| DON'T LOCK OUT SYSTEM GHOSTS |
|---|

Otherwise, the maximum number of DYNRESDF pages that may be allocated is reset as specified and the maximum user size is decreased by the amount of increase to maximum DYNRESDF pages.

If attempting to decrease the maximum number of DYNRESDF pages and the maximum user size would fall below 186 pages, the value specified must be equal to or greater than the number of DYNRESDF pages currently allocated. If not, the following message is displayed:

| CURRENT DYNRESDF PAGES > NEW MAXIMUM |
|---|

Otherwise, the maximum number of DYNRESDF pages that may be allocated is reset as specified and the maximum user size is increased by the amount of decrease to maximum DYNRESDF pages.

**RESDF** The RESDF command is used to redefine the RESDF memory segment. The RESDF command may be used to release all RESDF pages to the system or to obtain RESDF pages.

The format of the command is

RE[SDF][yyy][,xxxxx]

where

yyy specifies the number of pages in decimal.

xxxxx specifies the word address in hexadecimal of the first page in the real-time memory segment. This value must be equal to or greater than $10000_{16}$.

To release all RESDF pages, the following format should be used:

RESDF 0

This will cause all RESDF pages to be returned to the system. The maximum user size will be increased by the RESDF size.

If the RESDF memory segment is not currently allocated when this format of the RESDF command is used, the following message is displayed:

| NO RESDF PAGES ALLOCATED |
|---|

To re-establish the RESDF memory segment, the following format of the command should be used:

RESDF [yyy][,xxxxx]

If either the number of pages or the word address of the first page is not specified, the previous value of the parameter is used.

If the RESDF segment is currently allocated, the following message is typed:

| RESDF PAGES ALREADY ALLOCATED |
|---|

When this format of the RESDF command is used, the maximum user size will be decreased by an amount equal to the size of the RESDF segment to be allocated. Therefore, if the maximum user size would fall below 186 pages,

there must be no users on the system other than system ghosts and the system ghosts must be able to tolerate the decreased user size. Checks are also made to determine if the pages specified are available as described under the discussion of the GET command.

Otherwise, the pages specified are removed from the system and the maximum user size is decreased by an amount equal to the number of pages in the RESDF memory segment.

**END** The END command terminates PPS processing and has the format

EN[D]

### MONITOR DEFs

The following words are DEFed in the monitor root and may be used by the real-time programmer to gain information concerning the current allocation of real-time pages.

| | |
|---|---|
| RESDF | The size of the RESDF area currently allocated. If all RESDF pages have been returned to the system, the value is zero. |
| RESDFP | The word address of the first page in the RESDF area. |
| DYNRESDF | The number of DYNRESDF pages currently allocated. |
| MDYNRESDF | The maximum number of DYNRESDF pages that may be allocated. |
| PP:UPPC | The total number of RESDF and DYNRESDF pages currently allocated. |

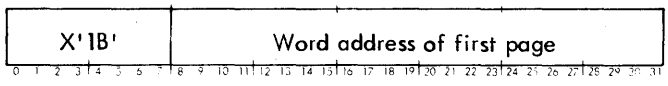
### RESDF MEMORY CAL

The real-time user may obtain information from the monitor concerning the current allocation of real-time memory segments by issuing the following CAL:

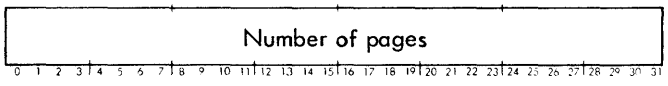
CAL 1,5 fpt

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

word 0




word 1




The system checks to see if the set of pages specified in the FPT are currently allocated real-time pages. On return from the CAL, the condition code setting will be as follows:

| 1 | 2 | 3 | 4 | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | The pages specified are the RESDF segment. |
| 0 | 0 | 0 | 1 | The pages specified are a DYNRESDF segment. |
| 0 | 0 | 1 | 0 | All pages are currently allocated real-time pages but are not a specific memory segment. |
| 0 | 1 | 0 | 0 | Some, but not all of the pages are currently allocated real-time pages. |
| 1 | 0 | 0 | 0 | None of the pages are currently allocated real-time pages. |

# 10. TRANSACTION PROCESSING FACILITIES

This chapter describes a procedure that was designed for use by Xerox in the development of the transaction processing facilities of CP-V. The procedure should never be included in any user-written programs. This chapter is intended for Xerox system programmers only.

## M:QUEUE PROCEDURE FORMAT

In transaction processing, the flow of transactions and reports is controlled through a single queue by a program called the System Queue Manager. The System Queue Manager is part of the CP-V monitor. The M:QUEUE procedure was developed for use in the System Queue Manager and requires transaction processing authorization (via the Super processor).

The format of the M:QUEUE procedure is

M:QUEUE { [*] dcb name / [*] list loc / [*] list id } , { UNLOCK / DEFINELIST / PUT / GET / STATS / PURGE / LOCK } [,(option)]. . .

where

dcb name specifies the name of the DCB for UNLOCK and LOCK requests.

list loc specifies the location of the list of criteria pointers for PUT, DEFINELIST, and STATS requests.

list id specifies the id of a list for GET and PURGE requests.

UNLOCK activates usage of the queue and defines the queue owner.

DEFINELIST defines the criteria for subsequent GET requests (i.e., the GET lists).

PUT enters a transaction or report into the queue.

GET retrieves a transaction or report from the queue.

STATS returns the status of a transaction or report.

PURGE discards outstanding GET lists which are active for a given user and releases user-associated queue control tables.

LOCK ensures that the user is the queue owner and locks the queue from further use.

The basic options are as follows:

LSIZE, [*] value specifies the size of the list for PUT, DEFINELIST, and STATS.

BUF, [*]address specifies the buffer address for returning a queue entry for a GET request or for returning queue status information for a STATS request.

BSIZE, [*] value specifies the size of the area defined by the BUF option.

WAIT specifies that the caller wishes to wait for access to the queue prior to resuming execution.

ECB, [*]address specifies the address of an ECB to be posted when a queue event occurs. A queue event may be: the arrival of an entry to the queue which satisfies an active GET list; the availability of the queue (when the WAIT option was not indicated on the original queue request); or queue space availability.

The following option is applicable only to the GET request:

INDEX, [*] value specifies the word displacement within the GET list to start the search for a criteria match.

The following option is applicable only to the PUT request:

{HIGH / LOW} specifies the priority for PUT requests.

The following options are applicable only to the UNLOCK request:

{OLD / NEW} specifies whether the queue is a new or existing file.

BACKUP specifies that the queue is to be kept up-to-date on secondary storage, (i.e., whenever a queue block is modified in core it is to be written to disk).

QPAGES, [*] value specifies the maximum number of core pages which can be used for queue blocks and queue manager work pages.

QSAT, [*] value specifies the percentage of queue capacity for acceptance of high priority PUTs only.

KEYMAX, [*] value specifies the maximum number of bytes required to contain any name (trancode) presented for enqueueing (1-13 may be specified).

RECOVER specifies queue unlock for recovery mode.

The following option is applicable only to the STATS request:

COUNT specifies that the number of occurrences in the queue of a specified criterion is to be returned in the second halfword of SR1.

The following option is applicable only to the LOCK request:

PAUSE specifies that the queue lock is temporary and current users may continue processing their current outstanding requests when the queue is unlocked.

## M:QUEUE FUNCTION PARAMETER TABLES (FPTS)

Calls generated by the M:QUEUE procedure have the form

```
CAL1,7     fpt
```

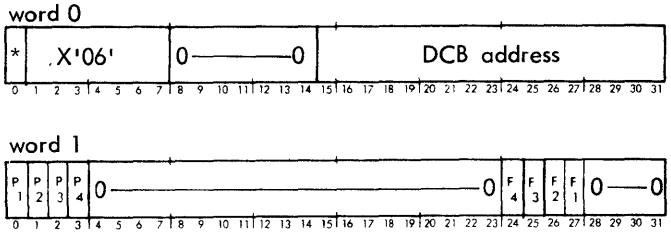
where fpt points to word 0 of an FPT. The code in the first byte of word 0 is as follows:

| FPT Code | Function |
|---|---|
| X'06' | UNLOCK |
| X'07' | DEFINELIST |
| X'08' | PUT |
| X'09' | GET |
| X'0A' | STATS |
| X'0B' | PURGE |
| X'0C' | LOCK |

The various FPT formats are described in the sections that follow.

### QUEUE UNLOCK REQUEST

The format of the FPT for the UNLOCK request is:




where

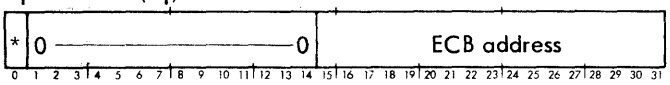
$F_1 = 1$ means WAIT option specified.

$F_2 = 1$ means BACKUP option specified.

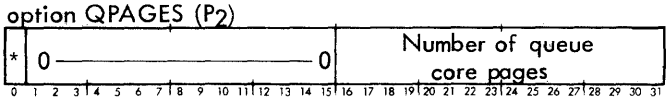
$F_3 = 1$ means NEW option specified.

$F_4 = 1$ means RECOVER specified.

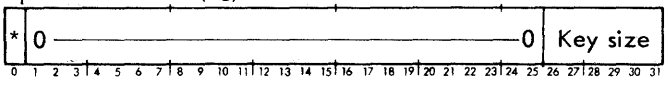
option ECB ($P_1$)



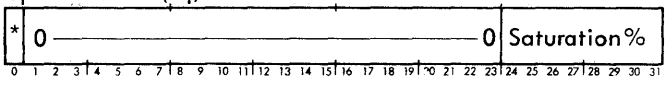

option QPAGES ($P_2$)




option KEYMAX ($P_3$)



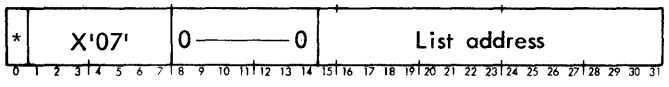

option QSAT ($P_4$)



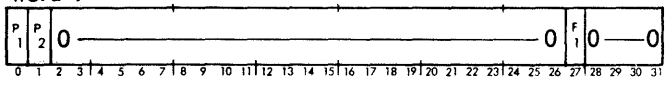

### QUEUE DEFINELIST REQUEST

The format of the FPT for DEFINELIST is:

word 0

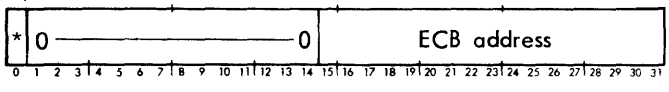



word 1

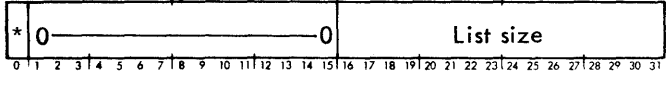



where $F_1 = 1$ means WAIT option specified.

option ECB ($P_1$)



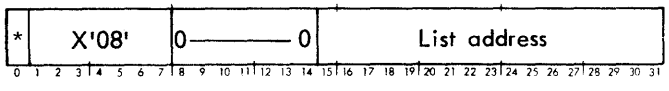

option List Size ($P_2$)



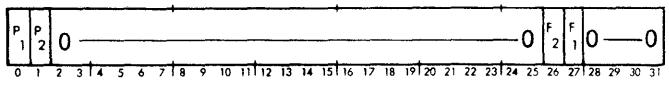

### QUEUE PUT REQUEST

The format of the FPT for PUT is:

word 0




word 1



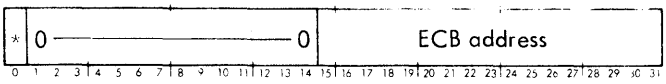

where

$F_1 = 1$ means WAIT option specified.

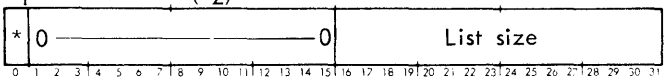
$F_2 = 0$ means low priority request.

$F_2 = 1$ means high priority request.

option ECB ($P_1$)



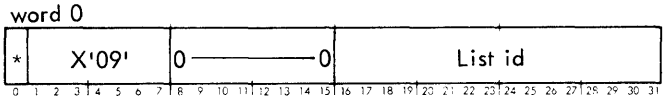

option List size ($P_2$)



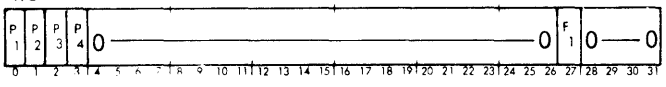

### QUEUE GET REQUEST

The format of the FPT for GET is:

word 0

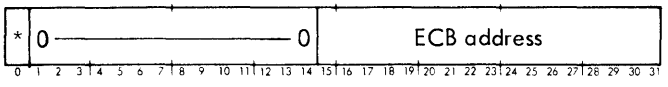



word 1

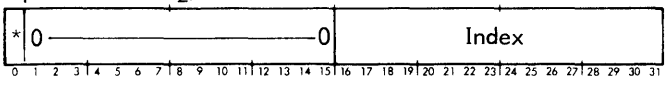



where $F_1$ = 1 means WAIT option specified.

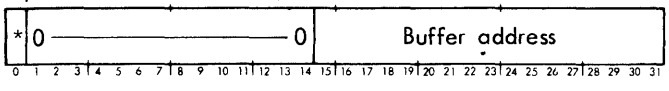
option ECB ($P_1$)



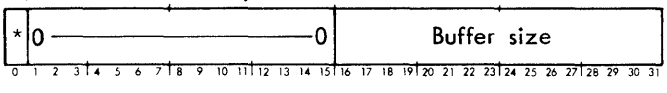

option Index ($P_2$)




option Buffer address ($P_3$)



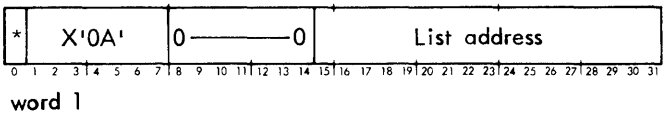

option Buffer size ($P_4$)



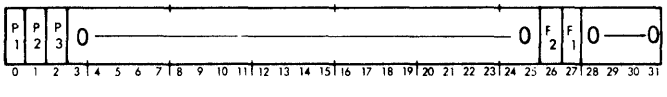

### QUEUE STATS REQUEST

The format of the FPT for STATS is:

word 0




word 1

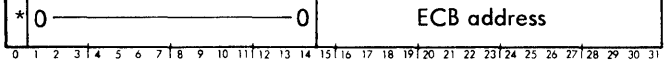



where

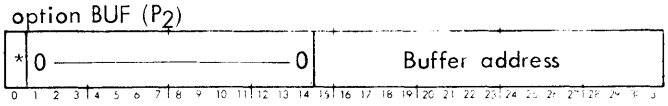
$F_2$ = 1 means COUNT option specified.

$F_1$ = 1 means WAIT option specified.

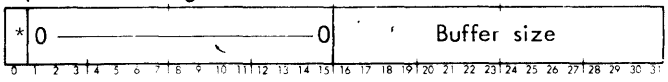
option ECB ($P_1$)




option BUF ($P_2$)



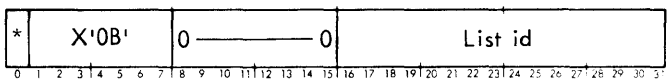

option BSIZE ($P_3$)



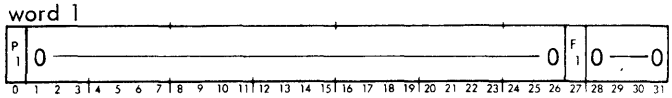

### QUEUE PURGE REQUEST

The format of the FPT for PURGE is:

word 0

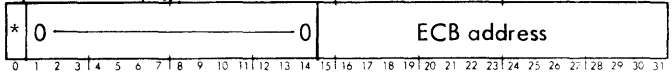



word 1




where $F_1$ = 1 means WAIT option specified.

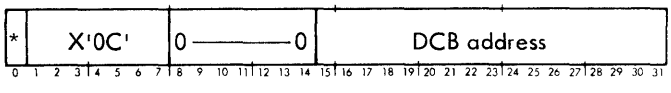
option ECB ($P_1$)



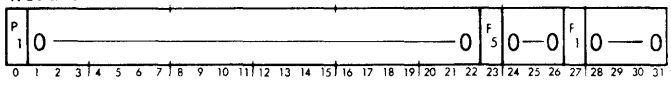

### QUEUE LOCK REQUEST

The format of the FPT for LOCK is

word 0




word 1

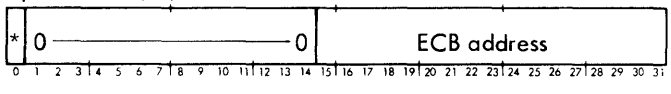



where

$F_1$ = 1 means WAIT option specified.

$F_5$ = 1 means PAUSE option specified.

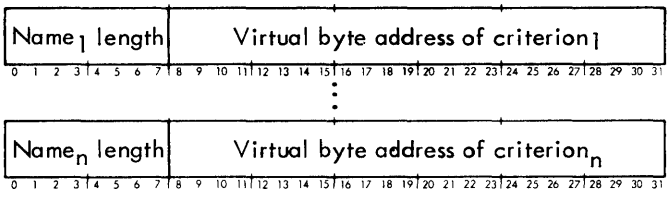
option ECB ($P_1$)




## LIST FORMATS

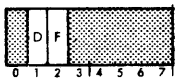
### DEFINELIST OR STATS LIST

The format of the DEFINELIST or STATS list is:



The criterion is in TEXT format with periods, followed by a flag byte. At least one period must appear in the criterion name. (The format of criterion is described in the CP-V/TP Reference Manual, 90 31 12.)

The flag byte has the format



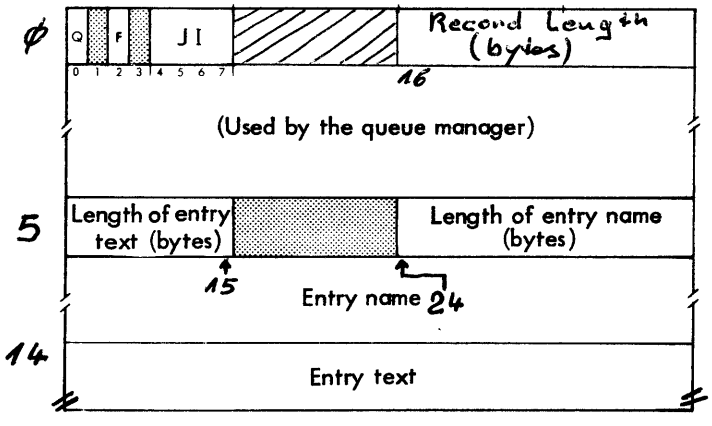

where

F is set to one if failed entries are acceptable (i.e., the system is to GET the transaction regardless of whether or not it was successful).

D is set to one if the entry is to be destroyed after it has been read.

### GET MESSAGE

The format of the GET message is:




where

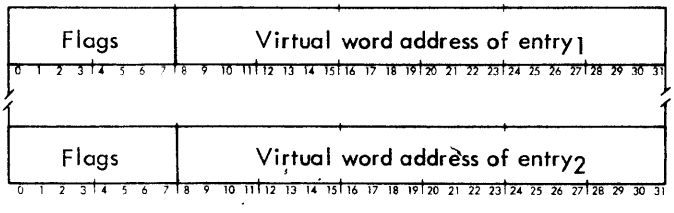
Q indicates queued and is always set to one.

F indicates failed, if set to one.

JI are journalization indicators. Although these bits are kept in this status byte, the queue manager does not use this information. The information is stored here for use by other transaction processors.

### PUT LIST

The format of the PUT list is:




where the entries are in journal record format. (See the GET message, above, for this format.)

The first four bits of the flags field have the following meaning:

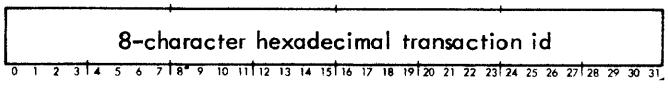
| Q | - | F | P | |
|---|---|---|---|---|
| 0 | - | - | 1 | Delete in-progress entry. |
| 1 | - | 0 | 0 | Insert an entry into the queue. |
| 1 | - | 1 | 1 | Mark an entry failed. |
| 1 | - | 0 | 1 | Put an in-progress entry back into the queue. |
| 1 | - | 1 | ϕ | Insert a pre-failed entry into the queue. |

The JI field contains journalization indicators. Although these bits are kept in this status byte, the queue manager does not use this information. The information is stored here for use by other transaction processors.

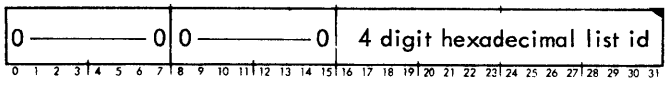
## M:QUEUE PROCEDURE OUTPUT PARAMETERS

### SR1 INFORMATION

UNLOCK: Transaction id returned in SR1

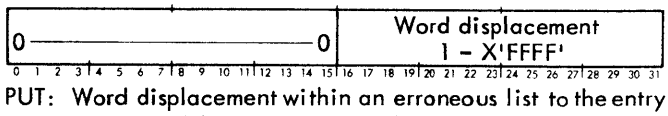



DEFINELIST: List-id returned in SR1



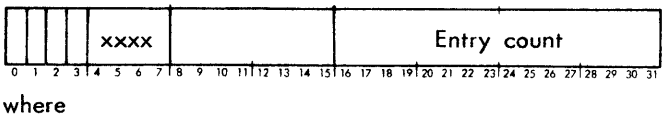

GET: Word displacement within the list to the criterion for which an entry has been stored in the caller's buffer. The format of the entry itself is given in the List Formats section.

Displacement returned in SR1




PUT: Word displacement within an erroneous list to the entry in error. (The SR1 format is the same as for GET.) If no errors occur, SR1 is meaningless.

STATS: The status of a queue entry and, optionally, a count of such entries are returned in SR1




where

Bit 0 = 1 means entry queued.

Bit 1 = 1 means reserved.

Bit 2 = 1 means entry in failed status.

Bit 3 = 1 means entry in progress, i.e., given to a TIC or TPC.

### ECB INFORMATION

ECB completion codes for a posted GET request are:

X'00' An entry is present. Request it.

X'01' An entry has been placed in the caller's buffer.

An unposted ECB indicates that a queue matching entry is unavailable. Error codes are returned in the ECB for requests using the ECB option.

### CONDITION CODE SETTINGS

When the M:QUEUE procedure is performed, the following condition code settings may result:

| CC1 | CC2 | Status |
|---|---|---|
| 0 | 0 | Normal return. |
| 1 | - | Queue unavailable or request cannot be satisfied. |
| - | 1 | ECB wait is meaningful. |

### QUEUE ERROR CODES

Errors detected by the system Queue Manager result in error notification to the caller or a user abort. The error code for M:QUEUE CALs is X'BC'. The code is communicated to the caller in SR3 and, if the ECB option is specified, in the ECB. The code is contained in byte 0 of SR3, and the subcode is contained in bits 8-14, ~~and the content of the EPT word 0, bits 15-31 is returned in the rightmost 17 bits of SR3. Therefore SR3 may contain the dcb address, listhead or listid depending upon the queue request~~. The error subcodes are listed in Table 44.

Table 44. M:QUEUE Error Subcodes

| Subcode | Meaning |
|---|---|
| 01 | Illegal queue service requested (e.g., an unlock is requested and the queue is not locked or the caller is not an authorized TP user). The task is aborted. |
| 02 | An event not associated with the queue has occurred for the user (e.g., M:INT, abort, ESCape or BREAK). |

Table 44. M:QUEUE Error Subcodes (cont.)

| Subcode | Meaning |
|---|---|
| 03 | Error return from get physical work page (abort during unlock processing only). |
| 07 | Queue saturated; i.e., index core space or queue secondary storage space is unavailable. |
| 08 | Queue lock or unlock caller does not have the required privilege. The task is aborted. |
| 09 | DCB not open for a lock or unlock request. The task is aborted. |
| 0A | Space is not available to define a list. |
| 10 | Bad memory address. The task is aborted. |
| 11 | Queue locked. |
| 12 | Queue physical page space is not available. |
| 13 | Error in the list presented for a PUT, DEFINELIST, or STATS request. |
| 14 | Entry not found for a queue request requiring an existing entry. |
| 15 | I/O error during control/index transfer for an unlock request. The task is aborted. |
| 16 | I/O error during a data block transfer. |
| 17 | Queue busy. |
| 20 | Queue GET or PURGE request for a non-existent GET list. |

# APPENDIX A. OPERATIONAL LABELS

Table A-1. Standard Operational Labels and Default Device Assignments

| Operational Label | Batch Device | On-Line Device | Ghost Device |
|---|---|---|---|
| C | Card reader | Terminal | Operator's console |
| OC | Operator's console | Terminal | Operator's console |
| LO | Line printer | Terminal | Line printer |
| LL | Line printer | Terminal | Line printer |
| DO | Line printer | Terminal | Line printer |
| PO | Card punch | None | Card punch |
| BO | Card punch | None | Card punch |
| LI | Card reader | None | Operator's console |
| SI | Card reader | Terminal | Operator's console |
| BI | Card reader | None | Operator's console |
| SL | Line printer | Terminal | Line printer |
| SO | Card punch | None | Card punch |
| CI | Card reader | None | Operator's console |
| CO | Card punch | None | Card punch |
| AL | Card punch | None | Card punch |
| EI | Card reader | Terminal | Operator's console |
| EO | Card punch | None | Card punch |
| UC | Operator's console | Terminal | Operator's console |

Table A-2. Batch Assignment of Operational Labels

| Device | Oplabel |
|---|---|
| Line printer | LO, LL, DO, SL, LP |
| Card reader | C, LI, SI, BI, CI, EI, CR |
| Card punch | PO, BO, SO, CO, AL, EO, CP |
| Operator's console | OC, UC |
| 9-track magnetic tape | 9T |
| 7-track magnetic tape | 7T |
| Default tape | MT |
| None | NO, ME |

Table A-3. On-Line Assignment of Operational Labels

| Device | Oplabel |
|---|---|
| User's terminal | C, OC, LO, LL, DO, SI, SL, EI, UC, ME, CR |
| Card punch | CP |
| Line printer | LP |
| 9-track magnetic tape | 9T |
| 7-track magnetic tape | 7T |
| Default tape | MT |
| None | NO, PO, BO, LI, BI, SO, CI, CO, AL, EO, PR, PP |

# APPENDIX B. PHYSICAL DEVICE NAMES

A physical device name is indicated by yyndd.

where

yy specifies the type of device (see Table B-1).

n specifies the IOP letter for Sigma computers (see Table B-2) or cluster/unit for the Xerox 560 (see Table B-3).

dd specifies the device number (see Table B-4), in hexadecimal.

