XEROX

Xerox Control Program for Real-Time (CP-R)

Xerox 550 and Sigma 9 Computers

Real-Time and Batch Processing
Reference Manual

90 30 85D

April 1976

File No.: 1X13
Printed in U S A

REVISION

This publication is a revision of the Xerox Control Program for Real-Time (CP-R)/RT,BP Reference Manual, Publication Number 90 30 85C (dated November 1974). This revision incorporates changes that reflect version D00 of the CP-R system.

RELATED PUBLICATIONS

Title	Publication No.
Xerox 550 Computer Reference Manual	90 30 77
Xerox Sigma 9 Computer Reference Manual	90 17 33
Xerox Control Program for Real-Time (CP-R)/OPS Reference Manual	90 30 86
Xerox Control Program for Real-Time (CP-R)/System Technical Manual	90 30 88
Xerox Control Program for Real-Time (CP-R)/RT,BP User's Guide	90 30 87
Xerox Character-Oriented Communications Equipment/Reference Manual (Models 7611-7616/7620-7623)	90 09 81
Xerox Mathematical Routines/Technical Manual	90 09 06
Xerox Assembly Program (AP)/LN,OPS Reference Manual	90 30 00
Xerox SL-1/Reference Manual	90 16 76
Xerox Extended FORTRAN IV/LN Reference Manual	90 09 56
Xerox Extended FORTRAN IV/OPS Reference Manual	90 11 43
Xerox Extended FORTRAN/Library Technical Manual	90 15 24
Xerox Availability Features (CP-R) Reference Manual	90 31 10

Manual Content Codes: BP - batch processing, LN - language, OPS - operations, RP - remote processing, RT - real-time, SM - system management, 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 xi

COMMAND SYNTAX NOTATION xii

GLOSSARY xiii

1. INTRODUCTION 1

Operating System 1
CP-R Terms and Processes 1
Job 1
Task 1
Load Module 1
Program 1
Foreground 1
Background 1
Terminal Job Entry System 2
Temp Stacks 2
Data Control Block 2
Function Parameter Table 2
Task Control Block 2
Program Control Block 2
Roll-Out/Roll-In 2
Reentrant Subroutine 2
Philosophy of Operation 2
Real-Time Processing 2
Batch Processing 3
CP-R Functions 3
Secondary Storage Utilization 3
Job Accounting 4
Public Library 4
Symbionts 5
Control Task 5
Overlays 5
Memory Protection 5
Disk Write Protection 6
Language Processors 6
Service Processors 6
Overlay Loader 6
RADEDIT 6
Edit 7
Background Job Organization 7
Foreground Job Organization 7
Input/Output Specifications 8
Physical Device Names 9
Disk File Identifiers 10
Operational Labels 11
Resolving I/O Medium Name Ambiguities 11

2. CONTROL COMMANDS 12

Job Control Processor 12
System Control Commands 13
JOB 13
ASSIGN 13
LOAD 15
ATTEND 15
MESSAGE 15
LIST 16
PAUSE 16
CC 17
LIMIT 17
STDLB 17
ROV 17
RUN 18
INIT 18
SJOB 18
BATCH 19
ALLOBT 19
Dump Control Command 20
PMD 20
Input Control Commands 20
EOD 20
FIN 20
Utility Control Commands 21
PFIL, PREC 21
SFIL 21
REWIND 21
UNLOAD 22
WEOF 22
DAL 22
Processor Control Commands 22
Processor Interface with CP-R 23
JCP Messages 24

3. OPERATOR COMMUNICATIONS 26

CP-R Messages 26
Trap Handler Messages 26
Output Message Formats 31
Operator Key-In 31
Combined Key-Ins 40
Device Control 41
I/O Key-In Format 41
Card Reader Messages/Key-Ins 42
Card Punch Messages/Key-Ins 42
Disk Pack Messages/Key-Ins 42
Disk Data Protection 43
Line Printer Messages/Key-Ins 43
Magnetic Tape Messages/Key-Ins 43

4. INPUT/OUTPUT OPERATIONS 44

Permanent Files 44
Temporary Files 44
File Organization 45
Blocked Files 45
Unblocked Files 45
Compressed Files 45
Disk Access Methods 45
Sequential Access 45
Direct Access 45
Device Access 45
Disk Pack Files 45
Extensible Files 46
I/O Queueing 46
I/O Cleanup and I/O Start 46

Sharing DCBs Among Tasks 46
Sharing I/O Devices Among Tasks 47
Sharing Disk Files Among Tasks and Jobs 47
I/O End Action 47
Reserving I/O Devices for Foreground Use 48
Device Preemption 48
Direct I/O Execution (IOEX) 48
Keyboard-Printer Edited I/O 49
Logical Devices 49
Operational Labels 49
Data Control Blocks (DCBs) 49
DCB Creation 49
DCB Assignment 50
DCB Format 50
Error and Abnormal Conditions 54
I/O System Calls 54
Open a File 54
OPEN 54
Close a File 55
CLOSE 55
Read a Data Record 55
READ 55
Write a Data Record 56
WRITE 56
Rewind, Unload, and Write EOF Functions 58
REW 58
UNLOAD 58
WEOF 58
File and Record Positioning Functions 59
PFIL, PREC 59
ALLOT, DELETE, and Truncate Functions 60
ALLOT 60
DELETE, TRUNCATE 62
Allot or Delete Symbiont File 63
JOB 63
Print and Type Functions 64
PRINT, TYPE 64
DEVICE 65
Device Control Functions 66
Check Correspondence of DCB Assignments 66
CORRES 66
ASSIGN 66
GETASN 67
IOEX 69
STDLB 71
STOPIO, STARTIO 72

5. USER-TASK SCHEDULING AND OPERATION 74

Scheduling and Loading Programs 74
Loading and Terminating Foreground Secondary Tasks 74
Queueing Primary Foreground Program Run Requests 74
Loading and Releasing Primary Foreground Programs 75
Loading and Executing Background Programs 75
Task Control Block (TCB) 75
Primary Task Control Block Format 75
Program Control Block (PCB) 76
PCB Format 76
User Temp Stack 77
CP-R Temp Stack 77
Master and Slave Modes 77
Overlay Segment Operations 77
Trap Handling 78
CAL Handling 78
Return Functions 78
Interrupt Control 79
Connecting and Disconnecting Primary Tasks to Interrupts 79
Arming, Disarming, Enabling, and Disabling 79
End-Action 79
System Function Call Formats 80
KJOB 80
SJOB 81
SETNAME 82
RUN 83
RLS 84
INIT 84
SCHED 85
INT 87
PC 87
EXTM 87
SIGNAL 88
POLL 90
POST 91
ENQ 92
DEQ 93
TRAP 94
JTRAP 94
TRTY 95
TEXIT 95
TRTN 95
EXIT 95
EXDA 95
TERM 95
ABORT 96
CONNECT, ARM, DISARM, DISCONNECT 96
CALRTN 99
ENABLE, DISABLE, TRIGGER 99
START 99
STOP 100
STATUS, MODIFY 100
MASTER, SLAVE 102
SEGLOAD 102
WAIT 103
TIME 103
GETTIME 104
STIMER 104
ERRSEND 105
ALARM 106
RECALARM 106

6. CP-R MEMORY MANAGEMENT 107

Real Memory Allocation 107
CP-R System Memory 107
Foreground Private Memory 107
Foreground Preferred Memory 107
Secondary Task Memory 109
Primary Task Memory Allocation 109
Virtual Memory Allocation 109
System Virtual Memory 109
Task Virtual Memory 109

Reserved Pages 109
Access Protection 110
Software Segmentation 110
Sharing Segments 112
Segment Access Protection 113
Secondary Task Structures 113
Linking a Program in Real Versus Virtual Memory 114
Linking a Program using Simplified Memory Management 115
Memory Management System Calls 115
ACTIVATE 115
DEACTIVATE 116
ERASE 116
GETPAGE 116
RELPAGE 117
LOCK 118
UNLOCK 118
PREFMODE 119
Roll-Out/Roll-In 119

7. ASYNCHRONOUS OPERATION CONTROL 121

Check Completion 121
CHECK 121
DELFPT 122
WAITALL 123
WAITANY 124
TEST 124

8. CP-R DEBUG SERVICE 126

Debug Call 126
DEBUG 126
Method of Operation 127
Debug Input 127
Debug Output 127
Debug Trap Control 128
Debug Commands 128
Debug Command Syntax 128
A 129
C 129
M 129
I 129
N 130
P 130
D 130
E 130
R 131
B 131
L 131
Q 131
W 131
Debug Snapshot 131
S 131
Debug Error Messages 132

9. CP-R MEDIA SERVICE 134

Key-In Conversions 134
Foreground Conversions 134
MEDIA Processing 134
MEDIA Service Call 134
MEDIA 134
MEDIA Key-Ins 136
Conventions 137
Messages 137

10. OVERLAY LOADER 138

Overview 138
Functional Flow 138
Limitations 138
Overlay Load Modules 138
Overlay Structures 139
Overlay Restrictions 139
Overlay Control Commands 140
Syntax 140
Order of Control Commands 140
!OLOAD 140
:ROOT 142
:SEG 143
:LIB 145
:INCLUDE 145
:EXCLUDE 146
:COMMON 146
:RES 146
:LCOMMON 147
:MODIFY 148
:ASSIGN 149
:PUBLIB 149
:LMHDR 150
Program File 150
Root Segments 150
Overlay Segments 151
Temporary Disk Files 151
Loader-Generated Items 151
Program Control Block 151
Data Control Block 151
DCBTAB 151
OVLOAD Table 151
Temp Stacks 151
External Definitions 151
Libraries 152
System and User Library Libraries 152
Assembly Language 152
Entry Address 152
System and User Libraries on Disk 152
Constructing and Maintaining Library 153
Public Libraries 153
Calling the Public Libraries 153
Library Protection 153
Releasing a Public Library 153
Forming a Public Library 153
Map 153
Error Diagnostics 157
User Load-Time Assigns 165
M:DCB and F:DCB 165
Run-Time Assigns 165
Load-Time Assigns 165
FORTRAN Interface 165
Common Allocation 165
CONNECT 166

Calling Overlay Segments ____ 166
Main Program Name and Entry ____ 166
Labeled COMMON Names ____ 166
Blank COMMON Names ____ 166
Core Layout at Execution Time ____ 166

11. RADEDIT 168

Operating Characteristics ____ 168
File Allocation ____ 168
Skipping Bad Tracks ____ 168
System and User Library Files ____ 169
Algorithms for Computing Library File Sizes ____ 169
Disk Area Protection ____ 170
Calling RADEDIT ____ 170
Command Formats ____ 170
RADEDIT Commands ____ 170
:ALLOT ____ 170
:COPY ____ 171
:DPCOPY ____ 172
:DELETE ____ 173
:CLEAR ____ 173
:SQUEEZE ____ 173
:TRUNCATE ____ 174
:MAP ____ 174
:SMAP ____ 174
:LMAP ____ 175
:CATALOG ____ 176
:DUMP ____ 176
:XDMP ____ 177
:SAVE ____ 178
:RESTORE ____ 178
:BDSECTOR ____ 178
:GDSECTOR ____ 179
:END ____ 179
Error Messages ____ 179
Disk Restoration Messages ____ 179

12. PREPARING THE PROGRAM DECK 184

AP Examples ____ 184
Assemble Source Program, Listing Output ____ 184
Assemble Source Program, Listing Output, Load and Go Operations ____ 184
Assemble and Load a Program Written for Unmapped (RBM) Background Execution ____ 184
Assemble from Compressed Deck with Source and Updates, Listing Output ____ 184
Assemble Source Program, Compressed Output on Cards, Listing Output ____ 185
Assemble Source or Compressed Program in Batch Mode, Listing Output ____ 185
Assemble Source Program, Compressed Output on Disk File, Listing Output ____ 185
Assemble Compressed Deck from Disk File, Source Updates from Cards, Listing Output ____ 185
Assemble Source Program, Write Compressed Output on 9-Track Tape, Listing Output ____ 186
Assemble Compressed Program from 9-Track Tape, Listing Output ____ 186
FORTRAN Job Examples ____ 187
Combined FORTRAN Compilations, Plus FORTRAN Compile and Execute ____ 187
Compile and Execute FORTRAN Source Program with Real-Time Linkages ____ 188
Compile and Execute Program Using LS, BO Default Options ____ 189
Compile a FORTRAN Program and Setup for Execution in Foreground Area ____ 190
Overlay Loader Examples ____ 191
Batch, Using GO Links ____ 191
Segmented Background Job ____ 192
Foreground Job Examples ____ 193
Load and Execute Foreground Program ____ 193
Load and Execute Segmented Foreground Program ____ 194
Load Two Foreground Secondary Programs Sharing a Read-and-Execute Procedure ____ 195
Load Foreground Secondary Program with a Segment Fixed in Real Memory ____ 195
Load Program with Dynamically-Allocated Segment ____ 196

13. LINE EDITOR (EDIT) 197

Introduction ____ 197
Calling Edit ____ 197
Subject File Format ____ 197
Sequence Numbers ____ 197
File Identifiers ____ 197
Input/Output Conventions ____ 197
Multiline Records ____ 198
Break Function ____ 198
Error Response ____ 198
EDIT Commands ____ 198
Command Structure ____ 198
File Commands ____ 199
EDIT ____ 199
SAVE ____ 199
END ____ 199
SEQ ____ 199
BP ____ 199
Command Stream Control ____ 200
GO and RET ____ 200
Record Editing Commands ____ 201
IN ____ 201
IS ____ 202
DE ____ 202
TY ____ 202
TC ____ 202
TS ____ 203
MD ____ 203
MK ____ 203
FD ____ 204
FT ____ 204
FS ____ 204
RN ____ 204
CM ____ 205
SE ____ 205
SS ____ 205
ST ____ 205

Intrarecord Editing Commands ____ 205
Y and N ____ 206
CL ____ 207
S ____ 207
D ____ 207
P ____ 207
F ____ 208
O ____ 208
E ____ 208
R or L ____ 208
L ____ 208
R ____ 208
A ____ 209
C ____ 209
DE ____ 209
TS ____ 210
TY ____ 210
JU ____ 210
NO ____ 210
RF ____ 211
Messages ____ 211
Edit Command Summary ____ 211

14. TERMINAL JOB ENTRY 220

Terminal Executive Language ____ 220
EDIT ____ 220
MUST ____ 220
The Terminal ____ 220
Terminal Operations ____ 220
Initiating and Ending On-Line Sessions ____ 222
Typing Lines ____ 222
Typing Commands ____ 224
Detecting and Reporting Errors ____ 224
Interrupting CP-R ____ 224
Paper Tape Input ____ 225
Summary of Teletype Functions ____ 225
Terminal Executive Language ____ 225
Major Operations ____ 226
Minor Operations ____ 226
Error Handling ____ 226
TEL Commands ____ 226
TABS ____ 226
OFF ____ 226
MESSAGE ____ 226
STDLB ____ 226
MEDIA ____ 226
BATCH ____ 226
JOB ____ 227
CANCEL ____ 227
SETNAME ____ 227
RUN ____ 227
INIT ____ 227
DEBUG ____ 227
EXIT ____ 227
STOP ____ 228
START ____ 228
EXTM ____ 228
QUIT ____ 228
CONTINUE or GO ____ 228
Interrupting, Resuming, and Terminating Executions ____ 228
MUST Operations ____ 228
TJE Account Maintenance ____ 228
Standard Symbols, Codes and Correspondences ____ 229
Standard Symbols and Codes ____ 229
Standard Character Sets ____ 229
Control Codes ____ 229
Special Code Properties ____ 229

15. SYSTEM GENERATION 238

BOOT26 ____ 238
SYSGENLOAD ____ 238
SYSGEN/SYSLOAD ____ 238
SYSGEN ____ 238
SYSLOAD ____ 238
Memory Allocation ____ 238
Disk Allocation ____ 246
Tables Allocated and Set by SYSGEN ____ 248
Input Parameters ____ 249
SYSGEN Control Commands ____ 250
:PROC ____ 250
:MONITOR ____ 250
:RESERVE ____ 251
:CHAN ____ 252
:DEVICE ____ 253
ALLOBT ____ 256
MDEF ____ 256
MOD ____ 257
:COC ____ 257
:STDLB ____ 258
:CTINT ____ 258
:INTLB ____ 258
:PUNCH ____ 259
:FIN ____ 259
:SITE ____ 259
:COMMENT ____ 259
:SYSLD ____ 259
SYSLOAD ____ 260
MAP Option ____ 260
All Option ____ 261
FAST Option ____ 261
Override Option (OVR) ____ 261
Update Option ____ 261
Allocation of SP Area ____ 262
SYSGEN and SYSLOAD Alarms ____ 262
Loading System Processors and User Programs ____ 262

16. HARDWARE CONFIGURATION GUIDELINES 266

Introduction ____ 266
Hardware Interrupt Requirements ____ 266
Main Storage (Memory) Requirements ____ 268
Memory Space Requirements for CP-R ____ 268
Memory Space Requirements for CP-R Processors ____ 268
Memory Space Requirements for User-Foreground Programs ____ 269
Secondary Storage Requirements ____ 269
BT Area Storage Requirements ____ 270
User Secondary Storage Requirements ____ 270
Removable Disk Packs and Cartridges ____ 270

Peripheral Equipment Requirements and Options ____ 272
Additional Peripheral Options ____ 272
Miscellaneous Hardware Options ____ 272
Register Blocks ____ 272
Power On/Off ____ 272

INDEX 331

APPENDIXES

A. SYSTEM CP-R 273

System CP-R Procedure References ____ 273
Default Form ____ 273
In-Line FPT Form ____ 273
FPT-Only Form ____ 273
CAL-Only Form ____ 273
Parameter Field ____ 273
System CP-R Errors ____ 274
CAL-Parameter Groups ____ 274
Specific CAL-Parameter Groups ____ 274

B. XEROX STANDARD COMPRESSED LANGUAGE 289

C. REAL-TIME PERFORMANCE DATA 290

Response to Interrupts by Centrally Connected Tasks ____ 290
I/O Interrupt ____ 290
Interrupt Inhibits ____ 290
Secondary Task Dispatch Time ____ 290
Console Interrupt ____ 290
Overlay Loading ____ 290

D. COOPERATIVES AND SYMBIONTS 291

Cooperative ____ 291
Symbionts ____ 291
Symbiont-Cooperative Housekeeping ____ 291

E. JCP LOADER 293

Loading Nonsegmented Programs ____ 293
Loading Segmented Programs ____ 293

F. SYSTEM INITIALIZATION AND PATCHING 294

Input Options ____ 294
Quick Patches ____ 294
Patch Command Formats ____ 295
Patch System Overlay or JCP ____ 295
Patch Simulation Routine ____ 295
Patch CP-R Monitor ____ 295
Patch System Tables ____ 295
Modify Patch Area ____ 295
Trace Command Formats ____ 295
Clear Command Format ____ 296
IEND Command Format ____ 296
System Patching and Tracing Diagnostics ____ 296

G. CHARACTER-ORIENTED COMMUNICATIONS ROUTINES 297

System Interface ____ 297
User Program Interface ____ 298
Preparation and Use of COC ____ 298
COC Commands and Calling Sequences ____ 298
Call CSTART (Initiate COC Activity) ____ 298
Call CSET (Set Line Table Parameters) ____ 299
Call CWRITE (Write Output to a Terminal) ____ 300
Call CREAD (Accept Input from a Terminal) ____ 301
Call CMOVE (Move Input to a User Area) ____ 302
Call CHECK (Check I/O Line Status) ____ 303
Call CSTOP (Terminate COC Activity) ____ 303
I/O Interrupt Routines ____ 304
Input Interrupt Processing ____ 304
Output Interrupt Processing ____ 304
End-Action Processing ____ 304
Special Action Processing ____ 304
Break Signal Handling ____ 305
Escape Sequence Processing ____ 305
COC Database ____ 305
COC Parameters ____ 305
Line Control Tables ____ 305
Line State Table Format ____ 305
Terminal Mode Table Format ____ 306
Terminal Type Table (COCTERMN) ____ 307
Buffer Pool ____ 307
Standard Input/Output Translation Tables ____ 307
Input Special Action Table ____ 307
Escape Sequence Tables ____ 311
Output Conversion (EBCDIC-USASCII) ____ 311

H. JOB MANAGEMENT 313

Concept of a Job ____ 313

I. TASK MANAGEMENT 314

Disconnecting Primary Tasks from Interrupts ____ 314
Event Management ____ 314

J. RESOURCE MANAGEMENT 316

K. PERIODIC SCHEDULING 318

L. ERROR LOGGING 319

M. SYSTEM ALARM PROCEDURES 320

N. CP-R SERVICE CALLS 321

O. ERROR AND ABNORMAL CODES 322

Type Completions (TYC) 322
Error Codes 322

P. VOLUME INITIALIZATION 328

Introduction 328
Loading VOLINIT 328
VOLINIT Commands 328

FIGURES

1. Overlay Structure 7
2. Loading Overlay Loader from Cards 16
3. Display Format 40
4. Error Summary Example 40
5. Task Status Format 41
6. User Temp Stack Format at CAL Processor Entry 98
7. TCB for Centrally Connected CALs 98
8. Example of Memory Organization for 64K System 108
9. CP-R System Memory 108
10. Virtual Memory Organization 110
11. Segment States 112
12. Tree Structure Options 113
13. An Overlay Program 139
14. Overlay Example 144
15. Object Module from GO File 145
16. :LIB Command Usage 145
17. :EXCLUDE Command Usage 146
18. DSECT Allocation Example 147
19. :MODIFY Command Items Example 149
20. Typical PROGRAM Map 155
21. Blank COMMON Allocation by Default 165
22. Blank COMMON Option 166
23. Standard Core Layout of a Program 167
24. Permanent Disk Area Before Squeezing 169
25. Permanent Disk Area After Squeezing 169
26. MAP SP Output Example 175
27. A Multiline Record 198
28. Model 33 Teletype Terminal Keyboard 221
29. SYSGEN Map Example 239
30. Disk Allocation Example 247
31. Linking the CP-R System Processors (Abbreviated) 265
32. CP-R Lowest-Cost, Minimum Configuration 266
33. Typical Configuration 267
34. CP-R Disk File Management 271
A-1. Specific CAL-Parameter Groups 274
D-1. Information Flow Through Cooperative and Symbionts 292

TABLES

1. I/O Device Type Codes 8
2. Channel or Cluster/Unit Designation Codes 8
3. Device Designation Codes 8
4. Disk Area Codes 9
5. System Operational Labels 9
6. JCP Messages 24
7. CP-R Messages and Responses 26
8. Standard Operator Key-Ins 32
9. Terminal Job Entry Key-Ins 37
10. Symbiont Key-Ins 38
11. Media Conversion Key-Ins 39
12. System DCBs 50
13. Line Printer Format Control Codes 52
14. IOEX Function Status Returns 70

15. Segment States Relative to a Task ____ 111
16. Real and Virtual Memory Program Linkage ____ 114
17. Roll-Out Levels ____ 120
18. Debug Error Messages ____ 132
19. Overlay Loader Diagnostics ____ 157
20. RADEDIT Error Messages ____ 179
21. Disk Restoration Messages ____ 182
22. Edit Messages ____ 211
23. Edit Command Summary ____ 213
24. On-Line User Processors ____ 220
25. Summary of Teletype Services ____ 225
26. CP-R 8-Bit Computer Codes (EBCDIC) ____ 230
27. CP-R 7-Bit Communication Codes (ANSCII) ____ 231
28. CP-R Symbol-Code Correspondences ____ 232
29. ANSCII Control-Character Translation Table ____ 236
30. Disk Area Default Sizes ____ 247
31. GO, OV, X1-X9 Default Sizes ____ 248
32. System Device Model Numbers and Parameters (Disk Devices) ____ 254
33. System Device Model Numbers and Parameters (Non-Disk Devices) ____ 255
34. SYSGEN and SYSLOAD Alarm Messages ____ 262
35. Interrupt-Structure Summary ____ 268
36. Processor Availability in Sample Configuration ____ 268
37. Comparison of Secondary Storage Devices ____ 269
F-1. System Patching Diagnostics ____ 296
G-1. COC I/O Commands ____ 298
G-2. COC Parameters ____ 305
G-3. Line Control Table Items ____ 306
G-4. Input Translation Table ____ 308
G-5. Input Conversion ____ 308
G-6. Special Action Table ____ 309
G-7. Output Conversion (EBCDIC-USACII) ____ 312
O-1. Type Completion Codes ____ 322
O-2. Monitor Error and Abnormal Codes ____ 323
O-3. TYC, BUSY, and R10, Byte 0 Settings ____ 327

PREFACE

This manual is the principal source of reference information for the real-time and batch processing features of CP-R (i.e., job control commands, system procedures, I/O procedures, program loading and execution, hardware interrupt and software interface, and service processors). The purpose of the manual is to define the rules for using background processing, real-time features, hardware mapping, and virtual memory capabilities. Manuals offering other levels of information regarding CP-R features are outlined below.

- The Control Program for Real-Time (CP-R)/User's Guide, 90 30 87, describes how to use the various batch and real-time features that are basic to most installations. It presents the information in a semitutorial format that offers the user a job-oriented approach to learning the features of the operating system.

- The Control Program for Real-Time (CP-R/OPS Reference Manual, 90 30 86, is the principal source of reference information for CP-R computer operators. It defines the rules for operator communication with the system (i.e., key-ins and messages) system start-up and initialization, job and system control, peripheral device handling, and recovery procedures.

- The Control Program for Real-Time (CP-R)/Systems Technical Manual, 90 30 88, describes the internal functions of the operating system. It is intended for use with the listings supplied with each CP-R system for purposes of system maintenance.

- The Availability Features (CP-R) Reference Manual, 90 31 10, describes the availability features that will rapidly identify a system problem as to either hardware or software malfunction and then further define the problem via software diagnostic criteria.