Table B-1. Standard I/O Device Type Codes

| yy | Device Type |
|---|---|
| 7T | 7-track magnetic tape |
| 9T | 9-track magnetic tape |
| CP | Card punch |
| CR | Card reader |
| TY | Typewriter |
| LP | Line printer |
| DP | Disk pack |
| DC | Magnetic disk |
| ME | CP-V terminal |
| RB | Remote processing data set controller |
| XP | Optical character printer |
| MO | Message mode communications equipment |
| MC | Remote assist terminal (maintenance console) |

Table B-2. Sigma IOP Designation Codes

| IOP Letter (n) | Unit Address |
|---|---|
| A | 0 |
| B | 1 |
| C | 2 |
| D | 3 |
| E | 4 |
| F | 5 |
| G | 6 |
| H | 7 |

Table B-3. Xerox 560 Cluster/Unit Matrix

| Unit Number | Cluster Number | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | A | B | H | N | T | Z | 5 | * |
| 1 | $ | C | I | O | U | 0 | 6 | * |
| 2 | # | D | J | P | V | 1 | 7 | * |
| 3 | @ | E | K | Q | W | 2 | 8 | * |
| 4 | : | F | L | R | X | 3 | 9 | * |
| 5 | * | G | M | S | Y | 4 | ␣ | * |
| 6 | * | * | * | * | * | * | * | * |
| 7 | * | * | * | * | * | * | * | * |

*Reserved

Table B-4. 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 9 through F). |

# APPENDIX C. CP-V SCREECH CODES

### Screech Code – 01

Called From: SCHED, MM

Message: USERS – PAGE CHAIN INCONSISTENT

Registers: When called from SCHED:

R0 - 0 if circular or unlinked chain; otherwise, the Link number index in chain.
R1 - Link register.
R2 - Next page chain link.
R4 - User being scheduled.
R7 - Address of Chain Head, Tail, and Count Table.
SR4 - Offending page number.

When called from MM (T:XPGVI):

R1 - Zero.
R3 - Physical page number.
R7 - Virtual page number.

Remarks: The requested virtual page in the user virtual map chain (JB:LMAP) can't be found. See T:PGCHK in SCHED. Effective when SS1 set.

### Screech Code –02

Called From: SCHED

Message: REPORTED EVENT INCONSISTENT WITH USER'S CURRENT STATE

Registers:
R3 - Previous state.
R4 - User number (T:RE, T:RCE).
R5 - User number (T:RUE).
R6 - Event number.
R7 - Line number (T:RCE).
SR4 - Return address for reschedule.

Remarks: The contents of R3 through R7 are dependent upon the called entry point. If R4 = S:CU, the call was T:RE. If R7 is the line number of the user in R4, the call was T:RCE. If R4 = R5, the entry is T:RUE.

### Screech Code – 0A

Called From: DPSIO, TSIO

Message: OPCODE IN SWAP COMMAND CHAIN IS INVALID

Registers: Case 1, command list security checks – SS4 set:

R1 - Incorrect command list order code if not equal to R3.
R2 - Incorrect command list entry address (IOCD).
R3 - Order code of first IOCD in command list.
R4 - Swap device index.
R6 - Command list beginning address.
R7 - Swapper function code.

Case 2, Unrecoverable read error during inswap:

R1 - Inswap user number.
R7 - DCT index.
SR1 - Incorrect command list entry address (IOCD).
D1 - Order code.

**Screech Code – 0B**

Called From: DPSIO, TSIO

Message: INCORRECT ORDER CODE IN SWAP COMMAND LIST

Registers: R1 - Incorrect order code; not seek.
R2-R7 - See case 1 of screech code 0A above.

Remarks: SS4 must be on for check.

**Screech Code – 0C**

Called From: DPSIO, TSIO

Message: ATTEMPT TO SWAP MONITOR'S MEMORY

Registers: R1 - Buffer address.
R2-R7 - See case 1 of screech code 0A above.

Remarks: SS4 must be on for check.

**Screech Code – 0D**

Called From: TSIO

Message: HALT FLAGS MISSING IN SWAP COMMAND LIST

Registers: R0 - FLAGS byte from TIC command.
R1 - TIC order code.
R2-R7 - See case 1 of screech code 0A above.

Remarks: SS4 must be set to check. FLAGS must not have command chaining set and must have interrupt-on-zero-byte-count or channel-end set.

**Screech Code – 0E**

Called From: TSIO

Message: I/O REQUEST WITH NULL COMMAND LIST

Registers: R4 - Swap device index.
R6 - Command list beginning address.
R7 - Swapper function code.

Remarks: Not checked for pack-only swappers.

### Screech Code – 0F

Called From: DPSIO, TSIO

Message: INPUT FUNCTION CODE IS INVALID

Registers: R2 - Swapper function code.
D4 - X'0F'.

Remarks: SS4 must be on to check. Function code not between one and five exclusively.

### Screech Code – 10

Called From: COC, ECBBLK

Message: BAD COC BUF POOL, OR BAD BUF ADR ON RELEASE REQUEST

Registers: R2 - Logical line number.
R4 - Buffer address.
R6 - Return address from buffer return call.

Remarks:
1. On a COC buffer release, an invalid relative buffer address was specified (address 15 or HRBA'4 + 15).
2. On a COC buffer GET or RELEASE, an invalid relative buffer address was found in the free pool chain. If the COC module was assembled with the COCGBUG and COCPBUG flags set (normally they're not), and sense switch 4 is set, the entire free pool chain is checked on each PUT and GET operation. (The R4 and R6 contents listed above are valid only at entry and exit times.)

### Screech Code – 11

Called From: COC

Message: INVALID INTERNAL CONTROL CODE TRANSLATE REQUEST

Registers: R1 - DCB address.
R2 - Line number.
R5 - Character.
R7 - Byte address of user buffer.
SR2 - Return address.
SR3 - Output translation table address.

Remarks: The cause is a translate table error (e.g., 2741 N/L on non-2741 line), or a bad input buffer chain. R1, R7, and SR4 are not always set.

### Screech Code – 12

Called From: COC

Message: COC – BAD INPUT BUF LINKAGE ON RELEASE REQUEST

Registers: R0 - Removal point.
R1 - DCB address.
R2 - Line number.
R3 - COC number.
R4 - Current release point.
SR3 - Output translate table address.

SR4 - Callei's return; RTN+1 = activation.
D3 - Return address.

Remarks: The COC input buffers are being released, and there is a conflict between the insertion and removal points and the chain. R0, R1, R3, and R4 are not always set.

**Screech Code – 13**

Called From: COC

Message: COC – OUTPUT BUF LINKAGE OR CHARACTER COUNT BAD

Registers:
R1 - DCB address.
R2 - Line number.
R3 - COC number.
R4 - Removal point (usually negative).
R5 - Character.
SR4 - Output count; usually = -1.

Remarks: The output count and buffers are inconsistent. This may be caused by extended interrupt pulse or clobbered COC tables – usually COCOC, COCOI, or COCOR. R1 is not always set.

**Screech Code – 14**

Called From: THEUNCOC

Message: COC ROUTINE CALLED IN NON-COC SYSTEM

Registers:
SR2 - BAL adr if 14-03.
SR4 - BAL adr if 14-01 or 14-02.
D4 - BAL adr if 14-04.

Remarks: The subcode indicates which routine was called:

| | |
|---|---|
| 14-01 | COCIO |
| 14-02 | COCOFF |
| 14-03 | COCSENDX |
| 14-04 | ECHOCR2 |

**Screech Code – 17**

Called From: IOQ

Message: INVALID DISK ADDRESS PASSED FOR AN I/O INSTRUCTION

Registers:
R1 - IOQ7,R3 = DCTX = 0.
R2 - DCB address.
R3 - Queue index.
SR1 - Seek address from CDA,R2.
D4 - X'17'.

Remarks: Caused by an invalid DCT index. R2 and SR1 are not always set. If the invalid address is on a RAD or disk, DSCVT will have been called and R2 and SR1 will be set.

**Screech Code – 19**

Called From: BUFF

Message: INVALID BUFFER ADDRESS PASSED FOR RELEASE

Registers:
R1 - Index to BUFLIMS.
R2 - Head of respective buffer pool.
R5 - JIT address.
SR4 - Link return address.
D3 - Buffer address.
D4 - X'19'.

Remarks: Occurs both on releasing and acquiring buffers of most types (CPOOL, SPOOL, and MPOOL).

**Screech Code – 1A**

Called From: CLS

Message: ACCOUNT DIRECTORY INACCESSIBLE

Registers: –

Remarks: The account directory is bad and the monitor is unable to reconstruct it. All files are lost.

**Screech Code – 1B**

Called From: Swapper

Message: USERS PAGE CHAIN NON ZERO AT SWAP COMPLETION

Registers:
R1 - Inswap user number (S:ISUN).
R2 - Physical byte address of JIT.
R3 - UB:US, 1 (user state).
R4 - Physical page head.
R5 - Physical page tail.
R6 - Physical page count.
SR4 - Count of swapper free page chain (S:FPPC).

Remarks: Swappers' free page pool must be nonzero at end of inswap. S:FPPH, S:FPPT contain head and tail of pages just allocated to the inswap user.

**Screech Code – 1D**

Called From: T:OV

Message: REQUESTED OVERLAY NUMBER IS OUT OF RANGE

Registers:
R2 - Overlay name.
R3 - Overlay name.
R4 - 0.
D4 - X'1D'.

Remarks: Requested monitor overlay is not in shared processor table.

**Screech Code – 1F**

Called From: SWAPPER

Message: NOT ENOUGH PAGES TO PERFORM THIS SWAP

Registers: R3 - Page to release.
SR1 - Deficient page count.

**Screech Code – 21**

Called From: MM

Message: ATTEMPT TO SET ACCESS CONTROLS ON NON-EXISTENT VIRTUAL PAGE

Registers: R6 - Number of pages to set.
R7 - Virtual page number.
SR4 - Link register.

**Screech Code – 22**

Called From: TYPR

Message: PRIVATE VOLUME ALLOCATION ERROR

Registers: R2 - SN count.
R3 - DCB volume number.
R4 - SYSID (0 = EXCLusive use).
R6 - DCB address.
SR4 - Return address.
D2 - DCB:SNT.
D4 - X'22'.

Remarks: Error in allocation. The specified entry in AVRTAB is not found or has bad flags.

**Screech Code – 23**

Called From: CSE57, CSE59, CSEX560, CSECOM

Message: INVALID ENTRY TO CSE HANDLERS

Registers: –

Remarks: Entry was made to an unused slot of the CSE branch vector for this machine.

**Screech Code – 24**

Called From: CSEHAND

Message: INSTRUCTION EXCEPTION TRAP IN MASTER MODE

Registers: –

Remarks: A trap X'4D' occurred while in the master mode. A slave mode trap causes a normal user job step abort. All relevant information is in the in-core error log buffer.

**Screech Code – 25**

Called From: CSEHAND

Message: UNRECOVERABLE WATCHDOG TIMER TRAP

Registers: –

Remarks: Sigma 9 and Xerox 560 systems will attempt recovery from watchdog timer traps resulting from I/O instructions without screeching. All relevant information is in the in-core error log buffer.

**Screech Code – 26**

Called From: CSEHAND

Message: CSE TRAP DURING MFI, PFI HANDLING

Registers: –

Remarks: During MFI handling on a Sigma 9 or during MFI or PFI handling on a Xerox 560, a CSE trap (X'46', X'4C', X'4D') occurred. All relevant information is in the in-core error log buffer.

**Screech Code – 27**

Called From: CSEHAND

Message: PROCESSOR FAULT INTERRUPT

Registers: –

Remarks: A processor fault interrupt interrupt occurred for which continued operation is unlikely. All relevant information is in the in-core error log buffer. (Xerox 560 systems only.)

**Screech Code – 28**

Called From: CSEHAND

Message: MEMORY PARITY ERROR – MEMORY ALTERED

Registers: –

Remarks: A memory parity error correction caused memory to be altered. Continuation without recovery is not possible. Caused by interrupt X'56' on Sigma 6 or 7 or trap X'4C' in Sigma 9 or Xerox 560. All relevant information is in the in-core error log buffer.

**Screech Code – 29-00**

Called From: CSEHAND

Message: TRAP 4C – BUS CHECK FAULT

Registers: –

Remarks: A Sigma 9 bus check fault or a Xerox 560 miscellaneous trap X'4C' occurred while in the master mode. All relevant information is in the in-core error log buffer.

**Screech Code – 29-01**

Called From: CSEHAND

Message: TRAP 4C – MAP PARITY ERROR

Registers: –

Remarks: A map register parity error occurred on a Sigma 9 or Xerox 560 while in the master mode. All relevant information is in the in-core error log buffer.

**Screech Code – 29-02**

Called From: CSEHAND

Message: TRAP 4C – REGISTER BLOCK PARITY ERROR

Registers: –

Remarks: A register block parity error occurred on the Xerox 560 while in the master mode. All relevant information is in the in-core error log buffer.

**Screech Code – 29-03**

Called From: CSEHAND

Message: TRAP 4C – WRITELOCK REGISTER PARITY ERROR

Registers: –

Remarks: A write lock register parity error occurred on the Xerox 560 while in the master mode. All relevant information is in the in-core error log buffer.

**Screech Code – 2C-00**

Called From: ADD

Message: BATCH SCHEDULING ERROR – MBS/CCI ERROR

Registers:
R1 - (S:CUN) current user number.
R2 - Device type.
R3 - Context block address.
R5 - 0.
R6 - User's DCB address (M:C).
SR2 - OPNLD + .14.
SR3 - Context block address.
SR4 - OPNLD + .40.
D1 - BA (OPNLD + .1E7) + .28.
D2 - BA (CONTXT BLK + SCFQARGS) + .28.
D3 - Device type mnemonic text.

Remarks: Register contents significantly different from above indicate the monitor wandered into GETI in ADD. Otherwise, a batch user has been created and has read a card before MBS selected him to be run. Actually all recorded 2C's have been CCI attempting to start a second job. Problem is either CCI read past FIN or a MBS/GETI communication problem (e.g., GIB:UN clobbered).

### Screech Code – 2D-00

Called From: COOP

Message: COOPERATIVE BUFFER MANAGEMENT ERROR

Registers:
R1 - BUFLIMS index for screech code 19.
R2 - .BC11.
R3 - Context block.
SR4 - COOP + .18D.
D3 - 0.

Remarks: At context block initialization a buffer was allocated for the context block. This buffer has been lost through core clobbering or mismanagement of a buffer chain. The particular user cannot continue.

### Screech Code – 2D-01

Called From: COOP

Message: SYMBIONT/COOP FILE DEVICE INACCESSIBLE

Registers:
R0 - COOP + .19B.
R1 - Context block physical address.
R4 - (DCT3(DCTX)) will appear in the format XX1X XXXX.
SR4 - COOP + .15C.
D1 - .XXFF0300 + DCTX (X means could be any value).
D2 - BA (COOP BUFFER).
D3 - .400.
D4 - Disk address.

Remarks: The symbiont/coop file device containing this user's file is down. If there are many file devices for symbiont/coop only, this user should be aborted. If only one symbiont/coop file device exists, it is pointless to run the system with that device down.

### Screech Code – 2D-02

Called From: COOP

Message: USERS COOP CONTEXT BLOCK CHAIN LOST

Registers:
R1 - BUFLIMS index for screech code 19.
R2 - .BC10.
SR2 - OPNLD + .137.
SR4 - OPNLD + .139.
D3 - 0.

Remarks: Similar to 2D-00 but detected at context block open time. Particularly alarming because this check immediately follows the code which allocates context blocks.

### Screech Code – 2D-03

Called From: SACT

Message: COOP CONTEXT BLOCK POINTERS CLOBBERED

Register:
R3 - 0.
R6 - User DCB address.
SR1 - FCN in leftmost 8 bits; DCB address in rightmost 24 bits.
SR4 - Exit from COOP.

Remarks: Either J:USCDX or context block 0 (special pointers) were clobbered.

**Screech Code – 2D-04**

Called From: SUPCLS

Message: COOP DATA BUFFERS MISALLOCATED

Registers: D3 - Buffer being released, including spare buffer index in byte 0.
R5 - Context block 0 address and DBPOOL which is the address of the free context buffer list.
R2 - SV:LSIZ.
SR4 - Return address to caller of RCBUFF.

Remarks: An attempt was made to release a COOP data buffer when the free data buffer pool was full. Either the free data buffer pool has been clobbered or too many buffers have been allocated meaning some other COOP data area has been clobbered.

**Screech Code – 2E**

Called From: RDF

Message: POOL BUFFERS LOST – NONE ALLOCATED CURRENTLY

Registers: SR3 - DCB address for which buffer is needed.
D4 - X'2E'.

Remarks: An attempt was made to get an IPOOL or FPOOL buffer, but none were in the free pool and no open DCB had any. Probably either the DCB chain has been clobbered or one or more DCBs have been clobbered.

**Screech Code – 2E-01**

Called From: RA

Message: INCONSISTENCY IN READ-AHEAD TABLES

Registers: R12 - Disk address.

Remarks: An attempt was made to add an AIR block to the tables when it was already there.

**Screech Code – 30**

Called From: PFSR

Message: UNBALANCED POWER ON/POWER OFF INTERRUPT PAIRS

Registers: –

Remarks: Unbalanced power on/power off interrupt pairs, more of one than another (usually power on, or else system would hang in wait; i.e., B $-1).

**Screech Code – 31**

Called From: IORT

Message: INVALID RESOURCE TYPE

Registers: SR4 - ADDRESS + 1 where discovered.

Remarks: Invalid resource type found.

**Screech Code – 32-00**

Called From: IOQ

Message: DCB DOESN'T CONTAIN A VALID DCT INDEX

Registers: R2 – Address of DCB.

Remarks: The DCT index is not present in DCB.

**Screech Code – 34-00**

Called From: TPQ1

Message: TRANSACTION PROCESSING FAILURE

Registers: –

Remarks: The System Queue Manager for transaction processing has discovered an unrecoverable state while processing transactions.

**Screech Code – 41-01**

Called From: RTROOT

Message: FAILED TO FIND USER'S STATE (M:INTSTAT)

Registers: R2 – Address of ICB being checked.

Remarks: Probably results from a state having been added to SCHED without updating the four masks used by the M:INTSTAT routine (WAIT:MASK, EXU:MASK, IOWAIT:MASK, BLCKD:MASK).

**Screech Code – 41-10**

Called From: RTNR

Message: BAD IOEX CALL TO NEWQ

Registers: Set for BALR, 11 NEWQNW.

Remarks: NEWQNW returned to BAL+1.

**Screech Code – 41-11**

Called From: RTNR

Message: UNABLE TO RETURN PRE-EMPTED DEVICE

Registers: –

Remarks: RTNR's call to RMAOV was invalid.

**Screech Code – 43-01**

Called From: CLOCK4

Message: NO ICBS CHAINED INTO RTICBCLKHDR

Registers: –

Remarks: This is probably caused by overwriting lowcore.

**Screech Code – 43-02**

Called From: CLOCK4

Message: ICBCLK FIELD OF ICB NEGATIVE

Registers: R2 - Address of bad ICB.
R10 - Current timer increment.

Remarks: The ICBCLK field of an ICB should never go negative.

**Screech Code – 43-03**

Called From: RTNR, CLOCK4

Message: NO BACK-LINK FOUND IN DE-CHAINED ICB

Registers: R2 - Current ICB (the one being de-chained).
R4 - Forward link (next ICB in chain).

Remarks: A back-link of zero implies that the current ICB is SYSICB1 (the 1-second CLOCK3 ICB). This ICB should never be de-chained (i.e., de-activated).

**Screech Code – 46-21**

Called From: PV

Message: PRIVATE VOLUME LOGIC INCONSISTENCY

Registers: SR4 - Address where error was detected.

Remarks: Numerous modules call PVERR.

**Screech Code – 49**

Called From: TYPR

Message: RESOURCE PREALLOCATION INCONSISTENT WITH REQUESTS

Registers: R3 - 0.
R3 - Reel number.
D4 - X'49'.

Remarks: The user was preallocated the resource (according to his job context), but when the system got to the point of actual allocation, it found that none of that resource was available. Either the job context specifying preallocation is damaged or the system context recording actual allocation is damaged.

**Screech Code – 56**

Called From: MOCIOP

Message: UNABLE TO RELEASE PHYSICAL WORK PAGE

Registers: The registers at the time of the trap.

Remarks: Originates in the MOCIOP module when unable to release a physical work page locked in core during transaction processing I/O on a message-oriented controller (e.g., 7605).

**Screech Code – 61 – (TRAP Cell)**

Called From: INITRCVR

Message: TEL OR CCI HAS TRAPPED

Registers: Registers at time of trap.

Remarks: The trap occurred while operating mapped, slave, and with TEL-in-control set. The subcode is the trap location.

**Screech Code – 62**

Called From: SCHED

Message: USER PROGRAM TOO LARGE FOR PHYSICAL MEMORY

Registers: R0 - Pages freed.
R4 - Inswap user (S:ISUN).

Remarks: R0 > SL:CORE. User got swapped out but now can't fit back in. Pages may be released but not reported. The JIT in-core flag = 0. (UH:FLG X'200'.)

**Screech Code – 63**

Called From: DPSIO

Message: INSUFFICIENT INFORMATION AVAILABLE TO SWAP THIS USER

Registers: R2 - IOCD.
R6 - Command list address.
R7 - Function code.
D4 - X'63'.

Remarks: Insufficient data to compute function, follow-on function code invalid, or flags not set properly. (Disk pack-only swappers.)

**Screech Code – 6A**

Called From: MM

Message: ATTEMPT TO RELEASE VIA M:CVM FROM USER W/O PROPER PRIVILEGE

Registers: R1 - X'80'.
R5 - Address of top of dynamic data or bottom of command.
R6 - Number of pages to release.
R7 - Virtual page number.
SR1 - Number of pages released.
SR2 - First page to release.
SR3 - Increment or decrement to next page.
SR4 - Link.
D1 - CC.
D2 - CC mask.

Remarks: Virtual page outside of user's area (BUP-EUP) was obtained by an M:CVM CAL, but the user lacks required privilege (X'80') to release it.

**Screech Code – 6B**

Called From: MM

Messages: ERROR IN SPARE BUFFER TABLES

Registers: R11 - Address in buffer subroutine within MM (T:GBUF, T:RBUF, etc.) which detected the error.

Remarks: Usually due to bad input from the calling routine.

**Screech Code – 6B**

Called From: SWAPPER

Message: ERROR IN SPARE BUFFER TABLES

Registers: R6 - BA (window page).
R14 - Physical page assigned to window.

Remarks: Page mapped into window is not contained in the spare buffer pool.

**Screech Code – 79**

Called From: ENTRY

Message: MONITOR COMMITTED A STACK TRAP

Registers: Registers at time of trap.

Remarks: Master bit on in PSD, overflow, underflow, or pointer to stack lost.

**Screech Code – 79-01**

Called From: T:OV

Message: MONITOR STACK TRAP

Registers: Registers at time of trap.

Remarks: OSTACK overflow.

**Screech Code – 7C**

Called From: ALTCP

Message: ALTCP CALLED TO SERVICE A CAL THAT DOESN'T BELONG TO ALTCP

Registers:
R3 - Register field of CAL.
R6 - First word PLIST.
R7 - Address of PLIST + 1.
SR1 - Code.
SR4 - Exit address (usually TRAPEXIT).

Remarks: A CAL1,1 or CAL1,2 was passed to ALTCP but should have been handled by CALPROC.

**Screech Code – 7E – (TRAP Cell)**

Called From: INITRCVR

Message: MONITOR HAS TRAPPED

Registers: Registers at time of trap.

Remarks: Subcode is trap location. For traps that occur at locations less than X'8000' (JOVVPA), the 15 cells preceding the trap location and the trap location are stored in the monitor JIT at X'8DFO' – X'8DFF'.

**Screech Code – 87**

Called From: ALLYCAT

Message: ALLOCATION BUFFERS CONTAIN INVALID WORD COUNT

Registers:
R1 - Stack number.
R2 - Stack count.

Remarks: Either low core has been clobbered or someone has changed ALLYCAT's in-core data.

**Screech Code – 88**

Called From: SCHED

Message: ALLYCAT CLOBBERED ONE OF THE ALLOCATION BUFFERS

Registers:
R1 - Stack index.
R3 - Stack count.

Remarks: ALLOCAT end-action has discovered a discrepancy in the granule/cylinder stacks.

**Screech Code – 89-00**

Called From: ALLYCAT

Message: ALLYCAT'S HGP CHAIN CLOBBERED

Registers:
R7 - Invalid HGP chain address.
R9 - ALLOCAT internal link register.

Remarks: ALLOCAT data (HGPs and TABLES) has been destroyed.

**Screech Code – 93**

Called From: DPSIO, TSIO

Message: TDV COMMAND ADDRESS DOESN'T POINT TO COMMAND LIST

Registers: R1 - 0.
SR1 - Command list address from TDV.
SR2 - TDV status.
D2 - Command list pointer (S:BECL,R1).

Remarks: IOP/memory failure; extraneous entry to TSIO/DPSIO not generated within CLIST.

**Screech Code – 94**

Called From: DPSIO, TSIO

Message: COMMAND LIST CLOBBERED DURING WRITE CHECK

Registers: SR1 - Incorrect command list entry address.
SR2 - TDV status.
R12 - Order code from incorrect command list entry.

Remarks: Can't find seek or TIC within next five command list entries following error entry on write or write check.

**Screech Code – 95**

Called From: DPSIO, TSIO

Message: UNRECOVERABLE I/O ERROR READING USER'S JIT

Registers: R1 - Inswap user number (S:ISUN).
R7 - DCT index.
SR1 - Command list address from TDV status.
SR2 - TDV status.

**Screech Code – 96**

Called From: DPSIO, TSIO

Message: UNRECOVERABLE I/O ERROR READING SHARED PROCESSOR

Registers: R1 - Inswap user number (S:ISUN).
R7 - DCT index.
SR1 - Command list address from TDV status.
SR2 - TDV status.

# APPENDIX D. XEROX 560 REMOTE ASSIST STATION

## INTRODUCTION

The Remote Assist Station (RAS) and the associated routines comprise the CP-V interface for on-line remote assistance for both software and hardware analysts. This facility provides an on-line connection to the operating system without requiring the use of any of the normal communications equipment. The RAS user has access to ELLA for listing and analyzing the contents of the system error log file (ERRFILE) and to ANLZ and Delta for examining crash dumps and the running monitor.

## HARDWARE INTERFACE

The Remote Assist Station may be any ASCII terminal capable of connecting to the provided data set (Bell 103A or its equivalent). The data set is connected to the Remote Channel Interface of the System Control Processor. (See the Xerox 560 Computer Reference Manual, 90 30 76.) To use the interface on-line, the REMOTE CHANNEL switch on the Xerox 560 System Control Panel must be in the I/O position. This connects the remote channel to address X'0B' on the MIOP in cluster zero, through which the CP-V interface communicates. This address must be SYSGENed as the Maintenance Control (MC) device. The hardware performs character translation from ASCII to EBCDIC (and vice versa) to make the terminal appear as an EBCDIC device. The translation tables are depicted in Table D-1. The left side of the table shows standard ASCII characters. The corresponding entries in the right side of the table show EBCDIC translation.

## SOFTWARE INTERFACE

CP-V provides an on-line communications interface enabling the remote analyst to log onto the Maintenance Console much as if he were connected to a COC terminal line. The interface is initiated at the Operator's Console (OC) by a special form of the GJOB key-in after the RAS is connected to the dial-up modem:

```
!GJOB LOGON,MC
```

This key-in causes LOGON to print a salutation to the MC resource requesting the RAS user to enter his account and name. The RAS user must be explicitly authorized via Super to use the MC resource. The following is an example of such a Super authorization:

```
-C RAS,ASSISTANCE ↵
--O$PR=A0 ↵          (required for running diagnostic
                      programs)
--OMMC=1 ↵           (authorizes on-line use of the MC
                      resource)
--OM9T=1 ↵           (required for mounting tape dumps)
-- ↵
-END ↵
```

LOGON verifies the OMMC authorization before it accepts the account and name, and will not allow the user to log on if he does not have this resource authorization.

If the user's account and name are accepted, the user is logged on as a non-COC on-line user and LOGON exits to TEL which issues a prompt for input (!).

## PROCESSOR RESTRICTIONS

The MC authorization causes a :PROCS entry to be created for the RAS user which restricts him to the following list of processors:

ANLZ

Delta

ELLA

No other processors or programs are allowed at the RAS. Except for these processor restrictions, TEL will accept most of its commands (e.g., SET, PRINT, MESSAGE, QUIT, GO). However, commands regarding terminal type and status will be ignored.

## COMMUNICATIONS RESTRICTIONS

The communications link to RAS uses a small resident handler in conjunction with the hardwired micro-coded controller to provide a terminal interface. Due to the limitations of the hardware and size restrictions on the software handler, some compromises have been made. The following list outlines the major characteristics of the communications interface:

1. The MC device is a message mode device, requiring either LINE FEED, RETURN, CONTROL X, or CONTROL H to end each input. LINE FEED and RETURN generate an X'15' (NL) character. CONTROL X and CONTROL H generate an X'08' (EOM) character which is used to cancel an input line so that the line may be retyped.

2. Although the RAS terminal is connected in full-duplex, the MC device operates in half-duplex, echoplex mode, allowing I/O transfer a line at a time in only one direction. When a read is pending, characters typed will be echoed to the print mechanism of the

Table D-1. ASCII to EBCDIC Translate Table

| ASCII | | | | | | | | | EBCDIC | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | NUL | DLE | SP | 0 | @ | P | ` | p | 0 | NUL 00 | DLE 10 | SP 40 | 0 F0 | (3) | P D7 | ¢ 4A | P D7 |
| 1 | SOH | DC1 | ! | 1 | A | Q | a | q | 1 | SOH 01 | DC1 11 | ! 5A | 1 F1 | A C1 | Q D8 | A(4) C1 | Q DB |
| 2 | STX | DC2 | " | 2 | B | R | b | r | 2 | STX 02 | DC2 12 | " 7F | 2 F2 | B C2 | R D9 | B C2 | R D9 |
| 3 | ETX | DC3 | # | 3 | C | S | c | s | 3 | ETX 03 | DC3 13 | # 7B | 3 F3 | C C3 | S E2 | C C3 | S E2 |
| 4 | EOT | DC4 | $ | 4 | D | T | d | t | 4 | EOT 04 | DC4 14 | $ 5B | 4 F4 | D C4 | T E3 | D C4 | T E3 |
| 5 | ENQ | NAK | % | 5 | E | U | e | v | 5 | ENQ 09 | NAK 0A | % 6C | 5 F5 | E C5 | U E4 | E C5 | U E4 |
| 6 | ACK | SYN | & | 6 | F | V | f | v | 6 | (1) | SYN 16 | & 50 | 6 F6 | F C6 | V E5 | F C6 | V E5 |
| 7 | BEL | ETB | ' | 7 | G | W | g | w | 7 | BEL 07 | ETB 17 | ' 7D | 7 F7 | G C7 | W E6 | G C7 | W E6 |
| 8 | BS | CAN | ( | 8 | H | X | h | x | 8 | (2) | (2) | ( 4D | 8 F8 | H C8 | X E7 | H C8 | X E7 |
| 9 | HT | EM | ) | 9 | I | Y | i | y | 9 | HT 05 | EM 19 | ) 5D | 9 F9 | I C9 | Y E8 | I C9 | Y E8 |
| A | LF NL | SUB | * | : | J | Z | j | z | A | NL 15 | SUB 1A | * 5C | : 7A | J D1 | Z E9 | J D1 | Z E9 |
| B | VT | ESC | + | ; | K | [ | k | { | B | VT 0B | ESC LB | + 4E | ; 5E | K D2 | I 4F | K D2 | B2 |
| C | FF | FS | , | < | L | \ | l | ¦ | C | FF 0C | FS 1C | > 6B | < 4C | L D3 | ¢ 4A | L D3 | I 4F |
| D | CR | GS | - | = | M | ] | m | } | D | NL 15 | GS 1D | - 6D | 7E | M D4 | ¬ 5F | M D4 | } B3 |
| E | SO | RS | . | > | N | ^ | n | ~ | E | SO 0E | RS 1E | . 4B | > 6E | N D5 | ^ 6A | N D5 | ¬ 5F |
| F | SI | US | / | ? | O | _ | o | DLE | F | SI 0F | US 1F | / 61 | ? 6F | O D6 | – 6D | O D6 | ¢ 4A |

Notes: (1) Used by Diablo Centaur terminal EOT/ACK protocol.

(2) Used to cancel line, echoes as ← and reissues read unless Delta is in control in which case is input as OA (LF).

(3) Causes previously typed character to be ignored (Rubout character).

(4) Lower case input is echoed lower case, but translated to upper case for program input.

terminal. If a read is not pending, characters typed are not echoed and are ignored.

3. The BREAK key may be depressed at any time to indicate a BREAK signal. The MC handler causes a BREAK event to be issued for the user and counts successive BREAKs. If the user issues four successive BREAKs, the handler causes a CONTROL Y event (i.e., an escape to TEL). The BREAK key cancels any current I/O operation to the terminal.

4. In order to detect line drop or disconnect, input requests will time out in three minutes and output messages will time out in 20 seconds. A failure to respond to a read within three minutes causes the RAS user to be logged off.

5. Records output to the RAS terminal have a maximum size restriction of 140 bytes. Trailing blanks in an output record are suppressed by the MC handler. Records output through DCBs other than M:UC have a RETURN/LINE FEED appended to them. Records written through M:UC must contain their own carriage control characters.

6. The MC handler does not simulate tabs nor does it affect pagination.

7. Individual characters may be erased on input by typing @ characters for each character to be erased (e.g., 'ANE@LZ' results in 'ANLZ'). Complete lines may be erased by ending the line with CONTROL X or CONTROL H which causes the handler to echo

   (#) (↵)

   and to reissue the read that was in operation.

8. End-of-file condition is set upon receipt of the three character sequence CONTROL F (^).

9. Lower case letters are echoed in lower case but are input to the program as upper case.

10. When Delta issues a read, speciel action takes place by the handler to simulate the Delta activation character set. Special Activation characters (CONTROL I , ) = /) should be immediately followed by a RETURN or LINE FEED. For commands which usually end with a RETURN, either a RETURN or a LINE FEED is valid. Commands which normally end in LINE FEED should be ended with CONTROL X or CONTROL H. Line erasure is effected by ending the line of input with ? RETURN.

# APPENDIX E. ERRFILE FORMATS

ERRFILE is a keyed file built and updated by ERR:FIL for use by diagnostic programs. The file contains one record for each error entry in the file created by ERRLOG.

The keys for this file contain the Julian date in packed decimal, the time of the error in EBCDIC, and a sequence number for errors with the same time tag. This sequence number is reset to zero for each entry with a new time tag. The format of the key is

| 08 | yy | 0d | dd |
|---|---|---|---|
| h | h | m | m |
| n | | | |

where

08 is the number of bytes in the key.

yy0ddd is the Julian date in packed decimal.

hhmm is the time (hours and minutes) in EBCDIC.

n is the sequence number.

The first record of ERRFILE is the key of the last record in ERRFILE and has a key of zero.

While copying records into ERRFILE, consistency and error checks are made on the input data. If any errors or inconsistencies are found, "copy error" records are written and a "copy error" counter in the summary record is incremented. The error and consistency checks, recovery actions taken, and the format of the copy error records are described below. The terminology used in the error record formats is defined in Table E-1.

Table E-1. Error Record Terminology

| Term | Meaning |
|---|---|
| Account | The doubleword used to identify a user's collection of files. |
| AIO CC | A 4-bit field representing the condition codes as returned by the hardware in response to an AIO instruction. |
| AIO Status | A 16-bit field representing the status as returned by the hardware in response to an AIO instruction. |
| Alternate I/O Address | A 16-bit value representing an alternate physical I/O address by which a dual-access device can be referenced. |
| Bytes Remaining | A 16-bit field representing the Remaining Byte Count (RBC) field as returned by the hardware in response to a TDV instruction. |
| Consecutive, Keyed, Random | Methods of organizing user files in CP-V (refer to the CP-V/BP Reference Manual, 90 17 64). |
| Count of Entries Identical to Previous Entry | The number of error log records which are identical to one previously logged for identical reasons (excludes time records). |
| Count of Entries Lost | The number of error log records lost when logging becomes temporarily impossible for any reason. |
| Current Command Doubleword | A 64-bit value representing the command doubleword currently being processed for a device (indicated by the TDV status DW). |
| Caller's Address | The address back to which the error logging routine is returned when logging is complete; used in isolating software faults. |
| DCT Index | The 8-bit value indicating the order in which the device is configured into the system (at SYSGEN). |
| DCT Index of Symbiont Device | The 8-bit value indicating the order in which the device associated with the symbiont is configured into the system (at SYSGEN). |

Table E-1. Error Record Terminology (cont.)

| Term | Meaning |
|---|---|
| Effective Address | A 32-bit value representing the final address computed for the instruction pointed to by the instruction address (IA) in the PSD. |
| Error Subcode | An 8-bit field indicating which of several types of file inconsistencies has occurred (see CP-V/BP Reference Manual, 90 17 64). |
| File Name | The TEXTC name used to identify a collection of user data on secondary storage. |
| Granule | The unit of secondary storage allocation equal to 2048 bytes (usually 2 sectors). |
| HIO CC | A 4-bit value (bits 0-3 of designated byte) representing the condition codes as returned by the hardware in response to an HIO instruction. |
| HIO Status | A 16-bit value representing the status as returned by the hardware in response to an HIO instruction. |
| I/O Address | A 16-bit value representing the physical I/O address. |
| I/O Count | A 32-bit value representing the number of SIO instructions executed for the device. |
| Julian Day | A 16-bit value representing the Julian day of the year (e.g., March 1 would be represented as X'3D') when the error was logged. |
| Length | A 8-bit value in the second byte of the error log record representing the number of useful 32-bit words contained in the error log record. It includes the first word in the count. |
| Memory Status Words (Sigma 9 only) | Each word is a 32-bit value representing data returned by the hardware in response to an LMS instruction. |
| MFI (Sigma 6 or 7 only) | A 4-bit value representing the current state of the memory fault indicators returned by the hardware in response to an RD instruction. All memory fault indicators will be reset. |
| Mode | A 16-bit value representing the manner in which the file was last referenced (see CP-V/BP Reference Manual, 90 17 64). |
| Model Number | A 16-bit value representing the model number assigned by Field Engineering to uniquely identify peripheral devices (e.g., 7242 would be represented as X'7242'). |
| Number of Parity Errors | A 16-bit value representing the number of bad locations causing memory parity errors (only the first 14 bad locations are entered in the log if the number of errors is greater than 14). |
| Primary I/O Address | A 16-bit value representing the physical I/O address by which a device can be referenced (see Alternate I/O Address). |
| PSD | A 64-bit value representing the program status doubleword. |
| Real Address | A 32-bit value representing the actual memory address (in a mapped system, this is the same as the address in the IA field of the PSD). |
| Recovery Count | An 8-bit value initialized to zero at system initialization and incremented by the value one for every system recovery. |
| Relative Sector Address | A sector is 256 words. Each sector on a given device is numbered zero through device end. CP-V maintains file pointers by relative sector number, thereby simplifying the logic necessary to address different devices. |
| Relative Time | A 32-bit value representing milliseconds since midnight. Resolution is 2 msec. |

Table E-1. Error Record Terminology (cont.)

| Term | Meaning |
|---|---|
| Relative Time Resolution | An 8-bit value, n, such that actual relative time resolution 2 msec. (e.g., n = 1 for a resolution of 500HZ or 2 msec.). |
| Retries Remaining | An 8-bit value representing Retry Request minus the number of entries attempted. The range is between Retry Request and -1. A value of -1 indicates the operation was terminated due to retry count rundown. |
| Retry Request | An 8-bit value representing the maximum number of retries after which a device error is returned to the requester. This value is obtained from the requester's DCB. |
| Screech Code | The code used by CP-V to identify the system failure which has occurred. |
| Screech Subcode | An 8-bit field identifying which type of a specific and similar set of system failures has occurred. (See Software check codes in the CP-V/OPS Reference Manual, 90 16 75.) |
| Seek Address | The physical disk address last used to access this device. |
| Sense Information | The diagnostic information returned from the device as a result of sending a "sense" order to the device. |
| SIO CC | A 4-bit value (bits 0 - 3 of designated byte) representing the condition codes as returned by the hardware in response to an SIO instruction. |
| SIO Status | A 16-bit value representing the status as returned by the hardware in response to an SIO instruction. |
| Site Identification | A 64-bit field containing the site ID from the SYSGEN :MON card left justified with blanks on the right. |
| Startup Type | An 8-bit field indicating which of several types of system initialization was used. See the SYSTEM STARTUP error record ( type X'18'). |
| Subchannel Status | The status of the I/O subchannel received from the hardware as a result of a TDV instruction. |
| Symbiont File | A CP-V system special file for buffering data between the CPU and slower speed line printers, card punchers, etc. |
| TDV CC | A 4-bit value (bits 0 - 3 of designated byte) representing the condition codes as returned by the hardware in response to a TIO instruction. |
| TDV Current Command DA | A 24-bit field representing the current command doubleword address used in obtaining the device status with a TDV instruction. |
| TDV Status Doubleword | A 24-bit field representing the subchannel status, as current command doubleword, device status, and byte count as returned by the hardware in response to a TDV instruction. |
| TIO CC | A 4-bit value (bits 0 - 3 of designated byte) representing the condition codes as returned by the hardware in response to a TIO instruction. |
| TIO Status | A 16-bit value representing the status as returned by the hardware in response to a TIO instruction. |
| Trap CC | A 4-bit value (bits 0 - 3 of designated byte) representing the condition codes as returned by the hardware when certain traps occur. |

Table E-1. Error Record Terminology (cont.)

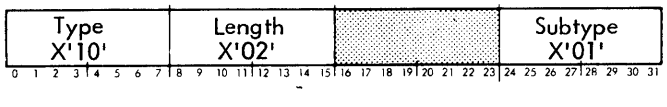
| Term | Meaning |
|---|---|
| Trapped Instruction | A 32-bit value representing the contents of the location pointed to by the instruction address (IA) in the PSD. |
| Type | An 8-bit value in the first byte of the error record which identifies the type of record. |
| Unit Address | A 6-bit value (bits 2 - 7 of designated byte) representing the address by which a processor can be referenced; the value is composed of a 3-bit cluster number followed by a 3-bit unit number. |
| Unit Type | An 8-bit value specifying the type of processor. Bit 0 of the designated byte indicates the presence of the processor in the current operational configuration (0 = present, 1 = not present). |
| User ID | A 16-bit value which is a unique number assigned by the system to the particular job or on-line session. |
| User Number | An 8-bit value which is the index into internal system tables used to access user specific information. |
| Version | The version identifier of the system running (i.e., A00, B00, etc.). This field is one byte in length. The letter of the version is stored in the first four bits and the number of the version is stored in the second four bits. |
| Volume Serial Number | A 4- or 6-byte field supplied by a user to identify either a tape or private pack. |
| Year | A 16-bit binary value representing the current year minus 1900 (e.g., 1973 is represented as X'49'). |

READ ERROR

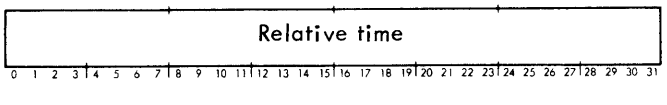
If the condition codes set by T:RDERLOG indicate a read error, a copy error record (Read Error) is written and copying of the record is attempted. If inconsistencies are found in the record, a copy of the bad record is placed in the ERRFILE file, followed by the End Read Error record. If no inconsistencies are found, the record is processed normally and the Read Error record remains in the ERRFILE file. The record formats are

Read Error

word 0

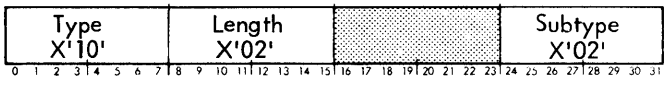



word 1

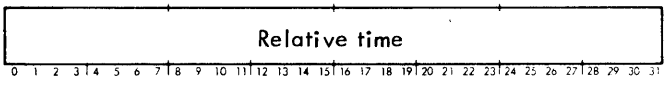



End Read Error

word 0



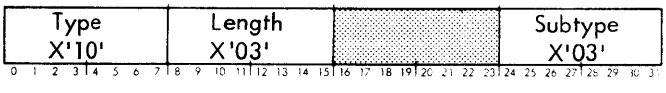

word 1



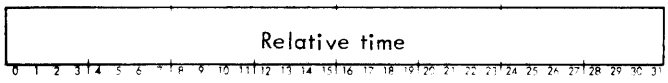

ERRLOG RECORD LENGTH ERROR

If the length of the ERRLOG record is greater than 64 words, a copy error record followed by the ERRLOG record is written on ERRFILE. No attempt is made to copy this record in the detailed format. The record format is

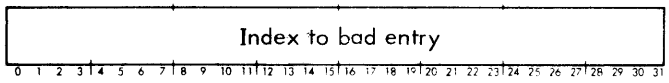
word 0



word 1



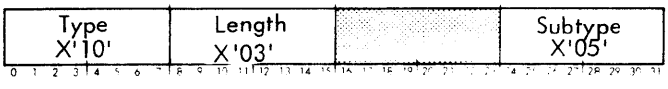

word 2



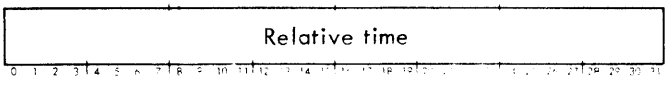

INCORRECT TIME

If the time of an entry is out of sequence, i.e., if it is earlier than the time of the last record and the data has not changed, a copy error record is written on ERRFILE followed by the ERRLOG record. The time of this entry is then used for the key and processing continues. The record format is

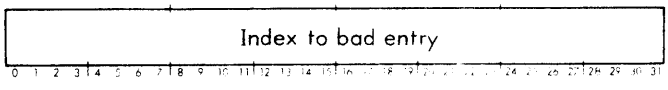
word 0




word 1




word 2

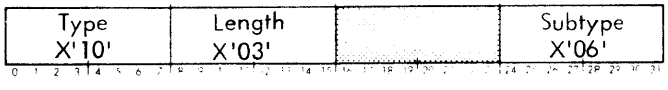



where index is the displacement within the ERRLOG record of the first word of erroneous entry.

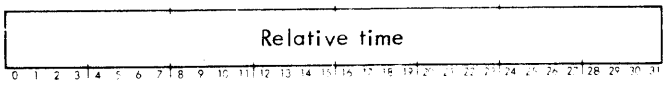
ILLEGAL ENTRY TYPE

If the entry type is not one of the legal types, a copy error record followed by the ERRLOG record is written on ERRFILE. No attempt is made to copy the remainder of the record. The record format is

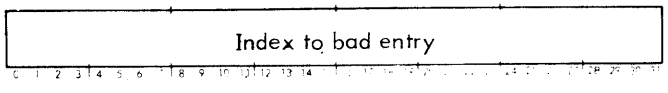
word 0




word 1




word 2




where index is the displacement within the ERRLOG record of the first word of erroneous entry.

Note: Errors that occur while booting have a time tag of 24XX but the keys of these records contain the current date and 0011 for the time.

If read or write errors are detected while reading or writing ERRFILE and SUMFILE, they are ignored.

Whenever I/O errors or certain unusual conditions occur, an entry will be made into the ERRLOG file. This entry will contain any information pertinent to the condition.

Word 0 of each entry will have a code indicating which error or unusual condition is present along with the number of words in the entry (including word 0). Time (hhmm) and Device Name (yyndd) are in EBCDIC.

There are no error log entries for the following two interrupts.

MEMORY FAULT INTERRUPT

The Memory Fault Interrupt (MFI) is triggered when an error is detected during a memory access by either the CPU or an IOP. If the MFI is triggered by the CPU, a parity error trap will also occur unless the error is a Loop Check Parity error or Overtemperature condition. The parity error trap routine performs error recovery, logs the error, and clears the MFI to avoid duplicate processing. The MFI service routine therefore expects to only handle errors detected during an IOP memory access and Loop Check and Overtemperature errors. The Loop Check and Overtemperature errors are processed by the memory parity program and the system recovery program is entered with code X'23'. The other errors are logged by the device handler, which also perform the required recovery.