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

COMMAND SYNTAX NOTATION

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

Notation	Description
lowercase letters	Lowercase letters identify an element that must be replaced with a user-selected value. CRndd could be entered as CRA03.
CAPITAL LETTERS	Capital letters must be entered as shown for input, and will be printed as shown in output. DPndd means "enter DP followed by the values for ndd".
[ ]	An element inside brackets is optional. Several elements placed one under the other inside a pair of brackets means that the user may select any one or none of those elements. [KEYM] means the term "KEYM" may be entered.
{ }	Elements placed one under the other inside a pair of braces identify a required choice. $\left\{\begin{matrix}A\\id\end{matrix}\right\}$ means that either the letter A or the value of id must be entered.
...	The horizontal ellipsis indicates that a previous bracketed element may be repeated, or that elements have been omitted. name[,name]... means that one or more name values may be entered, with a comma inserted between each name value.
⋮	The vertical ellipsis indicates that commands or instructions have been omitted. MASK2 DATA,2 X'1EF' ⋮ BYTE DATA,3 BA(L(59)) means that there are one or more statements omitted between the two DATA directives.
Numbers and special characters	Numbers that appear on the line (i.e., not subscripts), special symbols, and punctuation marks other than dotted lines, brackets, braces, and underlines appear as shown in output messages and must be entered as shown when input. (value) means that the proper value must be entered enclosed in parentheses; e.g., (234).
Subscripts	Subscripts indicate a first, second, etc., representation of a parameter that has a different value for each occurrence. $sysid_1, sysid_2, sysid_3$ means that three successive values for sysid should be entered, separated by commas.
Superscripts	Superscripts indicate shift keys to be used in combination with terminal keys. c is control shift, and s is case shift. L^{cs} means press the control and case shift (CONTROL and SHIFT) and the L key.
Underscore	All terminal output is underscored; terminal input is not. !RUN means that the exclamation point was sent to the terminal, but RUN was typed by the terminal user.
ESC RET LF	These symbols indicate that an ESC (ESC), carriage return (RET), or line feed (LF) character has been sent. !EDIT RET means that, after typing EDIT, a carriage return character has been sent.

GLOSSARY

active foreground program — a program is active if it is resident in memory, connected to interrupts, or in the process of being entered into the system via a IRUN control command.

asynchronous events — independent events that may be taking place concurrently to task execution (e.g., a read from magnetic tape or an STIMER service call).

background area — that area of core storage allocated to batch processing. This area may be checkpointed for use by foreground programs.

background program — any program executed under monitor control in the background job. These programs are entered through the batch processing input stream.

batch job — a job that is submitted to the batch job stream with an operator keyin or through an on-line terminal (using the BATCH command).

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

centrally connected interrupt — an interrupt that is connected to a monitor interrupt routine that first saves the environment of the system and then switches the environment to that of the task that gets control when the interrupt occurs.

checkpointed job — a partially processed background job that has been saved in secondary storage along with all registers and other "environment" so that the job can be restarted when a foreground task no longer requires the borrowed background memory and all currently active foreground activity has completed.

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 (see "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 executing program that contains the information used by the monitor in the performance of an I/O operation.

dedicated memory — core memory locations reserved by the monitor for special purposes such as traps, interrupts, and real-time programs.

directly connected interrupt — an interrupt which, when it occurs, causes control to go directly to the task; e.g., execution of the XPSD instruction in the interrupt location gives control to the task rather than first going to a monitor interrupt routine.

disk pack — a secondary storage system of removable rotating memory. For CP-R purposes, disk pack and RAD are synonymous unless otherwise noted.

dispatcher — a set of CP-R routines that schedule secondary tasks on a software priority basis. There may be more than one dispatcher in a given system.

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

end record — the last record to be loaded in an object module or load module.

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

execution location — a value defining the origin of a relocatable program, to set the address at which program loading is to begin.

external definition — a symbolic name that is declared to be "knowable" outside the range of the object module in which it is defined; a "global" symbol. An external definition allows the specified symbolic name to be used in external references (see below).

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

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

foreground area — that portion of memory dedicated specifically for foreground tasks and programs.

foreground program — a load module that contains one or more foreground tasks.

foreground task — a body of procedural code that is associated with an interrupt (primary task) or controlled by a CP-R dispatcher (secondary task).

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

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

granule a block of disk sectors containing a specified number of words.

idle state the state of the monitor when it is first loaded into core memory or after encountering a !FIN control command. The idle state is ended by means of a !C key-in.

installation control command any control command used during system generation to direct the creation of a CP-R system.

job a job is a collection of one or more related tasks. A foreground job is composed of one or more related foreground tasks. A background job is composed of one or more job steps each comprised of a single task. Tasks executed in the background are not connected to interrupts. Background job steps are processed through the batch input stream. The job concept is used for resources (such as files or devices) that are shared at the job level.

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

keyword a word, consisting of from one to eight characters that identifies a particular operand used in a control command or operator key-in.

load item in object modules, a load-control byte followed by any additional bytes of load information pertaining to the function specified by the control byte.

load map a listing of significant information pertaining to the storage locations used by a program.

load module an executable program formed by using relocatable object modules (ROMs) and/or library object modules as source information. Primary load modules contain one or more primary tasks; secondary load modules contain a single secondary task.

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

long wait when a secondary task performs an asynchronous service with wait, the task may specify that the WAIT is to be 'long-wait'. Long wait tasks are selected for roll-out before non long-wait tasks, overriding the strictly priority driven roll-out algorithm.

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

object deck a card deck comprising one or more object modules and control commands.

object language the standard binary language in which the output of a compiler or assembler is expressed.

object module the series of records containing the load information pertaining to a single program or subprogram. Object modules serve as input to the Overlay Loader to form the load module.

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

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

OV file is a temporary disk file that contains an executable program formed by the Overlay Loader if a program file name was not specified at load time. Used primarily to test new programs or new versions of programs.

Overlay Loader a processor that links elements of overlay programs (ROMs) and forms executable programs.

overlay program a segmented program in which the segment currently being executed may overlay the core storage area occupied by a previously executed segment.

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

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

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

primary reference an external reference that must be satisfied by a corresponding external definition (capable of causing loading from the system library).

primary task a body of code that is connected to a hardware interrupt and that gains control of the CPU when that interrupt becomes active.

Program Trap Conditions (PTC) two words that indicate trap status (set or reset) and trap exit address respectively.

Relocatable Object Module (ROM) a program, or subprogram in the form generated by a standard processor such as Macro-Symbol, FORTRAN, etc.

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

ROM Relocatable Object Module (see above).

secondary reference an external reference that may or may not be satisfied by a corresponding external definition (not capable of causing loading from the system library).

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

secondary task a body of code whose execution is scheduled by a CP-R dispatcher.

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

source deck a card deck comprising a complete program or subprogram in symbolic EBCDIC format.

source language a language used to prepare a source program (and therefrom a source deck) suitable for processing by an assembler or compiler.

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.

synchronous events events that must take place sequentially.

system library a group of standard routines in object language format, any of which may be included in a program being created.

task see "primary task" and "secondary task".

taskname identical to a load module name (lmn) unless redefined by the SETNAME CAL. All tasks within a load module have the same taskname.

temp stack push-down stacks created by the Overlay Loader and used by the monitor and system library routines.

time-sliced secondary tasks may be 'time-sliced' in which case CP-R will cause the task to relinquish control of the CPU after having executed for a defined period of time (a quantum of usually 100 milliseconds). After the expiration of a task's quantum the task is placed at the end of the CP-R dispatch queue in priority order. This in effect causes time-sliced tasks of equal priority to be selected for execution in a 'round-robin' fashion. The default priority for time-sliced tasks is X'FFFF'.

1. INTRODUCTION

OPERATING SYSTEM

The Xerox Control Program for Real-Time (CP-R) for Xerox 550 and Sigma 9 Computers is the major control element in an installation's operating system. Operating in a real-time environment, CP-R provides for concurrent background/foreground processing with emphasis on foreground operations.

The operating system consists of CP-R, language translators, service programs, batch (background) user's programs, terminal job entry system, and real-time (foreground) user's program. In general, CP-R governs the order in which the other programs are executed and provides services common to all of them.

The number, types, and version of programs in an operating system vary, depending upon the exact requirements at a particular installation. Each operating system should consist of a closely integrated set of service and control routines and processing programs specifically 'tailored' for a given range of applications.

As the requirements of an installation increase, the operating system can be enlarged, modified, or updated. The ability to adapt to new requirements is inherent in the system design. Once a system is generated, it can quickly be expanded to include users' programs, data, and system libraries.

A user's program and data may either be temporarily incorporated in the operating system or remain a part of the system for an extended period of time.

The operating system is self-contained and requires operator intervention only under exceptional conditions. Operating procedures are given in the CP-R/OPS Reference Manual, 90 30 86.

CP-R TERMS AND PROCESSES

The following items are either unique to the CP-R system or have specific meaning within the CP-R context. Terms and processes not defined below are fully explained in the appropriate chapter.

JOB

A job is a collection of one or more tasks which the user has specifically organized to share resources (e.g., operational label definitions, nonsharable peripherals, file table entries). A job may be initiated (SJOB) and terminated (KJOB) by system service calls or operator key-in. Each job in CP-R is associated with a user supplied account name.

TASK

A task is a body of procedural code that is capable of gaining control of the CPU. A primary task is one that is associated with a specific interrupt and gains control of the CPU when that interrupt becomes active. A secondary task gains control of the CPU when scheduled by a CP-R software dispatcher.

LOAD MODULE

A load module is a body of procedural code and data that is identifiable by name. A load module is created at link-edit time by the Overlay Loader (OLOAD) from object modules, and exists after the link-edit operation in memory-image form on disk storage. A load module is identified by name so that it may be loaded or terminated on request. Background load modules are loaded by control commands; foreground load modules can be loaded or terminated upon request through operator key-in, control command, or a system call from a foreground task.

Load modules when loaded into memory become tasks and retain the identification of the load module as their task-name. Primary modules may contain more than one primary task by virtue of multiple connections to interrupts. Secondary load modules always contain only a single task.

PROGRAM

For the purposes of this manual a program is synonymous with a load module.

FOREGROUND

The foreground is the set of all tasks in the system that are typically associated with real-time processes and are specifically not executing in the background job. The activation sequence of foreground tasks is controlled on a priority basis by the hardware-interrupt system and the CP-R software-scheduling services. Foreground tasks are guaranteed memory protection from background processes.

BACKGROUND

The background is a job in which noncritical tasks can execute so as to use any available CPU time after the real-time requirements are satisfied. In contrast to foreground tasks, background tasks are executed serially and their sequence is controlled by control commands. They

may be thought of as connected to the 'null' interrupt level, below that of all other tasks in the system.

TERMINAL JOB ENTRY SYSTEM

Terminal Job Entry (TJE) is a terminal support system that fulfills the user's need for local and remote job entry. A user may log-on, create files, edit files, submit batch jobs, build multi-task environments and debug the environment via the terminal. TJE is a real-time system that allows the user to create any foreground structure required.

TEMP STACKS

Temp stacks are sets of memory locations allocated by the Overlay Loader; one of which is used for dynamic temporary storage for system functions and the other when FORTRAN Library subroutines are called. The latter set of memory locations is also available as temporary storage for the user. The temp stacks are "push down" stacks, that is, they operate in a last-in/first-out fashion, and are allocated to a load module.

DATA CONTROL BLOCK

A DCB is a table located in the calling program that contains information used by CP-R in the performance of an I/O operation. DCBs are the means by which I/O information is communicated between a user's program and the system. The information required for a particular I/O operation is either contained in the associated DCB or is given in a call. The specific information needed for an I/O operation depends on the organization of the data involved and the type of operation to be performed.

The device used for I/O operation is determined by the contents of the associated DCB when the I/O operation is requested by the executing program.

There are both system DCBs and user-created DCBs. The system DCBs need only be coded as external references in a system processor or user program; the Overlay Loader will satisfy these external references at link time by furnishing a copy of the appropriate DCBs in the program's root. If a user is not satisfied with the standard DCB parameters furnished by the Overlay Loader, the system DCBs can be coded into the user program's root and the DCB name declared as an external definition.

FUNCTION PARAMETER TABLE

An FPT is a table through which a program communicates with a system service function.

TASK CONTROL BLOCK

A task control block (TCB) is a table containing task-associated parameters. The space for this table is allocated by the system for secondary tasks and by the user for primary tasks. The entries in this table are used and maintained by CP-R.

PROGRAM CONTROL BLOCK

The PCB is a table containing program-associated parameters. The PCB is constructed by the Overlay Loader at link time.

ROLL-OUT/ROLL-IN

Roll-out is the process by which the memory resources acquired by a low-priority task (foreground or background) is made temporarily available for use by a higher-priority task. This is accomplished by saving an 'in-memory' image of the task and its context on secondary storage while its memory space is required for higher-priority task.

Roll-in is the process by which a rolled-out task is restored to memory. This is accomplished by bringing the rolled-out task image back into memory, on a priority basis, whenever sufficient memory resources are available.

REENTRANT SUBROUTINE

A reentrant subroutine can be called by several different tasks. During execution of such a subroutine, a higher priority task can interrupt and call the same subroutine. When the higher priority task has completed execution, control is returned to the subroutine at the interrupted point. Since a reentrant subroutine does not perform any instruction modification and uses the temp stack or a virtual-memory nonshared context segment for scratch storage, processing continues as though the subroutine had never been reentered.

PHILOSOPHY OF OPERATION

The system provides for two levels of operation:

1. Real-time, foreground processing.
2. Batch, background processing.

REAL-TIME PROCESSING

Real-time processing, the most critical aspect of multi-usage, involves reacting to external events (including clock pulses) typically within microseconds.

Real-time programs can be either automatically loaded every time the system is booted from secondary storage or loaded and initiated as needed. The first method is used when the real-time process normally remains unchanged and is constantly operative. The second approach is used when real-time operations are executed periodically or irregularly, as in an experimental laboratory.

A real-time process is assigned machine facilities on a dedicated basis at initiation time. These facilities include disk and memory residency, I/O channels, peripheral devices, and external interrupt lines. Such allocation remains in force until either the process or the computer operator terminates the program.

During SYSGEN, a user can reserve a portion of his foreground area for communication between primary real-time programs. Locations in this area are called foreground mailboxes. The start of this area can be referenced through the system label FP:MBOX. Upon encountering an external reference to FP:MBOX, the Overlay Loader will satisfy standard symbolic references to a mailbox area.

The system provides foreground programs with the facility for direct I/O operations (called IOEX operations), wherein the user furnishes the basic hardware commands and does the necessary error checking and recovery. This type of I/O operation provides decreased overhead and greater flexibility as compared to indirect I/O operations.

Foreground programs can be loaded for execution from a background job stack by operator key-in, by terminal job entry commands, or through a system call by a foreground program, providing the program to be loaded is already on secondary storage in load-module format. Foreground programs are responsible for initializing the interrupt system and connecting tasks to interrupts. Foreground tasks can be processed compatibly and concurrently with a background production job stack.

BATCH PROCESSING

The system is capable of processing a continuous series of background jobs with little operator intervention. Reducing the need for operator participation ensures faster throughput, and makes operations less subject to error.

CP-R FUNCTIONS

CP-R controls and coordinates batch (background) and real-time (foreground) processing. Efficient operation is assured by minimizing system overhead in response to an interrupt, and by preserving the relative priority of the tasks.

Reentrant service functions perform I/O and control the interrupt system; other service functions load and initialize foreground tasks.

Some portions of CP-R are resident to ensure continuous operation. Other portions of CP-R are brought into main memory from secondary storage as required to perform specific functions. These portions are structured as overlays, thus minimizing main memory requirements. In addition, processing programs are retrieved from secondary storage and they too can capitalize on overlay techniques to minimize main memory requirements.

Secondary scratch storage for service programs, processors, and user programs is provided, and secondary storage also accommodates permanent user files. (Permanent user files on secondary storage are acquired through a processor called RADEDIT which provides media conversion, listing of files, and other services in addition to file allocation.)

SECONDARY STORAGE UTILIZATION

A disk storage unit is essential for operation of the system. It minimizes main memory requirements by storing all system, processor, and user-program overlays, yet provides very fast access whenever an overlay is called into memory.

DISK AREAS

During system generation, secondary storage is divided into large blocks called disk areas. These areas represent functional groupings of disk files. All files within a given area have the same software write protection. Since the entire CP-R system is disk oriented, every job will directly or indirectly involve the use, modification, allocation, or release of disk files. Listed below are the standard disk areas and the types of files that normally occupy them.

- System Program (SP) area contains CP-R, the Account Inventory file, the Scheduler file, and the set of language translators used by the local installation, such as AP and FORTRAN. The area also contains the System Library (i.e., FORTRAN Library/Run-time), and RADEDIT and Overlay Loader service processors. All translators and service processors are called by user to execute in the background.

- Foreground Program (FP) area contains a collection of foreground programs and an optional user library and Public Libraries. User-library routines are included in the user's load module at "link time". Public Libraries are groups of run-time, reentrant routines shared by a number of programs.

- Background Program (BP) area contains the set of permanent user programs that execute in the background.

- User areas (any two-character name) allow flexible subdivision of the disk space to aid the individual installation in controlling disk storage use. User disk areas provide both use control and file name uniqueness for the files within them. Use control is determined by the area protection type. This may be public, background, foreground, system, or IOEX. Public and background areas may be read or altered by either background or foreground users. Additionally, public areas provide a pool of space for area-independent file activity. Normally, foreground areas may be altered only by foreground users, and system areas may not be altered. However, both types may be read by any user. IOEX areas may be used only by IOEX (Direct Device Control) services. Files may not be defined in them. File name uniqueness is determined by the combination of file name,

disk file account name, and disk area name. This allows several files to be defined with the same name and account, as long as they are in different disk areas. (The data areas, D1 through DF, are considered to be user areas.)

- Checkpoint (CK) area is used by CP-R for roll-out/roll-in file storage and to save an image of memory after a system alarm condition.

- Input Symbiont (IS) and Output Symbiont (OS) areas contain files that are defined and maintained by symbiont routines. Files in the IS and OS areas can not be accessed by users.

- IOEX Access (XA) area contains no directory and hence no files. This area can be written only by IOEX and should normally be the only area of the disk that IOEX is allowed to access.

- Background Temp (BT) area contains temporary (scratch) files (X1 through Xn where n is a SYSGEN parameter) used by background programs for intermediate storage in processing. Their use is identical to scratch tapes on magnetic tape units. Temp files are automatically destroyed when a new !JOB command is encountered in a job stack, and thus there is no way to save data on temp files from one job to another. They may, however, be saved across job steps.

  The GO and OV files are special temporary files in the BT area. The GO file receives relocatable object modules (ROMs) formed by a language processor if the GO option was specified or by default. It is used by FORTRAN IV and AP programs. The OV file receives executable programs formed by the Overlay Loader, and is used primarily to test a new program that has no permanent file defined, or to test a new version of a program without destroying the current version. A program in the OV file is called for execution via an !ROV control command.

  Note that both GO and OV are used for communication between job steps but not jobs. There is no monitor protection for these files between one job and another.

JOB ACCOUNTING

Background job accounting is an option selected at SYSGEN. To correctly calculate the elapsed time for a background job, all primary foreground tasks must be centrally connected (see "Connecting Real-Time Tasks to Interrupts" in Chapter 4). Otherwise, the foreground task's execution time will be included in the elapsed time of the background job.

At the beginning of a background job, the date and start time (in hours and minutes) will be logged on the LL device. At the end of a background job, the total CPU time of the job (in hours, minutes, and seconds) will also be logged on LL. This information plus the account number and user name is then written on the "AL" file in the Background Data area of the disk. The "AL" file must be defined via the RADEDIT; it must be in the D1 area of the disk, and allocated a minimum size of 256 words. This file can be purged periodically by the operator.

The Account Inventory (AI) file located in the SP area is a blocked file containing 80-byte records which represent the authorized accounts and user names in a given system. The account and user name records in the file must be arranged alphabetically with account records preceding user name records. An account record contains the account name (1 to 8 EBCDIC characters) beginning in column 1. The user name record contains the account name beginning in column 1 followed by a comma followed by the name field (1 to 12 EBCDIC characters). One or more user name records may follow an account record. Both account and user name records must have EDIT line numbers in columns 73-80.

For example:

```
Column
1 2 3 4 5 6 7 8 9 10 11 12 13                    73 74 75 76 77 78 79 80
A 9 5 2 . . . . . . . . . . . . . . . . . . . . 1 . 0 0 0
A 9 5 2 , C 3 2 1 0 5 . . . . . . . . . . . . . 2 . 0 0 0
A 9 5 2 , N 5 6 . . . . . . . . . . . . . . . . 3 . 0 0 0
K 1 7 3 0 2 1 . . . . . . . . . . . . . . . . . 4 . 0 0 0
K 1 7 3 0 2 1 , A B 3 2 . . . . . . . . . . . . 4 . 1 0 0
K 1 7 3 0 2 1 , X 9 . . . . . . . . . . . . . . 5 . 0 0 0
K 1 7 3 0 2 1 , Z Z Z . . . . . . . . . . . . . 1 0 . 0 0 0
```

The AI file is used to validate the TJE user's log-on information. It is also used to validate the account and user name supplied on a background JOB card.

PUBLIC LIBRARY

If a CP-R system has several programs that share a group of subroutines, this set of subroutines can be collected in main memory in a "Public Library". This preselected set is loaded by the Overlay Loader into a previously defined file on the disk. The Loader also writes the names and entry points of all the routines into the same disk file. Then, whenever the Overlay Loader loads a foreground or background program that references one of the "public" routines, it links the appropriate branch to the Public Library copy instead of loading a separate copy. This can represent a considerable saving in space for a large system. An installation may define as many Public Libraries as needed.

If the appropriate Public Library is not presently resident it will be automatically loaded into its specified location whenever a program is loaded that uses it. If the Public Library is for secondary (mapped) tasks, it may include a common-virtual context segment that is loaded into different real space for each task using it, i.e., a nonshared context segment.

SYMBIONTS

Symbiont routines transfer data from the card reader to disk storage and from disk storage to the line printer. Input cooperatives intercept card read commands in user programs and transfer data from disk storage where it has been stored by symbiont routines. Output cooperative routines intercept output directed from user programs to a line printer and transfer the data to disk storage. Symbiont service is available only to background jobs.

CONTROL TASK

The CP-R Control Task performs the following functions:

1. "I/O cleanup" and "I/O start" when any of these functions are deferred from the I/O interrupt task because of priority considerations.
2. Loading, initialization and release of primary foreground tasks.
3. Sequencing of background programs.
4. Console interrupt and operator key-in processing.
5. Dumps for the background and operator.
6. Error Log filing.
7. Scheduling for periodic initialization of primary and secondary foreground tasks.

The CP-R Control Task is connected to the lowest priority dispatcher in the system at system generation time.

OVERLAYS

A large portion of the control program is overlayed to minimize main memory residence requirements. The overlays may be selectively SYSGENed as resident in order to enhance system performance on an installation-specific basis.

MEMORY PROTECTION

CP-R provides memory protection for all input operations except direct input, and provides write protection for disk files. The hardware write-lock and access-control features furnish memory protection for all non-I/O operations.

CP-R provides two levels of memory protection against faulty secondary (mapped) tasks, and one level of memory protection against faulty primary (unmapped) tasks:

- Memory accesses by secondary tasks are controlled both by means of the hardware write-lock feature and the access-protection feature associated with the memory map.
- Memory accesses by primary tasks are controlled by the write-lock feature.

For the purposes of write-lock protection, real memory is divided – at SYSGEN – into a variable number of partitions; each partition falls into one of four classes of memory, as follows:

- System memory: used exclusively by CP-R and given a write lock of 11. (One partition only.)
- Foreground private memory: used exclusively by primary tasks and given a write lock of 10.
- Foreground preferred memory: used by both primary tasks and secondary tasks, and given a write lock of 00 (i.e., unlocked).
- Secondary task memory: used by secondary tasks only, both foreground and background, and given a write lock of 01.

The several kinds of tasks that are possible under CP-R are given a write key based on the classes of memory that they are allowed to access (with respect to write-lock protection only):

- Control-program services execute with a "skeleton" write key of 00, and thus can modify memory of any class.
- Primary tasks execute with a write key of 10, and can modify private and foreground-preferred memory only.
- Secondary tasks, both foreground and background, execute with a write key of 01 and can modify foreground-preferred and secondary-task memory only.

For secondary tasks, which always execute in mapped mode, the hardware memory map affords an additional level of protection. The map effectively prohibits such a task from referencing any address outside its assigned virtual range and therefore completely protects all real memory not allocated to it. Further, through the assignment of access-control codes (read only, read/execute, read/write/execute to each of the virtual pages assigned to a task, the system automatically prevents undebugged programs from accidentally executing invalid types of access, e.g., from executing in a data-only section, or from writing into a procedure-only section. Memory-access control codes are assigned jointly by the user and the system.

Background tasks causing protection violations are aborted. Foreground programs are aborted unless they have requested trap control (see TRAPS system call).

Memory protection on all input operations performed via service functions (except IOEX) is guaranteed by the software. The system checks the validity of the input area on all read operations to ensure that the area is wholly contained in the calling program's write-lock area. If an attempt is made to read into an invalid area of main memory, an error condition is returned to the error address specified by the user. If no address is specified, the job is aborted. An "FG" key-in is required before a foreground program can be loaded from the background job stack; this protects the foreground from an error in a background job stack.

DISK WRITE PROTECTION

CP-R furnishes software write-protection of disk files in addition to that furnished by the hardware write-protect switches. Normally, background programs are allowed to write only in the Background Data area or Background Temp areas of the disk. Foreground programs are allowed to write into the Foreground and Background Data areas. The foreground user is responsible for ensuring that IOEX (direct I/O) writes only in the IOEX-Access area of the disk. The System Program area, Foreground program area, and Background Program area can be written into only if the "SY" key-in is in effect for the job in which the writing task is defined.

Similarly, the Overlay Loader and RADEDIT (background processors) are allowed to write in nonbackground areas only if the "SY" key-in is in effect for the background job. Any disk write-protection violation will result in a write-protection error indication return, and the write order will not be carried out.

All disk files can be read by a user without restriction. There is no system-furnished read protection for disk files.

LANGUAGE PROCESSORS

The following language translators are available for inclusion in the operating system:

Extended FORTRAN IV

SL-1

Assembly Program (AP)

Extended FORTRAN IV is a three-pass compiler that is a superset of ANS FORTRAN, incorporating many features not found in other FORTRANs.

SL-1 is a simulation language to solve differential equations as the fundamental procedure in simulating parallel, continuous systems. An extensive set of macros permit the user to simulate wide variety of linear and nonlinear elements through the use of single-operator statements. These prototype statements are inserted into the user program each time a macro is referenced by name.

Assembly Program (AP) is a three-pass assembly processor with a fourth pass that generates and prints a cross reference listing. The first pass accepts input written in symbolic format, compressed format, or compressed format with symbolic corrections or updates. Pass 1 then produces an encoded program used by the other passes for further processing, depending on the options specified on the !AP control command. The first pass is required for all AP runs; the other three passes are optional. The final assembly pass (Pass 3) outputs the program in standard object language.

SERVICE PROCESSORS

Service processors provide routines for performing frequently used functions. The service programs include the Overlay Loader, RADEDIT, and Edit.

OVERLAY LOADER

The Overlay Loader (a background processor) is used to create overlay-program load modules for later execution in either the foreground or background. Thus, if a foreground program can tolerate a slight delay in reading overlay segments into memory for execution, either foreground or background programs of virtually unlimited size can be constructed even though memory size is restricted. For example, a 1400-word overlay can be input in about 50 milliseconds, using a Model 7204 Disk; that is, the time required to bring in an overlay for execution is the time of the one disk access required to read the overlay. Since a load module is stored on disk in "memory image form", it can be loaded very quickly as one logical record per segment. A load module formed by the Overlay Loader can be entered permanently into the System Program area, Foreground Program area, or Background Program Area, or it can be loaded on a temporary file in the Background Temp area of the disk.

An overlay structure, as illustrated in Figure 1, consists of a permanently-resident root segment and any number of overlay segments. A blank COMMON and labeled COMMON data area can be established for use by the root and overlay segments. Each segment is created by the Loader from one or more object modules output by a language processor. The Loader will build the Program Control Block, the OVLOAD table (used to load the overlay segments at execution time), allocate or build DCBs, and allocate the temp stacks. It will also load library modules to satisfy unsatisfied references encountered in the loading process. A maximum of two libraries can be searched. Library search and loading are extremely fast, due to special tables that are added to the library files at the time the library is created on disk.

The overlay segments must be explicitly defined at link-edit time and explicitly called at execution time. All segments in a path may communicate with each other via REF/DEF linkages, but it is the user's responsibility to ensure that any segment referenced is currently in memory.

RADEDIT

RADEDIT controls disk allocation for areas containing permanent disk files and performs utility functions for all areas.

RADEDIT performs the following functions:

1. Adds or deletes entries in the permanent file directories.
2. Compacts disk areas by relocating files and updating/compacting directories to regain space within an area.
3. Maps permanent disk file allocations.

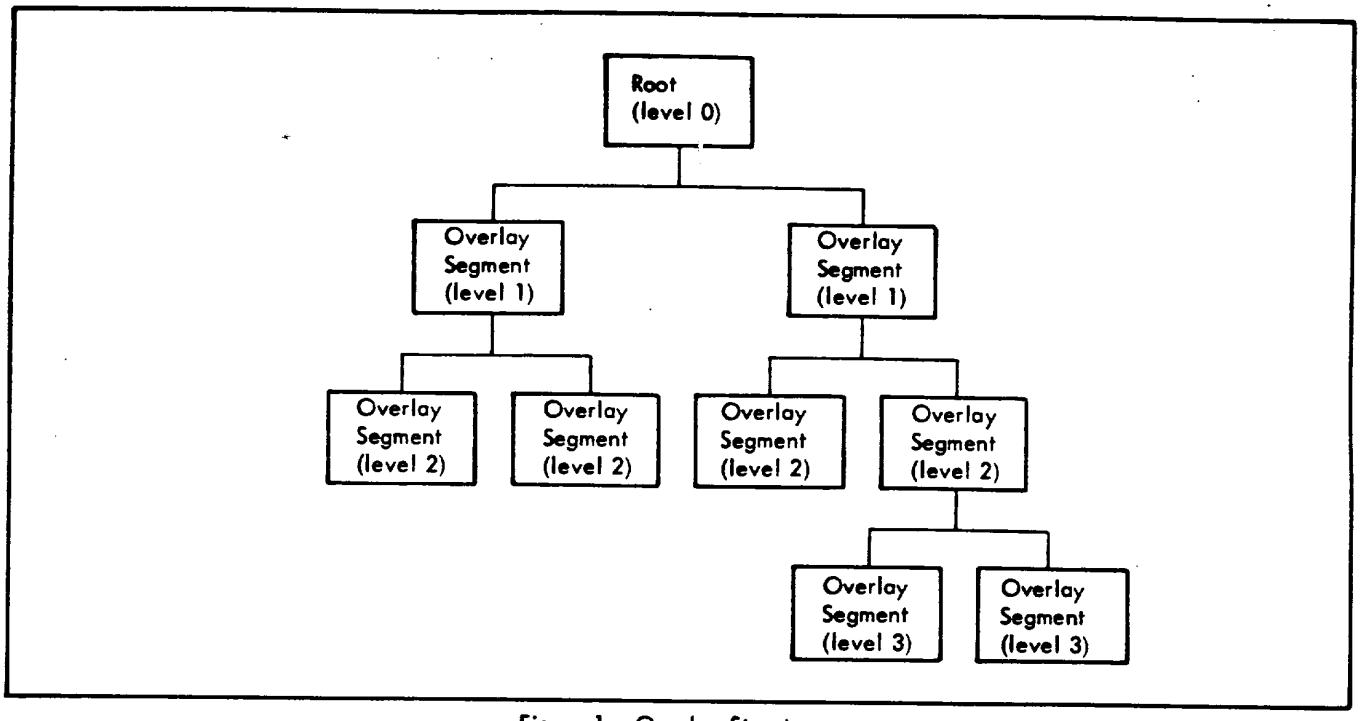


Figure 1. Overlay Structure

4. Builds and maintains library files on the disk for use by the Overlay Loader.
5. Copies permanent disk files.
6. Saves the contents of disk areas in self-reloadable form.
7. Restores disk areas previously saved.
8. Dumps the contents of permanent disk files or areas.

When used from a TJE job, not all of these functions are available. (See Chapter 14.)

EDIT

The Edit processor is a line-at-a-time text editor designed for on-line or batch 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 which is built and maintained only by the Edit processor.

Between editing sessions, the data may be saved on standard CP-R files of blocked, unblocked, or compressed organization and can be accessed by any other program or processor.

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 (see Chapter 13).

BACKGROUND JOB ORGANIZATION

The user controls the execution of a background job by means of control commands. These control commands are interpreted by either the Job Control Processor (JCP), Overlay Loader, or RADEDIT; they specify

- Processors required and the options to be used.
- Input/output devices required and their specific assignments.
- Loading and execution requirements.
- Libraries and supporting services required.
- Program modification and debugging requirements.

A batch job is a sequence of control commands that may be executed independently. Each such background job is independent of any other job and consists of one or more directly or indirectly related job steps. A job step results in the execution of a processing program such as a language translator, service program, or user's program.

FOREGROUND JOB ORGANIZATION

The user may define a set of one or more related foreground tasks as a job. Such a job may specify that certain peripheral devices or files are private for its use, or the job may gain exclusive control of a device on a temporary basis. All tasks within a foreground job may access the device defined as private.

INPUT/OUTPUT SPECIFICATIONS

Throughout the CP-R system, specifications of input data sources and output data destinations are made by giving a physical device name (see Tables 1-3), a disk file identifier (see Table 4), or an operational label (see Table 5).

Table 1. I/O Device Type Codes

yy	Device Type
TY	Typewriter.
LP	Line printer.
CR	Card reader.
CP	Card punch.
9T	9-track magnetic tape.
7T	7-track magnetic tape.
DC	Fixed-head disk.
DP	Disk pack unit.
PL	Plotter.
NO	Not a standard device. Used as a special purpose device for IOEX.
LD	Logical device.
LN†	Remote Terminal.

†In this case the device name will be LNxxx where xxx represents the decimal line number.

Table 2. Channel or Cluster/Unit Designation Codes

Specified Character (n)	Significance		
	Sigma 9 Unit Address	Xerox 550	
		Cluster Address	Unit Address
A	0	0	0
B	1	1	0
C	2	1	1
D	3	1	2
E	4	1	3
F	5	1	4
G	6	1	5
H	7	2	0
I	-	2	1
J	-	2	2
K	-	2	3
L	-	2	4
M	-	2	5

Table 2. Channel or Cluster/Unit Designation Codes (cont.)

Specified Character (n)	Significance		
	Sigma 9 Unit Address	Xerox 550	
		Cluster Address	Unit Address
N	-	3	0
O	-	3	1
P	-	3	2
Q	-	3	3
R	-	3	4
S	-	3	5
T	-	4	0
U	-	4	1
V	-	4	2
W	-	4	3
X	-	4	4
Y	-	4	5
Z	-	5	0
0	-	5	1
1	-	5	2
2	-	5	3
3	-	5	4
4	-	5	5
5	-	6	0
6	-	6	1
7	-	6	2
8	-	6	3
9	-	6	4
blank	-	6	5
$	-	0	1
#	-	0	2
@	-	0	3
:	-	0	4

Table 3. Device Designation Codes

Hexadecimal Code (dd)	Device Designation
00≤dd≤7F (single devices)	Refers to a device number (00 through 7F).
80≤dd≤FF (multiple devices)	Refers to a device controller number (8 through F) followed by a device number (0 through F).

Table 4. Disk Area Codes

Code	Area
SP	System Program area.
FP	Foreground Program area.
BP	Background Program area.
BT	Background Temp area.
XA	IOEX Access area.
CK	Checkpoint area.
D1 : DA : DF	Data area (number of data areas is defined at SYSGEN).

PHYSICAL DEVICE NAMES

Physical device names are of the form

$$\left[yy\left[\begin{matrix}ndd\\xxx\end{matrix}\right]\right]^{\dagger}$$
$$0$$

where

yy identifies the type of device (see Table 1).

n identifies the channel or cluster/unit number (see Table 2).

dd identifies the device number (see Table 3).

---

†See Command Syntax Notation for significance of brackets.

Table 5. System Operational Labels

Label	Standard or Optional	Reference	Comments
BI	Standard	Binary input	Used to input binary information.
BO	Standard	Binary output	Used by processors to output binary information.
C	Standard	Control command input	Used by the system and processors to read control commands.
CI	Standard	Compressed input	Used by AP.
CO	Standard	Compressed output	Used by AP.
DI	Optional	Debug input	Used by the Debug facility.
DL	Optional	Debug listing	Used by the Debug facility.
DO	Standard	Diagnostic output	Used by the system for postmortem dump and diagnostic messages and by AP for diagnostic messages.
DP	Optional	Debug patch	Used by the Debug facility.
ER	Optional	Error logging	Used by Error Logger.
LL	Standard	Listing log	Used by the system and AP to log control commands and other system messages.

Table 5. System Operational Labels (cont.)

Label	Standard or Optional	Reference	Comments
LO	Standard	Listing output	Used for object listings from assemblies and compilations.
MO	Optional	Media output	Used by Media facility.
OC	Standard	Operator's console	Used by the system for key-ins and operator control. (Always assigned to a keyboard/printer.)
P1	Optional	Debug utility 1	Used by the Debug facility.
P2	Optional	Debug utility 2	Used by the Debug facility.
SE	Optional	Save environment	Used by Alarm Saver.
SI	Standard	Symbolic input	Used by processors to input source (symbolic information).
SO	Standard	Symbolic output	Used by SL-1 and AP.

xxx identifies the line number of a communication line.

and

0 identifies the null device.

The null device does not exist as such, but may be used whenever a device of the form yyndd is valid. Input and output requests to the null device cause a normal type completion and zero actual record size to be posted. No data is moved to or from the I/O buffer.

Note that a physical device name may be the name of a logical device (LDndd).

DISK FILE IDENTIFIERS

Disk file identifiers have two basic forms:

1. area,name

where

area is the name of any disk area defined on the system.

name is the name of the file.

This form of file identifier may occur only in a parenthesized group, for example, as in the background command:

LOAD (OUT,SP,RADEDIT)

or the keyin:

BATCH (D1,JOBFILE)

Files in accounts other than the system account may not be identified in this form.

2. [name][.[area]][.[account]]

where

name is the name of the file. If omitted, the area must be specified, and the whole area is the identified medium.

area is the name of the disk area in which the file resides, and may be any area defined on the system. If omitted, the file is assumed to reside in a public area and the area will be determined by the system.

account is the name of the account in which the file is defined. Account names must have at least one character, and at most, eight.

The account name may be omitted, in which case, a default is provided. If the area name is also omitted, the account name defaults to that in which the task which opens the file is running. If the area name is specified, the account name defaults to the system account.

Files of the same name may be defined as long as they differ in either account name or area name, however, if files of the same name and account are defined in different public areas, one of them will be inaccessible except by explicit specification of its area name, since by default, the one found first is accessed.

The Background Temp area (BT), is a special area in that its file names are restricted to the set, GO, OV, and X1 through X9 and account name is not required. Its primary purpose is for work files for the background processors. Its files are temporary and will be discarded at the beginning of the next job.

Files in other areas are permanent files and will be kept by the system until they are explicitly deleted.

OPERATIONAL LABELS

Operational labels (oplabels) are names used to symbolically reference physical devices and disk files. All oplabels are defined at SYSGEN (see Chapter 15) and are given their permanent default, or system initialization assignments. System oplabels, as listed in Table 5, are treated specially:

First, they are created automatically during SYSGEN. The Standard System Operational Labels are always created in all SYSGENS. The optional system operational labels are used by optional features in the CP-R operating system and are created only if their associated feature is included in the system. Any user desired oplabel must be defined in SYSGEN. Default oplabel assignments are also made in SYSGEN.

Second, they are used by the CP-R system and its processors as the standard assignments for standard DCBs. For example, processors producing binary output do so to the M:BO DCB, and unless explicitly changed, the DCB is assigned to the BO oplabel. Where the output actually goes is a function of the assignment of the BO oplabel.

Oplabels are considered job level resources. That is, each job within the system has its own separate set of oplabel assignments. The oplabel assignments for the CP-R job are used as the default assignments for other jobs. When a new job is activated, this set of default assignments becomes the initial assignments in the new job. Any reassignments in that new job affect only that job. The assignments are retained until changed or the job terminates.

The default assignments in the CP-R job may be changed (see STDLB key-in). The resultant set of assignments becomes the default set for any subsequently activated jobs.

RESOLVING I/O MEDIUM NAME AMBIGUITIES

When a command permits more than one I/O medium type to be named in the same field, certain ambiguities may arise as to which type of name is intended; for instance, '9TA80' could be either a file name or a device name. The following rules resolve this type of ambiguity:

1. Names preceded or followed by periods are parts of file identifiers. This means '.SI' is not an operational label, and '9TA80.' is not a device.

2. Names followed but not preceded by periods are file names, for example, '9TA80.' is a file name, not a device. 'SI.' is also a file name, not a disk area name or an operational label.

3. A name is a disk area name if it follows a period, it does not follow another area name in the same file identifier, it has two characters, and it is defined as an area name in the system.

4. A name is a disk file account name if it follows a period and is not selected by rule 3.

5. A name is an operational label if and only if it has two characters, it is defined as an operational label in the system, and it is not excluded by rule 1.

6. A name is a device name if and only if it has five characters, it is defined as a device in the system, and it is not excluded by rule 1.

7. Names which are not typed by rules 1-6 are tested first as area names in file ID format 1, then as file names (with unspecified area and account) in file ID format 2.

2. CONTROL COMMANDS

Background processing is controlled and directed by means of control commands. These commands effect the construction and execution of programs and provide communication between a program and its environment. The environment includes CP-R and the assembly language processors, compiler language processors, service processors, the operator, and the peripheral equipment.

Control commands have the general form

```
/ Imnemonic specification
```

where

I is the first character of the record and identifies the beginning of a control message.

mnemonic is the mnemonic code name of a control function or the name of a processor. The name, consisting of up to 8 characters, may begin any number of spaces after the I character, except for an EOD command.

specification is a listing of required or optional specifications. This may include keywords, consisting of up to 8 characters, labels, and numeric values appropriate to the specific command.

For illustrative purposes within this manual only, the options of the specification field may be shown enclosed by brackets, braces, and/or parentheses. In actuality, only parentheses may be used. Parentheses are mandatory only when an option has two or more subfields (see Command Syntax Notation).

One or more blanks must separate the mnemonic and specification fields, but no blanks can be embedded within a field. A control command is terminated by the first blank after the specification field, or, if the specification field is absent and a comment follows the command, the command is terminated by a period after the blank that follows the mnemonic field. Annotational comments detailing the specific purpose of a command may be written following the command terminator, but no control command record can contain more than 80 characters.

A control command can be continued from one record to the next by using a semicolon to replace the comma as a subfield terminator in the specification field of the command. Column one of the continuation card must contain either an exclamation mark (if the control command is to be read by the Job Control Processor), or a colon (if the command is to be read by the Overlay Loader or RADEDIT). See the control command examples given later in this chapter for an illustration of the proper use of the semicolon.

Control commands are usually input to the monitor via punched cards; however, any input device(s) may be designated for this function. All control commands are listed on the output device designated as the listing log (normally a line printer). In this manner, the monitor keeps a record of the progress of the background. When a job is aborted, all control commands skipped over until the next JOB command is encountered are listed on LL with a "greater than" character (>) in column one.

Note that in all control commands, excluding processor control commands, the first three characters after the I character are sufficient to define any mnemonic code or keyword.

Control commands may be categorized as follows:

System Control	Debug Control
JOB	PMD
ASSIGN	
LOAD	Utility Control
ATTEND	
MESSAGE	PFIL
PAUSE	PREC
CC	SFIL
LIMIT	REWIND
STDLB	UNLOAD
ROV	WEOF
RUN	DAL
ALLOBT	
MODIFY (special)	Processor Control
LIST	
INIT	Processor name
SJOB	
BATCH	
Input Control	
EOD	
FIN	

JOB CONTROL PROCESSOR

All control commands are read from the "C" (oplabel) device by the Job Control Processor (JCP). The JCP is a special processor loaded into the background by CP-R upon the initial "C" key-in. The JCP is also reloaded into the background following each job step within a job. A job step is defined as all control commands required for the set-up and execution of a single processor or user program within a job stack.

The JCP processes each control command until a request is made to execute a processor or user program, at which time the appropriate program is read into the background and

given control. A detailed description of the JCP interface with the system processors or user programs is given later in this chapter under "Processor Control Commands".

SYSTEM CONTROL COMMANDS

JOB Each background job to be processed by the system must begin with a JOB control command. The JOB command signals the completion of the previous job, if any, and the beginning of a new one. The JOB command causes the temporary assignments of all operational labels (except the "C" operational label) to be reset to their permanent assignments.

The form of the JOB command is

```
!JOB[account number,name[,priority]]
```

where

account number identifies the account or project. It consists of from 1 to 8 alphanumeric characters.

name identifies the user. It consists of from 1 to 12 characters.

priority specifies the priority of the job and determines the order in which jobs are run in a symbiont system. Legal values are:

0 hold in job queue until priority is changed by a PRIORITY key-in.

1-7 lowest to highest priority. The default value is 1.

Note that commas separate the optional subfields. The account number must precede the name, and both fields must be present if either one is present. If the priority is present, the number and name fields must be present. The account and name information are verified against the information in the AI file and if not valid the job is aborted.

Example:

```
!JOB 12345,JOBSAMP1
```

The above example defines the account number for the job as 12345, and the user as JOBSAMP1. The job will have priority 1, by default.

ASSIGN The ASSIGN control command allows the changing of the default assignment of a Data Control Block (DCB) in the next background program or processor to be loaded. ASSIGN commands must appear prior to the RUN or Processor name command to which it will apply. They are effective for that job step only.

The assignment can be to a physical device, a disk file, or an operational label. If an error is detected in the command, the entire command is rejected and must be input again.

The form of the ASSIGN control command is

```
!ASSIGN (dcb,{fid|device|oplabel})[,(option),...]
```

where

dcb is the name (not exceeding eight characters in length) of the DCB to be assigned. It must be the first subfield following ASSIGN and must be followed by a specification (see below). The first two characters of a user's DCB must be "F:" (e.g., F:PRINT or F:BI). The first two characters of a system DCB name are "M:" (see Table 12).

fid is the CP-R file identifier for a file to which the DCB is to be assigned. File identifier format is defined in Chapter 1.

device specifies a physical device name or a numeric zero, "0", the null device.

oplabel specifies one of the operational labels defined during SYSGEN (see Table 5).

The options below are used only if the user creates the DCB or changes some of the DCB's parameters. Note that DCB parameters not specified on the ASSIGN command are not changed from their initial value. The initial values of the DCB parameters depend upon how the DCB was created.

Parameters of system DCBs have standard default values. DCBs allocated by the Overlay Loader (F:DCBs) are set to all zeros. User created DCBs have the initial values specified by the user.

Mode (may be any or all of the following):

{BCD} specifies the EBCDIC or automatic device mode.

{BIN} specifies the binary device mode.

{VFC} specifies vertical format control.

{NOVFC} specifies no vertical format control.

{PACK} specifies that the packed binary or unpacked binary mode is to be used for 7-track magnetic tape. PACK and UNPACK are not valid unless BIN is specified.

{UNPACK}

[D1600 / D800]	specifies density for tape drives with program-controllable density (1600 bpi or 800 bpi).
[ASCII / EBCDIC]	specifies tape data code for tape drives with program-controllable data encoding.
[DRC / NODRC]	specifies whether transfers with a keyboard-printer are to be in direct record control mode (no editing) or not.

Number of recovery tries

TRIES,value specifies the maximum number of recovery tries to be attempted for an I/O operation. The value must be less than 256.

Default record length

RECL,value specifies the default record length in bytes. The value n must be $1 \leq n \leq 32,767$. This record length is used for all requests referencing the DCBs that do not explicitly specify a record length in the FPT.

Byte displacement of first byte in buffer

BTD,n specifies the byte displacement of the first byte of the buffer. The value n is subject to the limits $0 \leq n \leq 3$. This value is the byte at which the buffer begins within the word specified as the buffer word address. This byte displacement is used for all requests referencing the DCB which do not have BTD parameters in their FPTS.

Examples:

1. Assign listable output to a magnetic tape:

```
!ASSIGN (M:LO,9TA81),VFC
```

This example assigns the M:LO DCB to a 9-track magnetic tape. Vertical format control is also specified, so the first byte in each record is a format control byte for the line printer.

2. Assign binary output to the GO file:

```
!ASSIGN (M:BO,BT,GO)
```

This example assigns the M:BO DCB to the GO file.

3. Assign source input to a disk file in the D1 data area:

```
!ASSIGN (M:SI,PRESTORE.D1)
```

This example assigns the M:SI DCB to the disk file PRESTORE, which is in the D1 area. This type of assignment could be used to assemble a source program that had been prestored onto a disk file.

4. Build a user DCB that was left empty at load time:

```
!ASSIGN (F:XX,7TAE0),PACK,(TRIES,3),
  (RECL,80),BIN
```

This example builds a user DCB, F:XX, and also assigns F:XX to a 7-track magnetic tape. The packed binary mode (PACK) will be used in accessing the tape, and a maximum of three recovery tries (TRIES,3) will be attempted for a possible tape parity error. The default record size to be read or written is 80 bytes (RECL, 80).

5. Assign a user DCB to read nonstandard binary cards:

```
!ASSIGN (F:INP,CRA03),BIN
```

This example assigns the user DCB, F:INP, to the card reader, and specifies that the binary mode is to be used in reading the cards. This type of assignment would be used to change an existing DCB to read nonstandard binary cards.

6. Assign a user DCB to a public file in the user's own account:

```
!ASSIGN (F:FILE,FILENAME)
```

This example assigns the DCB named 'F:FILE' to a file named 'FILENAME' in the user's account, in whichever public disk area it is found.

7. Assign a system DCB to a file in a specified account and area:

```
!ASSIGN (M:BI,ROM.D4.MYACCNT)
```

This example assigns the M:BI DCB to a file named 'ROM' in account 'MYACCNT' in area D4.

LOAD The LOAD control command directs the JCP Loader to load a background program on the disk and absolutize it for its main memory execution location.

The form of the LOAD command is

```
!LOAD [(option),(option)]
```

where the options are

IN,fid/name specifies the input device as a system physical device name, a system operational label, or a disk file from which the object modules will be loaded. The default input device is the one assigned to the BI operational label.

OUT,fid/name specifies the output device as an operational label or a disk file on which the loaded program is written. The default output device is the OV file.

EXLOC,value specifies the execution location (in hexadecimal) of the program being loaded. The default location will be the start of background.

SEG,value specifies the decimal number of overlay segments that follow the root. The default value is zero, which means only a root is being loaded.

MAP specifies that a map of the loaded program be output to the LO device. The default is no map.

The primary function of the JCP Loader is to load the Overlay Loader at SYSGEN time. However, the JCP Loader will load any nonoverlaid background program or one with a simple tree structure under certain restrictions (see Appendix E).