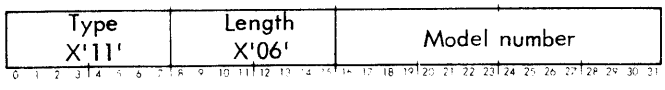
PROCESSOR FAULT INTERRUPT

The Processor Fault Interrupt is not enabled in CP-V. Errors that cause this interrupt in a monoprocessor system are handled by the I/O Interrupt Routines.

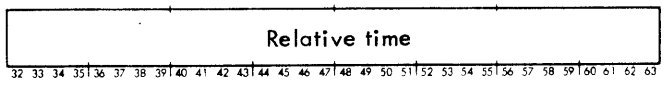
SIO FAILURE

This record is logged when CC1 and/or CC2 are set after execution of the SIO instruction.

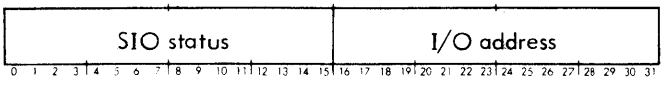
word 0



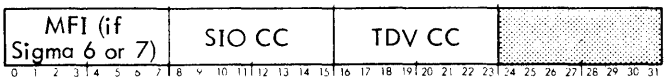

word 1



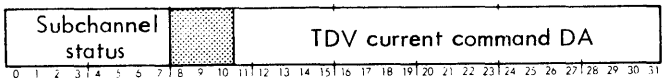

word 2



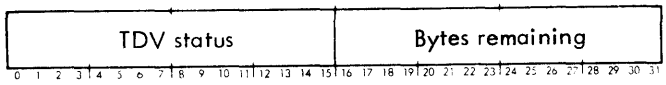
word 3




word 4



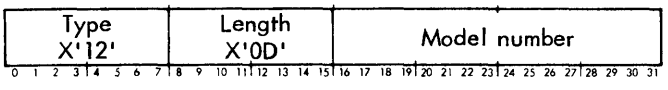

word 5



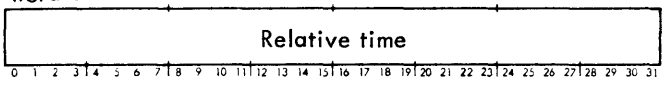

TIME OUT

This record is logged when the I/O interrupt does not occur within a specified time period in response to an I/O instruction.

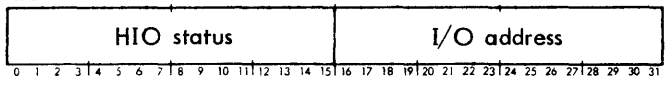
word 0



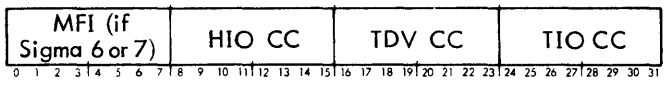

word 1



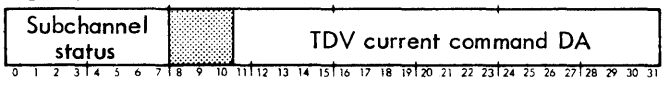

word 2



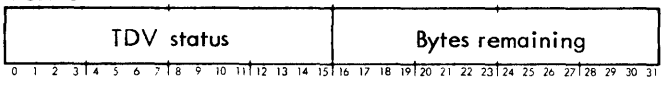

word 3



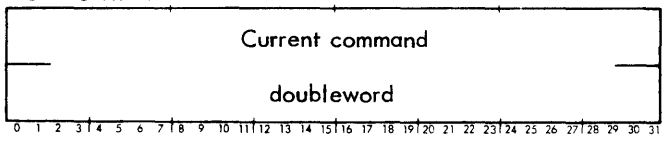

word 4



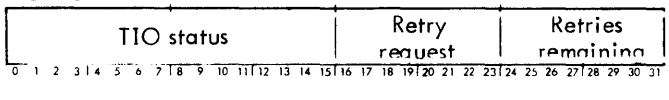

word 5



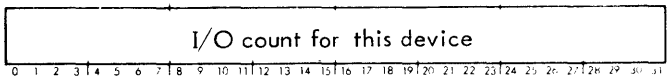

words 6 and 7



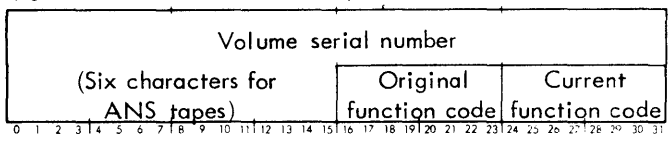

word 8



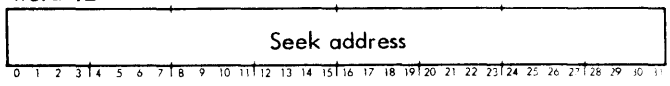

word 9




words 10 and 11



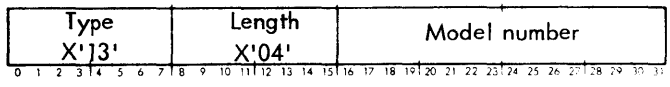

word 12



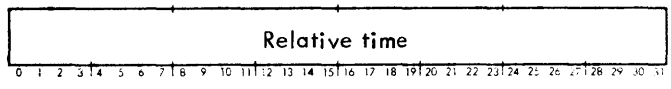

UNEXPECTED INTERRUPT

This record is logged when an interrupt, other than an attention interrupt, is received from a known device for which no I/O operations have been started by the system.

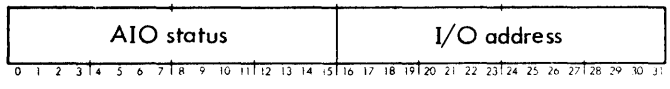
word 0



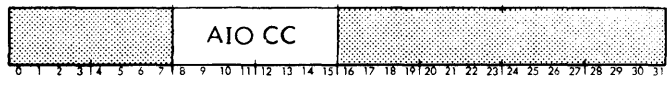

word 1




word 2



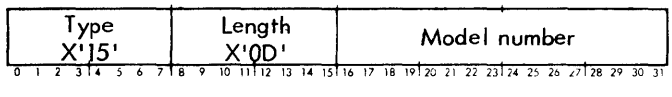

word 3



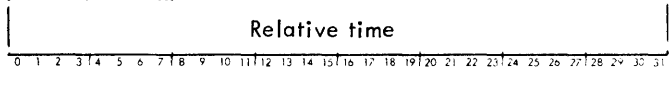

DEVICE ERROR

This record is logged when general analysis of the status received from an AIO, TDV, or TIO indicates an error which resulted from the I/O operation.

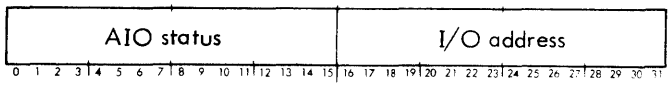
word 0



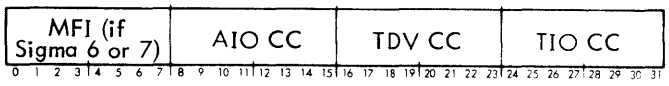

word 1



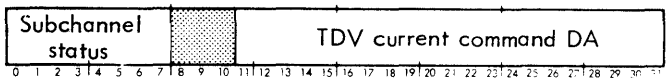
word 2



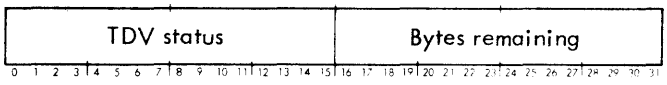

word 3



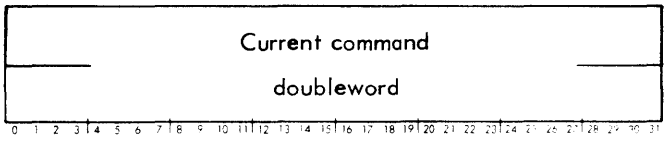

word 4



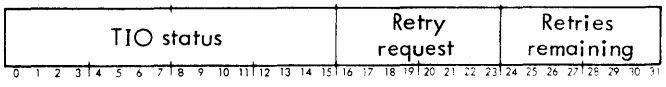

word 5



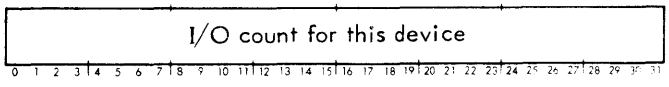

words 6 and 7



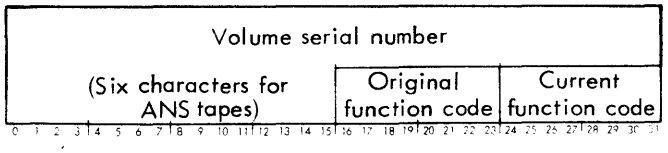

word 8



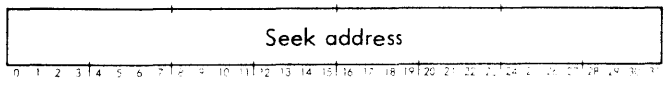

word 9




words 10 and 11



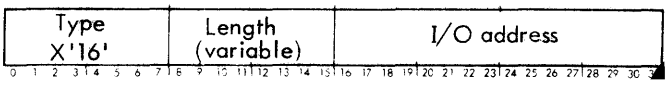

word 12



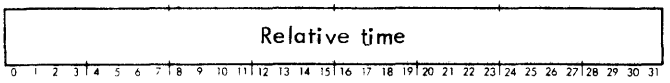

## SECONDARY RECORD FOR DISK PACK, RAD, AND TAPE

This record is generated as a result of a previous device error and contains device status which corresponds to the information contained in the Device Error record (type X'15') preceding this record.

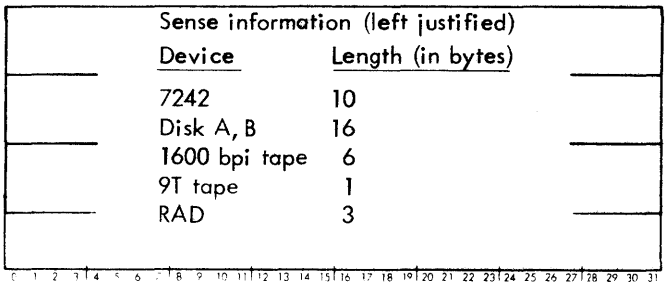
word 0




word 1




words 2 and following

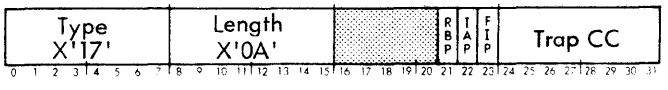



The I/O address links the secondary record to the corresponding device error entry.

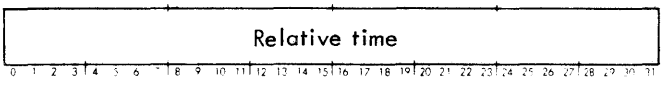
## HARDWARE ERROR

This record is logged when a hardware error has been detected, the type of error being indicated by the Trap CC. For Sigma 6 and 7, this record is generated as a result of the memory parity interrupt associated with location X'56' For Sigma 9 and Xerox 560 this record is generated as a result of the parity error trap associated with location X'4C'.

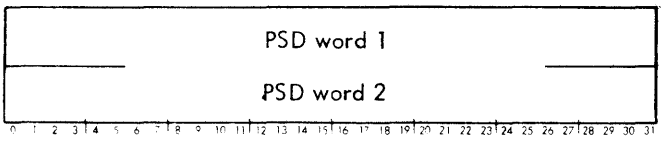
word 0



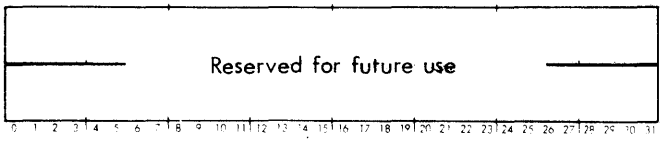

word 1



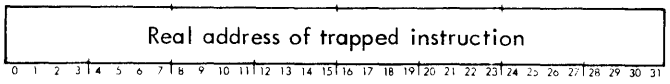

words 2 and 3



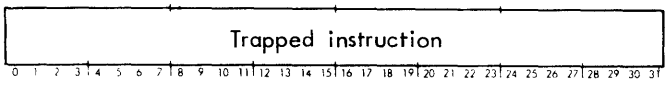

words 4 and 5



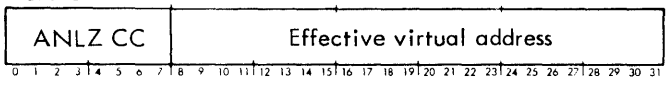
word 6



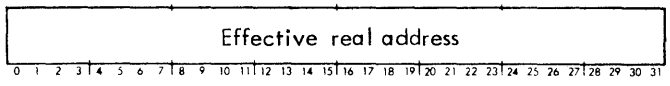

word 7




word 8




word 9




where

FIP indicates, when set, that a parity error occurred while fetching the instruction (causing a trap 4C) on a Sigma 9 or Xerox 560, or that a memory parity occurred (causing a machine interrupt using location 56) on a Sigma 6 or 7.

IAP indicates, when set, that a parity error occurred due to an indirect address fetch. Words 8 and 9 will be zero in this case.

RBP indicates, when set, that a parity error is present in the associated R-block registers. (Xerox 560 only.)

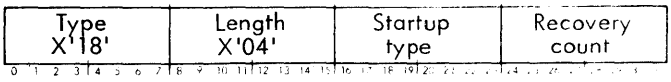
ANLZ CC specify the addressing type for the effective real address (words 8 and 9). If the instruction is an immediate type, these address fields will be zero. The ANLZ CC settings are:

| Bit 0 | Bit 1 | Bit 2 | Bit 3 | |
|---|---|---|---|---|
| 0 | 0 | - | 0 | Byte |
| 0 | 0 | - | 1 | Immediate, byte |
| 0 | 1 | - | 0 | Halfword |
| 1 | 0 | - | 0 | Word |
| 1 | 0 | - | 1 | Immediate, word |
| 1 | 1 | - | 0 | Doubleword |
| - | - | 0 | - | Direct addressing |
| - | - | 1 | - | Indirect addressing |

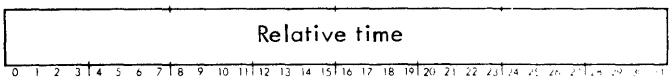
## SYSTEM STARTUP

This error is logged at system initialization and at every recovery.

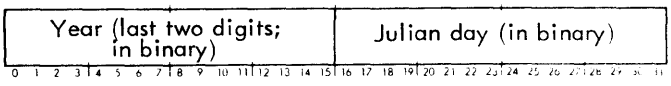
word 0



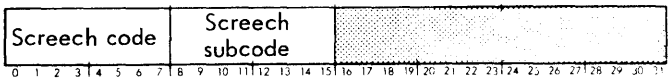

word 1




word 2




word 3




where

screech code and screech subcode are defined in the CP-V/OPS Reference Manual, 90 16 75.

recovery count is set to 0 for initial startup as defined by startup types 1, 2, or 3 below.

startup type specifies the type of startup.

1 - Initial PO boot

2 - PO boot with files

3 - System device boot (no recovery)

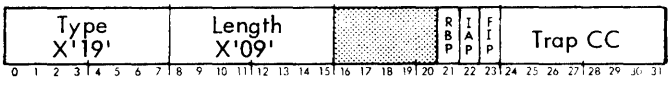
4 - System recovery

5 - Operator recovery

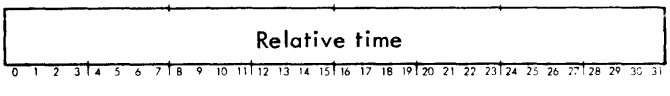
## WATCHDOG TIMER

This record is generated as a result of the instruction watchdog timer runout trap associated with location X'46'.

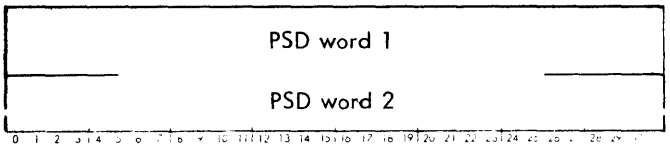
word 0



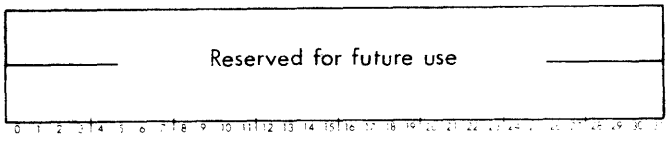

word 1



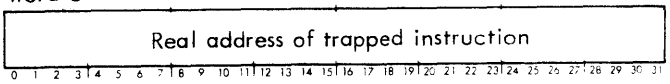

words 2 and 3



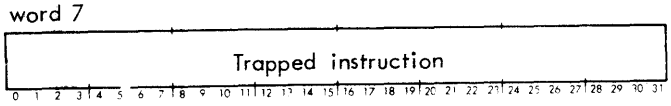
words 4 and 5



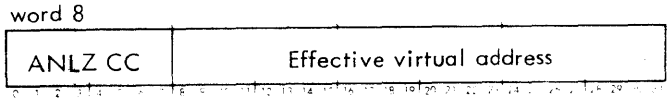

word 6



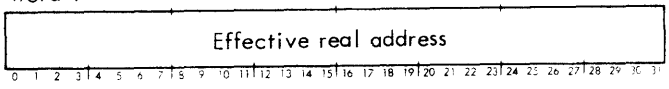

word 7




word 8




word 9

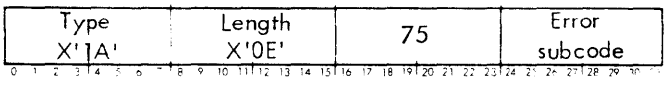



where FIP, IAP, RBP, and ANLZ CC have the same meaning as for the hardware error record (X'17').

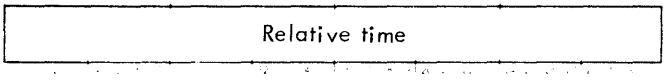
## FILE INCONSISTENCY ERROR

This record is logged if the system detects files which are inconsistent in that the associated file links do not match or are otherwise incorrect.

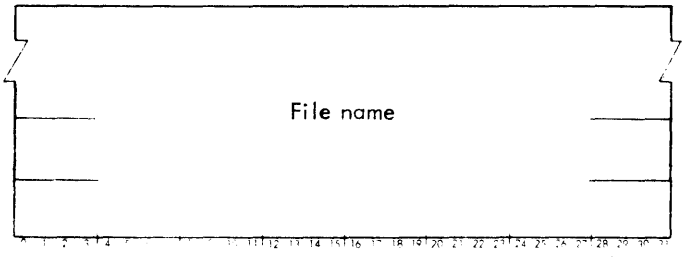
word 0



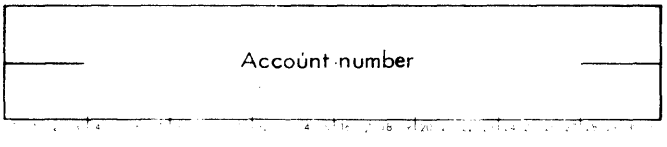

word 1



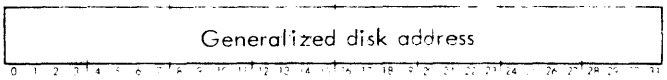

words 2 through 9



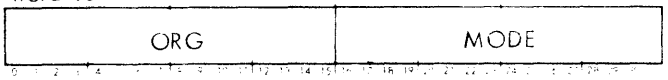

words 10 and 11




word 12




word 13




where

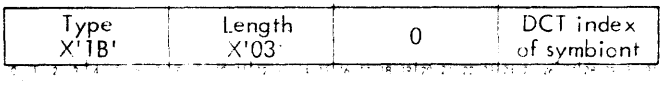
ORG is set to 1 for consecutive, 2 for keyed, and 3 for random.

MODE is set to 1 for IN, 2 for OUT, 4 for INOUT, and 8 for OUTIN.

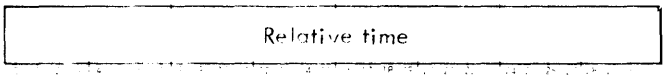
## SOFTWARE-DETECTED SYMBIONT INCONSISTENCIES

This record is logged if the system detects files which are inconsistent in that the associated file pointers do not match or are otherwise incorrect.

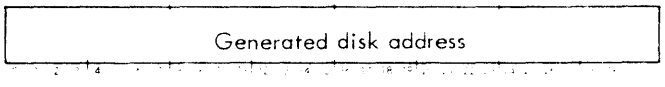
word 0




word 1



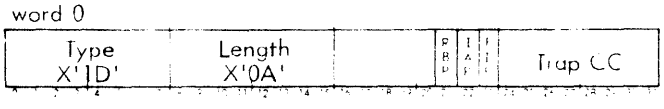

word 2



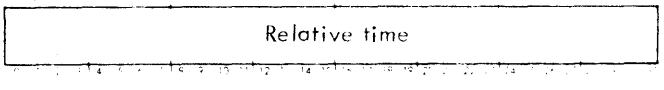

## INSTRUCTION EXCEPTION

This record is logged when program executions traps to location X'4D' on a Sigma 9 or Xerox 560 due to an instruction exception condition.

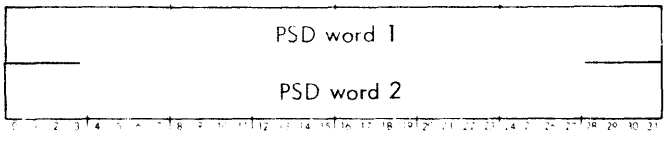
word 0



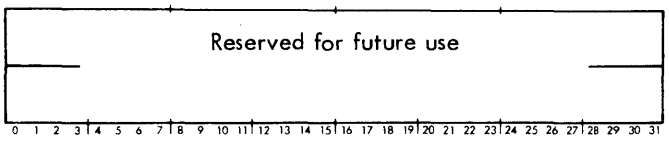

word 1



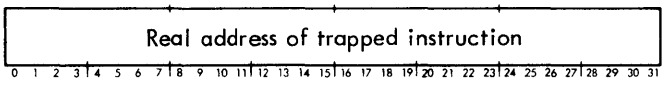

words 2 and 3



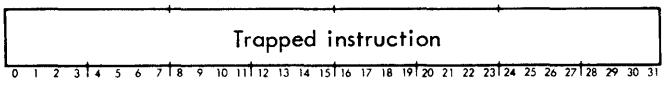
words 4 and 5



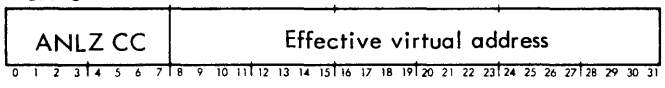

word 6



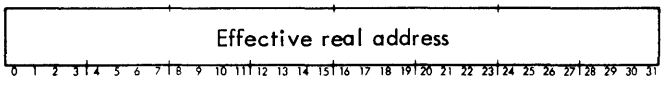

word 7




word 8




word 9

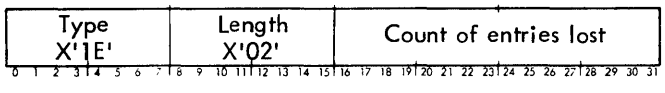



where FIP, IAP, RBP, and ANLZ CC have the same meanings as for the hardware error record (X'17').

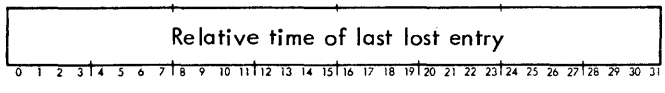
## LOST ENTRY INDICATOR

This record is entered when buffering constraints make error logging temporarily impossible. The newest entries are lost.

word 0



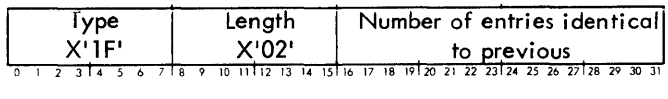

word 1



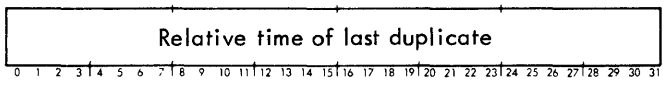

## DUPLICATE ENTRIES

This record is logged if duplicate error log entries are generated.

word 0



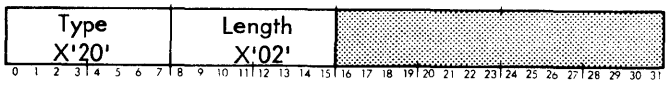

word 1



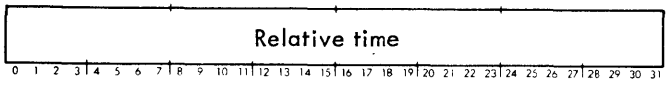

## POWER ON

This record is generated as a result of the power on trap associated with location X'50'.

word 0



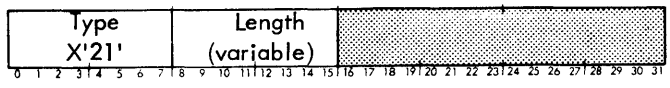

word 1



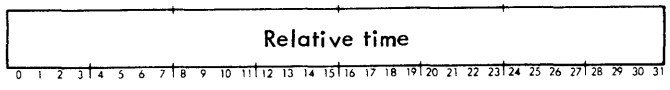

## CONFIGURATION RECORD

This record is logged at system startup.