Example:

Load Overlay Loader from cards:

```
!LOAD (IN,CRA03),(OUT,SP,OLOAD),
(SEG,6),MAP
```

This command would be used to load the Overlay Loader onto its permanent file (OLOAD) in the SP area of the disk. Six overlay segments (SEG, 6) are specified and a MAP of the load is requested. The complete deck structure required to perform the load is illustrated in Figure 2.

ATTEND The ATTEND control command is used during a programmer-attended run and indicates that CP-R is to go into a WAIT condition after a WAIT system call or after an abort from the background. After an unsolicited key-in of "C", background processing will continue from the point of the wait. If the ATTEND control command is not specified and an abort or error condition occurs, or if a WAIT system call is made, the system does not pause for operator intervention but skips all control commands, binary records, and data until a JOB or FIN command is encountered. When in skipping mode, all control commands encountered will be listed on the LL device, with a greater than character (>) replacing the exclamation mark in column 1. The default mode of operation (no ATTEND command present) is used for closed-shop batch processing and there are no halts between jobs after an abort.

The form of the ATTEND command is

```
!ATTEND
```

The effect of an ATTEND command exists for one job only. Normally, the ATTEND command immediately follows the JOB command.

MESSAGE The MESSAGE control command is used to type a message to the operator. The message will be typed on the OC device and normal processing will continue after the message is output.

The form of the MESSAGE command is

```
!MESSAGE message
```

where message is any comment to the operator, up to a full card image (80 columns). The message may contain any desired characters, including blanks, but may not be continued from one record to the next. Two or more MESSAGE control commands may be used in immediate succession.

Note that the entire card image, including !MESSAGE, will be output to LL and OC.

Example:

```
!MESSAGE SEND ALL SAVE TAPES TO JOHN SMITH
```

The above example would cause the following message to be output on the LL and OC devices:

```
!!MESSAGE SEND ALL SAVE TAPES TO JOHN SMITH
```

Note: All CP-R messages to the operator begin with two exclamation characters.

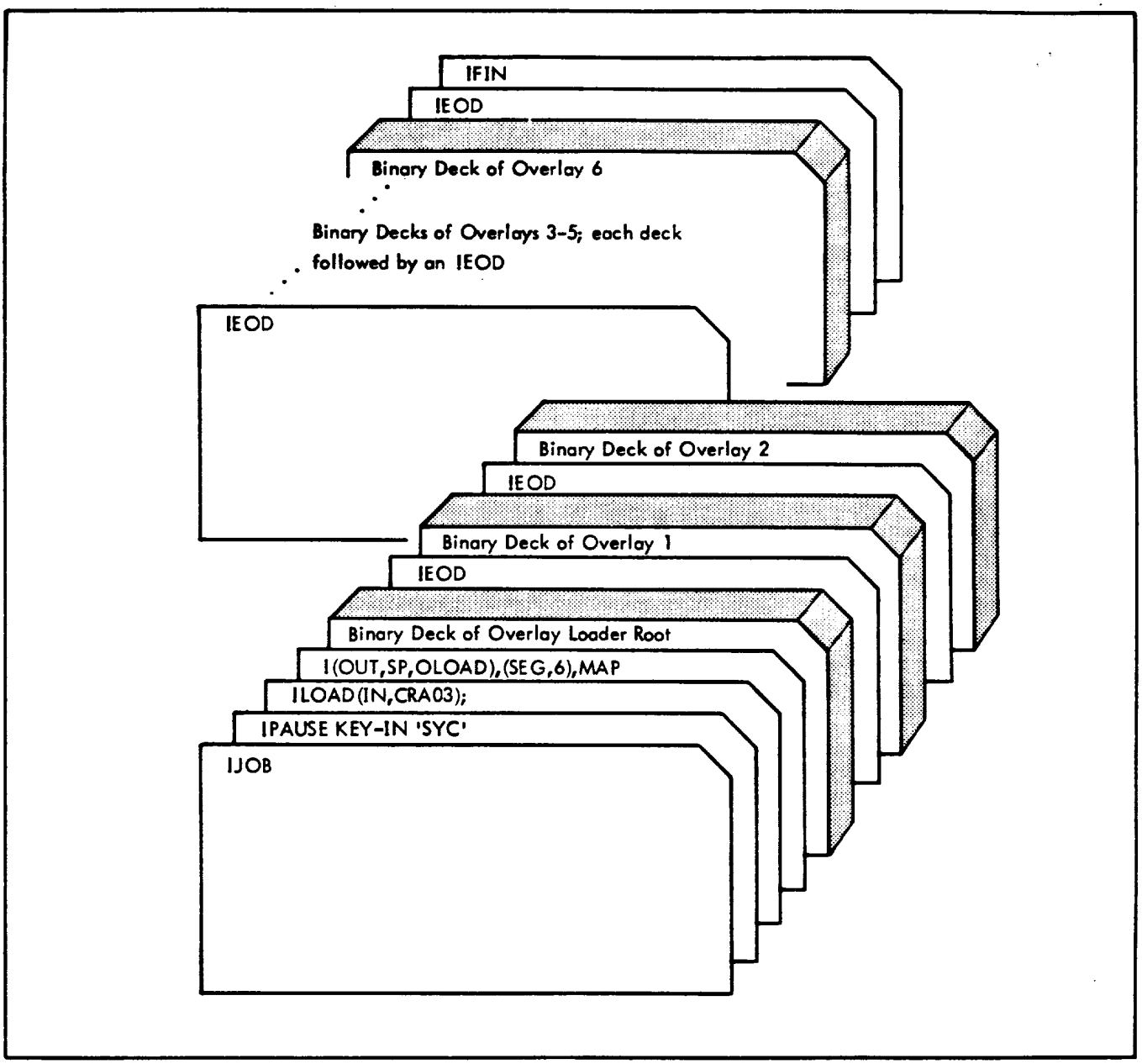


Figure 2. Loading Overlay Loader from Cards

LIST The LIST command is identical to the MESSAGE command above, except that the message is listed on LL only, and not on LL and OC.

The form of the LIST command is

```
!LIST message
```

where message may be any comment up to a full card image of 80 columns.

PAUSE The PAUSE control command is similar to the MESSAGE command except that the JCP will enter a suspended state after the message is output to OC to give the operator time to carry out the instructions in the message. Processing is continued after an unsolicited key-in of "C".

The form of the PAUSE command is

```
!PAUSE message
```

where message is any comment to the operator, up to a full card image (80 columns). PAUSE does not require ATTEND mode.

Example:

```
!PAUSE KEYIN SYC
```

The above example would cause the monitor to pause execution with the following message output on LL and OC,

```
!!PAUSE KEYIN SYC
```

giving the operator time to key in SYC, which would permit the user to override the write protection on the disk and continue the background job.

CC The CC control command removes typewriter override of the C device (see TY key-in description). The next control command will be read from the C device instead of the typewriter.

The form of the CC control command is

```
!CC
```

The CC control command has the same effect as the CC key-in, and can be used whenever the JCP has control.

LIMIT The LIMIT control command is used to set a maximum allowable execution time and maximum allowable number of line printer pages for a background program. If the job exceeds either limit, the background is aborted with a postmortem dump (if the dump option was specified via a PMD control command).

The form of the LIMIT control command is

```
!LIMIT n [,m]
```

where n specifies the maximum allowable execution time in minutes and m specifies the maximum allowable number of line printer pages.

STDLB The STDLB command is used to change the assignment of an operational label (with the exception of the OC – Operator's Console) for the current background job. The operational labels being changed receive the new assignments, which stay in effect until the next JOB command is encountered.

The form of the STDLB command is

```
!STDLB (label,{fid|device|oplabel})[,(label,{fid|device|oplabel})...]
```

where

label specifies the operational label to be assigned. It must have been defined during SYSGEN (see Table 5).

fid specifies the identity of a file, in the format defined in Chapter 1.

A disk file Xn in the BT area must have been defined by the JCP prior to its use in this command. (See the ALLOBT command.)

device specifies a physical device name or a numeric zero, "0", the null device.

oplabel specifies another operational label is to be used. "Label" will receive the same assignment as "oplabel" has currently.

Notes:
1. The C oplabel cannot be assigned to the null device.
2. If an error is detected, all assignments preceding the one in error will have been made.

Example:

Change temporary assignments of operational labels:

```
!STDLB (BO,BT,GO),(CO,COMPRESS.D2),(LO,9TA80)
```

This example could be used for an assembly to change the binary output to the GO file, the compressed output to the COMPRESS file in the D2 area of the disk, and the listable output to a 9-track magnetic tape.

ROV The ROV command (RUN OV) causes execution of the program (either foreground or background) on the OV file.

The form of the ROV command is

```
!ROV
```

Only primary programs may be loaded into Foreground with this command. The loading of any program into the foreground area via an ROV control command must be preceded by an FG key-in (see Chapter 3). A foreground program loaded by !ROV is given the name OV and runs under the CP-R job. There may be only one such program resident at any time.

If a priority other than the default value of X'7F' (as defined under the RUN control command) is to be associated with the foreground program, the IRUN BT, OV, priority command should be used. The default value X'7F' is the lowest priority.

RUN The RUN control command causes the named foreground or background program to be executed.

The form of the RUN command is

```
IRUN fid[,priority][,DEBUG]
```

where

fid specifies the identity of the load module to be executed. The loading of any program into the foreground area via a RUN control command must be preceded by an FG key-in and must be a primary program. It will run in the CP-R job.

priority is a hexadecimal number in the range 0 through X'7F', to be assigned to the foreground RUN request if the memory space is currently being used by another foreground program. The lowest priority, X'7F', is the default value.

DEBUG specifies that the program is to run under the control of the CP-R Debug facility.

INIT The INIT control command causes the named foreground task to be read into memory and initiated. The INIT command has the form

```
IINIT taskname[,(JOB,jobname)][,PRI][,STOP]
      [,(PRIO,xxxx)][,DEBUG][,TS]
```

where

taskname is the file identifier for the Load Module, with the exception that the specified file name may be converted to an actual file name by lookup in the Job Program Table. If no match is found in the JPT, the specified file name is used.

JOB is a keyword and indicates that the task is to be run under a job other than the CP-R job. If the JOB option is not specified, the task will be run under the CP-R job.

jobname specifies the name of the job under which, the task is to be run. It may not be "BKG".

PRI identifies the task as a primary task. If the PRI option is not specified, the task is run as a secondary task.

STOP specifies that the task is to be left in suspended state after load. The default is to execute directly after load. This option is invalid for a primary task.

PRIO is a keyword indicating that task priority is specified. This option is valid for secondary tasks only. The default is to run under the lowest-priority dispatcher, at the lowest software priority.

xxxx is the hexadecimal priority value for the task. The first two characters specify the interrupt level, minus X'4F', of the dispatcher for the task. The last two characters specify the software priority for the task.

DEBUG indicates that the task is to be initiated under control of Debug.

TS indicates that the task is to be time-sliced.

Note: IINIT command may not be used to start a background program.

SJOB The SJOB control command creates a foreground job. It sets up job controls and table entries but does not initiate any task in the job. The SJOB command has the form

```
ISJOB jobname[,(DEBUG,TYndd)][,(ACCT,xxx)]
```

where

jobname is the name of the job to be started.

DEBUG indicates that a Debug control-console device address is specified.

TYndd is the address that Debug will use for communication with the user when any task in the job is given control under Debug.

ACCT indicates that an account number (xxx) is being supplied.

xxx The account number the named JOB is to be associated with.

BATCH The BATCH command allows a background job to cause a file to be entered in the background stream as a control deck for a later job. This command is available only in systems generated for symbiont-supported background.

The form of the batch command is

```
!BATCH fid
```

where

fid is the identifier for the file to be entered into the batch stream. File identifier format is defined in Chapter 1. A batch file may contain only one background job.

ALLOBT The ALLOBT control command is used to define the files in the BT area of the disk, and overrides any JCP default definitions. The files input on the ALLOBT command will receive the specified sizes and formats. The files defined via an ALLOBT command will stay in effect only for the current job step unless the SAVE option is invoked. If the SAVE option is used, the ALLOBT command will stay in effect for the entire job (any input for the GO or OV files will always stay in effect for the entire job).

The form of the ALLOBT command is

```
!ALLOBT (FILE,nn)[,(option),(option). . .]
```

where

FILE,nn specifies the name of the background temp file to be allocated. Legal names for nn are X1,X2,...,X9,GO or OV.

The options are

FORMAT,value specifies the format of the file; U for unblocked, B for blocked, C for compressed.

The default is unblocked for all files except GO; the default for GO is blocked.

FSIZE,value specifies the decimal length of the file in logical records. If ALL is input for a value, the remainder of the BT area will be allocated for this file. An ALL input is allowed only once and is only allowed for Xi files (not GO or OV). A check is made for overflow of the BT area at the time the ALLOBT command is input. The default value is 1000 records. Note that the file size in sectors is computed using the logical record size and not the granule size.

RSIZE,value specifies the decimal number of words per logical record. This field is only meaningful for blocked or unblocked files since the monitor compresses records of compressed files into 256-word blocks. Blocked files have a default record size of 128 words, and unblocked files have a default record size equal to the granule size. Note that if RSIZE > 128, unblocked organization will always be given to the file.

GSIZE,value specifies the decimal number of words per granule. This field is only used in directly accessing a file. The default granule size will be the size of a disk sector.

SAVE specifies that this file is to be saved throughout the job and not reallocated between job steps.

Example:

Change the default assignments of the background temp files:

The group of ALLOBT commands

```
5 !ALLOBT (FIL,OV), (FSIZ, 0)
4 !ALLOB (FIL,X4), (FSI,ALL)
  !(GSIZE, 180)
3 !ALLO (FILE,X3), (FSIZE,20);
  !(FSIZE, 100), (RSIZE,30)
2 !ALL (FILE,X2), (FORMAT,B);
  !(FSIZE, 1000), SAVE
1 !ALLOBT (FILE,X1), (FORMAT,C);
```

could be used by a background program to achieve the following results:

1. The X1 temp file would be a compressed file that could hold approximately 1000 EBCDIC cards. This file would be saved throughout the entire job.
2. The X2 temp file would be a blocked file that could hold a maximum of 100 binary cards.
3. The X3 temp file would be an unblocked file containing 40 sectors (assuming a 7204 disk) with a granule size of 180 words or two sectors.

4. The X4 temp file would be an unblocked file with a record and granule size of 90 words (assuming a 7204 disk) and would be allocated the remainder of the Background Temp area.

5. The OV file would not be allocated.

After inputting this series of ALLOBT commands, the background temp area would have the following layout (assuming a 7204 disk):

X2	X3	X4	X1	GO
36 Sectors	40 Sectors	n Sectors	120 Sectors	Default Size

Note that X4 receives n sectors, where n is the remainder of the area after all other files have been allocated. X1 is allocated at the opposite end of the BT area since it will be saved throughout the entire job.

The formula used to calculate the number of sectors for X2 is

$$\frac{RSIZE}{256} \times FSIZE \times 3$$

where 256 is the number of words per blocking buffer and 3 is the number of sectors (assuming a 7204 disk) necessary to contain a blocking buffer.

The formula used to calculate the number of sectors for X1 is

$$\frac{FSIZE}{25} \times 3$$

where it is assumed that 25 cards can be compressed into a 256-word blocking buffer. The number 3 is the number of sectors necessary to contain a blocking buffer.

DUMP CONTROL COMMAND

PMD The postmortem dump (PMD) control command causes the system to dump a specified area of memory if a background job is aborted during execution. Such a dump is termed "postmortem" because it is performed after the background program has been aborted, terminated normally, or not executed at all for any reason. The dump is always output on the DO device. In the case of an abort the time to perform the dump is not included in the total time on the LIMIT control card. Note that the PMD command must precede the RUN command.

The form of the PMD command is

```
!PMD[U][,T][,(from,to[,T])][,(from,to[,T])]. . .
```

where

U specifies that an unconditional dump at the end of the job is to be output even if there were no errors. If U is absent, the dump occurs only if the job is aborted.

T specifies (when it precedes address ranges) that the dump will be listed in both hexadecimal and text for all address ranges that follow. If the general T is absent, the [,(from,to,T)] option will dump that address range in hexadecimal and text, and the [,(from,to)] option will dump that address range in hexadecimal only.

from specifies the location (in hexadecimal) at which dumping is to begin. If no locations are specified, the entire background is dumped.

to specifies the last location (in hexadecimal) to be dumped. The last location must ≥ first location.

A maximum of four location pairs is processed and only the last PMD command is honored within a job step. If an error occurs anywhere on the command, the entire command must be reentered.

Example:

Request a postmortem dump:

```
!PMD U,(1200,1300),(2000,3000,T)
```

This example requests an unconditional dump at the termination of the next program to run in the background. Locations 1200_{16} through 1300_{16} will be dumped in the standard hexadecimal format, and 2000_{16} through 3000_{16} will be dumped in both hexadecimal and text. The output will be on the DO device.

INPUT CONTROL COMMANDS

EOD The user may define blocks in a data deck by inserting EOD control commands at the end of each block. When an EOD command is encountered, the system returns an EOD status. Any number of EOD commands may be used in a job and for any reason.

The form of the EOD command is

```
!EOD
```

Note that EOD control commands must not have any spaces between the exclamation character and the mnemonic.

FIN the FIN control command is used to specify the end of a stack of jobs. When the FIN command is encountered, the system writes it on the listing log to inform the operator that all current jobs have been completed, types "BEGIN IDLE" on OC, and then enters the idle state.

All time preceding the FIN command is charged to the previous job, if job accounting is being performed. All time from the FIN command to the next JOB command is not charged.

The form of the FIN command is

```
!FIN
```

UTILITY CONTROL COMMANDS

The utility control commands described below allow the user to manipulate magnetic tape files.

PFIL,PREC The file and record positioning commands are used to position a device within its current file. The PFIL command, will leave the device positioned before the file mark when moving in the forward direction and at the first record of the file when positioning backwards. For the PREC command, no adjustment is made when an EOT, BOT, or EOF mark is encountered. Only background devices (not dedicated to the foreground or IOEX) can be positioned.

The forms for the PFIL and PREC control commands are

```
{!PFIL }
{!PREC} name[,BACK][,n]
```

where

name specifies a system device name or operational label of the device that is to be positioned. This must be the first item in the specification field.

BACK specifies that the direction of the positioning is backward. The default is forward.

n specifies the number of records to skip. The n parameter applies only to the PREC command and not to PFIL. The default is skip one record. The PFIL command always refers to one file. (Any BOT, EOT, EOF marks encountered will terminate the PREC command.)

Examples:

1. Position the BO oplabel to the end of the data:

```
!PFIL BO
```

This example could be used to position the BO oplabel file so additional object modules could be added to those already existing.

2. Position a magnetic tape:

```
!PREC BO,30
!STDLB (BO,7TAE0)
```

This example would position magnetic tape on 7TAE0 30 records forward from its current position.

SFIL The skip file command is used to skip one or more files on a magnetic tape unit. The SFIL command leaves the device positioned beyond the specified EOF in the direction of the tape movement. If a BOT condition occurs, the device is positioned at the first record following the BOT marker.

The form of the SFIL control command is

```
!SFIL name[,BACK][,n]
```

where

name specifies a system device name or operational label of the device that is to be positioned. This must be the first item in the specification field.

BACK specifies that the direction of the positioning is backward. The default is forward.

n specifies the number of files to skip. The default is one file.

Example:

Skip tape files:

```
!SFIL 9TA82,BACK,4
```

This example would cause back skipping of four files on the designated 9-track magnetic tape.

REWND The REWIND command is used to rewind a magnetic tape. It has no effect on other devices.

The form of the REWIND command is

```
!REWIND name
```

where name specifies a system device name or operational label of the device that is to be rewound.

Example:

Rewind a tape

```
!REWIND 7TAE0
```

This example would rewind the designated 7-track tape.

UNLOAD The rewind manual (UNLOAD) command causes the specified magnetic tape to be rewound in manual mode. Operator intervention will be required to use the device again (i.e., depressing the START switch on a tape drive).

The form of the UNLOAD command is

```
!UNLOAD name
```

where name specifies a system device name or operational label of the device that is to be rewound in manual mode.

Example:

Unload a magnetic tape:

```
!UNLOAD 9TA83
```

This example would cause the designated 9-track tape to be rewound in manual mode.

WEOF The write end-of-file (WEOF) command causes an end-of-file mark to be written on the output device if an EOF is appropriate for the device. For magnetic tape, a tape mark is written; for cards, an EOD is written. The WEOF command is ignored for all other devices.

The form of the WEOF command is

```
!WEOF name[,n]
```

where

name specifies a system device name or operational label of the device that is to receive the EOF.

n specifies the number of end-of-files to write. The default is one.

Example:

Write end-of-file on magnetic tape:

```
!WEOF 9TA81,2
```

This example would write two EOFs on the designated 9-track magnetic tape.

DAL The Dump Accounting Log command causes the contents of the Accounting Log to be printed on the LO device. The Accounting Log is kept on the AL file on the D1 area of the disk. An option exists to purge the file after the dump is completed. Note that an SY key-in is required to purge the AL file.

The form of the DAL command is

```
!DAL [PAL]
```

where PAL specifies that the Accounting Log is to be purged after the dump is completed.

PROCESSOR CONTROL COMMANDS

A processor control command indicates to the system that control is to be transferred to the specified processor. It may also specify the types of input to be accepted and the types of output to be produced by the processor.

Processors can be created, updated, and deleted under normal batch operations, and there are no restrictions as to how many and what kind of processors may be added to the system.

User programs in any area of the disk can be called by

```
!RUN fid
```

where fid is the file identifier for the load module to be executed.

All system processors and user processors in the SP area or system processor alternate area can be called for execution by the control command:

```
!name parameters
```

where

name is the disk file name of the processor to be executed (e.g., FORTRAN, SL-1, or AP. Note that the disk file name for the Assembly Program should be AP, since the JCP does special allocation of the BT area if !AP is encountered.

parameters are optional parameters interpreted by each processor. Normally, at least one input option and one output option must be specified.

PROCESSOR INTERFACE WITH CP-R

The standard system processors available under CP-R are

Assembly Program (AP)

Overlay Loader

RADEDIT

Edit

Extended FORTRAN IV

Dump Analysis Program (ANALYZE)

SL-1

Error Log Lister and Analysis (ELLA)

Device Exerciser

System processors and any user processors should follow these common ground rules:

1. All processors must reside on the SP area or system processor alternate area of the disk to be callable by a lname command.
2. All processors must operate in the background.
3. All system DCBs (M:dcb) should be identified as a primary reference in the processor, since at load time the Overlay Loader will furnish the processor with a copy of the system DCBs.
4. All processors with overlay segments need only make the explicit call to SEGLOAD to load the segments. The DCB used to load segment M:SL will be furnished by the Overlay Loader.
5. CP-R will furnish the start address and end address of unused background memory to any processor that needs this information. The two addresses will be in the following locations, and should be defined via the EQU directive in the processor:

Location	Mnemonic	Description
X'153'[†]	K:BPEND	LWA +1[††] of the background program's loaded area; that is, this cell contains the FWA[††] the processor can use for a dynamic table area.
X'141'[†]	K:BCKEND	LWA of usable background memory for the processor; that is, this cell contains the LWA the processor can use for a dynamic table area.

6. If a processor has parameters to process from the "lname" control command (where "name" is the processor's name) the address of the buffer containing the control command is in cell X'144'. That is,

Location	Mnemonic	Description
X'144'[†]	K:CCBUF	Address of control card buffer.

7. A processor must perform its own vertical format control of the printer if format control is required. That is, the processor must set the VFC (vertical format control) bit in the DCB via the Device Format Control call and ensure that the first byte output to the printer is a format control byte. If a processor (i.e., AP) outputs a title at the top of each page, the number of lines to print per page is contained in the following system cell:

Location	Mnemonic	Description
X'174' byte 0	K:PAGE	Number of lines per page to print.

8. If a processor uses scratch files (background temp files X1-X9) and desires a different record size, granule size, or organization that is given by default by the JCP, the processor must make the appropriate system call on the Device Mode function. By calling the Device Mode function, the processor can set the file organization (blocked or compressed) and the appropriate record size and granule size. The background temp file default assignments by the JCP are described below.
9. In general the processor should terminate input from SI when an end-of-file status is sensed on SI. To terminate, the processor should make a system call on EXIT. EXIT will close all the processor's DCBs and close all open disk files.
10. All processors using the GO file should open GO and then do a file skip (PFIL function call) on GO so the GO file is properly positioned to receive additional data. The Job Control Processor will purge the GO file upon reading a JOB control command.

[†]All these addresses are in bits 15-31 with bits 0-14 containing zeros.

[††]LWA and FWA are the last word address and first word address.

JCP will allocate the Background Temp area for all Xi files, where, 1 i 9. The GO and OV files will receive their SYSGEN defined sizes unless overridden with an ALLOBT command. The GO file will be defined as a blocked file with a logical record size of 120 bytes; the OV file will be unblocked with the record and granule sizes equal to the sector size. The Background Temp area that remains after GO and OV have been allocated will then be equally distributed among the Xi files. All Xi files will be given unblocked organization with the record and granule sizes equal to the disk sector size. The user can override any of these defaults via an ALLOBT command. If the user desires not to have the Xi files reallocated between processors, the SAVE option on the ALLOBT command can be used.

JCP MESSAGES

The messages itemized in Table 6 are output by the Job Control Processor on the LL device.

Table 6. JCP Messages

Message	Meaning
!!ACCT LOG I/O ERROR X'xx'	An I/O error that could not be automatically processed was encountered during JCP execution. CP-R will enter a WAIT state if the !ATTEND control command was first read in; otherwise, an abort and a skip to the next job will occur.
!!BI CKSM ERR !!BI SEQ ERR	JCP Loader encountered a checksum or sequence error on a binary card during the loading process.
!!BT OVERFLOW	Insufficient Background Temp space to execute the requested background program. The job is aborted.
!!CC ERROR, BT OVERFLOW	The file size input on an !ALLOBT command is greater than the available Background Temp space.
!!CC ERROR, FG KEY-IN REQUIRED	A request has been made to run a foreground program without previously inputting an FG key-in. The !RUN or !ROV command must be reentered after the FG key-in is input.
!!CC ERROR, ILL. RELOCATION OF BT	An improper !ALLOBT command was input to change a Background Temp (BT) scratch file that was designated as a "saved" file prior to this job step.
!!CC ERROR IN ITEM xx	An error exists in a JCP control command in the indicated item. Every item (except the ! character) followed by a blank or comma is counted in determining the item in error.
!!D1 AREA FGD, CAN'T UPDATE AL FILE	JCP could not process accounting information because the D1 area on disk has been reserved for foreground. CP-R will enter WAIT state if an !ATTEND control command was specified; otherwise, an abort and a skip to the next job will occur.
!!EOT ON FILE xxxxxxxx	End-of-Tape status was returned from an attempt to read or write the indicated disk file.
!!ERR, CONTROL BYTE = xx	JCP Loader is not equipped to process the indicated control byte.
!!FILE xxxxxxxx NONEXIST.	The indicated disk file was never allocated via the RADEDIT or was never written into.
!!ILL. DEFINE FIELD ITEM	JCP Loader has encountered a define field item that it is not equipped to handle.
!!ILLEGAL BINARY CARD	An EBCDIC card was read by the JCP Loader where a binary card was expected.

Table 6. JCP Messages (cont.)

Message	Meaning
!!ILL. EXPRESSION	JCP Loader has encountered an expression that it is not equipped to evaluate (a mixed resolution expression). The load will be aborted.
!!ILL. NEG. ORG ITEM	JCP Loader has encountered an origin item that it is not equipped to handle (an origin item that moves the load location counter in a negative direction). The load will be aborted.
!!I/O ERROR X'xx', LOC 'xxxxx'	JCP encountered an irrecoverable I/O error.
!!LMI FULL, CAN'T LOAD xxxxxxxx	The indicated foreground program cannot be loaded because insufficient space exists in the Load Module Inventory table.
!!NOT A TASK LOAD MODULE	A RUN command has named a file without a valid load module header. The RUN command is aborted.
!!NOT ENUF SPACE FOR LOAD	JCP Loader is unable to complete the load because of insufficient background space.
!!PUB LIB, CAN'T LOAD xxxxxxxx	The designated program on the !RUN command is a Public Library and cannot be executed via a !RUN command.
!!SCHED FILE OVERFLOW	File SCHED in the SP area is full thus discarding all future requests for periodic scheduling. File should be copied and enlarged.
!!SCHED FILE TYC 'xx' ON 'operation'	The type completion 'xx' was received on the 'operation' indicated or the file SCHED in SP. Refer to Appendix N for TYCs.
!!SCHING FOR JOB CMD	The present job has been aborted and the JCP is searching the job stack for the next !JOB or !FIN command.
!!'taskname' IN 'jobname' MISSED 'xx' CYCLES	The task indicated in the job indicated missed 'xx' initiations by the periodic scheduler.
!!'taskname' IN 'jobname' TYC 'xx' [DELETED]	The task indicated in the job indicated received the type completion 'xx' when an INIT was attempted by the periodic scheduler. The entry was deleted if indicated. Refer to Appendix N for TYCs.
!!TOO MANY ASSIGNS	JCP has encountered more than 48 !ASSIGN commands in a job step.
!!TOO MANY CONTROL SECT.	JCP Loader has encountered more than one nonstandard control section. The load will be aborted.
!!TOO MANY DCB'S	The maximum number of M: and F: DCBs was exceeded during the loading process. Approximately 27 DCBs can be accommodated by the system. The excess DCBs will not be stored in the DCB table or the disk file header.
!!UNSATISFIED REF xxxxxxxx	Indicated REF was not satisfied during the loading process. This alarm occurs only on LL if no map was requested, or on LO if a map was requested.
!!UNSATISFIED REF's DURING LOAD	This message is typed to the operator on OC at the end of a load if any unsatisfied REFs were encountered during the loading process.
!!WARNING: NONEXIST. FILE xxxxxxx	JCP has processed an !ASSIGN command that assigned a DCB to a nonexistent file. JCP will continue normal processing.

3. OPERATOR COMMUNICATIONS

Communications between the operator and the system take place through operator key-ins (solicited and unsolicited) and monitor printouts (CP-R messages).

CP-R MESSAGES

CP-R and associated processors (as selected during SYSGEN) output messages on the OC device whenever operator action is required or to inform the operator as to the status of events (including errors) taking place within the system. CP-R output messages are listed and described in Table 7.

TRAP HANDLER MESSAGES

In addition to the messages listed in Table 7, the following messages are output by the trap handler upon occurrence of the various traps if the user does not specify his own trap handling:

!!ARITH. FAULT AT xxxxx ID = xxxxxxxx

!!BREAK ERROR AT xxxxx ID = xxxxxxxx

!!MEM. PARITY ERR AT xxxxx ID = xxxxxxxx

!!MEM. PROT. ERR AT xxxxx ID = xxxxxxxx

!!NONEXIST. ADD. AT xxxxx ID = xxxxxxxx

!!NONEXIST. INST. AT xxxxx ID = xxxxxxxx

!!PRIVILEGE INST. AT xxxxx ID = xxxxxxxx

!!STACK OVERFLOW AT xxxxx ID = xxxxxxxx

!!UNIMPLE. INST. AT xxxxx ID = xxxxxxxx

!!WDOG TIMER RUNOUT AT xxxxx ID = xxxxxxxx

Note that the message "ARITH. FAULT AT xxxxx" is output for the fixed point arithmetic overflow trap, the floating-point fault trap, and the decimal arithmetic fault trap. The message

"ERRxx ON CAL AT xxxxx ID = xxxxxxxx"

is output if a user program furnishes an invalid parameter while attempting to use a service function. ID identifies the trapped task.

Table 7. CP-R Messages and Responses

Message[†]	Meaning	Operator Action
!!yyndd **** message	"yyndd" is the control device and job name for the terminal job. "Message" is the message sent by the terminal user to the operator.	Action is determined by the content of the terminal user's message.
!!yyndd ATTENTION INTERRUPT	An attention interrupt was received from the specified device.	No action is required.
!!yyndd ERROR[†††]	An irrecoverable error has occurred.	No key-in response required except for card reader. For card reader error, remove last card in output hopper for jam and replace it or a duplicate to input hopper. Key in CRndd R ↵. If card or reader cannot be fixed, key in CRndd E ↵ to inform the requesting task the card reader erraged.

[†]Messages beginning with !!yyndd in the message column will vary according to device type. Therefore the second words in such messages are listed alphabetically.

[††]If background, requires an I/O key-in to continue or retry I/O operation on the device; if foreground, the operation is errored and no key-in is required or expected.

[†††]Key-in may be required depending on device type.

Table 7. CP-R Messages and Responses (cont.)

Message	Meaning	Operator Action
!!yyndd ERROR, NOT OPERATIONAL	Device went not operational during I/O operation.	No action required.
!!yyndd ERROR, POSITION LOST	A magnetic tape unit either returned inconsistent status or the tape position is indeterminant following a tape operation.	No action required.
!!yyndd I/O TIMED OUT[tt]	An I/O interrupt failed to return from the device used for an I/O operation in the software timeout period allowed.	Unless the timeout occurred in a foreground task, the message always requires operator action. If the cause and/or correction is unknown, key-in yyndd E(NL). If the cause is known, key in either yyndd R(NL) or yyndd C(NL) to specify whether the I/O operation should be or should not be retried. (R specifies retry; C specifies continue.)
!!yyndd KEY-IN PENDING	An I/O operation is waiting for an operator key-in on the indicated device. This message is repeated at intervals.	Supply the appropriate key-in.
!!yyndd MANUAL	Device was in manual mode at SIO initiation.	Ready the device. No key-in is required.
!!yyndd SIO REJECT, CC = 10 —[tt]	SIO instruction returned CC1 and CC2 as nonzero (10).	Correct condition and key in yyndd R(NL). If condition cannot be repaired, key in yyndd E(NL) to inform task the I/O has errored.
!!yyndd SIO REJECT, CC = 01 —[tt]	SIO instruction returned CC1 and CC2 as nonzero (01).	Correct condition and key in yyndd R(NL). If condition cannot be repaired, key in yyndd E(NL) to inform task the I/O has errored.
!!yyndd TEST MODE	Device went into test mode during I/O operation.	No action required.
!!yyndd UNRECOGNIZED[tt]	SIO instruction returned CC1 and CC2 as nonzero (11).	Correct condition and key-in yyndd R(NL). If condition cannot be repaired, key in yyndd E(NL) to inform task the I/O has errored.
!!yyndd WRITE PROTECTED[tt]	An attempt was made to write to a write protected device or disk track.	If the write is to be permitted then change the write protected status of the device or track and key in yyndd R(NL). If the write is not to be permitted, key in yyndd E(NL).
!!ALARM xxxxxxxxx	A system inconsistency was detected, following the ALARM message. The reason is indicated in the text.	See Chapter 5, "Availability Operating Procedures".
!!BACKGROUND IDLE	Background sequencing has been terminated because JCP read a !FIN command or encountered a critical error.	If more background jobs are to be run, key in C to restart background sequencing.

Table 7. CP-R Messages and Responses (cont.)

Message	Meaning	Operator Action
!!BACKGROUND WAIT	Background has executed a "WAIT" request.	Key in C ↵ to continue background processing.
!!BKG ABORT WAIT	Background has aborted in attended mode.	Key in "C" to continue, or "DB" to dump background memory.
!!BKGD JOB ident ON	Specified background symbiont job has been selected for execution.	No operator action required.
!!CANT OPEN ERRORLOG	The error log file was already in use by error list program, full RFT, no blocking buffer available, or a DED key-in was in effect.	See Chapter 2 Availability Manual.
!!CORE SAVED	Memory has been saved on the CK area following a system fault.	No action required.
!!CORE USED, CAN'T LOAD xxxxxxxx	The specified primary program cannot be loaded for execution because its required core space is already in use.	Key-in X ↵ to abort the job.
!!CP-R RESTARTED	CP-R is fully restarted following a system ALARM and an auto-restart. The date and time information have been maintained and need not be re-entered.	No action required.
!!CRndd CARD NOT FED††	The card reader was unable to feed a card correctly.	Correct or replace the card in the read hopper. Push RESET START and key in CRndd R ↵ to retry the operation.
!!DPndd IDLE	Indicated unit has no open files.	May be removed from the spindle.
!!ERRORLOG ERROR	An irrecoverable write error was encountered while attempting to write entries to the ER operational label.	Reassign ER oplabel to another output device or "0".
!!ERRORLOG FULL	While attempting to write entries to the ER operational label, an end-of-file, end-of-data, or end-of-tape was encountered.	See Chapter 2 for Error Log purging procedures. Availability Manual.
!!FILE NAME ERR	A problem has occurred in attempting to open or close a disk file through an STDLB key-in.	Key-in X ↵ to abort the job.
!!I/O ERR, CAN'T LOAD xxxxxxxx	An I/O error occurred or no blocking buffer was available when attempting to load the specified foreground program for execution.	Key-in X ↵ to abort the job.
!!JOB account, name, priority, ident	An input job was added to a list of symbiont files. Message has some format as user's job card except for ident, which is added to the end.	No action required.
!!KEY ERR	CP-R cannot recognize an unsolicited key-in response.	Retry the key-in.

Table 7. CP-R Messages and Responses (cont.)

Message	Meaning	Operator Action
!!LMI FULL, CAN'T LOAD xxxxxxxx	The specified primary program cannot be loaded for execution because no room exists in the Load Module Inventory.	Key-in X ↵ to abort the job.
!!LOADED PROG xxxxxxxx	Specified primary programs were loaded by Foreground Loader for execution. Up to three program names will be output in one message.	No action required. Outputting of this message to OC may be prevented by setting appropriate assembly time software switch to bypass the code causing the message.
!!MEDIA ABORTED REQ xxxx:yyyy	MEDIA request xxxx was aborted due to reason yyyy where yyyy can be OPER - operator keyed in "X" NOMO - the MO oplabel is not defined OPNI - unable to open input file's DCB OPNO - unable to open output file's DCB PREP - an irrecoverable error occurred during preprocessing BUFS - insufficient job reserved pages for blocking buffers DEV - a fatal error occurred during the copy SPEC - invalid specifications were detected	None. If request xxxx was initiated by an operator or MEDIA key-in and the cause of the abortion removed, the key-in can be reentered.
!!MEDIA MOUNT TAPES FOR xxxx	MEDIA request xxxx is ready to begin.	Mount tape or tapes required. When ready, key-in MEDIA I to initiate copy.
!!MEMORY FAULT	During a memory status scan, memory status was found which indicated that an error was detected by the memory unit. An error log entry was recorded for this problem.	No action required.
!!NO PATCH AREA - CLEAR B S IGNORES MODIFY COMMAND	If this is OC, only the message portion preceding the hyphen is output. If not OC, the entire message is output.	If this is OC, input next command. If not OC, idle machine, increment address, and RUN.
!!NO TSPACE, CAN'T LOAD xxxxxxxx	A primary program load request could not be satisfied because TSPACE was not available for building FPTs or for reading in the header.	Key in X ↵ to abort the job.
!!NONEXIST., CAN'T LOAD xxxxxxxx	Specified program cannot be loaded for execution because it does not exist on disk, or required Public Library does not exist on disk.	Retry. If message is output again, key-in X ↵ to abort the job.

Table 7. CP-R Messages and Responses (cont.)

Message	Meaning	Operator Action
!!PATCH LOC ERR-CORRECT AND CLEAR B $	If this is OC, only the message portion preceding the hyphen is output. If not OC, the entire message is output.	If this is OC, input next command. If not OC, idle machine, increment address, and RUN.
!!PAUSE commands	A !PAUSE command has been read. Comments field will have operator instruction.	Press INTERRUPT switch and key in C ⏎ to continue reading from the job stock after performing required action.
!!PLEASE KEY-IN DATE-TIME	CP-R has been booted in and requires the date and time before other operations are allowed.	Enter DT␣ mm, dd, yy, hh, mm where mm - month (1-12) dd - day (1-31) yy - year (00-99) hh - hour (0-23) min - minute (0-59)
!!PROCESSOR FAULT	During a processor status scan, processor status was found which indicated that an error was detected by a processor. An error log entry was recorded for this problem.	No action required.
!!PUB LIB, CAN'T LOAD xxxxxxxx	Invalid request to load Public Library for execution. All Public Libraries must be automatically loaded by the system as needed.	Either retry the job or key in X⏎ to abort the job.
!!QUEUED AS NUM. xxxx	MEDIA copy request is accepted and is assigned ident number xxxx.	No action required.
!!RELEASED PROG xxxxxxxx	Specified primary program was released.	No action required. Message can be prevented by setting appropriate assembly time software switch to bypass code causing the message.
!!RLS NAME NA	Key-in request to release a foreground program whose name is not recognized by the system.	Retry.
!!SPURIOUS EXTERNAL INTERRUPT	An interrupt has been triggered but is not connected to a task.	No action is required.
!!Syyndd ERR xx	An irrecoverable error occurred. Symbiont activity on specified device was terminated.	No action required.
!!Syyndd IDLE	Specified symbiont device entered idle state.	No action required.
!!Syyndd UNAVAILABLE	Specified device is currently unavailable to the symbiont.	No action required.
!!TERMINAL JOB yyndd account, name OFF	Indicates the terminal user is now inactive. "yyndd" is the control device and job name for terminal job. "Account, name" is logon verified user account, name.	No action required.

Table 7. CP-R Messages and Responses (cont.)

Message	Meaning	Operator Action
!!TERMINAL JOB yyndd account, name ON	Indicates the terminal user is now active. "yyndd" is the control device and job name for terminal job. "Account, name" is logon verified user account and user name.	No action required.
!!UNABLE TO CLOSE DCB xxxxxxx	The specified DCB was not closed in the task during termination.	No action required.
!!XEROX CP-R VERSION xxxx	Output whenever system is booted in.	Message can be terminated by hitting BREAK key on OC.

OUTPUT MESSAGE FORMATS

Output messages are printed in three different formats on the Operator's Console. These formats are outlined below.

1. Messages without tabs are for operator's reference at a later time and are typically generated by foreground or background programs that wish to communicate with the operator. They are similar to system messages but have no time stamp.

2. System messages which begin with two exclamation marks (!!) will be preceded by a time stamp (hh:mm) in column 1 through 5, followed by a single tab. With normal tab settings the message will be positioned 1 inch to the right of the left margin.

3. Device control messages will have a time stamp in column 1 through 5, followed by four tabs. With normal tab settings the message will be positioned 4 inches to the right of the left margin.

OPERATOR KEY-IN

After the system has been initialized, operator key-ins permit the operator to control the execution of tasks by the system (i.e., foreground tasks, background job stream, and Control Task services).

With the exception of the DT key-in (which is requested by the system at the end of system initialization but prior to any other activity, if job accounting or error-logging has been requested), operator key-ins are unsolicited. That is, an operator key-in is initiated by the operator depressing the INTERRUPT switch on the Control Panel. (On a Xerox 550, the ATTENTION key may be substituted.) This action activates the Control Panel Task which, in turn, triggers the CP-R Control Task. When the Control Task becomes the highest priority task in the system (that is, when all foreground tasks are inactive), the system issues a prompt character (a dash, "-") to the Operator's Console and waits for operator input.

Via the OC device, the operator inputs appropriate information in the following sequence:

1. Optionally, types in an exclamation mark (!).

2. Types in the desired key-in and any associated parameters. (Refer to Tables 8 through 11 for listing and description of Standard, Terminal Job Entry, Symbiont, and Media Conversion key-ins.)

3. Types in a New Line character ⏎ to indicate the completion of a key-in.

If a typing error is discovered before the ⏎ character has been keyed in, the error may be corrected by the operator in one of the following ways:

1. If the erroneous character is the last character typed, key in one cent sign (¢) character, retype the last character correctly, and continue to type in the remainder of the key-in. Indicate completion of corrected key-in by keying in the ⏎ character.

2. If the erroneous character is within a few positions of the last character typed, key in an appropriate number of cent signs (starting with the last character, each cent sign deletes one character and performs a simulated backspace of one position) until the erroneous character has been effectively deleted. Retype correctly, all characters deleted and resume typing rest of key-in. Upon completion of key-in, key in the ⏎ character.

3. If the erroneous character is located toward the beginning of the key-in or many positions from the last character typed, it may be more expeditious to key in an EOM (End of Message) character. In this case, the entire key-in is deleted and the monitor is ready for a new key-in.

Table 8. Standard Operator Key-Ins

Key-In	Purpose
yyndd a ⓝⓛ	Controls system action following an abnormal condition during an I/O operation, where "yyndd" is the physical device name of the device involved, and "a" is a single character that requests a system action relative to the device as follows: a = C – continue "as is"; a = E – inform user program of the error and transmit record "as is"; a = R – retry the I/O operation; a = X – unconditionally abort the current operation on the device and error the request.
ALARM ⓝⓛ	Forces an operator-initiated system alarm. See also REBOOT key-in.
ATT [{OFF / ON}] ⓝⓛ	If OFF, the batch control command !ATTEND becomes illegal. If ON, both the !PAUSE and !ATTEND control commands become legal. This command allows the system to permit or reject batch jobs that depend upon the programmer being able to interact with his job. Default is ON. See also PAU key-in.
BMEM[n] ⓝⓛ	Change memory allocation for the background job. The "n" specifies the number of pages to be allocated to the background job. If "n" is not specified, its background job memory is restored to its SYSGEN-defined value. Allocations for background tasks using simplified memory management (SMM option on the !OLOAD command) will not take effect until the current background job step terminates.
BREAK jobname ⓝⓛ	Transfers control to user's break-receiver routine if the user established break control in the job via an INT service call. jobname is the name of the job to receive the break signal.
C [S/O] ⓝⓛ	If no parameters: Continue processing in the background. If the background was in a wait or idle state, the system leaves that state and proceeds. If "S" parameter: Change the background control mode to symbiont, so the symbiont can start background when necessary. If "O" parameter: Change the background control mode to operator, so only the operator may start background. The symbiont can only start background from an idle state. It can not clear a background wait.
CC ⓝⓛ	Retransfer control back to the C device from OC. Used in conjunction with TY key-in.
CINT {location / label}, {D / A / T} ⓝⓛ	Disarm, arm and enable, or trigger specified interrupt. The "location" specifies the hex address of the interrupt; "label" specifies an interrupt label; "D" is used to disarm specified interrupt; "A" is used to arm and enable; "T" is used to arm, enable, and trigger the interrupt.
CKD {loaddr / loaddr,hiaddr} ⓝⓛ	Selectively dump system output saved in the CK area after a system alarm. The "loaddr" parameter specifies the lowest address to dump. The "hiaddr" parameter specifies the highest address to dump. Default cases for "loaddr" and "hiaddr" are 0 and high core respectively. Activation of the Control Panel Interrupt during the MAP portion of the dump stops map output and starts the DUMP portion. Activation of the interrupt during the DUMP portion terminates the dump.
COC ⓝⓛ	Job was halted because of error in control command. Continue from OC with correct control command (after depressing the Control Panel Interrupt Key).

Table 8. Standard Operator Key-Ins (cont.)

Key-In	Purpose
CRD {loaddr / loaddr,hiaddr} ⊛	Identical to CKD key-in above except that the dump is read from the SE oplabel rather than the CK area (see CRS key-in below).
CRS ⊛	Preserve an alarm dump by copying it from the CK area to the SE oplabel in 1024-byte records.
DEBUG taskname[,(JOB,jobname)] ⊛	Causes the named task in the named job to be run under DEBUG control.
{DED / UND} yyndd, {F / X / N / D / R} [{1 / ,D}] ⊛	DED dedicates a device, device controller, or IOP. UND undedicates a previously dedicated device, device controller, or IOP. F defines the device to be dedicated to the foreground; X prevents device use and aborts any existing requests; N prevents device use but does not abort existing requests for use; D restricts a device for diagnostic use and aborts any existing requests; R specifies that the disk pack will be removed (DED) or has been replaced (UND) from unit DPndd; 1 dedicates or undedicates all devices on IOP n (of yyndd); D dedicates or undedicates only device yyndd. If neither 1 nor D is specified, all devices on the same multiunit controller are dedicated or undedicated.
DISPLAY[LL] ⊛	Display current status of all tasks in the system on the LL or OC device. Output is to the OC device unless LL is specified. A typical display output is shown in Figure 3.
{DM / DF / DB} [from,to][,T][,S] ⊛	Dumps the contents of specified memory onto the device that is permanently assigned to the DO oplabel. DM specifies a real-memory dump with default boundaries being zero and the end of the monitor. DF specifies a real-memory dump with default boundaries being the limits of the first Foreground Private Memory partition. DB specifies a virtual-memory postmortem dump of background with default boundaries as the limits of Task Virtual Memory. DF or DM requests are performed immediately. A DB dump request is not performed until the next task termination occurs in background (either for JCP or any other processor or user program). If "from, to" is absent, the entire default area will be dumped; if present, the first word address in hex and last word address in hex of the selected area are defined. If ",T" is absent, core memory is dumped in hexadecimal; if present, core memory is dumped in hexadecimal and EBCDIC. 'S' specifies the dump will be in a short-line format at four words per line.
DT mo,day,yr,hr,min ⊛	Input of current data and time. Example: DT 8, 17, 69, 22, 30.
ELOG {ON / OFF / PURGE} ⊛	Turns error logging procedures on and off. Eliminates the majority of the execution time overhead associated with error logging but does not prevent gathering of the error statistics for the ESUM display. PURGE clears the error log and all error, log and I/O counts.
{ERRSEND / ESEND} text ⊛	Creates an error log entry containing the supplied text (56-byte maximum).
ESUM[LL] ⊛	Display a device error summary on OC or LL. A typical error summary is shown in Figure 4.

Table 8. Standard Operator Key-Ins (cont.)

Key-In	Purpose
EXTM taskname[,(JOB,jobname)] Ⓜ	Terminate a task in a job. The "taskname" is the name of the task to be terminated. "JOB" is a keyword that indicates that the job under which the task is running will be specified. (If JOB is absent, jobname defaults to the CP-R job.) The "jobname" is the name of the job under which the task to be terminated is running.
FG Ⓜ	Permit loading of foreground program from background job stack for execution via a !RUN or !INIT control command.
INIT fid[,(JOB,jobname)][,PRI] [,STOP][,(PRIO,xxxx)] [,DEBUG][,TS] Ⓜ	Read named task into memory and initiate it. The fid is the file identifier for the load module file. "JOB" is a keyword that indicates that the task is to run under a job other than the CP-R job (if "JOB" is not specified, the task runs under the CP-R job). The "jobname" specifies the name of the job under which the task is to be run. PRI specifies the task is primary and if not specified the task is run as secondary. STOP is an option that specifies the task (secondary only) is to be left in suspended state after load. The default is to execute directly after load. PRIO is a keyword to specify task priority (for secondary tasks only). Default is to run under lowest-priority dispatcher at lowest software priority. This is X'EFEF' for nontimesliced tasks or X'FFFF' for timesliced tasks. xxxx is the task's priority value (in hex). The first 2 characters specify the interrupt level minus X'4F' of the task's dispatcher, and the last 2 characters specify the task's software priority. DEBUG indicates the task is to be initially under Debug control. TS indicates that the task is to be time-sliced.
INTLB label, loc (hex) Ⓜ	Change assignment of interrupt labels.
KJOB jobname Ⓜ	Terminates all tasks in the named job.
PAU [ON / OFF] Ⓜ	If OFF, both the !PAUSE and !ATTEND batch control commands become illegal and any I/O operation which would otherwise have made a request for an operator's key-in to correct an I/O problem is aborted with a TYC4. If ON, only the !PAUSE command becomes legal. This command allows the system to permit or reject batch jobs that depend upon operator interaction (e.g., tape mounting) to continue. Default is ON. See also ATT key-in.
Q [(SLICE,n) / (QMIN,t) / (QSWA,t) / (QMAX,t)] ,... Ⓜ	If no parameters are used, the current settings of the time-slicing controls are reported. These parameters are number of slices per second; and slice length, QMIN, QSWA, and QMAX in milliseconds. If parameters are specified, the corresponding controls are set as specified. The parameter "n" is a decimal number of slices per second that must be between 1 and 500, and must divide 500 evenly. The parameter "t" is milliseconds (decimal) and must be 1 or more.
. Q30 [NORETRY / RETRY / NOPARITY / PARITY / NOWDT / WDT] [,...] Ⓜ	This command is available on a Xerox 550 system only. It affects the contents of the Q30 register as follows: NORETRY: set bit 0 — No instruction retry RETRY: reset bit 0 NOPARITY: set bit 1 — No parity checks PARITY: reset bit 1 NOWDT: set bit 2 — No watchdog timer WDT: reset bit 2 Default is no change for each bit.