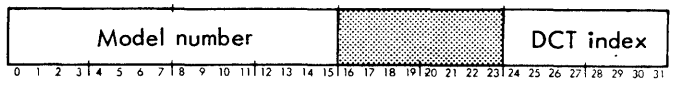
word 0



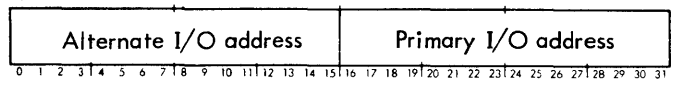

word 1



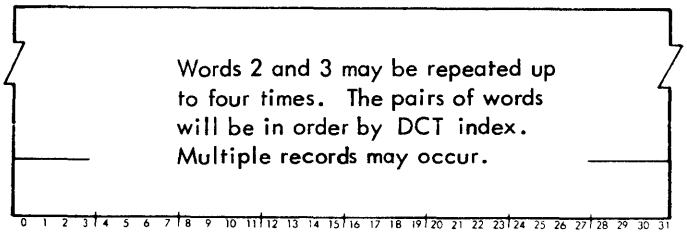

word 2




word 3



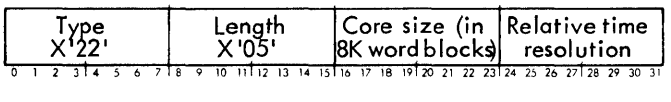

additional words



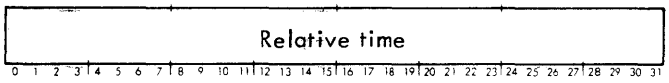

## SYSTEM IDENTIFICATION

This record is entered at system startup and recovery and is entered after the CONFIGURATION RECORD (type X'21').

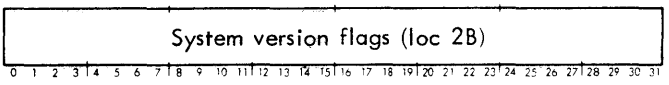
word 0



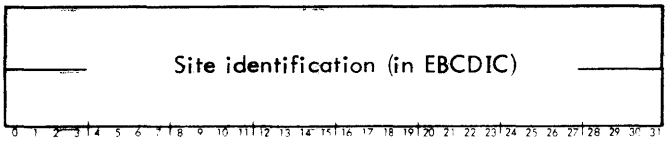
word 1




word 2



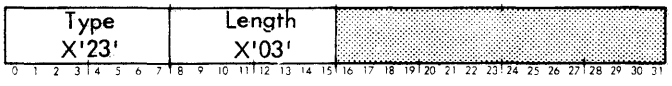

words 3 and 4



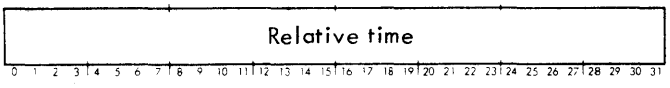

## TIME STAMP

This record is entered once each hour on the hour.

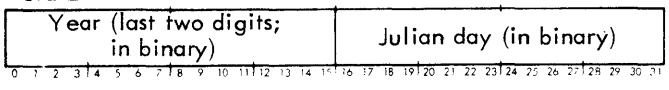
word 0




word 1



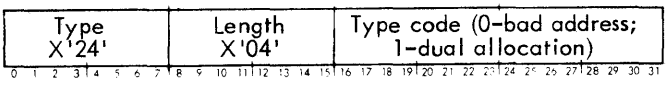

word 2



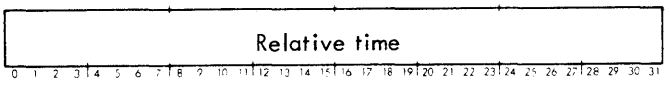

## BAD GRANULE RELEASE

This record is logged if the granule being released contains an invalid disk address or has already been released (dual allocation).

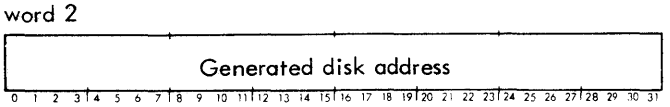
word 0



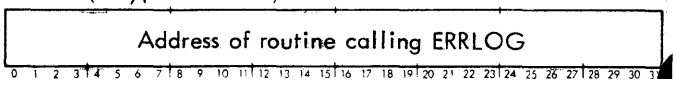

word 1




word 2

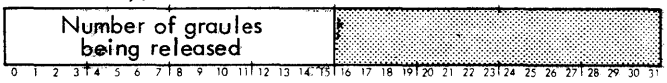



word 3 (if type code = 0)




or

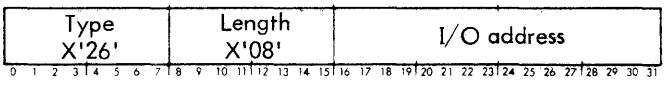
word 3 (if type code = 1)



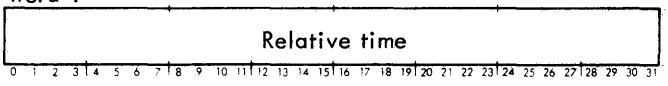

## REMOTE PROCESSING ERROR

This record is logged when an error occurs in the transmission of data to or from a remote processing workstation.

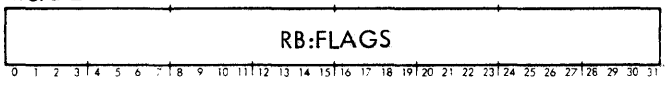
word 0



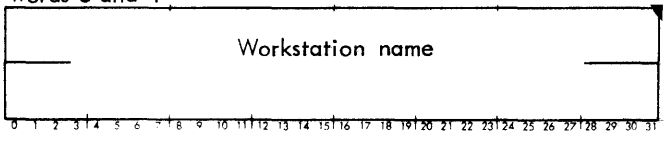

word 1



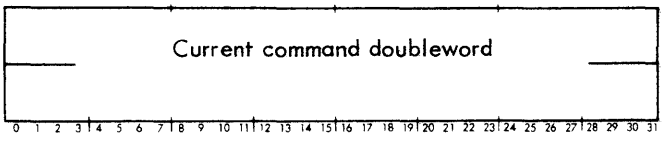

word 2



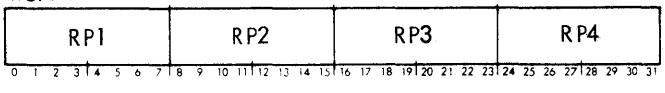

words 3 and 4




words 5 and 6




word 7




where

type identifies the type of error log record.

length specifies the number of 32-bit words contained in the error log record.

I/O address is a 16-bit address representing the physical I/O address.

relative time represents milliseconds since midnight. Resolution is 2 msec.

RB:FLAGS specifies the contents of RB:FLAG at the time of the error. RB:FLAG is described in the CP-V/Data Base Technical Manual, 90 19 95.

workstation name specifies the workstation name (in TEXT format, left-justified and padded with blanks) if the terminal is logged on.

current command doubleword specifies the command doubleword of the I/O that was taking place when the error occurred. For Xerox 7670 RBTs, the current command doubleword contains the second command doubleword used to write the text of an output message and is meaningful only for RP1 = 0, 1, A, or B.

RP1, RP2, RP3, and RP4 have specific meaning for the type of remote workstation associated with the record. The meanings are listed in Tables E-2 through E-7.

Table E-2. Xerox 7670 RBT – RP1, RP3, and RP4

| RP1 Value | Meaning | Corresponding RP3 Meaning | Corresponding RP4 Meaning |
|---|---|---|---|
| 1 | First character in record not SOH. | Current character position. | Offending character. |
| 2 | Incorrect parity on SEL. | Current character position. | Offending character. |
| 3 | Incorrect block protect. | Current character position. | Offending character. |
| 4 | Third character in record not STX. | Current character position. | Offending character. |
| 5 | RBBAT COMBUF or MPOOL unavailable for log-on. | Meaningless. | Meaningless. |
| 6 | Incorrect character parity. | Current character position. | Offending character. |
| 7 | Record trailer character not ETX. | Current character position. | Offending character. |
| 8 | Incorrect block check parity. | Current character position. | Offending character. |
| 9 | Incorrect block check. | Current character position. | Offending character. |
| A | Communication line time-out. | Meaningless. | Meaningless. |
| B | NAK received. | Response received reading for ACK. (RP3 and RP4 combine to be a halfword). | |
| C | Garbled ACK or NAK. | Response received reading for ACK. (RP3 and RP4 combine to be a halfword). | |

Table E-3. Xerox 7670 RBT – RP2

| RP2 Value | Meaning (Current Function Code) |
|---|---|
| 0 | Write card punch. |
| 1 | Write line printer. |
| 2 | Send ACK. |
| 3 | Write TOF (Block protect 0). |
| 4 | Write TOF (Block protect 1). |
| 5 | Write SPACE (Block protect 0). |
| 6 | Write SPACE (Block protect 1). |
| 7 | Read card reader. |
| 8 | Write TOF (log-on). |
| 9 | Read card reader (special). |
| A | Read ACK card punch. |
| B | Read ACK line printer. |
| C | Read ACK TOF (Block protect 0). |
| D | Read ACK TOF (Block protect 1). |
| E | Read ACK SPACE (Block protect 0). |
| F | Read ACK SPACE (Block protect 1). |
| 10 | Write EOT. |
| 11 | Write DC1. |
| 12 | Write ACK (special). |
| 13 | Write NAK. |
| 14 | Write NAK (special). |
| 15 | Write BEL (on error). |

Table E-4. IBM 2780 RBT – RP1 and RP4

| RP1 Value | Meaning | Corresponding RP4 Meaning |
|---|---|---|
| 1 | Disconnect due to | |
| | a. EOT on read. | EOT |
| | b. Use of 2780 on IRBT only system. | ENQ |
| 2 | Line timeout. | Same as RP2. |
| 3 | ENQ not received on logon read. | Character received. |
| 4 | No EOT after EOF sent. | Character received. |
| 5 | a. ENQ in text mode. | Character received. |
| | b. No ENQ answering WACK. | Character received. |
| | c. ENQ answer to ACK of EOF. | Character received. |
| 6 | NAK received. | Character received. |
| 8 | CRC failed on input. | Last character CRCed. |
| 9 | Unknown response reading for ACK. | Character received. |
| A | Trailer character not ETB or ETX. | Character received. |
| C | Header character not STX. | Character received. |

Table E-5. IBM 2780 RBT – RP2 and RP3

| Value | RP2 (Current Function Code) | RP3 (Calling Function Code) |
|---|---|---|
| 0 | Disconnect. | Software error – should not occur. |
| 1 | Write data. | Write. |
| 2 | Send ENQ. | Send ENQ (Wait). |
| 3 | Send ACK 0. | Read. |
| 4 | Send WACK. | Send WACK (Wait). |

Table E-5. IBM 2780 RBT – RP2 and RP3 (cont.)

| Value | RP2 (Current Function Code) | RP3 (Calling Function Code) |
|---|---|---|
| 5 | Write data. | Write EOF. |
| 6 | Send ENQ. | Request to output. |
| 7 | Read for ACK, ENQ, EOT (depends on RP3). | POL for input. |
| 8 | Read for ENQ. | Logon. |
| 9 | Read. | Software error – should not occur. |
| A | Send NAK. | Software error – should not occur. |
| B | Send ACK 1. | Software error – should not occur. |
| C | Send EOT. | Software error – should not occur. |

Table E-6. IRBT – RP1 and RP4

| RP1 Value | Meaning | Corresponding RP4 Meaning |
|---|---|---|
| 0 | Recoverable block check error. | Difference (mod 16) between expected and received BCBs. |
| 1 | Catastrophic block check error (NAK sent in case of line error). | Difference (mod 16) between expected and received BCBs. |
| 2 | Communication line time-out. | Same as RP2. |
| 3 | Read for ENQ timed-out (logon). | Same as RP2. |
| 4 | Received ACK O instead of SIGNON at logon. | ACK O |
| 5 | Inappropriate line bid (not ENQ-master, not ACK O-slave). | Line bid received. |
| 6 | NAK received. | NAK. |
| 7 | Read timed out. | Same as RP2. |
| 8 | Incorrect CRC. | Last character CRCed. |
| 9 | Trailer character not ETB. | Offending character. |
| A | Leader character not STX. | Offending character. |
| B | Lost data. | First character after IDLE. |
| C | Garbled ACK O-NAK. | First character of message. |

Table E-7. IRBT – RP2 and RP3

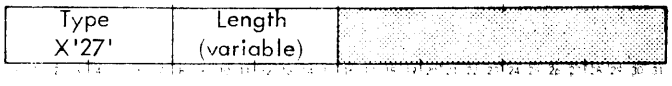
| Value | RP2 (Current Function Code) | RP3 (Calling Function Code) |
|---|---|---|
| 0 | Disconnect. | Software error – should not occur. |
| 1 | Write block. | Write block – read block. |
| 2 | Write ACK. | Write ACK – read block. |
| 3 | Write block. | Write block (Wait-a-bit) – Read special. |
| 4 | Write Wait-a-bit. | Write Wait-a-bit – Read special. |
| 5 | Read block. | Software error – should not occur. |
| 6 | Send NAK. | Software error – should not occur.† |
| 7 | Send ENQ. | Logon as Slave. |
| 8 | Read for ENQ. | Logon as Master. |
| 9 | ACK 0 to ENQ. | Logon as Master after ENQ Read. |
| A | Read logon record. | Software error – should not occur. |
| B | NAK logon record. | Software error – should not occur.† |

†If errors with the same RP1 code occur consecutively, this code may appear in the RP3 field for the second and subsequent consecutive errors, replacing another legal RP3 code.

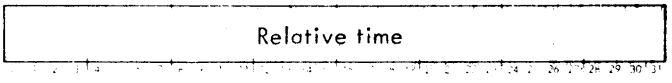
OPERATOR MESSAGE

This record is interjected as the result of an operator ERRSEND key-in or by a diagnostic program. It is generally used to describe unusual conditions surrounding a particular error.

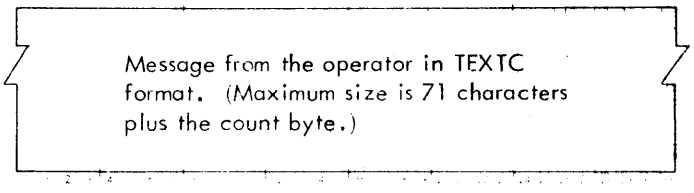
word 0




word 1



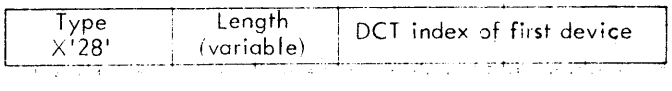

word 2



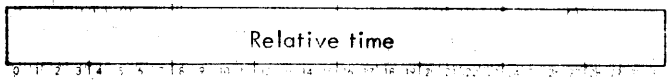

I/O ACTIVITY COUNT

This is recorded once per hour and at recovery.

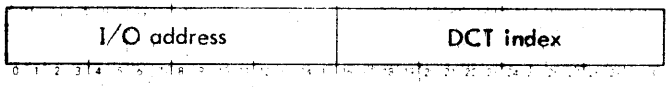
word 0



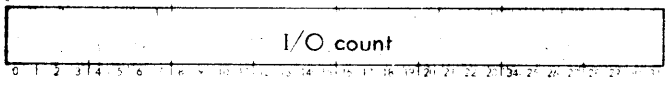

word 1



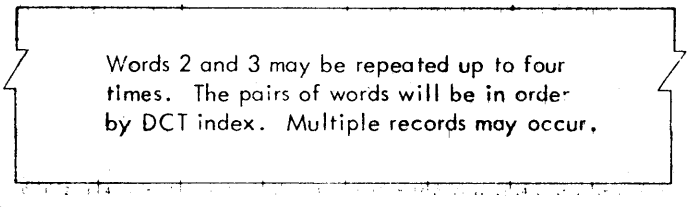

word 2




word 3



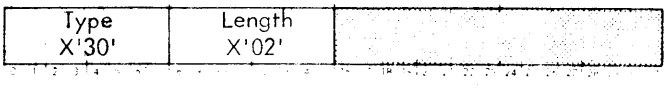

additional words



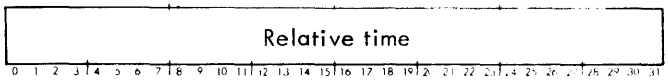

PFI PRIMARY RECORD

This record is logged when program execution is interrupted to location X'56' on the Xerox 560 due to a Processor Fault Interrupt condition.

word 0



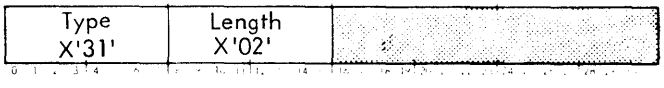
word 1



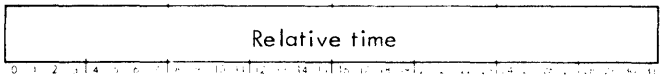

MFI PRIMARY RECORD

This record is logged as a result of the memory fault interrupt associated with location X'57' on a Sigma 9 or Xerox 560.

word 0



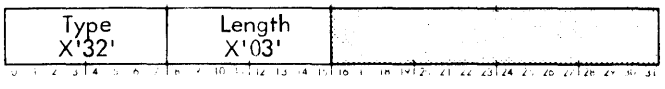

word 1




SECONDARY RECORD FOR POLL INFORMATION

This record is logged to record specific information obtained by issuing a POLL instruction subsequent to detecting hardware errors. One record is produced per valid poll status received.

word 0



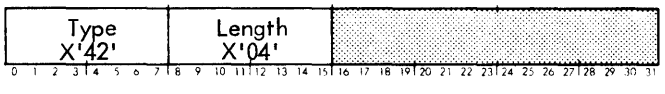

**Unit type =**

**1 - Basic Processor**
**2 - Memory Interface**
**3 - Processor Interface**
**4 - Multiplexor IOP**
**5 - Rotating Memory Processor**
**6 - Not Used**
**7 - System Control Processor**

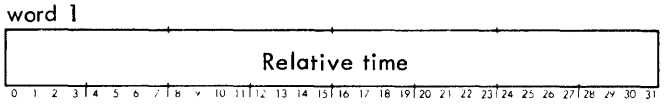
XEROX 560 MEMORY PARITY SECONDARY RECORD

This record is logged to record specific information returned in response to an LMS instruction subsequent to detecting hardware errors.

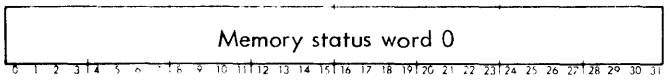
word 0



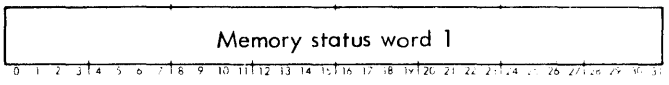

word 1




word 2



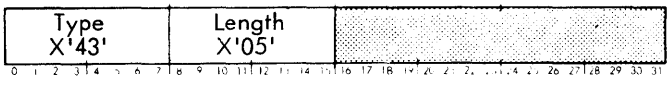

word 3



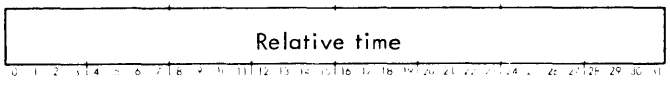

~~SIGMA 6/7~~ 9 MEMORY PARITY SECONDARY RECORD

This record is logged as a result of the memory fault interrupt associated with location X'57' or the memory parity trap associated with location X'4C' on the Sigma 9 ~~Xerox 560~~. This record follows record type X'17' and record type X'31'.

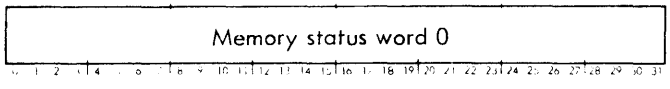
word 0



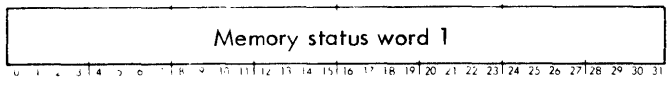

word 1



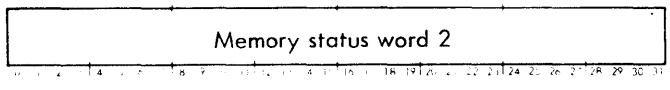

word 2




word 3



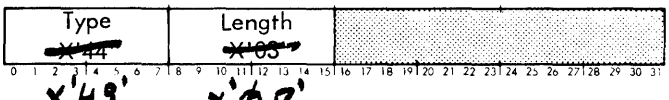

word 4



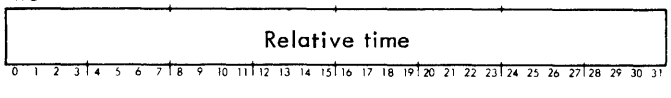

~~SIGMA 6/7~~ MEMORY PARITY SECONDARY RECORD

This record is logged to record specific information obtained by scanning memory to attempt to isolate locations which cannot sustain correct parity.

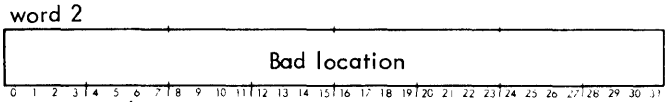
word 0




word 1




word 2



word 2

| Number of bad locations in memory |
|---|

word 3

| Logical AND of addresses of the bad locations |
|---|

word 4

| Logical OR of addresses of the bad locations |
|---|

word 5

| Logical AND of contents of the bad locations |
|---|

word 6

| Logical OR or contents of the bad locations |
|---|

word 7

| LMS CC | Address of lowest location with bad parity |
|---|---|

word 8

| Contents of lowest location with bad parity |
|---|

word 9

| LMS CC | Address of highest location with bad parity |
|---|---|

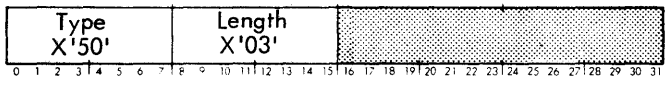
word 10

| Contents of highest location with bad parity |
|---|

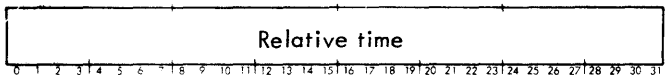
ENQUEUE TABLE OVERFLOW

This record is logged when an Enqueue CAL has been rejected because there are insufficient unused entries in the Enqueue tables.

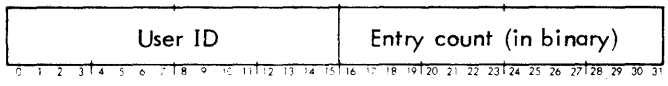
word 0




word 1




word 2

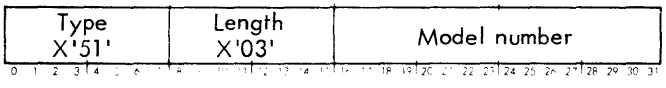



Entry count is the number of entries in the enqueue table belonging to the specified user at the time the error log entry was made.

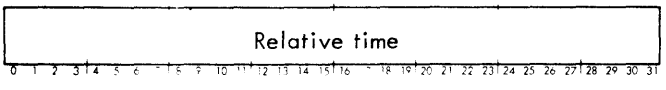
PARTITIONED RESOURCE

This entry is logged when a resource is partitioned via the SYSCON processor by the operator.

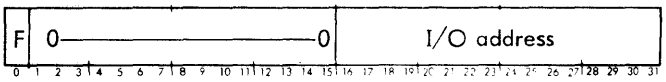
word 0




word 1




word 2




where

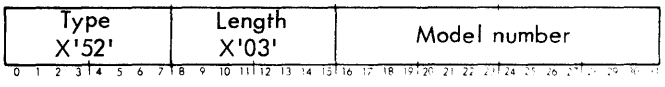
F = 0 for device entry.

F = 1 for controller entry.

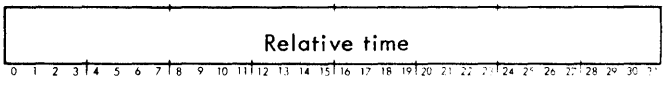
RETURNED RESOURCE

This entry is logged when a resource is returned from being partitioned via the SYSCON processor by the operator.

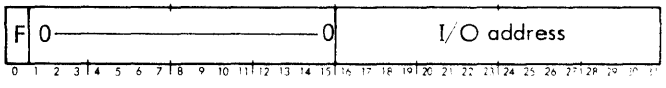
word 0




word 1




word 2




where

F = 0 for device entry.