Table 8. Standard Operator Key-Ins (cont.)

Key-In	Purpose
Q31 [option,...] ⊜	This command allows simple examination and modification of the address stop controls (Q31 register) of a Xerox 550 system. It is not available on a Sigma 9 system. The "options" are any of the following: (ADDR, x) Set address to "x" (hexadecimal). (Symbol, x) Set address to "x" plus the value of "symbol". "Symbol" may be any symbol found in the symbol table in module CRS2 or any overlay name. (STM, x) Set address to "x" plus X'6000'. (ROOT, x) Set address to "x" plus X'600E'. INST Compare execution access addresses. WRITE Compare write access addresses. ANY Compare any addresses accessed. ALL (Same as ANY.) PAGE Compare page addresses. WORD Compare word addresses. REAL Compare real addresses. VIRT Compare virtual addresses. STOP Stop execution on compare. HALT (Same as STOP.) BEEP Briefly sound audio alarm on compare. CLEAR Reset all flag bits (but save the address). Any bit or field not specified is not changed. If no options are specified, the Q31 register content is typed as a hexadecimal number.
REBOOT ⊜	Forces an operator-initiated system alarm and an automatic reboot of the system. See also ALARM key-in.
RLS taskname ⊜	Terminate a foreground primary program running under the CP-R job.
RSY [jobname]	Removes SY key-in for the specified job, returning to normal software disk write protection. Default for 'jobname' is 'BKG' which is the background job. Note that the SY key-in is automatically removed from background whenever a !JOB or !FIN command is processed.
RUN taskname[, priority] ⊜	Load and execute a foreground program running under the CP-R job. Only primary tasks can be loaded with this key-in. The name of the foreground file to be loaded must be input.
SCHED fid [, (JOB,jobname)][, PRI] [, (PRIO, xxxx)][, (STRT,time)] [, (INTV,interval)][, DELE[TE]] [, TS]	Schedule a task for periodic INIT. The "fid" is a CP-R file identifier with one variation: the file name is actually a task name that may be the same as the file name or related to it by a SETNAME CAL. If neither an account nor an area is specified, the defaults are area FP and the system account. If an area name is specified, the default is the system account. Specification of a whole area is an error. JOB is a keyword indicating that the task is to be INITed under a job other than the CP-R job. The default is the CP-R job. PRI specifies a primary task; the default is secondary. PRIO is a keyword to specify task priority for secondary tasks. Default is to run under lowest-priority dispatcher at lowest software priority. "XXXX" is the task's priority value in hex. The first two characters specify the interrupt level minus X'4F' of the task's dispatcher, and the last two characters specify the software priority. TS specifies a time-sliced task. DELETE specifies that the task is to be de-scheduled. STRT is a keyword indicating that the time for the first INIT to the task is specified. A value of zero or absence of the keyword causes an immediate INIT.

Table 8. Standard Operator Key-Ins (cont.)

Key-In	Purpose
SCHED fid (cont.)	The format of the time values for start time is as follows: [[[[[[yr,]mm,]dd,]hh,]min,]sec] where yr means year (e.g., 76) mm means month number (e.g., 9) dd means day (e.g., 5) hh means hour (e.g., 14) min means minute (e.g., 30) sec means second (e.g., 10) Values are optionally deletable from left to right; e.g., any value may be omitted provided that all parameters to its left are also omitted. Thus, if hh is omitted, yr, mm, and dd must also be omitted. The SCHED key-in uses current date/time values for omitted values. An example of the STRT option is: ...(STRT, 14, 30, 0)... which would cause the specified task to be INITed at 2:30 p.m. of the current year, month, and day. Please note that omission of the 0 (seconds) value would cause INIT at 14 minutes and 30 seconds of the current year, month, day, and hour. INTV is a keyword indicating that the period between INITs is specified in seconds. If the specified value is not an integral multiple of five, it is rounded up to the next highest integral multiple of five. A value of zero or absence of the parameter causes the INIT to be issued once at the specified STRT time. The presence or absence of the STRT and INTV keywords is interpreted as follows: INTV / STRT / Result present / absent / periodic INIT, starting at now + INTV present / present / periodic INIT, starting at STRT absent / absent / INIT once immediately absent / present / INIT once at STRT
SJOB jobname[, (DEBUG, TYndd)] ↵ [, (ACCT, xxx)]	Creates the named foreground job by setting up job controls and table entries but does not initiate any task in the job. DEBUG indicates a Debug control-console device is specified. TYndd is the address used by Debug for communication with the user. ACCT indicates an account is specified. xxx is the account the named JOB is to be associated with.
SNAP[FILE, fid] ↵	Saves core (the monitor) on the specified file or to the SE oplabel by default.
STAT taskname[, (JOB, jobname)]	Output the status of the specified job on OC. The "taskname" defines the name of the task from which status information is desired. The "JOB" is a keyword that indicates that the job under which the task is running will be specified. (If "JOB" is not present, jobname is defaulted to the CP-R job.) The "jobname" is the name of the job under which the task is running. The task status will have the format illustrated in Figure 5.

Table 8. Standard Operator Key-Ins (cont.)

Key-In	Purpose
STDLB label, {fid / device / oplabel} ⊕	Changes a current oplabel assignment. The new assignment will stay in effect until changed by another STDLB key-in or system is rebooted. "label" specifies the oplabel to be assigned, which must have been previously assigned at SYSGEN. "fid" specifies a disk file or disk area. "device" specifies a physical device name, or "0" for the null device. "oplabel" specifies one of the SYSGEN-defined oplabels.
SY[jobname] ⊕	Overrides normal software disk file write protection for the specified job. The default for 'jobname' is 'BKG' which is the background job. The SY privilege is removed by the RSY key in or job termination. For background, it is also removed automatically when a !JOB or !FIN command is processed.
TY ⊕	Transfer control from the C device to OC (typewriter) for reading control commands.
W ⊕	Suspend current background job and enter WAIT state.
X ⊕	Abort current background job. Message on OC and LL will show last location executed.
⊕	Delete this line. (On a Xerox 550, the combination "CONTROL" and "X" may be substituted.)
⊕	Ignore operator key-in request.
←	Delete last character. (On a Xerox 550, a "\" may be substituted.)

Table 9. Terminal Job Entry Key-Ins

Key-In	Purpose
CONTROL yyndd ⊕	Causes a control interrupt (equivalent of Yc) to take place for the named terminal job. It is used for jobs previously created by the LOGON key-in. Active terminal jobs will have been logged on OC as they become active.
LOGON TYndd ⊕	Causes device TYndd to be processed by the logon processor and so perform as a normal TJE user.
OFF ⊕	Prevents any new on-line user from logging on. An OFF key-in and an ON 0 key-in are equivalent.
ON{ALL / n} ⊕	Specifies the number of on-line users allowed on the system at any one time. When n users are on, no additional users are allowed to log on until a current user logs off. All makes all lines available.
SEND{yyndd / ALL}, message ⊕	Causes the text message to be sent to the TJE terminal specified by yyndd, or sent to all active terminal users if the ALL option is specified.

Table 10. Symbiont Key-Ins

Key-In	Purpose
BATCH fid ⓔ	Places the specified file on the symbiont input queue. If background control is in symbiont mode (see 'C' key-in), background is started. A batch file may contain only one background job.
DELETE ident ⓓ	Deletes symbiont files. All input and output symbiont files associated with the specified job ID will be deleted. If the specified job is still active, the DELETE key in has no effect.
DO ⓑ	Sets a switch to cause the symbiont system to delete each one of a job's files from the OS area as soon as it has been output. When the DO key-in is in effect, the R and B options of the Syyndd key-in have the same effect as the C option. If DO has not been keyed in, the switch is set such that all a job's output files are deleted when its last file has been output.
PRIORITY ident, priority ⓒ	Changes the priority of a job in the symbiont area, where "ident" is the job ID and "priority" is the new priority to be associated with the specified job. Priorities are expressed as hexadecimal values from 0 through 7, where 7 is the highest priority. A priority of 0 inhibits selection of a job for execution in the IS area and prevents output from a job waiting in the OS area.
RDO ⓐ	Used in conjunction with the DO key-in and causes deletion of a job's files in the OS area to occur when the job's last file has been output. See DO key-in.
SS ⓐ	Initiates symbiont input when only one symbiont input device exists. The key-in is not allowed if more than one symbiont input device exists. If background control is in symbiont mode (see 'C' key-in), each time symbiont input defines a new job, the symbiont insures that background is started.
Syyndd, option ⓐ	This symbiont key-in gives the operator control of the symbionts, where "yyndd" is the physical address of a symbiont device and "option" specifies the action to be taken and may be one of the following: I — initiate symbiont I/O on the specified device. Output symbionts do not require this key-in as they are self starting unless an "L" or "T" is in effect. S — suspend symbiont activity for the specified device. C — continue symbiont activity for a previously suspended device. B[,n] — continue symbiont activity for a previously suspended device. Before the output is continued, the output file is backspaced n line printer pages. The default will be one line printer page (a line printer page is approximately 37 records). If the device is not a line printer or if the DO key-in is in effect, B has the same effect as C. R — restart symbiont activity for a previously suspended device. Symbiont activity will start from the beginning as if it had not been suspended. If the DO key-in is in effect or if this is an input symbiont, R has the same effect as C. L — lock out the symbiont from future activity after this file. After completing the current file, the symbiont terminates. An input symbiont will terminate when the next !JOB or !FIN card is read. An Syyndd, I key-in is required to restart symbiont activity on the device.

Table 10. Symbiont Key-Ins (cont.)

Key-In	Purpose
Syyndd, option (cont.)	T has the same effect as the "L" option except the device is removed from the symbiont pool if it was not dedicated to symbionts at SYSGEN. Q save the current output file and terminate. What remains of the file is returned to the output queue and the symbiont is locked immediately. The entire file is saved if the symbiont is not in DO mode. When the system assigns a new symbiont output device to the file, the output operation is continued from where it was stopped. Q is useful in moving a file from a down device to one that is working. X release the current job file and begin processing the next job file. If background control is in symbiont mode (see C key-in), each time symbiont input defines a new job, the symbiont insures that background is started.

Table 11. Media Conversion Key-Ins

Key-In	Purpose
MEDIA { I / L / S / X } ⏎	Either to control the operation of the MEDIA task or to specify a MEDIA copy of an input source to an output destination, where control operations are: I = initiate the MEDIA task or resume operation if it has been stopped. L = prevent the start of any new copy operations after the completion of the current operation. S = suspend the current operation. X = abort the current operation and do not post processing.
MEDIA ({ FILE, fid / IN, { op / yyndd } }) [, ({ SFILE, n / ALL / DEL / REW / UNLOAD }), ...] , ({ FILE, fid / OUT, { op / yyndd } }) [, ({ NVFC / SPACE, n / ADD / SFILE, n / WEOF, n / REW / UNLOAD }), ...] ⏎	Copy operations request MEDIA to copy a file on a disk tape, or cards to another file, which may be on a disk, card punch, tape, printer, or, if the input is on a disk, a keyboard printer. One or more options may be given to specify pre- and post-processing of the input and output mediums, where SFILE, n skips a tape n files forward before the copy begins. ALL continues copying successive files until a double end-of-file is found. DEL deletes a disk file after the copy. REW and UNLOAD rewinds (unloads) a tape after the copy. NVFC inhibits use of the first byte in each record as a VFC byte in printed files. SPACE, n causes n lines to be skipped between each printed line when NVFC is given. ADD causes the input file(s) to be added to the end of a tape already containing files. WEOF, n writes n EOTs on the output tape after the copy.

The display has the following format:

```
PRI  TASK     TASK TASK   JOB    TASK   TASK
SEC  NAME     PRIO STAT   NAME   FWA    LWA
 S CTRLTASK   FEFE  10   CPR     00000 03E4D
 S MMEXEC     FFFF  20   CPR     06000 1DFFF
 S BKG        FFFF  20   BKG     06000 1DFFF
 P CTRLTASK   FEFE  80   CPR     00000 03E4D
```

where

P indicates a primary task.

S indicates a secondary task.

TASK PRIO is the hexadecimal priority value associated with the task.

TASK STAT is a representation of the task's status, as follows:

80 - task is primary
40 - task is rolled out
20 - task is stopped
10 - task is in execution
08 - task is in initialization
04 - task is suspended
02 - task is time-sliced
01 - task is swapped
00 - task is executable

FWA and LWA stand for first-word address and last-word address respectively.

Figure 3. Display Format

The analysis and subsequent action from an operator's key-in is performed at the Control Task priority level. If the operator key-in is not recognized as a valid input, the following message is output on the OC.

```
!!KEY ERR
```

In which case, the operator should retype the input correctly. Note that if the typewriter is busy at the time of the Control Panel Interrupt (i.e., waiting for an input to complete), the operator must complete the input before the system will output the prompt character.

COMBINED KEY-INS

To expedite operator key-ins, the following combinations of key-ins are recognized:

Combined Form	Result
FGC	Execute FG and C key-ins.
SYC	Execute SY and C key-ins.
SFC, FSC	Execute FG, SY, and C key-ins.
TYC	Execute TY and C key-ins.

```
09:23     OCT 25,'73
YYNDD     MDL#     ACCESSES      ERRORS      ERR/1000

TYA01     7012           76           0             0
LPA02     7445         1037           0             0
CRA03     7140          110           0             0
CPA04     7160           13           0             0
9TA80     7322         1760           0             0
9TA81     7322            0           0             0
9TA82     7322          273           6            21
9TA83     7322            0           0             0
9TAD1     7333            0           0             0
9TAD2     7333            0           0             0
7TAE0     7372            0           0             0
7TAE1     7372            0           0             0
DPDF0     7242        25752           0             0
DPDF1     7242            0           0             0
DPDF2     7242            0           0             0
DPDF3     7242            0           0             0
DPBE4     7275            0           0             0
DCBF0     7212            0           0             0
DCCF0     7232         1651           0             0
DCCF1     7232            0           0             0
DCCF2     7232            0           0             0

    6 FILED LOGS,        0 LOGS LOST
```

where

YYNDD and MDL# correspond to parameters defined on the :DEVICE control commands input at SYSGEN.

ACCESSES is the number of SIOs issued for each job.

ERRORS is the number of error retries and error completions for each device.

ERR/1000 is the error rate computed as follows:

ERR/1000 = (ERRORS*1000)/ACCESSES if ACCESSES > 0.

ERR/1000 = 0 if ACCESSES = 0.

FILED LOGS is the total number of error log entries that have been successfully filed.

LOGS LOST is the number of log entries lost because error log filing could not take place.

Figure 4. Error Summary Example

STATUS xxxxxxxxxxxxxxxx PRIORITY xxxx
(binary) (hexadecimal)

where binary status bits are as follows:

Bits	Value	Meaning
0	1	Task in final termination.
1	1	Task connected to CAL2.
2	1	Task connected to CAL3.
3	1	Task connected to CAL4.
4	1	Background task.
5	1	Secondary task.
6	1	Task being aborted.
7	0	Task initiated via RUN.
	1	Task initiated via INIT.
8	1	Load to be performed.
9	1	Public Library used by primary tasks.
10	1	Public Library used by secondary tasks.
11	1	Release to be performed.
12	1	TEL control requested.
13	1	Task is loaded.
14	1	Task is run queued.
15	1	Break control requested.

Hexadecimal priority characters are as follows: First two hexadecimal characters correspond to interrupt level minus X'4F'. For secondary tasks this is the level of the dispatcher for the task. Last two hexadecimal characters are the software priority of a secondary task, or zero for a primary task.

Figure 5. Task Status Format

DEVICE CONTROL

If the system encounters an abnormal condition during an I/O operation, a pertinent message to the operator is output on the OC device. Such a message is of the form

```
!! name message
```

where

name is the physical device name, yyndd, or the disk file name.

message is the message string informing the operator of the specific condition that has been detected; for example:

ERROR (error was detected on operation)

or

MANUAL (device not ready)

I/O messages are discussed below, grouped according to the type of device to which they apply.

I/O KEY-IN FORMAT

After correcting the abnormal conditions, the operator responds by means of a key-in. The format for an I/O key-in is

yyndd a ↵

where

yyndd is the physical device name of the device involved in the I/O operation.

a is a single character that requests a system action relative to the device, as follows:

- C Continue "as is".
- E Inform the user program of the error and transmit record "as is".
- R Retry the I/O operation.
- X Abort the pending I/O.

↵ is the NEW LINE code.

CARD READER MESSAGES/KEY-INS

If the card reader fails to read properly, or if a validity error occurs, one of the following messages is issued:

!!CRndd ERROR

!!CRndd CARD NOT FED

A FAULT indicates that the error condition occurred prior to any data being transferred; an ERROR indicates that at least one byte was read. After correcting the condition, the operator responds with an I/O key-in message. The action character selected depends on the circumstances causing the error condition.

If a feed check error or a power failure occurs, CP-R outputs one of the following messages (depending on where in the read cycle the error took place):

!!CRndd ERROR

!!CRndd CARD NOT FED

!!CRndd TIMED OUT

If the card in the hopper is damaged, the operator replaces it with a duplicate, presses the RESET button on the card reader, and responds with one of the following key-ins:

CRndd R ⓃⓁ

CRndd C ⓃⓁ

In the event of a power failure, the operator presses the RESET button on the card reader and responds with the key-in:

CRndd R ⓃⓁ

If the card stacker is full, if the hopper is empty, or if the device is in the manual mode, the following message is issued:

!!CRndd MANUAL

The operator corrects the condition and then presses the START button on the card reader.

CARD PUNCH MESSAGES/KEY-INS

Instead of outputting an error message when a punch error is first detected, the I/O handler attempts to punch a card x times (x = NRT, a DCB parameter specified by the user) before the following message is issued:

!!CPndd ERROR

This message indicates that the card punch is not functioning properly and the operator should reevaluate the job stack based on this knowledge. Improperly punched cards are routed to an alternate stacker.

If the input hopper is empty, the stacker is full, or the chip box is full (some devices), or if the device is in the manual mode, the following message is issued:

!!CPndd MANUAL

The operator corrects the condition and presses the START button on the card punch.

If a power failure or a feed check error occurs, the system outputs one of the following messages (depending on where in the cycle the error took place):

!!CPndd ERROR

!!CPndd TIMED OUT

If the card in the hopper is damaged, the operator removes it, presses the RESET button on the card punch, and responds with the key-in

CPndd R ⓃⓁ

In the event of a power failure, the operator presses the RESET button on the card punch and responds with the key-in

CPndd R ⓃⓁ

DISK PACK MESSAGES/KEY-INS

If the operator enters the key-in

DED DPndd,R ⓃⓁ

and there are no open files on the specified disk pack, CP-R outputs the message

!!IDLE

If there is at least one open file on the specified disk pack at the time the DED key-in is performed, CP-R will output the message

!!DPndd IDLE

when there is no longer an open file on the indicated disk pack. The operator may now remove the pack from the indicated unit, insert a different pack, and key in

UND DPndd,R ⓃⓁ

to allow use of the new pack.

DISK DATA PROTECTION

Software protection of the data on disk storage is provided on disk file and area accesses. Areas with write protection code 'S' (system) may not normally be written by any user program. Areas with protection 'F' (foreground) can not normally be written by background programs. Disk area protection is specified when the area is defined by SYSGEN. Write protection does not normally apply to device-access disk operations, but this type of access is normally permitted only to foreground programs. All software restrictions on disk access may be overridden for a specified job (including BKG, the background job) by use of the SY key-in. The message

!!yyndd WRT RESTRICTED

or

!!PAUSE KEY-IN SY

(if included in the background command stream) will be output on OC to inform the operator that access to a protected disk area is requested. The operator would not normally grant system privileges (key-in SY) unless he was assured that it was authorized for the requesting job.

LINE PRINTER MESSAGES/KEY-INS

When an irrecoverable print error is detected, the system outputs the following message:

!!LPndd ERROR

The I/O handler attempts to print a line x times (x = NRT, a DCB variable specified by the user) before outputting the above message. The operator's response after correcting the condition depends on the specific device and circumstances.

If the printer is out of paper, if the carriage is inoperative, or if the device is in the manual mode or off, the following message is issued:

!!LPndd MANUAL

The operator corrects the condition and presses the START button on the line printer.

If the line printer power is off, the system outputs the following message:

!!LPndd UNRECOG

If a printer went into test mode during an I/O operation, the following message is issued:

!!yyndd TEST MODE

The operator should correct the condition and respond with the key-in

LPndd R ㊓

If a printer became nonoperational during an I/O operation, the following message is issued:

!!yyndd NOT-OPERATIONAL

The operator should correct the condition and respond with the key-in:

!!yyndd R ㊓

MAGNETIC TAPE MESSAGES/KEY-INS

If an error occurs during the reading or writing of magnetic tape, the I/O handler attempts a recovery x times (x = NRT, a DCB variable). If the error is irrecoverable, the user is informed via an error return.

If a tape unit is addressed and there is no tape mounted or power is off, the following message is issued:

!!9Tndd UNRECOGNIZED

If an attempt is made to write on a tape unit without a write-permit ring, the following message is issued:

!!9Tndd WRITE PROTECTED

The operator's key-in response depends on the circumstances.

4. INPUT/OUTPUT OPERATIONS

The CP-R I/O system provides the user with the capability of performing input/output operations on standard Xerox peripheral devices. An I/O request is made through execution of a CAL1 instruction that addresses a Function Parameter Table (FPT), which in turn is a list of parameters that define the request. The FPT addresses a Data Control Block (DCB), which is a list of parameters that define the nature of the data file. The DCB then addresses a Device Control Table (DCT) entry or a RAD File Table (RFT) entry, depending upon whether the data file concerned is associated with a peripheral device or with a disk file. The DCT entry contains the device status parameters and the RFT entry contains the disk file parameters.

The CAL1 instruction and FPT must be generated at assembly or compilation time. Symbol, Macro-Symbol, or AP users must include both the CAL1 and the FPT in the source code. For FORTRAN users, the compiler generates the necessary CAL1s and FPTs.

For users of the Xerox AP processor, a set of high-level I/O procedures is provided. These procedures translate, at assembly time, to the requisite CAL1 instructions and appropriate FPTs (via assembly-system CPR). The procedure calls for I/O and other types of system services are described in Appendix A.

All DCBs are given names beginning with M: for system DCBs or F: for user DCBs. The DCBs may be included in the source code if desired. If not included, the Overlay Loader generates the DCBs necessary to satisfy any unsatisfied references to F: or M: DCB names. System DCBs generated by the Loader have default parameters; user DCBs generated by the Loader are left blank.

The correspondence between a DCB and a device or file can be established by using the !ASSIGN control command or ASSIGN service function. Other DCB parameters describing the data file may also be set by the !ASSIGN control command or by the DEVICE/FILE Mode service function.

Two types of Read/Write requests are provided. Type I requests have the completion status posted in the DCB. The disadvantage of this type of I/O operation is that a DCB cannot be shared among requests in different tasks because, in general, it is impossible to associate the completion status in the DCB with a specific request. For this reason TYPE II requests are provided.

Type II requests result in the completion status being posted in the FPT associated with the request. This enables several requests (perhaps in several tasks) to be in progress simultaneously on a given DCB. Type II requests require that the associated FPT must be in memory and not in a register.

The CHECK function tests for the completion of READ/WRITE requests that are performed without waiting for completion. In no-wait requests, the CHECK function must be used to cause the DCB or FPT to be posted with the completion code and actual record size.

PERMANENT FILES

Permanent files are defined through RADEDIT by use of the :ALLOT command or through the ALLOT service call. Data can be entered through RADEDIT or any program that uses the system I/O. At definition time, the following file parameters are given by the user:

File name (maximum eight characters).

Disk area (optional).

Disk file account (optional).

File organization (blocked, unblocked, compressed).

Record size (for blocked or unblocked files to be accessed sequentially).

Granule size (for files to be accessed directly).

File size.

In systems which do not include the disk file account option, the user must specify the area name on all disk file references, but may not use an account name. In systems which include the option, either the disk area name or the disk file account name or both may be omitted and defaults will be provided. In most cases, if an account name but not an area name is specified, the file will be defined/found in one of the public areas on the system (at least one of which must have been defined by SYSGEN, in order to support this default). If an area name is specified, but not an account name, the system account name is the default. If neither area nor account name is specified, the defaults are the account of the user and one of the public areas. These defaults allow several ways to chose between simplicity and detailed control in dealing with the disk file data base.

TEMPORARY FILES

Temporary files are in the Background Temp area and have the fixed names X_i ($1 \leq i \leq 9$), GO and OV. The size for these files can be set by using the !ALLOBT control command. If no !ALLOBT control command appears within a user job, the files assume default sizes that are set by the Job Control Processor. The files X_i should be considered as primarily for temporary use within a single job step, since they are all allocated from a single area with X_{i+1} beginning just above X_i. Therefore, changing the size of a file X_i can cause a change in location of files X_j for $j > i$. GO and OV are allocated from the top of the temporary file area downward. A change in the size of files X_i therefore has no effect on the position of these files on the disk.