F = 1 for controller entry.

# APPENDIX F. XEROX STANDARD OBJECT LANGUAGE

## INTRODUCTION

### GENERAL

The Xerox standard object language provides a means of expressing the output of any Xerox processor in standard format. All programs and subprograms in this object format can be loaded by the Monitor's relocating loader.[†] Such a loader is capable of providing the program linkages needed to form an executable program in core storage. The object language is designed to be both computer-independent and medium-independent; i.e., it is applicable to any Xerox computer having a 32-bit word length, and the same format is used for both cards and paper tape.

### SOURCE CODE TRANSLATION

Before a program can be executed by the computer, it must be translated from symbolic form to binary data words and machine instructions. The primary stages of source program translation are accomplished by a processor. However, under certain circumstances, the processor may not be able to translate the entire source program directly into machine language form.

If a source program contains symbolic forward references, a single-pass processor such as the Xerox Symbol assembler can not resolve such references into machine language. This is because the machine language value for the referenced symbol is not established by a one-pass processor until after the statement containing the forward reference has been processed.

A two-pass processor, such as the Xerox Meta-Symbol assembler, is capable of making "retroactive" changes in the object program before the object code is output. Therefore, a two-pass processor does not have to output any special object codes for forward references. An example of a forward reference in a Symbol source program is given below.

```
        .
        .
Y       EQU     $ + 3
        .
        .
        CI, 5   Z
        .
        .
        LI, R   Z
        .
        .
Z       EQU     2
        .
        .
        BG      Z
        .
        .
R       EQU     Z + 1
        .
        .
```

[†]Although a discussion of the object language is not directly pertinent to CP-V, it is included in this manual because it applies to some of the processors operating under CP-V.

In this example the operand $ + 3 is not a forward reference because the assembler can evaluate it when processing the source statement in which it appears. However, the operand Z in the statement

```
CI,5    Z
```

is a forward reference because it appears before Z has been defined. In processing the statement, the assembler outputs the machine-language code for CI,5, assigns a forward reference number (e.g., 12) to the symbol Z, and outputs that forward reference number. The forward reference number and the symbol Z are also retained in the assembler's symbol table.

When the assembler processes the source statement

```
LI,R    Z
```

it outputs the machine-language code for LI, assigns a forward reference number (e.g., 18) to the symbol R, outputs that number, and again outputs forward reference number 12 for symbol Z.

On processing the source statement

```
Z   EQU     2
```

the assembler again outputs symbol Z's forward reference number and also outputs the value, which defines symbol Z, so that the relocating loader will be able to satisfy references to Z in statements CI, 5 Z and LI, R Z. At this time, symbol Z's forward reference number (i.e., 12) may be deleted from the assembler's symbol table and the defined value of Z equated with the symbol Z (in the symbol table). Then, subsequent references to Z, as in source statement

```
BG  Z
```

would not constitute forward references, since the assembler could resolve them immediately by consulting its symbol table.

If a program contains symbolic references to externally defined symbols in one or more separately processed subprograms or library routines, the processor will be unable to generate the necessary program linkages.

An example of an external reference in a Symbol source program is shown below.

```
        REF     ALPH
        .
        .
        LI,3    ALPH
        .
        .
```

When the assembler processes the source statement

```
REF     ALPH
```

it outputs the symbol ALPH, in symbolic (EBCDIC) form, in a declaration specifying that the symbol is an external reference. At this time, the assembler also assigns a declaration name number to the symbol ALPH but does not output the number. The symbol and name number are retained in the assembler's symbol table.

After a symbol has been declared an external reference, it may appear any number of times in the symbolic subprogram in which it was declared. Thus, the use of the symbol ALPH in the source statement

```
LI,3    ALPH
```

in the above example, is valid even though ALPH is not defined in the subprogram in which it is referenced.

The relocating loader is able to generate interprogram linkages for any symbol that is declared an external definition in the subprogram in which that symbol is defined. Shown below is an example of an external definition in a Symbol source program.

```
        DEF     ALPH
        :
        LI,3    ALPH
        :
ALPH    AI,4    X'F2'
        :
```

When the assembler processes the source statement

```
DEF     ALPH
```

it outputs the symbol ALPH, in symbolic (EBCDIC) form, in a declaration specifying that the symbol is an external definition. At this time, the assembler also assigns a declaration name number to the symbol ALPH but does not output the number. The symbol and name number are retained in the assembler's symbol table.

After a symbol has been declared an external definition it may be used (in the subprogram in which it was declared) in the same way as any other symbol. Thus, if ALPH is used as a forward reference, as in the source statement

```
LI,3    ALPH
```

above, the assembler assigns a forward reference number to ALPH, in addition to the declaration name number assigned previously. (A symbol may be both a forward reference and an external definition.)

On processing the source statement

```
ALPH    A1,4    X'F2'
```

the assembler outputs the declaration name number of the label ALPH (and an expression for its value) and also outputs the machine-language code for AI,4 and the constant X'F2'.

## OBJECT LANGUAGE FORMAT

An object language program generated by a processor is output as a string of bytes representing "load items". A load item consists of an item type code followed by the specific load information pertaining to that item. (The detailed format of each type of load item is given later in this appendix.) The individual load items require varying numbers of bytes for their representation, depending on the type and specific content of each item. A group of 108 bytes, or fewer, comprises a logical record. A load item may be continued from one logical record to the next.

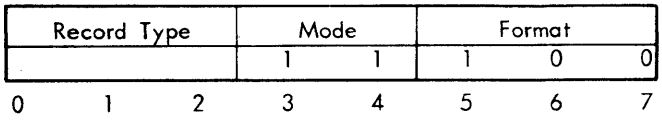
The ordered set of logical records that a processor generates for a program or subprogram is termed an "object module". The end of an object module is indicated by a module-end type code followed by the error severity level assigned to the module by the processor.

## RECORD CONTROL INFORMATION

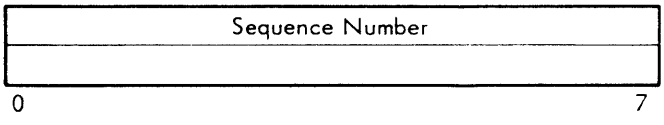
Each record of an object module consists of 4 bytes of control information followed by a maximum of 104 bytes of load information. That is, each record, with the possible exception of the end record, normally consists of 108 bytes of information (i.e., 72 card columns).

The four bytes of control information for each record have the form and sequence shown below.

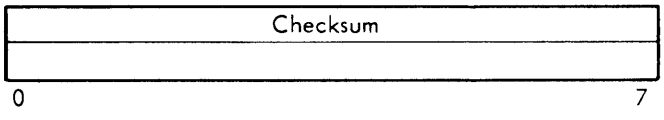
Byte 0



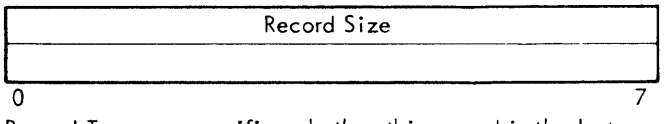

Byte 1




Byte 2




Byte 3




Record Type specifies whether this record is the last record of the module:

000 means last
001 means not last

Mode specifies that the loader is to read binary information. This code is always 11.

Format specifies object language format. This code is always 100.

Sequence Number is 0 for the first record of the module and is incremented by 1 for each record thereafter, until it recycles to 0 after reaching 255.

Checksum is the computed sum of the bytes comprising the record. Carries out of the most significant bit position of the sum are ignored.

Record Size is the number of bytes (including the record control bytes) comprising the logical record (5 ≤ record

size ≤ 108). The record size will normally be 108 bytes for all records except the last one, which may be fewer. Any excess bytes in a physical record are ignored.

### LOAD ITEMS

Each load item begins with a control byte that indicates the item type. In some instances, certain parameters are also provided in the load item control byte. In the following discussion, load items are categorized according to their function:

1. Declarations identify to the loader the external and control section labels that are to be defined in the object module being loaded.
2. Definitions define the value of forward references, external definitions, the origin of the subprogram being loaded, and the starting address (e.g., as provided in a Symbol/Meta-Symbol END directive).
3. Expression evaluation load items within a definition provide the values (such as constants, forward references, etc.) that are to be combined to form the final value of the definition.
4. Loading items cause specified information to be stored into core memory.
5. Miscellaneous items comprise padding bytes and the module-end indicator.

## DECLARATIONS

In order for the loader to provide the linkage between subprograms, the processor must generate for each external reference or definition a load item, referred to as a "declaration", containing the EBCDIC code representation of the symbol and the information that the symbol is either an external reference or a definition (thus, the loader will have access to the actual symbolic name).

Forward references are always internal references within an object module. (External references are never considered forward references.) The processor does not generate a declaration for a forward reference as it does for externals; however, it does assign name numbers to the symbols referenced.

Declaration name numbers (for control sections and external labels) and forward reference name numbers apply only within the object module in which they are assigned. They have no significance in establishing interprogram linkages, since external references and definitions are correlated by matching symbolic names. Hence, name numbers used in any expressions in a given object module always refer to symbols that have been declared within that module.

The processor must generate a declaration for each symbol that identifies a program section. Each object module produced by an assembler is considered to consist of at least one control section. If no section is explicitly identified in the source program, the assembler assumes it to be a standard control section (discussed below). The standard control section is always assigned a declaration name number of 0. All other control sections (i.e., produced by a processor capable of declaring other control sections) are assigned declaration name numbers (1, 2, 3, etc.) in the order of their appearance in the source program.

In the load items discussed below, the access code, pp, designates the memory protection class that is to be associated with the control section. The meaning of this code is given below.

| pp | Memory Protection Feature[†] |
|---|---|
| 00 | Read, write, or access instructions from. |
| 01 | Read or access instructions from. |
| 10 | Read only. |
| 11 | No access. |

Control sections are always allocated on a doubleword boundary. The size specification designates the number of bytes to be allocated for the section.

Declare Standard Control Section

Byte 0

| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Byte 1

| Access code | | | | Size (bits 1 through 4) | | | |
|---|---|---|---|---|---|---|---|
| p | p | 0 | 0 | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Byte 2

| Size (bits 5 through 12) | |
|---|---|
| | |
| 0 | 7 |

Byte 3

| Size (bits 13 through 20) | |
|---|---|
| | |
| 0 | 7 |

This item declares the standard control section for the object module. There may be no more than one standard control section in each object module. The origin of the standard control section is effectively defined when the first reference to the standard control section occurs, although the declaration item might not occur until much later in the object module.

[†] "Read" means a program can obtain information from the protected area; "write" means a program can store information into a protected area; and, "access" means the computer can execute instructions stored in the protected area.

This capability is required by one-pass processors, since the size of a section cannot be determined until all of the load information for that section has been generated by the processor.

Declare Nonstandard Control Section

Byte 0

| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Byte 1

| Access code | | | | Size (bits 1 through 4) | | | |
|---|---|---|---|---|---|---|---|
| P | P | 0 | 0 | | | | |
| 0 | 1 | 2 | 3 | 4 | | | 7 |

Byte 2

| Size (bits 5 through 12) | |
|---|---|
| 0 | 7 |

Byte 3

| Size (bits 13 through 20) | |
|---|---|
| 0 | 7 |

This item declares a control section other than standard control section (see above).

Declare Page Boundary Control Section

Byte 0

| Control Byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Byte 1

| Access code | | | | Size (bits 1 through 4) | | | |
|---|---|---|---|---|---|---|---|
| P | P | 0 | 0 | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Byte 2

| Size (bits 5 through 12) | |
|---|---|
| 0 | 7 |

Byte 3

| Size (bits 13 through 20) | |
|---|---|
| 0 | 7 |

This item declares a nonstandard control section beginning on a memory page boundary.

Declare Dummy Section

Byte 0

| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Byte 1

| First byte of name number |
|---|
| 0 7 |

Byte 2

| Second byte of name number[†] |
|---|
| 0 7 |

Byte 3

| Access code | | | | Size (bits 1 through 4) | |
|---|---|---|---|---|---|
| P | P | 0 | 0 | | |
| 0 | 1 | 2 | 3 | 4 | 7 |

Byte 4

| Size (bits 5 through 12) | |
|---|---|
| 0 | 7 |

Byte 5

| Size (bits 13 through 20) | |
|---|---|
| 0 | 7 |

This item comprises a declaration for a dummy control section. It results in the allocation of the specified dummy section, if that section has not been allocated previously by another object module. The label that is to be associated with the first location of the allocated section must be a previously declared external definition name. (Even though the source program may not be required to explicitly designate the label as an external definition, the processor must generate an external definition name declaration for that label prior to generating this load item.)

Declare External Definition Name

Byte 0

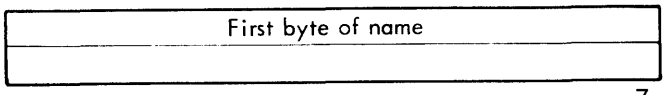
| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Byte 1

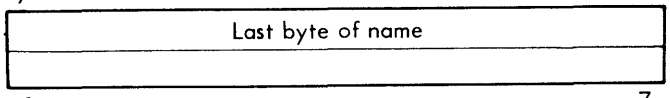
| Name length, in bytes (K) |
|---|
| 0 7 |

[†]If the module has fewer than 256 previously assigned name numbers, this byte is absent.

Byte 2



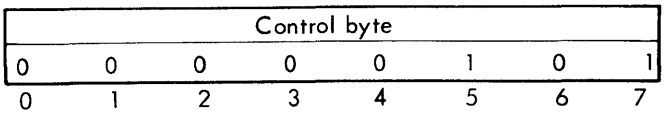

Byte K+1



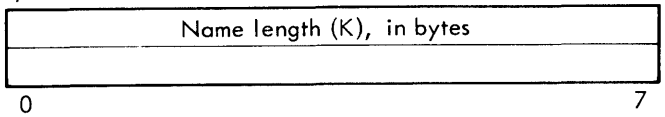

This item declares a label (in EBCDIC code) that is an external definition within the current object module. The name may not exceed 63 bytes in length.

Declare Primary External Reference Name

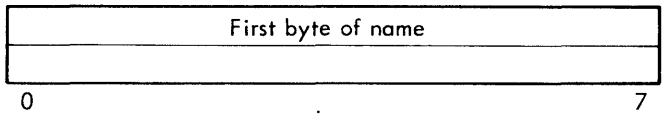
Byte 0



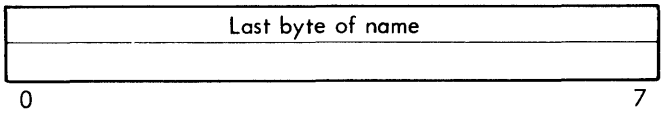

Byte 1




Byte 2




Byte K+1

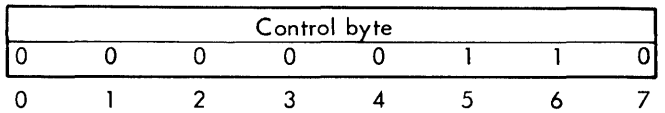



This item declares a symbol (in EBCDIC code) that is a primary external reference within the current object module. The name may not exceed 63 bytes in length.

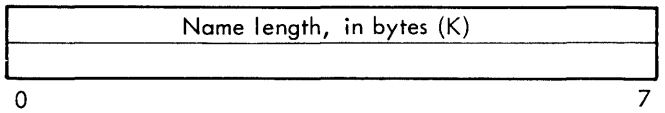
A primary external reference is capable of causing the loader to search the system library for a corresponding external definition. If a corresponding external definition is not found in another load module of the program or in the system library, a load error message is output and the job is errored.

Declare Secondary External Reference Name

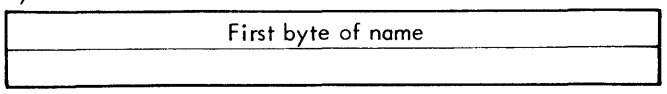
Byte 0



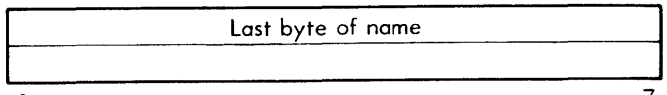

Byte 1




Byte 2




Byte K+1




This item declares a symbol (in EBCDIC code) that is a secondary external reference within the current object module. The name may not exceed 63 bytes in length.

A secondary external reference is not capable of causing the loader to search the system library for a corresponding external definition. If a corresponding external definition is not found in another load module of the program, the job is not errored and no error or abnormal message is output.

Secondary external references often appear in library routines that contain optional or alternative subroutines, some of which may not be required by the user's program. By the use of primary external references in the user's program, the user can specify that only those subroutines that are actually required by the current job are to be loaded. Although secondary external references do not cause loading from the library, they do cause linkages to be made between routines that are loaded.

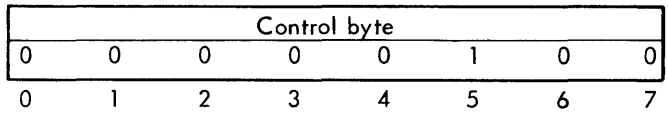
## DEFINITIONS

When a source language symbol is to be defined (i.e., equated with a value), the processor provides for such a value by generating an object language expression to be evaluated by the loader. Expressions are of variable length, and terminate with an expression-end control byte (see "Expression Evaluation" in this appendix). An expression is evaluated by the addition or subtraction of values specified by the expression.

Since the loader must derive values for the origin and starting address of a program, these also require definition.

Origin

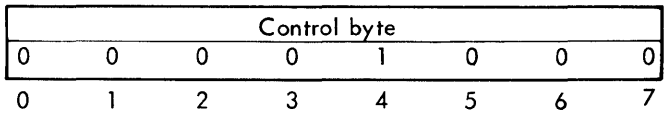
Byte 0




This item sets the loader's load-location counter to the value designated by the expression immediately following the origin control byte. This expression must not contain any elements that cannot be evaluated by the loader (see "Expression Evaluation" which follows).

Forward Reference Definition

Byte 0



Byte 1

| First byte of reference number |
|---|
| |

0 7

Byte 2

| Second byte of reference number |
|---|
| |

0 7

This item defines the value (expression) for a forward reference. The referenced expression is the one immediately following byte 2 of this load item, and must not contain any elements that cannot be evaluated by the loader (see "Expression Evaluation" which follows).

Forward Reference Definition and Hold

Byte 0

| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Byte 1

| First byte of reference number |
|---|
| |

0 7

Byte 2

| Second byte of reference number |
|---|
| |

0 7

This item defines the value (expression) for a forward reference and notifies the loader that this value is to be retained in the loader's symbol table until the module end is encountered. The referenced expression is the one immediately following the name number. It may contain values that have not been defined previously, but all such values must be available to the loader prior to the module end.

After generating this load item, the processor need not retain the value for the forward reference, since that responsibility is then assumed by the loader. However, the processor must retain the symbolic name and forward reference number assigned to the forward reference (until module end).

External Definition

Byte 0

| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Byte 1

| First byte of name number |
|---|
| |

0 7

Byte 2

| Second byte of name number[†] |
|---|
| |

0

This item defines the value (expression) for an external definition name. The name number refers to a previously declared definition name. The referenced expression is the one immediately following the name number.

Define Start

Byte 0

| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

This item defines the starting address (expression) to be used at the completion of loading. The referenced expression is the one immediately following the control byte.

## EXPRESSION EVALUATION

A processor must generate an object language expression whenever it needs to communicate to the loader one of the following:

1. A program load origin.
2. A program starting address.
3. An external definition value.
4. A forward reference value.
5. A field definition value.

Such expressions may include sums and differences of constants, addresses, and external or forward reference values that, when defined, will themselves be constants or addresses.

After initiation of the expression mode, by the use of a control byte designating one of the five items described above, the value of an expression is expressed as follows:

1. An address value is represented by an offset from the control section base plus the value of the control section base.

[†]If the module has fewer than 256 previously assigned name numbers, this byte is absent.

2. The value of a constant is added to the accumulated sum by generating an Add Constant (see below) control byte followed by the value, right-justified in four bytes.

   The offset from the control section base is given as a constant representing the number of units of displacement from the control section base, at the resolution of the address of the item. That is, a word address would have its constant portion expressed as a count of the number of words offset from the base, while the constant portion of a byte address would be expressed as the number of bytes offset from the base.

   The control section base value is accumulated by means of an Add Value of Declaration (see below) or Subtract Value of Declaration load item specifying the desired resolution and the declaration number of the control section base. The loader adjusts the base value to the specified address resolution before adding it to the current partial sum for the expression.

   In the case of an absolute address, an Add Absolute Section (see below) or Subtract Absolute Section control byte must be included in the expression to identify the value as an address and to specify its resolution.

3. An external definition of forward reference value is included in an expression by means of a load item adding or subtracting the appropriate declaration or forward reference value. If the value is an address, the resolution specified in the control byte is used to align the value before adding it to the current partial sum for the expression. If the value is a constant, no alignment is necessary.

Expressions are not evaluated by the loader until all required values are available. In evaluating an expression, the loader maintains a count of the number of values added or subtracted at each of the four possible resolutions. A separate counter is used for each resolution, and each counter is incremented or decremented by 1 whenever a value of the corresponding resolution is added to or subtracted from the loader's expression accumulator. The final accumulated sum is a constant, rather than an address value, if the final count in all four counters is equal to 0. If the final count in one (and only one) of the four counters is equal to +1 or -1, the accumulated sum is a "simple address" having the resolution of the nonzero counter. If more than one of the four counters hava a nonzero final count, the accumulated sum is termed a "mixed-resolution expression" and is treated as a constant rather than an address.

The resolution of a simple address may be altered by means of a Change Expression Resolution (see below) control byte. However, if the current partial sum is either a constant or a mixed-resolution value when the Change Expression Resolution control byte occurs, then the expression resolution is unaffected.

Note that the expression for a program load origin or starting address must resolve to a simple address, and the single nonzero resolution counter must have a final count of +1 when such expressions are evaluated.

In converting a byte address to a word address, the two least significant bits of the address are truncated. Thus, if the resulting word address is later changed back to byte resolution, the referenced byte location will then be the first byte (byte 0) of the word.

After an expression has been evaluated, its final value is associated with the appropriate load item.

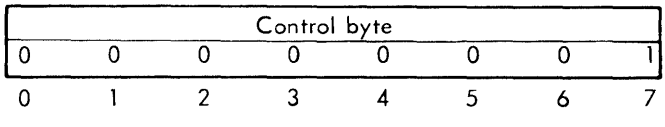
In the following diagrams of load item formats, RR refers to the address resolution code. The meaning of this code is given in the table below.

| RR | Address Resolution |
|---|---|
| 00 | Byte |
| 01 | Halfword |
| 10 | Word |
| 11 | Doubleword |

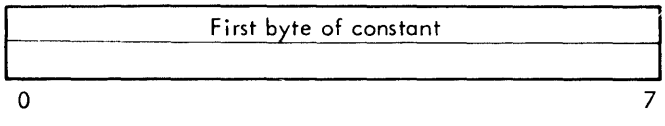
The load item discussed in this appendix, "Expression Evaluation", may appear only in expressions.

Add Constant

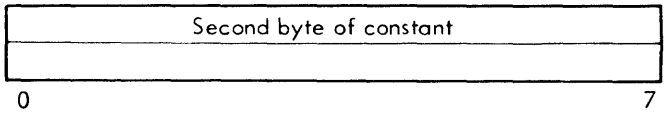
Byte 0




Byte 1




Byte 2



Byte 3

| Third byte of constant |
|---|
| |

0 7

Byte 4

| Fourth byte of constant |
|---|
| |

0 7

This item causes the specified four-byte constant to be added to the loaaer's expression accumulator. Negative constants are represented in two's complement form.

Add Absolute Section

Byte 0

| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 | R | R |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

This item identifies the associated value (expression) as a positive absolute address. The address resolution code, RR, designates the desired resolution.

Subtract Absolute Section

Byte 0

| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 0 | R | R |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

This item identifies the associated value (expression) as a negative absolute address. The address resolution code, RR, designates the desired resolution.

Add Value of Declaration

Byte 0

| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 | R | R |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Byte 1

| First byte of name number |
|---|
| |

0 7

Byte 2

| Second byte of name number† |
|---|
| |

0 7

†If the module has fewer than 256 previously assigned name numbers, this byte is absent.

This item causes the value of the specified declaration to be added to the loader's expression accumulator. The address resolution code, RR, designates the desired resolution, and the name number refers to a previously declared definition name that is to be associated with the first location of the allocated section.

One such item must appear in each expression for a relocatable address occurring within a control section, adding the value of the specified control section declaration (i.e., adding the byte address of the first location of the control section).

Add Value of Forward Reference

Byte 0

| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 1 | R | R |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Byte 1

| First byte of forward reference number |
|---|
| |

0 7

Byte 2

| Second byte of forward reference number |
|---|
| |

0 7

This item causes the value of the specified forward reference to be added to the loader's expression accumulator. The address resolution code, RR, designates the desired resolution, and the designated forward reference must not have been defined previously.