Since the size and location of temporary files can be changed through background job control commands, they must not be used by foreground programs.

FILE ORGANIZATION

BLOCKED FILES

Blocked files contain fixed length records whose length is less than or equal to 128 words. In blocked files, the largest possible integral number of records is combined into 256-word blocks. These blocks are basic units of data transmitted to and from the disk. As sequential READ requests are made to a blocked disk file, the blocks are read from the disk into blocking buffers as necessary, and the data records are transmitted to the user's input buffer.

Blocked organization is specified for a file when the file is defined. A file specified by the user as blocked, but having a record size greater than 128 words, will be given unblocked organization.

UNBLOCKED FILES

Unblocked files contain fixed length records. Each record begins on a sector boundary and requires some integral number of sectors that is the smallest possible integral number that can contain the record.

COMPRESSED FILES

Since many blanks occur in typical programming-language source code compression of EBCDIC data in disk files is accomplished by the removal of blank characters. Compressed files are blocked into 256 word blocks on the disk and the records are of variable length. No record crosses a block boundary. Special codes, imbedded in the compressed records, allow for proper decompression.

DISK ACCESS METHODS

SEQUENTIAL ACCESS

The sequential access method provides record-by-record access to the file in the same way that a data file on magnetic tape is accessed. A sequential access READ/WRITE request results in the next record in sequence being read or written. Sequential access can be used on blocked, unblocked, or compressed files.

DIRECT ACCESS

In the direct access method, the user furnishes the relative granule number of the start of the READ/WRITE request and the number of bytes to be transferred. The user is responsible for the organization of the file, including discrimination of logical records, maintenance of a key structure within the file, etc. Addressing files by granules allows the direct access method to be independent of the disk sector size. Granule size is specified by the user at file creation. Each granule begins on a sector boundary.

The user is not restricted to I/O operations whose length is less than or equal to the granule size. For requests of length greater than granule size, the I/O system transfers the requested number of bytes to or from the disk starting with the granule specified. An entire area may be treated as a single direct access file by using a file name of 0.

DEVICE ACCESS

In the device access method, the user reads or writes through a DCB which is assigned directly to a disk as a device. The assignment may be done with a standard ASSIGN command or with the DEVICE Set Index function. The "key" parameter, which must be present in the READ/WRITE FPT, is treated as a sector number which is relative to the absolute start of the disk. This method allows direct (random) access to any sector on the disk (except the flawed track pool). All other parameters in the READ/WRITE FPT are treated as they would be for any READ/WRITE call.

In this method, the disk may have areas defined on it, but having areas is not a requirement. Note that no error checking is provided which would prevent users of this method from interfering with other users who may be accessing the disk.

DISK PACK FILES

Although files on disk packs are logically and functionally identical to files on fixed-head disks, the physical characteristics of disk packs require that the I/O be treated differently in certain cases.

Initially, all transfers to or from a disk pack file are attempted as one complete I/O operation. If the hardware signals that the transfer encountered a flawed track, then the single I/O transfer will be broken into several I/O transfers. Each of these will be confined to no more than one track, thus allowing processing of alternate tracks for flawed-tracks. All such data transfers will be treated as follows:

1. The byte count will be truncated to end on the first track boundary.
2. The data transfer to the track boundary will be done and any flawed-track processing will be performed.
3. If the residue of the operation does not cross another track boundary, the operation will be completed as originally requested; otherwise, steps 1 and 2 will be repeated until the residue does not cause transfer over a track boundary.
4. If a flawed track is encountered, the disk pack error-recovery routine will read the header of the flawed track to determine its alternate. The data transfer will then be performed with the assigned alternate track.

   If the initial transfer encounters a cylinder boundary, the I/O request is modified to account for the data already transferred and the I/O operation is continued on the next cylinder.

EXTENSIBLE FILES

When a permanent disk file is ALLOTed as extensible (see ALLOT call and :ALLOT RADEDIT command), an extension to the file will be automatically allocated when an end-of-file condition is detected during a WRITE operation to the file. The WRITE request resumes using the new extent.

The extents will have the size specified when the file was ALLOTed with one exception. A direct access WRITE to an extensible file will cause an extent to be allocated which is large enough to contain the record but no smaller than the specified extent size. The user may request, however, that the extents be limited to the specified extent size by use of the FIX option on the :ALLOT RADEDIT command or by setting the F5 parameter in the ALLOT CAL.

If an end-of-data (EOD) is detected when reading an extensible file, a switch to the next extent in sequence occurs and the READ continues using the newly selected extent.

When a READ/WRITE is requested with no-wait, intermediate file selection or allocation is done with wait. However, the actual reading or writing of the data record is done without waiting.

I/O QUEUEING

The I/O system provides for queueing of all requests to I/O devices. That is, any I/O request (READ, WRITE, REW, IOEX, etc.) requiring a device to be accessed results in the request for the specific access being queued.

Device requests are queued on a controller basis (one queue per controller), and they are queued in order by priority of the task making the request. For example, a READ request to a card reader will be placed in the queue for the specified card reader controller, and its position in the queue is determined by the priority of the requesting task and the relative priorities of the requests already in the queue. Requests for a designated device from a specified priority level are queued by order of occurrence. The queues are chains of entries representing requests for actual I/O operations on devices. There is a single pool of free entries for all devices, and these entries are removed from the pool and linked to the controller queues as needed. The queue entry is returned to the free entry pool when a queued request is completed.

By using assembly options, the system may include queuing routines that optimize I/O transfers to disk packs and fixed-head disks. In the case of disk packs, the optimization minimizes arm motion when there are more than two requests in the queue. In the case of fixed-head disks, the optimization minimizes rotational latency when there are more than two requests in the queue.

At System Generation, the user may specify the maximum number of entries to be used for background requests to ensure that the background does not tie up all the queue entries, thus causing foreground requests to wait. Whenever a request is made and the free entry pool is empty (all queue entries in use), the request is made to wait until an entry is freed.

I/O CLEANUP AND I/O START

I/O Cleanup is the data processing performed between completion of the actual data transmission (signaled by occurrence of the I/O interrupt) and the completion of the request. It includes such functions as error testing, setup for error recovery, posting of completion status in the ECB, setting of indicators in the DCT, dequeuing the completed request, etc.

I/O Start is the operation of starting a device for the next request.

Under the CP-R I/O system, CPU time may be taken from a task to perform data processing for lower priority tasks. I/O Cleanup and I/O Start functions are performed at the various times and priority levels given below:

Depending upon SYSGEN options, I/O Cleanup may be done in one of three ways:

1. I/O Cleanup may be done at the I/O interrupt level. This assures fast I/O service but may delay service to real-time tasks that are connected below the I/O interrupt level.

2. I/O Cleanup may be deferred to a specified external interrupt level. This allows fast response to real-time tasks running at levels higher than the deferred level yet, giving maximum I/O service to lower priority tasks.

3. I/O Cleanup may be deferred to user or CP-R dispatcher levels at all times. In this case, I/O Cleanup is done upon:

   a. Entry to any CP-R dispatcher (all channels),

   b. For any I/O service performed with WAIT (only for requested channel), or

   c. For any I/O Start request (only for requested channel).

I/O Start is done either at the time the I/O request is made or following I/O Cleanup when there are still I/O requests pending for that channel.

SHARING DCBs AMONG TASKS

DCBs can be shared among several primary tasks within a given load module subject to the restriction that no task can make a Type I request on a DCB that is busy with another Type I request.

DCBs explicitly referenced by the user are allocated and created by the user either in the source code (for assembly language), the compiler, or the Overlay Loader. This means that each program has a private copy of all DCBs explicitly referenced, and no DCBs are shared among programs. The user program has the responsibility for coordinating the Open and Close functions for DCBs shared among primary tasks within a program. An Open request results in the DCB being opened if it is not already open. A Close request causes the DCB to be closed if it is not already closed. No attempt is made to balance Open and Close requests for a DCB to determine which Close request should actually cause the DCB to be closed.

SHARING I/O DEVICES AMONG TASKS

Any number of tasks within a given program can share any device by sharing a DCB assigned to the given device. For sequential type devices (i.e., card reader, card punch, line printer, magnetic tape, responsibility for positioning and/or determining the position of the device is left to the user. No attempt is made to analyze a request on a DCB to determine which task has made the request.

Sequential output devices (i.e., card punch, line printer, magnetic tape), can be shared by tasks (possibly in different programs) that use different DCBs. The sharing of output devices by different programs using different DCBs could be used for logging error conditions or alarms.

Sharing of sequential input devices (i.e., card reader, magnetic tape) should be accomplished through real-time requests on a single DCB. For example, a background user who wishes to use double buffering on a card reader can do so by using two real-time Read requests with two different FPTs.

Random access devices such as disks can be shared, using direct access, by tasks within different programs using various DCBs. The sharing can be performed without restriction other than those restrictions normally imposed on tasks sharing a DCB.

As DCBs are opened and closed, a count of the DCBs that are open and assigned to a device is kept. This count is incremented for every open request on a DCB assigned to the particular device, and is decremented for each Close request.

SHARING DISK FILES AMONG TASKS AND JOBS

Any number of primary tasks within a load module can share a disk file by sharing a DCB assigned to the file (subject to the conditions placed on tasks sharing DCBs discussed previously). A disk file shared in this manner can be accessed either sequentially or directly. Input/output requests are allowed.

Any number of primary tasks within a load module and any number of load modules within a job can share a disk file using different DCBs with the restriction that no sequential input requests can be allowed on a file shared in this manner. A count of the number of DCBs opened and assigned to a disk file is kept for each file. If the count is greater than 1, no sequential input from the file is allowed. The user programs have the responsibility of coordinating accesses to disk files shared in this manner.

Disk files can be shared by tasks in different jobs if certain procedures are followed. The file is made sharable by assigning an operational label to the file using the STDLB key-in. Executing a STDLB call from a task running in the CP-R job will accomplish the same thing. The file should not have been open at the time of the STDLB request. Each active job will now have the operational label assigned to the file (if the operational label assignment was not previously changed). Access to the file is done through a DCB assigned to the operational label. Access will be denied to a blocked or compressed format file which currently has a blocking buffer tied to it if it is determined that the blocking buffer is not in the caller's context. The restrictions that apply to the sharing of a file within a job also apply to the sharing of a file by tasks in different jobs.

An Open request on a DCB assigned to a disk file results in opening of the file if it is not already open. A Close request on such a DCB results in closing of the file only if the "Open DCB Count" for the file is 1 and no operational labels in any job are assigned to the file.

I/O END ACTION

Primary tasks may use I/O end-action. Three types of end-action are possible.

1. The user provides an end-action address in the FPT. A transfer to this address will be made following the occurrence of an I/O interrupt that signals completion of the data transfer. This end-action transfer is made by executing.

   BAL, 11 end-action address

   with the CPU in master mode, the I/O cleanup level active, and registers 5 through 9 loaded as follows:

5	AIO status
6	Upper halfword of TIO status, right-justified in register
7	Device type (DCT) index
8,9	TDV status (doubleword)

   Return from the end-action routine must be made by

   B *11

   It should be noted that since end-action may be performed with the I/O cleanup level high, all tasks whose priority is lower than that of the I/O cleanup level are effectively disabled for the duration of the end-action.

Since the end-action user can seriously degrade interrupt response for lower priority tasks, it is strongly recommended that this type of end-action not be used for applications where other techniques are satisfactory.

2. The user FPT contains either an interrupt number or interrupt label specifying a system interrupt. The system interrupt is triggered upon occurrence of an I/O interrupt that signals completion of the request (this interrupt will be triggered before the I/O interrupt is cleared). The task connected with the specified interrupt then performs the end-action function at the proper priority level. The user is responsible for connecting the interrupt and ensuring that it is armed and enabled.

   The I/O system sets a flag in the TCB to indicate that the trigger has been performed. The EXIT routine interrogates this flag before performing the EXIT for centrally connected tasks. If the flag is set, the occurrence of the interrupt (previously lost by the triggering of an active interrupt) will be simulated. If more than one I/O interrupt can occur while a task is active (or prior to such a task becoming active), the user is responsible for checking all possible devices from which an interrupt may have occurred. (The flag bit in the TCB indicates one or more interrupt attempts.) Directly connected tasks using this type of end-action must assume the responsibility for solving problems of this type since there is no TCB in which a flag bit can be set.

3. The user FPT contains the address of a location to contain the AIO status following the occurrence of an I/O interrupt. The user should initialize the signal location to zero before executing the I/O call. This type of end-action is useful when multiple-device I/O operations are being performed and the slower I/O is controlled by end-action types 1 and 2.

   Note that no end-action is taken for requests on a blocked or compressed disk file.

RESERVING I/O DEVICES FOR FOREGROUND USE

I/O devices can be reserved for exclusive use of the foreground program system through SYSGEN input, operator key-in, or through a system call from a foreground program. Reservation can be made either for a specific device, for all devices on a controller, or for all devices associated with a given IOP. When a device is reserved, it is specified that either all foreground requests will be allowed or no requests will be allowed.

Device reservation results in all background requests to the device being held in abeyance until the device is released for background use. The background user program is unaware that execution is suspended.

A count is kept of the number of reservations (STOPIO requests) of each type (either all foreground I/O allowed or no I/O allowed) for each device. As devices are reserved, the proper count is incremented; and as they are released, the count is decremented. A value greater than zero indicates that the device is reserved. The user must balance each STOPIO request with a STARTIO request so the system can maintain order.

When I/O requests are received by the system, the requests will be queued. Any request queued but currently not allowed will not be started.

The foreground user can specify in a STOPIO request that the in-process operation on the specified device be aborted through execution of an HIO.

DEVICE PREEMPTION

A user may preempt a device (receive I/O interrupt control) with the STOP ALL SYSTEM I/O or DEACTIVATE I/O system calls by specifying the end-action parameter on the call. All succeeding I/O interrupts cause the specified end-action to occur. A device may also be preempted by dedicating it to IOEX at SYSGEN or by use of DEDICATE yyndd, X key-in. Note that in the latter two cases, the monitor handles the interrupt.

DIRECT I/O EXECUTION (IOEX)

CP-R provides primary tasks with the capability of programming I/O devices by executing TIO, TDV, HIO, and SIO instructions with user supplied IOP command doublewords.

An SIO request will be placed in the I/O queue if the specified device is not preempted. At return from the call, the condition codes and status are set to simulate a successful SIO instruction. True hardware status for the I/O request may be obtained by the user by specifying BAL type end-action on the SIO request. If the SIO request cannot be queued for any reason, the condition codes and status are set to simulate an SIO failure. Note that queued IOEX requests must cause only one I/O interrupt to occur during the execution of the channel program.

If the device was preempted at the time of the request, the SIO instruction is executed and true hardware status and condition codes are returned to the user. If end-action is specified on the request, device preemption end-action is overridden.

An HIO request will use monitor routines to stop I/O activity on the device if it is not preempted. At return from the call, the condition codes and status are set to simulate a successful HIO instruction. If the device was preempted at the time of the request, the HIO instruction is executed and true hardware status and condition codes are returned to the user:

TIO and TDV instructions are always executed and true hardware status and condition codes are returned to the user.

KEYBOARD-PRINTER EDITED I/O

Data transfers involving a keyboard-printer (TYndd) are normally subject to editing. On output, this means that each record is automatically followed by a ⟨CR⟩ and ⟨LF⟩, and if VFC is set, the first character of the line is omitted. On input, editing is more complicated. First, before the input is requested, a prompt character is output. The prompt character is obtained from the Job Control Block (JCB) of the calling task's job, unless it is specified in the READ call. When the record has been read, it is scanned. Each time a "¢" is encountered, it and the prior character are deleted. If the record ends with an EOM, it is discarded and reread. Trailing carriage returns or line feeds are deleted, and the record is extended with blanks to the requested byte length.

The prompt character in the JCB defaults to no prompt. It can be reset using the PC (Prompt Character) service call.

Editing is suppressed by setting the DRC flag in the DCB.

LOGICAL DEVICES

Provision is made in SYSGEN to include logical devices. These are pseudo-devices which form a logical connection between read and write or write file mark I/O requests. They are SYSGENed as if they were real devices (including fictitious device addresses), and may be used as any other I/O devices.

Read and write requests are entered into the I/O queue normally. When the logical device finds a match between a read and a write request, the data transfer is made directly from the write buffer to the read buffer. Requests are handled on a first in, first out basis within priority and otherwise in order by priority. Actual record sizes are posted as usual. Write file mark requests result in an EOF TYC and abnormal code being posted for the read request.

Logical devices supply the capability of communicating between tasks via normal read/write services. It also provides the capability of intercepting or monitoring a data stream.

The user should be aware that I/O buffers are locked in main memory during any I/O operation, and that where very large buffers or very many outstanding I/O requests are used, this may result in a deadlock. This is particularly true of logical device requests which must be satisfied by another I/O request and not by independent action by a peripheral. Similarly I/O queue requests may be tied up and result in a deadlock condition.

Logical device requests are not subject to I/O timeouts. If desired, the user may supply a time interval parameter on the service request (P13). This will cancel the request after the specified time period and post a FPT error code of X'67'.

OPERATIONAL LABELS

Operational labels are used to lend flexibility in the assignment of DCBs to peripheral devices. Operational labels are logical I/O media whose relation to a device or file can be controlled at several levels.

Operational labels are assigned to devices or disk files. Each operational label has a permanent assignment and an assignment in each active job. When a job is created, its operational label table is derived from the permanent operational label table. Assignment of operational labels to devices or disk files is made in the following ways.

1. At System Generation, permanent assignments are made and remain in force until changed through STDLB key-in. The original system assignments are reinstated whenever the system is again booted and initialized.

2. The STDLB key-in can be used by the operator to change the system assignment of an operational label, which will result in a corresponding change in the operational label table for each active job, unless the assignment was previously changed within the job with a STDLB call.

3. The !STDLB control command can be inserted by the user to change the assignment of an operational label in the current background job. This change is in effect until the next !JOB command is encountered.

4. The STDLB service call may be used to change the assignment of an operational label in the user's job. Operational label assignments in other jobs are not affected unless the call originated from the CP-R job. In this case, the call has the same effect as an STDLB key-in.

At System Generation, the user may specify any number of optional operational labels with the provision that the optional labels be two characters in length. An entry is built in the operational label table for each optional oplabel, and each entry requires four bytes of system residence.

DATA CONTROL BLOCKS (DCBs)

DCB CREATION

The Overlay Loader creates the DCBs for FORTRAN programs that reference the standard FORTRAN operational labels 101 through 106, and 108 for their I/O requests. For other labels, the user must create DCBs using !ASSIGN control commands and machine language subroutines.

DCBs for assembly language programs are allocated and defined in the following ways:

1. User Created DCBs: The user may create his DCBs in the source code. The parameters defined at source time may be overridden by !ASSIGN control commands if the user follows the convention of defining the name of a DCB and beginning the name with F:.

   Warning: DCBs will not receive any memory protection, and assembly language users should exercise extreme care to prevent accidental alteration.

2. Loader Created DCBs: At the conclusion of the object module load and the library search and load, the Loader creates DCBs for any unsatisfied REFs beginning with M: or F:. For REFs to system DCBs, defined in Table 12, a copy of the standard DCB is included in the root portion of the load module. This DCB contains standard system parameters, including standard assignment to a system operational label. For example, M:LO is assigned to operational label LO. User DCBs (F:) are included in the load module but their parameters are left blank. The background user must define the parameters for F: DCBs through Overlay loader control commands at load time or !ASSIGN at run-time. Definition and assignment of F:DCBs in foreground programs should be made through Overlay Loader control commands.

3. !ASSIGN Command Created DCBs: !ASSIGN control commands can create DCBs in addition to defining or redefining parameters in existing DCBs. This DCB creation facility enables FORTRAN IV and FORTRAN IV-H programs to perform I/O using variables as operational labels. At run-time, the FORTRAN program evaluates such variables, converts the variable value to a DCB name and locates the DCB. For example, a FORTRAN variable with value 101 would result in an I/O operation using DCB F:101. The DCB must have been created in an assembly language subroutine or through an !ASSIGN control command.

DCB ASSIGNMENT

Most of the fields of a DCB can be modified at any stage in the creation and use of a program. Values may be assembled into user-created DCBs. They can be modified when the program is linked, using the Overlay Loader :ASSIGN command. For background programs, DCBs may be altered by the JCP !ASSIGN command when the program is loaded for execution. During execution, DCBs may be changed by Device Mode services. Those fields not subject to these means of modification are maintained by the monitor, and should not be changed by user code.

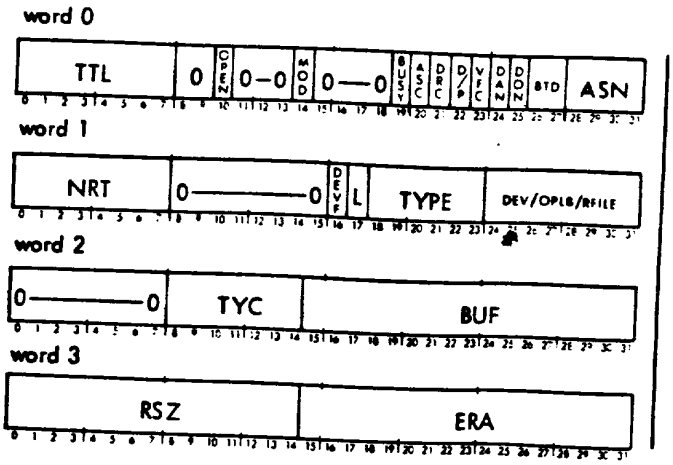
Table 12. System DCBs

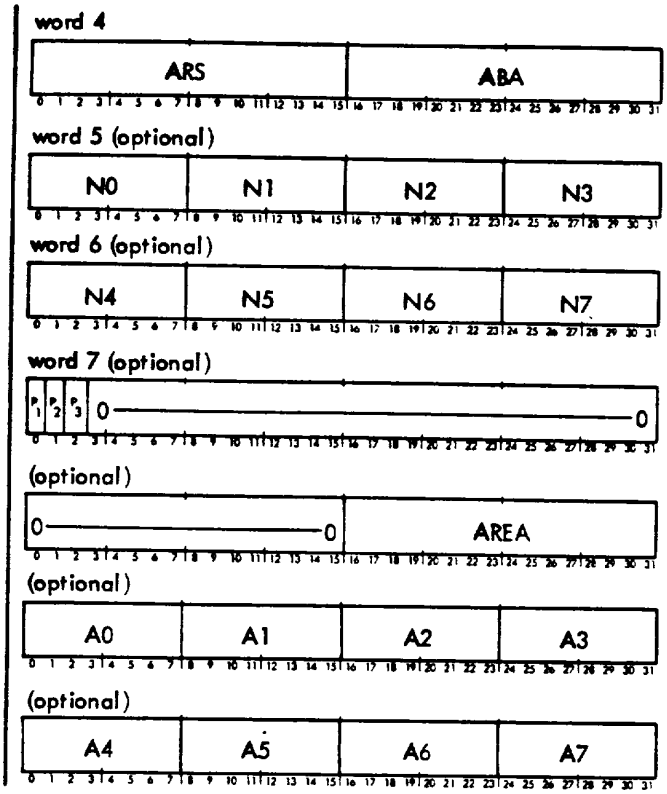
DCB Name	Op Label or Disk File Assignment	Comments
M:C	C	The first 12 DCBs are assigned to the standard operational labels.
M:OC	OC	
M:LO	LO	
M:LL	LL	
M:DO	DO	
M:CO	CO	
M:BO	BO	
M:CI	CI	
M:SI	SI	
M:BI	BI	
M:SO	SO	
M:PL	PL	
M:Xi(1≤i≤9)	Xi.BT	DCBs for Background Temp scratch file.
M:GO	GO.BT	DCB to write on GO file.
M:OV	OV.BT	Output DCB for Overlay Loader.
M:SL[t]	Appropriate program file	Input DCB for Segment Loader.

[t] The M:SL DCB does not have to be referenced by a program using overlays, since this DCB is automatically furnished by the Overlay Loader for any program with overlay segments.

DCB FORMAT

The format for a Data Control Block is given below:





where

Word 0

TTL specifies the total length set aside for the DCB. It must be five or greater (although there is currently no way to use any space past eleven words). It need not all be used.

TTL should be set when the DCB space is reserved. No system services modify it.

OPEN is the DCB open indicator. It must be set to zero before the DCB is opened. The I/O system sets the indicator to 1 when the DCB is opened.

MOD is the mode flag (0 for EBCDIC mode; 1 for binary). This flag has meaning only for I/O requests to COC lines, 7-track magnetic tape, card punch, or card reader. For requests to read a card reader, Mode flag 0 causes a Read Automatic.[†] Input from a card reader designated as the C device is always performed in automatic mode (mode flag is ignored). For COC lines MOD = 1 and DRC = 1 implies transparency (no EBCDIC conversion, no N/L, or other line or timing control characters).

BUSY is the DCB busy indicator that is set and maintained by the I/O system to indicate that a Type I request using the DCB is in progress. Any Type I request using a DCB that is made when the DCB is busy will result in an error.

ASC is the indication of ASCII mode tape operation for magnetic tape drives having program-controlled ASCII translation (0 indicates no translation; 1 indicates translation). This flag is ignored for other devices.

DRC is the indication of direct record control for keyboard-printer operation (0 indicates edited record transfers; 1 indicates direct record transfers). Keyboard/printer edited transfers are described earlier in this chapter. For transfers involving other devices, the flag is ignored. (See MOD description for COC line transparency.)

D/P indicates packed binary mode for 7-track tapes in conjunction with MOD (above) being set (0 indicates unpacked; 1 indicates packed if MOD is also 1). It also is used to indicate density selection on a write at load-point to a magnetic tape drive with program-controlled density (0 indicates 800 bpi; 1 indicates 1600 bpi). On other devices or under other circumstances than described, the flag is ignored.