Subtract Value of Declaration

Byte 0

| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 | 0 | R | R |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Byte 1

| First byte of name number |
|---|
| |

0 7

Byte 2

| Second byte of name number† |
|---|
| |

This item causes the value of the specified declaration to be subtracted from the loader's expression accumulator. The address resolution code, RR, designates the desired resolution, and the name number refers to a previously declared definition name that is to be associated with the first location of the allocated section.

Subtract Value of Forward Reference

Byte 0

| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 | 1 | R | R |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Byte 1

| First byte of forward reference number |
|---|
| 0 7 |

Byte 2

| Second byte of forward reference number |
|---|
| 0 7 |

This item causes the value of the specified forward reference to be subtracted from the loader's expression accumulator. The address resolution code, RR, designates the desired resolution, and the designated forward reference must not have been defined previously.

Change Expression Resolution

Byte 0

| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | R | R |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

This item causes the address resolution in the expression to be changed to that designated by RR.

Expression End

Byte 0

| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

This item identifies the end of an expression (the value of which is contained in the loader's expression accumulator).

## FORMATION OF INTERNAL SYMBOL TABLES

The three object code control bytes described below are required to supply the information necessary in the formation of Internal Symbol Tables.

In the following diagrams of load item formats, Type refers to the symbol types supplied by the object language and maintained in the symbol table. IR refers to the internal resolution code. Type and resolution are meaningful only when the value of a symbol is an address. In this case, it is highly likely that the processor knows the type of value that is in the associated memory location, and the type field identifies it. The resolution field indicates the resolution of the location counter at the time the symbol was defined. The following tables summarize the combinations of value and meaning.

Symbol Types

| Type | Meaning of 5-Bit Code |
|---|---|
| 00000 | Instruction |
| 00001 | Integer |
| 00010 | Short floating point |
| 00011 | Long floating point |
| 00110 | Hexadecimal (also for packed decimal) |
| 00111 | EBCDIC text (also for unpacked decimal) |
| 01001 | Integer array |
| 01010 | Short floating-point array |
| 01011 | Long floating-complex array |
| 01000 | Logical array |
| 10000 | Undefined symbol |

Internal Resolution

| IR | Address Resolution |
|---|---|
| 000 | Byte |
| 001 | Halfword |
| 010 | Word |
| 011 | Doubleword |
| 100 | Constant |

Type Information for External Symbol

Byte 0

| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Byte 1

| Type field | IR field |
|---|---|
| 0 4 | 5 7 |

Byte 2

| Name number |
|---|
| 0 7 |

Byte 3 (if required)

| Name number (continued) |
|---|
| 0 7 |

This item provides type information for external symbols. The Type and IR fields are defined above. The name number field consists of one or two bytes (depending on the current declaration count) which specifies the declaration number of the external definition.

Type and EBCDIC for Internal Symbol

Byte 0

| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Byte 1

| Type field | IR field |
|---|---|

0 4 5 7

Byte 2

| Length of name (EBCDIC characters) |
|---|

0 7

Byte 3

| First byte of name in EBCDIC |
|---|

0 7

Byte n

| Last byte of name in EBCDIC |
|---|

0 7

Byte n + 1, . . .

| Expression defining value of internal symbol |
|---|

0 7

This item supplies type and EBCDIC for an internal symbol. The load items for Type and IR are as above. Length of name specifies the length of the EBCDIC name in characters. The name, in EBCDIC, is specified in the required number of bytes, followed by the expression defining the internal symbol.

EBCDIC for an Undefined Symbol

Byte 0

| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Byte 1

| Length of name (EBCDIC characters) |
|---|

0 7

Byte 2

| First byte of name in EBCDIC |
|---|

0 7

Byte n

| Last byte of name in EBCDIC |
|---|

0 7

Byte n + 1, n + 2

| Two bytes of symbol associated forward reference number |
|---|

0 7

This item is used to associate a symbol with a forward reference. The length of name and name in EBCDIC are the same as in the above item. The last two bytes specify the forward reference number with which the above symbol is to be associated.

# LOADING

Load Absolute

Byte 0

| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | N | N | N | N |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Byte 1

| First byte to be loaded |
|---|

0 7

⋮

Byte NNNN

| Last byte to be loaded |
|---|

0 7

This item causes the next NNNN bytes to be loaded absolutely (NNNN is expressed in natural binary form, except that 0000 is interpreted as 16 rather than 0). The load location counter is advanced appropriately.

Load Relocatable (Long Form)

Byte 0

| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | Q | C | R | R |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Byte 1

| First byte of name number |
|---|

0 7

Byte 2

| Second byte of name number† |
|---|

0 7

This item causes a four-byte word (immediately following this load item) to be loaded, and relocates the address field according to the address resolution code, RR. Control bit C designates whether relocation is to be relative to a forward reference (C = 1) or relative to a declaration (C = 0). Control bit Q designates whether a 1-byte (Q = 1) or a 2-byte (Q = 0) name number follows the control byte of this load item.

†If the module has fewer than 256 previously assigned name numbers, this byte is absent.

If relocation is to be relative to a forward reference, the forward reference must not have been defined previously. When this load item is encountered by the loader, the load location counter can be aligned with a word boundary by loading the appropriate number of bytes containing all zeros (e.g., by means of a load absolute item).

Load Relocatable (Short Form)

Byte 0

| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | C | D | D | D | D | D | D |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

This item causes a four-byte word (immediately following this load item) to be loaded, and relocates the address field (word resolution). Control bit C designates whether relocation is to be relative to a forward reference (C = 1) or relative to a declaration (C = 0). The binary number DDDDDD is the forward reference number or declaration number by which relocation is to be accomplished.

If relocation is to be relative to a forward reference, the forward reference must not have been defined previously. When this load item is encountered by the loader, the load location counter must be on a word boundary (see "Load Relocatable (Long Form)", above).

Repeat Load

Byte 0

| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Byte 1

| First byte of repeat count |
|---|
| 0 7 |

Byte 2

| Second byte of repeat count |
|---|
| 0 7 |

This item causes the loader to repeat (i.e., perform) the subsequent load item a specified number of times. The repeat count must be greater than 0, and the load item to be repeated must follow the repeat load item immediately.

Define Field

Byte 0

| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Byte 1

| Field location constant, in bits (K) |
|---|
| 0 7 |

Byte 2

| Field length, in bits (L) |
|---|
| 0 7 |

This item defines a value (expression) to be added to a field in previously loaded information. The field is of length L ($1 \leq L \leq 255$) and terminates in bit position T, where:

T = current load bit position − 256 + K.

The field location constant, K, may have any value from 1 to 255. The expression to be added to the specified field is the one immediately following byte 2 of this load item.

## MISCELLANEOUS LOAD ITEMS

Padding

Byte 0

| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Padding bytes are ignored by the loader. The object language allows padding as a convenience for processors.

Module End

Byte 0

| Control byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Byte 1

| Severity level | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | E | E | E | E |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

This item identifies the end of the object module. The value EEEE is the error severity level assigned to the module by the processor.

## OBJECT MODULE EXAMPLE

The following example shows the correspondence between the statements of a Meta-Symbol source program and the string of object bytes output for that program by the assembler. The program, listed below, has no significance other than illustrating typical object code sequences.

Example

```
 1                                          DEF     AA,BB,CC   CC IS UNDEFINED BUT CAUSES NO
                                                               ERROR
 2                                          REF     RZ,RTN     EXTERNAL REFERENCES DECLARED
 3  00000                          ALPHA    CSECT              DEFINE CONTROL SECTION ALPHA
 4  000C8                                   ORG     200        DEFINE ORGIN
 5  000C8    22000000   N          AA       LI,CNT  0          DEFINES EXTERNAL AA; CNT IS A
                                                               FWD REF
 6  000C9    32000000   N                   LW,R    RZ         R IS A FORWARD REFERENCE;
 7                                  *                          RZ IS AN EXTERNAL REFERENCE, AS
 8                                  *                          DECLARED IN LINE 2
 9  000CA    50000000   N          RPT      AH,R    KON        DEFINES RPT; R AND KON ARE
10                                  *                          FORWARD REFERENCES
11  000CB    69200000   F                   BCS,2   BB         BB IS AN EXTERNAL DEFINITION
12                                  *                          USED AS A FORWARD REFERENCE
13  000CC    20000001   N                   AI,CNT  1          CNT IS A FORWARD REFERENCE
14  000CD    680000CA                       B       RPT        RPT IS A BACKWARD REFERENCE
15  000CE    68000000   X                   B       RTN        RTN IS AN EXTERNAL REFERENCE
16  000CF    0001       A          KON      DATA,2  1          DEFINES KON
17           00000003              R        EQU     3          DEFINES R
18           00000004              CNT      EQU     4          DEFINES CNT
19  000D0    224FFFFF   A          BB       LI,CNT  -1         DEFINES EXTERNAL BB THAT HAS
20                                  *                          ALSO BEEN USED AS A FORWARD
21                                  *                          REFERENCE
22  000C8                                   END     AA         END OF PROGRAM
```

CONTROL BYTES (In Binary)

Begin Record  Record number: 0

```
00111100  Record type: not last, Mode binary, Format: object language.
00000000  Sequence number 0
01100011  Checksum: 99
01101100  Record size: 108
```

} Record control information not part of load item

0302C1C1 (hexadecimal code comprising the load item)

```
00000011  Declare external definition name (2 bytes)  Name: AA   Declaration number: 1
```

0302C2C2

```
00000011  Declare external definition name (2 bytes)  Name: BB   Declaration number: 2
```

0302C3C3

```
00000011  Declare external definition name (2 bytes)  Name: CC   Declaration number: 3
```

} Source Line 1

0502D9E9

```
00000101  Declare primary reference name (2 bytes)  Name RZ      Declaration number: 4
```

0503D9E3D5

```
00000101  Declare primary reference name (3 bytes)  Name: RTN   Declaration number: 5
```

} Source Line 2

```
Begin Record  Record number: 0

              0A010100000320200002
001010        Define external definition
              Number 1
00000001      Add constant: 800   X'320'                          Source Line 5†
00100000      Add value of declaration (byte resolution)
              Number 0
00000010      Expression end

              040100000320200002
00000100      Origin
00000001      Add constant: 800   X'320'
00100000      Add value of declaration (byte resolution)          Source Line 4
              Number 0
00000010      Expression end

              4422000000
01000100      Load absolute the following 4 bytes: X'22000000'

              07EB0426000002
00000111      Define field
              Field location constant: 235   bits
              Field length: 4   bits                              Source Line 5
              Add the following expression to the above field:
00100110      Add value of forward reference (word resolution)
              Number 0
00000010      Expression end

              8432000000
10000100      Load relocatable (short form).  Relocate address field (word resolution)
              Relative to declaration number 4
              The following 4 bytes: X'32000000'

              07EB0426000602
00000111      Define field
              Field location constant: 235   bits
              Field length: 4   bits                              Source Line 6
              Add the following expression to the above field:
00100110      Add value of forward reference (word resolution)
              Number 6
00000010      Expression end

              CC50000000
11001100      Load relocatable (short form).  Relocate address field (word resolution)
              Relative to forward reference number 12
              The following 4 bytes: X'50000000'

              07EB0426000602
00000111      Define field
              Field location constant: 235   bits
              Field length: 4   bits                              Source Line 9
              Add the following expression to the above field:
00100110      Add value of forward reference (word resolution)
              Number 6
00000010      Expression end
```

†No object code is generated for source lines 3 (define control section) or 4 (define origin) at the time they are encountered. The control section is declared at the end of the program after Symbol has determined the number of bytes the program requires. The origin definition is generated prior to the first instruction.

| Begin Record | Record number: 0 | |
|---|---|---|
| | D269200000 | |
| 11010010 | Load relocatable (short form). Relocate address field (word resolution) | Source Line 11 |
| | Relative to forward reference number 18 | |
| | The following 4 bytes: X'69200000' | |
| | 4420000001 | |
| 01000100 | Load absolute the following 4 bytes: X'20000001' | |
| | 07EB0426000002 | |
| 00000111 | Define field | |
| | Field location constant: 235 bits | Source Line 13 |
| | Field length: 4 bits | |
| | Add the following expression to the above field: | |
| 00100110 | Add value of forward reference (word resolution) | |
| | Number 0 | |
| 00000010 | Expression end | |
| | 80680000CA | |
| 10000000 | Load relocatable (short form). Relocate address field (word resolution) | Source Line 14 |
| | Relative to declaration number 0 | |
| | The following 4 bytes: X'680000CA' | |
| | 8568000000 | |
| 10000101 | Load relocatable (short form). Relocate address field (word resolution) | Source Line 15 |
| | Relative to declaration number 5 | |
| | The following 4 bytes: X'68000000' | |
| | 08 | |
| 00001000 | Define forward reference (continued in record 1) | Source Line 16 |

| Begin Record | Record number: 1 | |
|---|---|---|
| 00011100 | Record type: last, Mode: binary, Format: object language. | |
| 00000001 | Sequence number 1 | Record Control |
| 11101100 | Checksum: 236 | Information |
| 01010001 | Record size: 81 | |
| | 000C010000033C200002 (continued from record 0) | |
| | Number 12 | |
| 00000001 | Add constant: 828 X'33C' | |
| 00100000 | Add value of declaration (byte resolution) | Source Line 16 |
| | Number 0 | |
| 00000010 | Expression end | |
| | 42001 | |
| 01000010 | Load absolute the following 2 bytes: X'0001' | |
| | 080006010000000302 | |
| 00001000 | Define forward reference | |
| | Number 6 | Source Line 17 |
| 00000001 | Add constant: 3 X'3' | |
| 00000010 | Expression end | |
| | 080000010000000402 | |
| 00001000 | Define forward reference | |
| | Number 0 | Source Line 18 |
| 00000001 | Add constant: 4 X'4' | |
| 00000010 | Expression end | |

```
Begin Record  Record number: 1

              0F00024100
00001111      Repeat load                                          } Advance to Word
              Repeat count: 2                                      } Boundary
01000001      Load absolute the following 1 bytes: X'00'           }

              08001201000000340200002
00001000      Define forward reference                             }
              Number 18                                            }
00000001      Add constant: 832  X'340'                            }
              Add value of declaration (byte resolution)           }
              Number 0                                             }
00000010      Expression end                                       }
                                                                   }
              0A0201000000340200002                                } Source Line 19
00001010      Define external definition                           }
              Number 2                                             }
00000001      Add constant: 832  X'340'                            }
00100000      Add value of declaration (byte resolution)           }
              Number 0                                             }
00000010      Expression end                                       }

              44224FFFFF
01000100      Load absolute the following 4 bytes: X'224FFFFF'     }

              0D0100000320200002
00001101      Define start                                         }
00000001      Add constant: 800  X'320'                            } Source Line 22
00100000      Add value of declaration (byte resolution)           }
              Number 0                                             }
00000010      Expression end                                       }

              0B000344
00001011      Declare standard control section declaration number: 0
              Access code: Full access.  Size 836  X'344'

              0E00
00001110      Module end
                  Severity level: X'0'
```

A table summarizing control byte codes for object language load items is given below.

| Object Code Control Byte | Type of Load Item |
|---|---|
| 0 0 0 0 0 0 0 0 | Padding |
| 0 0 0 0 0 0 0 1 | Add constant |
| 0 0 0 0 0 0 1 0 | Expression end |
| 0 0 0 0 0 0 1 1 | Declare external definition name |
| 0 0 0 0 0 1 0 0 | Origin |
| 0 0 0 0 0 1 0 1 | Declare primary reference name |
| 0 0 0 0 0 1 1 0 | Declare secondary reference name |
| 0 0 0 0 0 1 1 1 | Define field |
| 0 0 0 0 1 0 0 0 | Define forward reference |
| 0 0 0 0 1 0 0 1 | Declare dummy section |
| 0 0 0 0 1 0 1 0 | Define external definition |

| Object Code Control Byte | Type of Load Item |
|---|---|
| 0 0 0 0 1 0 1 1 | Declare standard control section |
| 0 0 0 0 1 1 0 0 | Declare nonstandard control section |
| 0 0 0 0 1 1 0 1 | Define start |
| 0 0 0 0 1 1 1 0 | Module end |
| 0 0 0 0 1 1 1 1 | Repeat load |
| 0 0 0 1 0 0 0 0 | Define forward reference and hold |
| 0 0 0 1 0 0 0 1 | Provide type information for external symbol |
| 0 0 0 1 0 0 1 0 | Provide type and EBCDIC for internal symbol |
| 0 0 0 1 0 0 1 1 | EBCDIC and forward reference number for undefined symbol |
| 0 0 0 1 1 1 1 0 | Declare page boundary control section |
| 0 0 1 0 0 0 R R | Add value of declaration |
| 0 0 1 0 0 1 R R | Add value of forward reference |
| 0 0 1 0 1 0 R R | Subtract value of declaration |
| 0 0 1 0 1 1 R R | Subtract value of forward reference |
| 0 0 1 1 0 0 R R | Change expression resolution |
| 0 0 1 1 0 1 R R | Add absolute section |
| 0 0 1 1 1 0 R R | Subtract absolute section |
| 0 1 0 0 N N N N | Load absolute |
| 0 1 0 1 Q C R R | Load relocatable (long form) |
| 1 C D D D D D D | Load relocatable (short form) |

# APPENDIX G. XEROX STANDARD COMPRESSED LANGUAGE

The Xerox standard compressed language is used to represent source EBCDIC information in a highly compressed form.

Meta-Symbol (along with several of the utility programs) accepts this form as input or output, will accept updates to the compressed input and will regenerate source when requested. No information is destroyed in the compression or decompression.

Records may not exceed 108 bytes in length. Compressed records are punched in the binary mode when represented on card media. Therefore, on cards, columns 73 through 80 are not used and are available for comment or identification information.

The first four bytes of each record are for checking purposes. They are as follows:

Byte 1 Identification (00L11000) L = 1 for each record except the last record, in which case L = 0.

Byte 2 Sequence number (0 to 255 and recycles).

Byte 3 Checksum which is the least significant 8 bits of the sum of all bytes in the record except the checksum byte itself. Carries out of the most significant bit are ignored. If the checksum byte is all 1's, do not checksum the record.

Byte 4 Number of bytes comprising record including the checking bytes (≤ 108)

The rest of the record consists of a string of 6-bit and 8-bit items. Any partial item at the end of a record is ignored.

The following six-bit items (decimal number assigned) comprise the string control:

| Item | Function | Item | Function |
|---|---|---|---|
| 0 | Ignore | 32 | O |
| 1 | Not currently assigned | 33 | P |
| 2 | End of line | 34 | Q |
| 3 | End of file | 35 | R |
| 4 | Use 8-bit character that follows† | 36 | S |
| 5 | Use n + 1 blanks (next 6-bit item is n) | 37 | T |
| 6 | Use n + 65 blanks (next 6-bit item is n) | 38 | U |
| 7 | Blank | 39 | V |
| 8 | 0 | 40 | W |
| 9 | 1 | 41 | X |
| 10 | 2 | 42 | Y |
| 11 | 3 | 43 | Z |
| 12 | 4 | 44 | . |
| 13 | 5 | 45 | < |
| 14 | 6 | 46 | ( |
| 15 | 7 | 47 | + |
| 16 | 8 | 48 | \| |
| 17 | 9 | 49 | & |
| 18 | A | 50 | $ |
| 19 | B | 51 | * |
| 20 | C | 52 | ) |
| 21 | D | 53 | ; |
| 22 | E | 54 | ¬ |
| 23 | F | 55 | - |
| 24 | G | 56 | / |
| 25 | H | 57 | , |
| 26 | I | 58 | % |
| 27 | J | 59 | _ |
| 28 | K | 60 | > |
| 29 | L | 61 | : |
| 30 | M | 62 | ' |
| 31 | N | 63 | = |

†Eight-bit characters are in uncompressed EBCDIC format (e.g., !@#?).

# APPENDIX H. 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 H-1. CP-V 8-Bit Computer Codes (EBCDIC)

| Least Significant Digits | Hexadecimal | Most Significant Digits 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Binary | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 0 | 0000 | NUL | DLE | LF only | ESC F | SP | & | - | ^ | | | | | SP | | - | 0 |
| 1 | 0001 | SOH | X-ON | FS | CAN | [7] | | / | •• | a | j | | \ [1] | A | J | | 1 |
| 2 | 0010 | STX | DC2 | GS | ESC X | ⊥ | | ⌜ | ¯ | b | k | s | { [1] | B | K | S | 2 |
| 3 | 0011 | ETX | X-OFF | RS | ESC P | | □ | | | c | l | t | } [1] | C | L | T | 3 |
| 4 | 0100 | EOT | DC4 | US | ESC U | L | | ⌡ | ≤ | 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 | | 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 | ≀ | | | v | i | r | z | | I | R | Z | 9 |
| A | 1010 | NAK | SUB | EOT | ESC C | ∉ [2] | ! | ⌒ [1] | : | | | | | [7] | | | × |
| B | 1011 | VT | ESC | BS | ESC LF | . | $ | , | # | | | | | | | | ÷ |
| 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 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 H-2. CP-V 7-Bit Communication Codes (ANSCII)

| Least Significant Digits: Decimal (rows) | Binary | Most Significant Digits: 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|
| | | x000 | x001 | x010 | x011 | x100 | x101 | x110 | 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 H-3. CP-V Symbol-Code Correspondences

| EBCDIC† Hex. | EBCDIC† Dec. | Symbol | Card Code | ANSCII†† | 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 Teletypes. |
| 14 | 20 | DC4 | 11-9-4 | 1-4 | device control 4 | |
| 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 made | |
| 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 LF | 9-8-3 | | line continuation | 3B toggles the backspace edit mode for 2741 terminals. |
| 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 | |

†Hexadecimal and decimal notation.

††Decimal notation (column-row).

Table H-3. CP-V Symbol-Code Correspondences (cont.)

| | EBCDIC† Hex. | EBCDIC† Dec. | Symbol | Card Code | ANSCII†† | Meaning | Remarks |
|---|---|---|---|---|---|---|---|
| A0 | 40 | 64 | SP | blank | 2-0 | blank | |
| | 41 | 65 | | 12-0-9-1 | | | 41, 43, 46, and 47 are unassigned. |
| | 42 | 66 | ⊥ | 12-0-9-2 | | decode | |
| | 43 | 67 | | 12-0-9-3 | | | |
| | 44 | 68 | ⌊ | 12-0-9-4 | | minimum | 42, 44, 45, 48, and 49 are APL characters for 2741 APL use only. |
| | 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. |
| AE | 4B | 75 | . | 12-8-3 | 2-14 | period | |
| BC | 4C | 76 | < | 12-8-4 | 3-12 | less than | |
| A8 | 4D | 77 | ( | 12-8-5 | 2-8 | left parenthesis | On 2741 APL, ¢ is ⊂ (subset). |
| AB | 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. |
| A6 | 50 | 80 | & | 12 | 2-6 | ampersand | On 2741 APL, & is ∩ (intersection). |
| | 51 | 81 | | 12-11-9-1 | | | 51, 52, 54, 57, 58, and 59 are unassigned. |
| | 52 | 82 | | 12-11-9-2 | | | |
| | 53 | 83 | ⍞ | 12-11-9-3 | | quad | 53, 55, and 56 are APL characters for 2741 APL use only. |
| | 54 | 84 | | 12-11-9-4 | | | |
| | 55 | 85 | ⍑ | 12-11-9-5 | | encode | |
| | 56 | 86 | ○ | 12-11-9-6 | | circular | |
| | 57 | 87 | | 12-11-9-7 | | | |
| | 58 | 88 | | 12-11-9-8 | | | |
| | 59 | 89 | | 11-8-1 | | | |
| A1 | 5A | 90 | ! | 11-8-2 | 2-1 | exclamation point | On Model 7670, ! is \|. On 2741 APL, ! is ⍟ (degree). On 2741 APL, $ is ∪ (union). |
| A4 | 5B | 91 | $ | 11-8-3 | 2-4 | dollars | |
| AA | 5C | 92 | * | 11-8-4 | 2-10 | asterisk | |
| A9 | 5D | 93 | ) | 11-8-5 | 2-9 | right parenthesis | |
| BB | 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. |
| AD | 60 | 96 | - | 11 | 2-13 | minus, dash, hyphen | |
| AF | 61 | 97 | / | 0-1 | 2-15 | slash | |
| | 62 | 98 | ⌈ | 11-0-9-2 | | maximum | 62, 64, 66, and 67 are APL characters for 2741 APL use only. |
| | 63 | 99 | | 11-0-9-3 | | | |
| | 64 | 100 | ⌊ | 11-0-9-4 | | down arrow | |
| | 65 | 101 | | 11-0-9-5 | | | |
| | 66 | 102 | ω | 11-0-9-6 | | omega | 63, 65, 68, and 69 are unassigned. |
| | 67 | 103 | ⊃ | 11-0-9-7 | | superset | |
| | 68 | 104 | | 11-0-9-8 | | | |
| | 69 | 105 | | 0-8-1 | | | |
| | 6A | 106 | ^ | 12-11 | 5-14 | circumflex | On Model 7670 ^ is ¬. On Model 7015 ^ is ∧ (caret). On 2741 APL, ^ is ⍲. On 2741 APL, ~ is ⍱. |
| AC | 6B | 107 | , | 0-8-3 | 2-12 | comma | |
| A5 | 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. |
| BE | 6E | 110 | > | 0-8-6 | 3-14 | greater than | |
| BF | 6F | 111 | ? | 0-8-7 | 3-15 | question mark | |
| | 70 | 112 | ∧ | 12-11-0 | | APL | 70-72, 74, 76, and 79 are APL characters for 2741 APL use only. |
| | 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 | |
| BA | 7A | 122 | : | 8-2 | 3-10 | colon | |
| A3 | 7B | 123 | # | 8-3 | 2-3 | number | |
| C0 | 7C | 124 | @ | 8-4 | 4-0 | at | |
| A7 | 7D | 125 | ' | 8-5 | 2-7 | apostrophe (right single quote) | |
| BD | 7E | 126 | = | 8-6 | 3-13 | equals | |
| A2 | 7F | 127 | " | 8-7 | 2-2 | quotation mark | |

†Hexadecimal and decimal notation.

††Decimal notation (column-row).

Table H-3. CP-V Symbol-Code Correspondences (cont.)

| EBCDIC[t] Hex. | EBCDIC[t] Dec. | Symbol | Card Code | ANSCII[tt] | 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 | | 12-11-0-8-1 | | | |
| 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 | On Model 7670, ] is !. On Model 7015, ] is ¬. |
| B6 | 182 | | 12-11-0-6 | | | B0 and B6 through BB are unassigned. |
| B7 | 183 | | 12-11-0-7 | | | |
| 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 ). |

[t] Hexadecimal and decimal notation.

[tt] Decimal notation (column-row).

Table H-3. CP-V Symbol-Code Correspondences (cont.)

| EBCDIC[t] Hex. | EBCDIC[t] Dec. | Symbol | Card Code | ANSCII[tt] | Meaning | Remarks |
|---|---|---|---|---|---|---|
| C0 | 192 | SP | 12-0 | 2-0 | blank | Output only. |
| C1 | 193 | A | 12-1 | 4-1 | | C1-C9, D1-D9, E2-E9 comprise the uppercase alphabet. |
| C2 | 194 | B | 12-2 | 4-2 | | |
| C3 | 195 | C | 12-3 | 4-3 | | |
| C4 | 196 | D | 12-4 | 4-4 | | |
| C5 | 197 | E | 12-5 | 4-5 | | |
| C6 | 198 | F | 12-6 | 4-6 | | |
| C7 | 199 | G | 12-7 | 4-7 | | |
| C8 | 200 | H | 12-8 | 4-8 | | |
| C9 | 201 | I | 12-9 | 4-9 | | |
| CA | 202 | | 12-0-9-8-2 | | | CA through CF are unassigned. |
| CB | 203 | | 12-0-9-8-3 | | | |
| CC | 204 | | 12-0-9-8-4 | | | |
| CD | 205 | | 12-0-9-8-5 | | | |
| CE | 206 | | 12-0-9-8-6 | | | |
| CF | 207 | | 12-0-9-8-7 | | | |
| D0 | 208 | | 11-0 | | | D0 is unassigned. |
| D1 | 209 | J | 11-1 | 4-10 | | |
| D2 | 210 | K | 11-2 | 4-11 | | |
| D3 | 211 | L | 11-3 | 4-12 | | |
| D4 | 212 | M | 11-4 | 4-13 | | |
| D5 | 213 | N | 11-5 | 4-14 | | |
| D6 | 214 | O | 11-6 | 4-15 | | |
| D7 | 215 | P | 11-7 | 5-0 | | |
| D8 | 216 | Q | 11-8 | 5-1 | | |
| D9 | 217 | R | 11-9 | 5-2 | | |
| DA | 218 | | 12-11-9-8-2 | | | DA through DF are unassigned. |
| DB | 219 | | 12-11-9-8-3 | | | |
| DC | 220 | | 12-11-9-8-4 | | | |
| DD | 221 | | 12-11-9-8-5 | | | |
| DE | 222 | | 12-11-9-8-6 | | | |
| DF | 223 | | 12-11-9-8-7 | | | |
| E0 | 224 | – | 0-8-2 | 2-13 | minus | Output only. E1 is unassigned. |
| E1 | 225 | | 11-0-9-1 | | | |
| E2 | 226 | S | 0-2 | 5-3 | | |
| E3 | 227 | T | 0-3 | 5-4 | | |
| E4 | 228 | U | 0-4 | 5-5 | | |
| E5 | 229 | V | 0-5 | 5-6 | | |
| E6 | 230 | W | 0-6 | 5-7 | | |
| E7 | 231 | X | 0-7 | 5-8 | | |
| E8 | 232 | Y | 0-8 | 5-9 | | |
| E9 | 233 | Z | 0-9 | 5-10 | | |
| EA | 234 | | 11-0-9-8-2 | | | EA through EF are unassigned. |
| EB | 235 | | 11-0-9-8-3 | | | |
| EC | 236 | | 11-0-9-8-4 | | | |
| ED | 237 | | 11-0-9-8-5 | | | |
| EE | 238 | | 11-0-9-8-6 | | | |
| EF | 239 | | 11-0-9-8-7 | | | |
| F0 | 240 | 0 | 0 | 3-0 | | |
| F1 | 241 | 1 | 1 | 3-1 | | |
| F2 | 242 | 2 | 2 | 3-2 | | |
| F3 | 243 | 3 | 3 | 3-3 | | |
| F4 | 244 | 4 | 4 | 3-4 | | |
| F5 | 245 | 5 | 5 | 3-5 | | |
| F6 | 246 | 6 | 6 | 3-6 | | |
| F7 | 247 | 7 | 7 | 3-7 | | |
| F8 | 248 | 8 | 8 | 3-8 | | |
| F9 | 249 | 9 | 9 | 3-9 | | |
| FA | 250 | X | 12-11-0-9-8-2 | | multiply | FA through FF are APL characters for 2741 APL use only. |
| 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. |

[t] Hexadecimal and decimal notation.

[tt] Decimal notation (column-row).

Table H-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 |
| ETX (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 H-4. ANSCII Control-Character Translation Table (cont.)

| Input | | | | | Output | |
|---|---|---|---|---|---|---|
| ANSCII | TTY Key | Echoed | Prog. Receives (EBCDIC) | Process | EBCDIC | Transmitted (ANSCII) |
| SYN (16)† | V^C | SYN | SYN | None | SYN† (16) | SYN (not transmitted for last character in user's buffer). |
| ETB (17)† | W^C | ETB | ETB | None | ETB (17) | ETB |
| CAN (18) | X^C | Back-arrow and CR/LF | None | Cancel input or output message. | CAN (18) | CAN |
| EM (19) | Y^C | Back-arrow and CR/LF | None | Monitor Escape/ Control to TEL | EM (19) | EM |
| SUB (1A) | Z^C | SUB | SUB | Input Complete | SUB (1A) | # (A3) |
| ESC (1B) | K^CS ESC PREFIX | None | None | Initiate escape sequence mode. | ESC (1B) | ESC |
| FS (1C) | L^CS | FS | FS | Input Complete | FS (1C) | FS |
| GS (1D) | M^CS | GS | GS | Input Complete | GS (1D) | GS |
| RS (1E) | N^CS | RS | RS | Input Complete | RS (1E) | RS |
| US (1F) | O^CS | US | US | Input Complete | US (1F) | US |
| } (7D) | ALT-MODE | } or None | } or None | } if model 37; as ESC if model 33, 35, or 7015. | }(B3) | }(7D) |
| ~(7E) | ESC (7015) | ~or None | ~or None | ~if model 37; as ESC if model 33, 35, or 7015 | ¬(5F) | ~(7E) |
| DEL (7F) | Rubout | \ | None | Rubout last character. | DEL (FF) | None |
| All ANSCII upper and lower case alphabetics are translated on input into the corresponding EBCDIC graphics as shown in Tables C-1 and C-2. All special graphics map as shown, allowing for Table C-1, Note 2, and the exceptions above for model 33 and 35. Lower case alphabetics map into corresponding EBCDIC upper case if the ESC U mode is set. Upper case alphabetics map into corresponding EBCDIC lower case if ESC) is set. | | | | | Alphabetic and symbol output translation is also as shown in Tables C-1 and C-2; for Models 33 and 35, and 7015 terminals, however, lowercase alphabetics are automatically translated to upper case. | |

†These characters are communication control characters reserved for use by hardware. Any other use of them risks incompatibility with future hardware developments and is done so by the user at his own risk.

Table H-4. Substitutions for Nonexistent Characters on 2741 Keyboards

| EBCDIC Character | APL Keyboard | Selectric Keyboard | EBCD Keyboard |
|---|---|---|---|
| > | > | , (upper case) | > |
| < | < | . (upper case) | < |
| ^ | ↑ | ¢ | ¢ |
| \| | \| | ° (degree) | \| |
| ¬ | ~ | ± | ¬ |
| # | ≠ | # | # |
| % | ρ | % | % |
| ¢ | ⊂ | ¢ | ¢ |
| @ | α | @ | @ |
| " | ∇ | " | " |
| ! | ∘ | ! | ! |
| & | ∩ | & | & |
| $ | ∪ | $ | $ |

# 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.


* command, ANLZ, 43
? command, DRSP, 104
↑ command, ANLZ, 43
2741 terminal,
  substitutions for nonexistent characters, 211


## A


A Programming Language 7
active interrupt, 119
ALL command, ANLZ, 40
Analyze (see ANLZ)
ANLZ, 39, 10, 22
ANLZ, batch mode, 39
ANLZ, command summary, 55, 54
ANLZ, commands, 39
  *, 43
  ↑, 43
  ALL, 40
  BF, 44
  CLOSE, 45
  COMPARE, 43
  DELTA, 44
  DISPLAY, 40
  DUMP, 45
  END, 45
  HELP, 45
  INPUT, 40
  IS, 45
  LINE FEED, 43
  loc, 40
  loc=value, 43
  loc1, loc2, 43
  LP, 44
  MAP, 43
  MONITOR, 43
  NODELTA, 44
  PRINT, 44
  ROWS, 44
  RUN, 40
  SEARCH, 44
  SMASK, 44
  SYMBOLS, 45
  SYMBOL/, 45
  UC, 44
  UNMAP, 43
ANLZ, ghost mode, 39
ANLZ, messages, 54
ANLZ, on-line mode, 39
ANLZ, output, 45
ANS COBOL, 7
ANS labeled tape, xi
ANSCII, 204, 202, 209
APL, 7
application processors, 10
armed interrupt, 119
automatic recovery, 22


## B


BASIC, 6
Batch (processor), 10
batch job, xi
batch processing, 1
BF command, ANLZ, 44
binary input, xi
booting, 23, 35
  from disk, 37
bootstrap I/O error recovery, 38
bootstrap operations (see booting)


## C


CCI, 93, 4
character sets, 202
CIRC, 11
cleared interrupt, 119
CLIS command, PPS, 64
CLOSE command, ANLZ, 45
cluster/unit matrix, 148
COBOL, 7
COBOL On-Line Debugger, 9
codes and correspondences, 205
command processor programming, 95
command processors, 3
command summaries,
  ANLZ, 55, 54
  DRSP, 107, 104
  ELLA, 85, 84
command syntax notation, x
COMPARE command, ANLZ, 43
compressed language, 295
concatenation, xi
conditional patch control commands, 33
conflicting reference, xi
Control (processor), 4
control message, xi
cooperative, xi
CP-V operating system, 3
crash analysis (see ANLZ)
control codes, 267
Control Command Interpreter, 93, 4
control commands, xi

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


Data Control Block (see DCB)
DCB, 90, xi
  diagnostic, 114, 109
DDCB, 114, 109
DEFCOM, 9
DELETE command,
  DRSP, 103
  GENMD, 33
Delta, 8
DELTA command, ANLZ, 44
Delta format patches, 23
DEV command, ELLA, 78
DEVDMP, 5
device designation codes, 148
device names, 148
Device Save/Restore processor, 5
device type codes, 148
diagnostic DCB, 114, 109
diagnostics (see on-line peripheral diagnostic facilities)
disabled interrupt, 119
disarmed interrupt, 119
DISP command, ELLA, 75
DISPL command, ELLA, 81
DISPLAY command,
  ANLZ, 40
  PPS, 138
DRSP, 102, 10
DRSP, command summary, 107, 104
DRSP, commands,
  ?, 104
  DELETE, 103
  END, 104
  ENTER, 102
  LIST, 103
  LISTALL, 103
  REPLACE, 103
DRSP, error messages, 104
DRSP, limitations and restrictions, 104
DUMP command, ANLZ, 45
Dynamic Replacement of Shared Processors (see DRSP)
DYNRESDF command, PPS, 139


## E


EASY, 4
EBCDIC, 203, 202
Edit (processor), 9
EDMS, 11
ELLA, 61, 10, 21
ELLA, command summary, 85, 84
ELLA commands,
  CLIS, 64
  DEV, 78
  DISP, 75
  DSPL, 81
  END, 76
  MOD, 78
  RSET, 76
  SET, 61
  SLIS, 71
  SUM, 75
  TIME, 77
  TYPE, 78
ELLA, error log entry headings, 66
ELLA, error log entry types, 71
ELLA, input/output assignments, 61
ELLA, input/output characteristics, 63
ELLA, interrupting execution, 64
ELLA, messages, 84
ELLA, predefined tasks, 81
ELLA, starting execution, 61
enabled interrupt, 119
END command,
  ANLZ, 45
  DRSP, 104
  ELLA, 76
  PPS, 140
:END command (boot-time), 28
ENTER command, DRSP, 102
ERR:FIL, 61, 21
ERRFILE file, 168, 61
ERRFILE file formats, 168
  bad granule release, 178
  configuration record, 177
  device error, 173
  duplicate entries, 177
  enqueue table overflow, 184
  errlog record length error, 171
  file inconsistency error, 175
  hardware errors, 174
  I/O activity count, 182
  illegal entry type, 172
  incorrect time, 172
  instruction exception, 176
  lost entry indicator, 177
  memory fault interrupt, 172
  memory parity secondary record, 183
  MFI primary record, 183
  operator message, 182
  partitioned resource, 184
  PFI primary record, 182
  power on, 177
  processor fault interrupt, 172
  read error, 171
  remote processing error, 178
  returned resource, 184
  secondary record for poll information, 183
  secondary records for device errors, disk pack, RAD, and tape, 174
  Sigma 6/7 memory parity secondary record, 183
  SIO failure, 172
  software-detected symbiont inconsistencies, 176
  system identification, 177
  system startup, 175
  time out, 173

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.


time stamp, 178
unexpected interrupt, 173
watchdog timer, 175
Xerox 560 memory parity secondary record, 183
ERRMSG file, 59
error detection and recovery, 20,21
error log display, 174
error log file (see ERRFILE)
Error Log Listing program (see ELLA)
error log, reading, 189
error log, writing, 189
error message file, 59
error messages (see messages)
error record terminology, 168
execution control processors, 8
Extended FORTRAN IV, 5
extension of output files, 95,xi
external reference, xi


## F


FDP, 9
fid, 91
File Restore processor, 5
files,
extension, 95,xi
identification, 91
shared, 116
FILL (processor), 4
Fix, 4
fixed monitor locations, 88
FLAG, 7
FORTRAN, 5
FORTRAN Debug Package, 9
FORTRAN Load and Go, 7
FPT, xi
FREE command, PPS, 138
FRES, 5
FSAVE, 5
function parameter table, xi


## G


GAC, 5
:GENDCB command, 32
General Purpose Discrete Simulator, 11
GENMD,
commands,
DELETE, 33
GENMD, 33
LIST, 33
error messages, 34,33
patches, 33
GENMD command, GENMD, 33
GET command, PPS, 138
ghost job, xi
ghost job, initiating, 190
global symbol, xi
:GO command (boot-time), 26
GO file, xi
GPDS, 11
Granule Accounting Cleanup processor, 5
granule, xi


## H


hardware-error diagnostic CALs, 85
initiate ghost job, 86
read error log, 86
write error log, 86
HELP command,
ANLZ, 45
ERR:LIST, 175


## I


I/O scheduling, 17
ICB, 119
initialization and start-up routines, 264
INPUT command, ANLZ, 40
interrupt connection and control services, real-time, 119
interrupt control block, 119
interrupt label, 119
IOP designation codes, 148
IS command, ANLZ, 45


## J


JIT, 88,xi
job information table, 88,xi
job step, xi


## K


key, xi
key-in, xi

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.

## L


Label, 5
language processors, 5, xi
libraries, 118
  public, 98
library load module, xi
LINE FEED command, ANLZ, 43
Link (processor), 8
linking loader, xi
  (see Link processor)
LIST command,
  DRSP, 103
  GENMD, 33
LISTALL command, DRSP, 103
Load (processor), 8
load map, xi
load module, xi
loc command, ANLZ, 40
loc=value command, ANLZ, 43
loc1, loc2 command, ANLZ, 43
log-on connection, 87
logical device, xi
logical device stream, xi
LOGON/LOGOFF, 3
LP command, ANLZ, 44


## M


M:BLIST, 111
M:CHKINT, 132
M:CLOCK, 125
M:COC, 136
M:CONNECT, 120
M:DCLOSE, 111
M:DDCB, 109
M:DISCONNECT, 121
M:DMOD#, 113
M:DOPEN, 110
M:EXCP, 133
M:EXU, 129
M:FPP, 130
M:GDG, 131
M:GJOB, 130
M:GJOBCON, 120
M:GPP, 130
M:HOLD, 124
M:INHIBIT, 122
M:INTCON, 122
M:INTRTN, 123
M:INTSTAT, 124
M:IOEX, 127
M:LOCK, 113
M:MAP, 113, 130
M:MASTER, 129
M:NEWQ, 133
M:QFI, 123
M:QUE, 135
M:QUEUE, 141
M:RDG, 131
M:RUE, 131
M:SIO, 112
M:SLAVE, 129
M:STARTIO, 127
M:STOPIO, 125
Manage, 8
MAP command, ANLZ, 43
master system tape, 23, 24
memory control, 89
memory layout, 19
memory management, 13
messages,
  ANLZ, 54
  ELLA, 84
  GENMD, 263
  on-line peripheral diagnostics, 114
  PASS0, 38
Meta-Symbol, 6
MOD command, ELLA, 78
monitor, 12, xii
MONITOR command, ANLZ, 43
monitor DEFs (for real-time), 140
monitor dump analysis program (see ANLZ)


## N


NODELTA command, ANLZ, 44


## O


object language, xii
object module, xii
on-line job, xii
on-line peripheral diagnostic facilities, 108, 22
  access device directly for diagnostics, 190
  abnormal codes and messages, 114
  DDCB, 114, 109
  M:BLIST, 111
  M:DCLOSE, 111
  M:DDCB, 109
  M:DMOD#, 113
  M:DOPEN, 110
  M:LOCK, 113
  M:MAP, 113
  M:SIO, 112
  PSECT directive, 108
  restrictions, 108
operational label, 147, xii
overlay loader, xii
  (see Load processor)
overlay program, xii
overlay restrictions, shared processors, 89

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.

## P


page allocation for real-time, 136
:PART command,
  boot-time, 26
partitioning resources, 26
PASS0, 38
PASS0, error messages, 38
patch, xii
patch control commands, conditional, 33
patch deck comment cards, 34
patch deck structure, 23
patch deck symbol table, 25
patch file creation, 35
patches, Delta format, 23
patching operations, 23
PCL, 9
Peripheral Conversion Language, 9
peripheral device (see device)
peripheral diagnostic facilities (see on-line peripheral diagnostic facilities)
physical device, xii
  (see device)
physical page allocation for real-time, 136
Physical Page Stealer (see PPS)
PPS, 137
PPS commands,
  DISPLAY, 138
  DYNRESDF, 139
  END, 140
  FREE, 138
  GET, 138
  RESDF, 139
preventive maintenance, 199
PRINT command, ANLZ, 44
procedures, 199, 195
  M:BLIST, 111
  M:CHKINT, 132
  M:CLOCK, 125
  M:COC, 136
  M:CONNECT, 120
  M:DCLOSE, 111
  M:DDCB, 109
  M:DISCONNECT, 121
  M:DMOD#, 113
  M:DOPEN, 110
  M:EXCP, 133
  M:EXU, 129
  M:FPP, 130
  M:GDG, 131
  M:GJOB, 130
  M:GJOBCON, 120
  M:GPP, 130
  M:HOLD, 124
  M:INHIBIT, 122
  M:INTCON, 122
  M:INTRTN, 123
  M:INTSTAT, 124
  M:IOEX, 127
  M:LOCK, 113
  M:MAP, 113, 130
  M:MASTER, 129
  M:NEWQ, 133
  M:QFI, 123
  M:QUE, 135
  M:QUEUE, 141
  M:RDG, 131
  M:RUE, 131
  M:SIO, 112
  M:SLAVE, 129
  M:STARTIO, 127
  M:STOPIO, 125
processor management, 18
processors,
  application, 10
  command, 3
  execution control, 8
  language, 5
  service, 9
  shared processor facilities, 87
  system management, 4
  user, 87, 13
program product, xii
prompt character, xii
protective mode, xii
PSECT directive, 108, 130
public library, 98, xii
public programs, 87


## R


RATES, 4
read error log, 189
real-time facilities, 119
  clock service, 125
  device preemption services, 125
  direct I/O services, 127
  dynamic physical page allocation, 136
  interrupt connection and control services, 119
  lock in core service, 124
  miscellaneous services, 130
reconfiguration and partitioning commands, 26
  :END, 28
  :GO, 26
  :PART, 28
  :REMOVE, 27
  :SAVE, 27
  :TYPE, 27
reconfiguration and partitioning command summary, 28
reconfiguration and partitioning messages, 30, 28
recovery, 20, 21
reentrant, xii
relative allocation, xii
relocatable object module (ROM), xii
remote diagnostic assistance, 165, 22
:REMOVE command (boot-time), 27
REPLACE command, DRSP, 103
RESDF command, PPS, 139

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.


RESDF memory CAL, 140
resident program, xii
response time, xii
ROWS command, ANLZ, 44
RSET command, ELLA, 76
RUN command, ANLZ, 40

## S

:SAVE command (boot-time), 27
scheduler, xii
scheduler inputs, 14
scheduler operation, 15
scheduler output. 15
scheduler status queues, 16
scheduling, 13
screech codes, 149
SEARCH command, ANLZ, 44
secondary storage, xii
semi-protective mode, xii
service processors, 9
SET command, ELLA, 61
shared file use, 95
shared processor, xii
shared processor facilities, 87
shared processor maintenance, 102
shared processor programming, 88
shared programs, 87
Simulation Language, 8
SL-1, 8
SLIS command, ELLA, 71
SMASK command, ANLZ, 44
software check codes (see screech codes)
Sort/Merge, 10
source language, xii
special shared processor, xii
specific allocation, xii
SR1, SR2, SR3 and SR4, xiii
static core module, xiii
STATS, 5
stream-id, xiii
  (see logical device stream)
SUM command, ELLA, 75
SUMFILE file, 174, 190
Summary (processor), 5
swap hardware organization, 17
swap-in, swap-out queues, 17
symbiont, xiii
symbol-code correspondences, 205
symbolic input, xiii
symbolic name, xiii
SYMBOLS command, ANLZ, 45
SYMBOL/ command, ANLZ, 45
symbols, graphic, 202
SYMCON, 80
SYSCON (processor), 5
SYSGEN, 9,xiii
SYSTEM DIAG, 109
system error log file (see ERRFILE)
system generation, 9,xiii
system integrity, 19
system library, xiii
system loading, 264
system management processors, 4
system programming facilities, 2
system register, xiii
SYSTEM RTPROCS, 119
SYSTEM SIG7, 109
SYSTEM SIG9, 109
system start-up and initialization, 264
system tape format, 23,24

## T

tape, master system, 23,24
task control block (TCB), xiii
TEL, 4
TEL, scan, 92
Terminal Executive Language, 4
terminal I/O, 93
TIME command, ELLA, 77
time-sharing, 1
transaction processing, 2
transaction processing facilities, 141
  list formats, 143
  M:QUEUE FPTs, 142
  M:QUEUE procedure format, 141
  M:QUEUE procedure output, 144
TYPE command, ELLA, 78
:TYPE command (boot-time), 27

## U

UC command, ANLZ, 44
UNMAP command, ANLZ, 43
unsatisfied reference, xiii
user processors, 87, 13
user status queues, 15

## V

virtual memory, special processors, 87
VOLINIT, 5

## W

waiting interrupt, 119
write error log, 189

## X

Xerox 560 cluster/unit matrix, 148
Xerox standard compressed language, 201
Xerox standard object language, 185
Xerox standard symbols, codes, and correspondences, 202

RANK XEROX DATA SYSTEMS
Rank Xerox Limited
York House
Empire Way
Wembley
Middlesex HA9 0QF
ENGLAND

Telephone 01 903 4521
Telecopier transceiver ext 284
Telex 27992

Rank Xerox and Xerox are trade marks

All specifications subject to change without notice.

Sigma is a trademark used by Rank Xerox for computers.