VFC is the vertical format indicator (0 indicates no format control; 1 indicates format control) specifying whether or not the first character of an output record is to be used to control vertical positioning for output to a line printer or keyboard/printer. Under format control, the line printer is given a "print with format" order. The keyboard/printer performs a preliminary new line (regardless of the format character) and outputs the record beginning with the second byte. The first byte is output as data on all other devices. VFC has no effect on other I/O operations. The format control codes are itemized in Table 13.

DAN indicates whether a disk area name is provided. If ASN is not 1, DAN is not used. If ASN = 1 and DAN = 0, the disk area index will be obtained from the TYPE field. If ASN = 1, DAN = 1, and P1 = 0, the disk area is unspecified (OPEN may provide a default). If ASN = 1, DAN = 1, and P1 = 1, the area name will be obtained from the AREA field.

DON indicates whether a device or operational label name is provided. If ASN is not 3, DON is not used. If ASN = 3 and DON = 0, the device or operational label index will be obtained from the DEV/OPLB/RFILE field. If ASN = 3 and DON = 1, the device or operational label name will be obtained from the N0-N7 fields, as described later.

BTD is the byte displacement specifying at which byte (0-3) in a buffer the data begins.

[†] See Chapter 3, Xerox Sigma Card Reader (Models 7120/7122/7140)/Reference Manual, 90 09 70.

ASN is the assignment type indicator (0 means null; 1 means disk file; 2 means not used; 3 means device or system operational label).

Table 13. Line Printer Format Control Codes

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

Word 1

NRT is the number of recovery tries to be allowed before outputting a device error message.

DEVF is an indicator specifying whether the device assignment (when ASN has value 3) in force is directly to a physical device or indirectly through an operational label (1 means direct; 0 means indirect). See TYPE and DEV/OPLB RFILE discussion below.

L is an indicator specifying whether the assigned device is a line printer or keyboard/printer. The indicator is set by the system at OPEN time.

TYPE is a field indicating the type of device that is directly assigned if ASN has value 3 and DEVF has value 1. In this case, TYPE is set when the DCB is opened, regardless of its prior content.

Value	Device
0	NO (IOEX)
1	TY or LN
4	CR
5	CP
6	LP
7	DC
8	9T
9	7T
10	CP (Low Cost)
11	LP (Low Cost)
12	DP
13	PL
14	DP
15	LP
16	9T
18	XX (special)
19	LD (logical device)

If ASN has value 1, TYPE specifies the area that contains the disk file. If DAN is 0, TYPE must be set before opening. If DAN is 1, TYPE is set by the OPEN service.

Value	Area
0	SP
1	FP
2	BP
3	BT
4	XA
6	} user defined areas
:	

(Value 5 refers to area CK, which is reserved for system use.)

DEV/OPLB/RFILE contains one of three:

1. The DCT index of the assigned device when the assignment is to a device (ASN equals 3 and DEVF equals 1). This must be set before opening if DON is 0. It is set by OPEN if DON is 1.

2. The operational label table index of the assigned operational label when the assignment is to an operational label (ASN equals 3 and DEVF equals 0). This must be set before opening if DON is 0. It is set by OPEN if DON is 1. The index values for standard system operational labels are

Label	Index Value
C	1
OC	2
LO	3
LL	4
DO	5
CO	6
BO	7
CI	8
SI	9
BI	10
SO	11

Optional operation labels may be provided by the user, or in connection with certain SYSGEN options. The user is responsible for determining the index values for his optional operational labels when specifying an operational label by index. These values are a function of the order in which the optional operational labels are specified at System Generation.

The index value for the devices are also a function of the order in which the devices are specified at System Generation.

3. When a DCB is assigned to a disk file (ASN equals 1), this field contains the index to the RFT (RAD File Table). This value is set when the DCB is opened. The RFT entry is created at OPEN if an entry does not already exist for the file.

Word 2

TYC is an indicator showing the type of completion for an I/O operation. TYC is set by the I/O system at the completion of each request that uses the DCB in a Type I mode (see discussion of Read and Write system calls below). The completion type codes are listed in Appendix N.

BUF is the address of the user buffer for requests whose FPTs do not include a buffer address.

Word 3

RSZ is the default record size in bytes ($1 \leq RSZ \leq 32{,}767$). The parameter is used as the byte count for Read/Write requests that do not include a byte count.

ERA is the address of the user's routine that handles errors associated with insufficient or conflicting information in the DCB or FPT. Zeros in this field are used to indicate that no user error routine exists (see discussion of error and abnormal returns below).

Word 4

ARS is the actual record size in bytes. The parameter is set by the I/O system when a request is completed. It is set in the DCB for Type I requests only. For I/O requests that will result in a data transfer of more than 32,767 bytes, a TYPE II request should be used since only 15 bits are available for posting ARS in Type I requests.

ABA is the address of the user's routine that handles abnormal conditions associated with insufficient or conflicting information in the DCB or FPT. Zeros are used to indicate that no user abnormal routine exists (see discussion of error and abnormal returns below).

Information past word 4 is used only by the OPEN and ASSIGN CALs. While a DCB is open, all information of value to CP-R is in words 0-4.

Words 5 and 6:

If ASN = 1 and TTL $\geq$ 7, words 5 and 6 will be accessed by the OPEN service for an 8-byte file name or numeric zero. Additionally, if TTL = 6, four bytes of file name or zero will be obtained from word 5 only.

If ASN = 3, DEVF = 0, and DON = 1, word 5 will be accessed by the OPEN service for an operational label in the second halfword. Word 6 need not be present.

If ASN = 3, DEVF = 1, and DON = 1, words 5 and 6 will be accessed by the OPEN service for a device name in N0-N4, with blanks in N5-N7.

Word 7:

If word 7 is used, words 5 and 6 must be present even if they are not used.

P1 is a presence bit for the word containing AREA (P1 = 0 if not present; P1 = 1 if present).

P2 is a presence bit for the word containing A0-A3 (P2 = 0 if not present; P2 = 1 if present).

P3 is a presence bit for the word containing A4-A7 (P3 = 0 if not present; P3 = 1 if present). However, if P2 is zero, A4-A7 will be ignored, regardless of the value of P3.

Words selected by the presence bits must follow word 7 in the indicated order. Words not selected must be omitted.

variable-position optional words:

AREA is a system or user disk area name. This parameter is used only during the OPEN service, as described in the paragraph on the DAN field. If it is omitted or zero when it is needed, OPEN provides a default.

A0-A7 is an account name, filled to eight characters with trailing blanks. The word containing A4-A7 may be omitted if it is all blanks. This parameter is accessed only during the OPEN service when ASN=1 and the file name is not numeric zero. If ASN=1 and the file name is nonzero, but A0-A7 are omitted, numeric zero, or all blanks, OPEN provides a default account name.

ERROR AND ABNORMAL CONDITIONS

Certain error codes are returned to the user's error or abnormal return routines upon occurrence of various conditions. At entry to these routines, the error code is contained in byte 0 of register 10, the DCB address is contained in the address field (low-order 17 bits) of register 10, and the address of the location following the CAL1 is contained in register 8. The previous contents of registers 8 and 10 are lost.

Foreground users must provide error and abnormal returns on all I/O requests with wait and on all CHECK requests. If background users omit the error and abnormal addresses, the system will take action as detailed below. The error codes are defined in Appendix N.

I/O SYSTEM CALLS

I/O system calls may be made only when a user has been given control under the following circumstances:

1. The background or foreground program loader has transferred control to the starting address of the user's program.

2. The user's centrally connected task has been given control upon the occurrence of its associated interrupt.

   Warning: Do not perform I/O system calls from a task connected to an interrupt of higher priority than the I/O interrupt.

In all of the FPT formats which follow, an asterisk in bit 0 indicates that indirect addressing is permitted. That is, if bit 0 contains a 1, bits 15 through 31 contain the address of the parameter rather than the parameter itself.

Whenever a return is made to an abnormal or error address, the error or abnormal code will be in byte 0 of register 10 and the address of the location following the call will be in register 8.

Calls for which F3 (wait indicator) is not available are either immediate or synchronous. That is, control will never be returned to the caller until the service has been completed (see Appendix I for more detailed information). Primary task users should recognize that during any wait for completion of the service all lower priority tasks are blocked.

If an error is detected in the call and no error address is available, the situation will be handled similar to a trap and the user will be aborted unless he has elected to do his own trap handling.

OPEN A FILE

OPEN The OPEN system call opens the data file if it is not already open. If the addressed DCB is assigned to a device (directly or through an operational label), a count is kept of the number of open DCBs assigned to the device.

If the DCB is assigned to a disk file, an entry is built in the RFT (RAD File Table) if one does not already exist and the index of the entry is placed in the DEV/OPLB/RFILE field of the DCB. A count of open DCBs assigned to the disk file is also maintained. The user may specify a buffer to be used in the File Directory search but this is not mandatory. The OPEN function will use available blocking buffers if such a buffer is not given.

When a DCB assigned to a file is opened, the following defaults for area and account names apply:

1. If neither account name nor area name is specified, the calling task's account is used, and the area may be any public area. This provides the simplest specification, and the user need not be concerned with the possibility of name conflicts other than within his own account.

2. If the account name is specified, but the area name is not, the file may be in any public area. This provides for area-independent file specification.

3. If the area name is specified but the account name is not, the system account will be used. This case provides compatibility for code written before the addition of file account names.

4. If the account name is all blanks, or numeric zero, it is treated as unspecified.

5. If the disk area name is numeric zero, it is treated as unspecified.

The error and abnormal addresses in the DCB may be set or changed at OPEN. The error and abnormal addresses will be copied from the FPT if present. The OPEN function causes the specified DCBs file-open indicator (OPEN) to be set to 1.

If the specified DCB is assigned to an operational label and the operational label, in turn, is assigned to a 7T device, PACK and BIN are set to 1.

If a DCB is already open (OPEN = 1) for device-assigned DCBs when the OPEN function is called, an abnormal condition is signaled (see Appendix N). The device indicator (DEV/OPLB/RFILE) of the DCB is checked for validity. If it references a valid operational label or physical device, the DCB is marked open; if the device indicator is invalid, the DCB is not marked open and an abnormal condition is signaled (see below for the OPEN call format).

For DCBs assigned (directly or through oplabels) to line printers or keyboard/printers, the L indicator in the DCB is set to 1.

If a DED DPndd,R key-in is in effect, the file will not be opened. A DCB abnormal condition with code X'2F' exists.

If the DCB is assigned to a device that has been declared to be nonsharable, the user will be given exclusive use of the device providing that it is currently available. If the device is already in use in another job, the user's DCB will not be OPENed and an error condition will be returned.

CLOSE A FILE

CLOSE The CLOSE function closes a DCB by setting the DCB open indicator (OPEN) to 0, which may result in closing the assigned data file on a device or disk file. The CLOSE function decrements the "open DCB count" in the proper DCT or RFT entry and if the count becomes zero, the data file is closed.

If the data file is to be closed and is a disk file opened for output, the directory entry for the disk file is updated with the information from the RFT entry and the entry is deleted from the RFT table. If the file is extensible, all extents past the one that contains the last record are deleted.

If the data file is to be closed and is a disk file opened for input only, the entry is deleted from the RFT table.

If the file is on a 'non-system' disk pack, a DED DPndd,R key-in has been made, and this is the last file to be closed, the system will output the message

!!DPndd IDLE

If the DCB is assigned to a device that has been declared to be nonsharable, the device will be marked as being available to other jobs providing that no other DCBs are open to the device.

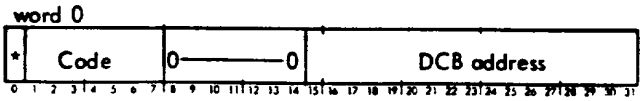
Closing other types of data files requires no action.

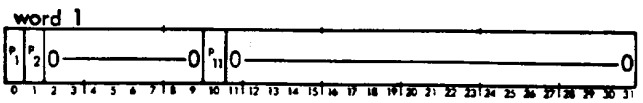
OPEN AND CLOSE SYSTEM CALL FORMAT

OPEN and CLOSE system calls have the format

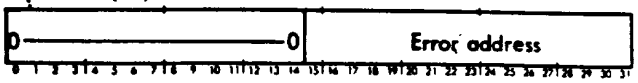
CALL, 1 address

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

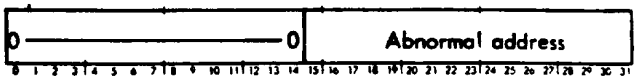




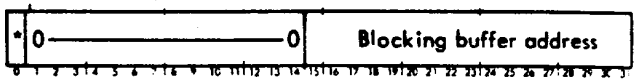
optional (P1)†



optional (P2)†



optional (P11)†



where

Word 0

Code is X'14' for OPEN, X'15' for CLOSE.

DCB address is the address of the associated DCB.

Word 1

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_2 is the abnormal address parameter presence indicator (0 means absent; 1 means present).

P_{11} is the blocking buffer address parameter presence indicator (0 means absent; 1 means present).

Word Options

Error address is the address of the entry to the user's routine that will handle error conditions.

Abnormal address is the address of the entry to the user's routine that will handle abnormal conditions.

Blocking buffer address is the address of a 257-word buffer to be used for file directory search if the DCB being closed is assigned to a disk file.

READ A DATA RECORD

READ The READ function causes the I/O system to read a data record into a user buffer from the device or file specified by the DCB.

An implicit OPEN will be performed on the DCB if the addressed DCB is closed when the READ request is made.

† In all FPTs for I/O functions where an optional parameter is not used, the parameter word must be omitted from the FPT and the corresponding presence indicator (P_n) set to 0.

READ requests may specify either a "wait for completion" or an "immediate return" condition. Foreground requests with wait must include error and abnormal returns in the FPT. Background requests can omit these addresses and have the system handle error and abnormal conditions.

Should the input record be physically longer than the specified buffer length, data is lost and the user is notified through an abnormal return with code 07.

Should the input record be physically shorter than the specified buffer length, the buffer is not filled and the actual record length is posted in the FPT or DCB.

Input from the card reader is performed either in automatic or binary mode. If the card reader is not the C device, the input mode is determined by the BIN flag in the DCB. The C device is always read in automatic mode. Foreground programs may not read the C device as this would disrupt the background job stream.

Input from the C device results in all control commands (! in column 1) being intercepted by the I/O system. Any control command other than !EOD causes an abnormal return with a code of 06 in register 10. The input record is kept in the CP-R control command buffer. If an attempt is made to read this same device again, an error return with code 54 is given (see Appendix N).

An !EOD record encountered from a card reader on a READ request results in an abnormal return with code 05.

For direct access input from disk files, the user includes a key in the FPT. All READ requests without a key parameter are assumed to be sequential access requests and result in the next record in order being input into the user buffer. For sequential input from blocked files, the request may not result in an actual disk access.

For sequential access input from compressed files, the I/O system decompresses the record in transmitting it to the user buffer.

Type II READ requests must include in their FPTs a completion status parameter in which the type of completion code and the actual byte count are posted.

A Type I READ request that finds the DCB busy with a previous Type I request results in an error condition (error code 48).

WRITE A DATA RECORD

WRITE The WRITE function causes the I/O system to write a data record from a user buffer to the device or disk file specified by the DCB.

WRITE requests may specify "wait" or "no-wait". As with READ requests, foreground WRITE requests specifying wait must include error and abnormal return addresses in the FPT.

For direct access output to disk files, the key address parameter is included in the FPT. All WRITE requests without a key address parameter present are assumed to be sequential access requests.

For output to compressed files, the I/O system compresses the record in transmitting it to the system blocking buffer.

Type II Write requests must include a completion status parameter word in their FPTs in which the I/O system will post the type of completion code and the actual byte count.

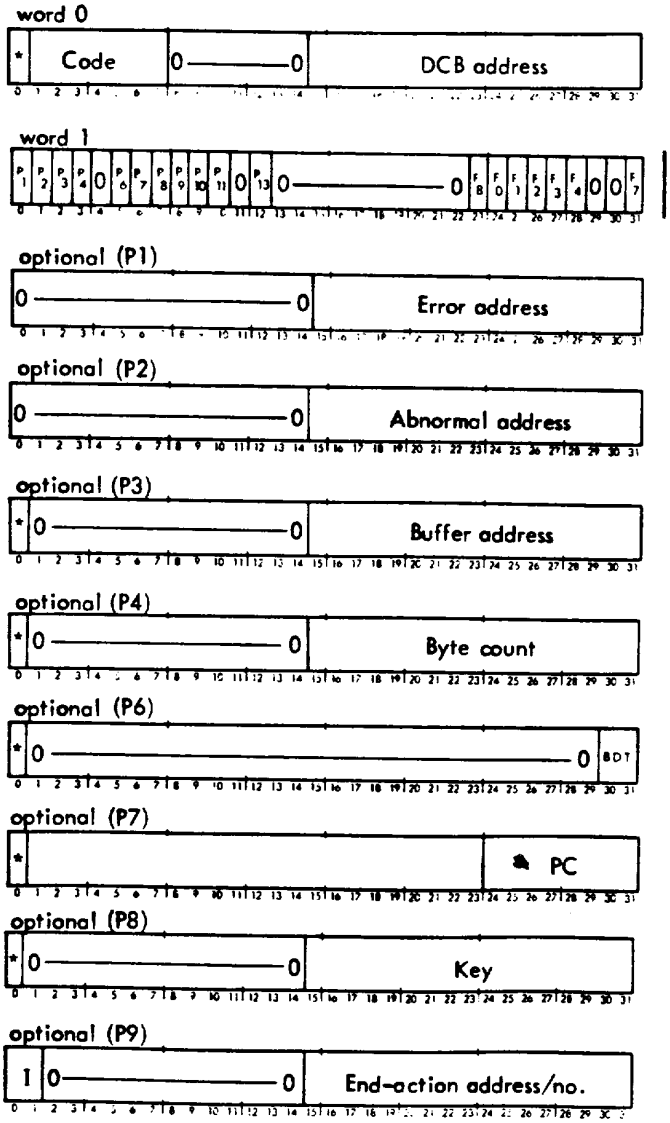
A Type I request that finds the DCB busy with a previous Type I request results in an error condition (error code 48).

READ AND WRITE FUNCTION CALL FORMAT

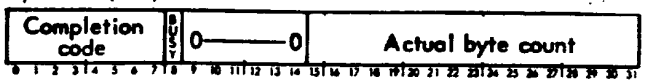
Calls for these functions are of the form

```
CAL1,1 address
```

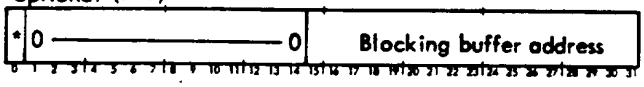
where address points to word 0 of the FPT shown below.



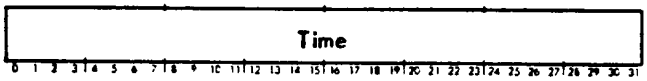
optional (P10)



optional (P11)



optional (P13)



where

Word 0

Code is X'10' for READ and X'11' for WRITE.

DCB address is the address of the associated DCB.

Word 1

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_2 is the abnormal address parameter presence indicator (0 means absent; 1 means present).

P_3 is the buffer address parameter presence indicator (0 means absent; 1 means present).

P_4 is the byte count parameter presence indicator (0 means absent; 1 means present).

P_6 is the byte displacement parameter presence indicator (0 means absent; 1 means present).

P_7 is the prompt character presence indicator (0 means absent; 1 means present).

P_8 is the key parameter presence indicator (0 means absent; 1 means present).

P_9 is the end-action parameter presence indicator (0 means absent; 1 means present).

P_{10} is the completion code parameter presence indicator (0 means absent; 1 means present).

P_{11} is the blocking buffer address parameter presence indicator (0 means absent; 1 means present).

P_{13} is the time interval parameter presence indicator (0 means absent; 1 means present).

$F_1 = 1$ specifies that I/O is to be allowed only if the device has been deactivated (i.e., marked "down"); this bit is for diagnostic-program usage.

$= 0$ specifies that I/O is to be allowed only if the device has not been deactivated; normal program usage.

F_2 is the direction indicator for READ (0 means forward; 1 means reverse). This indicator has effect on magnetic tape Read/Write operations only.

F_3 is the wait indicator (0 means no-wait; 1 means wait for I/O completion).

F_4 is the RAD Check-Write indicator (1 means a read or write on a RAD or disk pack will be followed by a Check-Write operation; 0 means a normal read or write will be performed).

$F_7 = 1$ is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

F_8 is a delete on post indicator (1 means delete the event when it is posted — no CHECK will be performed and the user's FPT/DCB will not be posted; 0 means a CHECK will be performed). F_8 has meaning only when $F_3 = 0$.

Word Options

Error address is the address of the entry to the user routine that will handle error conditions for requests specifying wait.

Abnormal address is the address of the entry to the user routine that will handle abnormal conditions for requests specifying wait.

Buffer address is the word address of the user buffer to be used in the I/O operation. Data is written from or read into this buffer. If this parameter is omitted, the buffer address is taken from the DCB (BUF). The buffer address may not be a register.

Byte count is the size in bytes of the data record. If this parameter is omitted, the record size is taken from the DCB (RSZ parameter).

BTD is the byte displacement (0–3) from the word boundary of the beginning of the data record. If this parameter is omitted and the Buffer address parameter is included in the FPT, value 0 is assumed for BTD. If both parameters are omitted from the FPT, the values of the DCB are used for both.

PC is a prompt character to precede a keyboard-printer read with editing. If not specified, the prompt character associated with the job is used.

Key is the number of the granule within a disk file or area to be accessed with the direct-access method. For the device-access method, key is the number of the sector to be accessed, relative to the beginning of a disk.

I,End-action address/no. I indicates the contents of the End-action address/number field. End-action is allowed only for foreground.

Value 0 indicates an end-action address.

Value 1 indicates an interrupt number.

Value 2 indicates an interrupt operational label.

Value 3 indicates a completion signal address.

End-action is taken only in the case of an actual I/O operation involving a peripheral device (i.e., end-action will not occur for blocked disk file I/O).

Completion status is the word wherein the I/O system posts the completion parameters for the request. Presence of this word indicates that the I/O request should be of Type II (see P_{10}). The I/O system initializes the completion code and actual byte count to zero before starting the operation. At completion of the request the actual byte count and the completion code are posted if the wait indicator was set. Otherwise, these parameters are posted when the service is CHECKed.

BUSY is the FPT busy indication that is set and maintained by the I/O system to indicate that a Type II request using the FPT is in progress.

Blocking buffer address is the address of a 257-word buffer to be used for file directory search if the DCB is being opened to a disk file.

Time interval is the maximum number of seconds that the caller will wait for the CALL to complete. If not specified the CALL will not be timed-out. Note that the I/O system may time out a device but this is independent from the use of the "Time Interval" parameter.

REWIND, UNLOAD, AND WRITE EOF FUNCTIONS

REW The Rewind function causes a data file to be positioned at its beginning if the file is on magnetic tape or disk. Rewind of a file on magnetic tape is accomplished by causing the tape drive to rewind to beginning of tape. Rewind of a disk file is accomplished by setting the file position parameter in the RAD File Table (RFT) so that the next sequential access request on the file results in the first record being accessed. A Rewind request for an extensible disk file results in the file being positioned at the first record of the first extent. A Rewind request for a data file on any other device results in no action being taken.

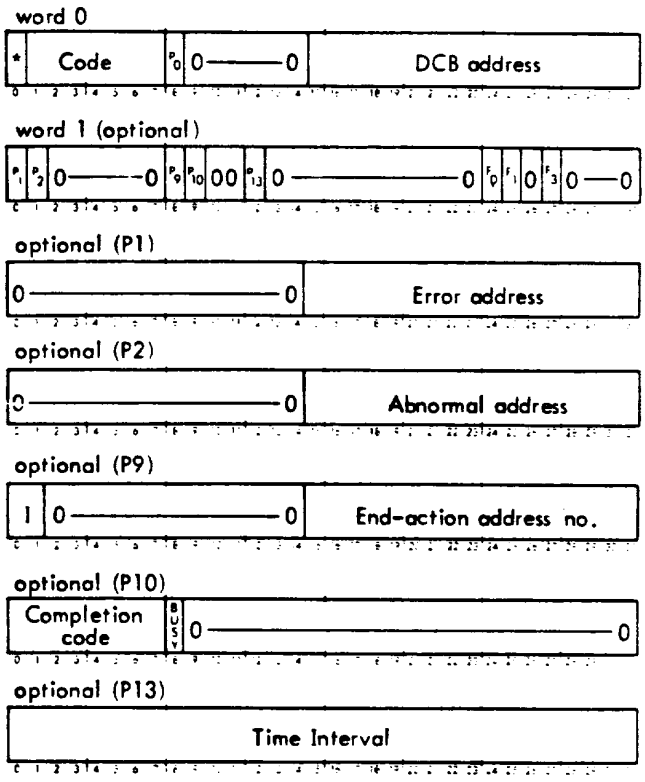
UNLOAD The Unload request results in the same action as Rewind except that magnetic tapes are rewound "off-line". When the rewind is concluded, operator action is required before the device can be used again.

WEOF Write End-of-File causes an EOF to be written if the addressed DCB is assigned to a magnetic tape unit. If the DCB is assigned to a card punch, and !EOD record is output. If the DCB is assigned to a disk file, an implicit EOF is written. If the disk file is extensible, all extents past the current one will be deleted. If the DCB is assigned to any other type of device, no action is taken.

Rewind (REW), Unload (UNLOAD) and Write End-of-File (WEOF) calls are of the form

CAL1, 1 address

where address points to word 0 of the FPT below.



where

Word 0

Code is X'01' for REWIND, X'02' for WEOF, and X'03' for UNLOAD.

$P_0 = 0$ means word 1 of the FPT is absent;

$= 1$ means word 1 of the FPT is present.

If 0, REW and UNLOAD are done without wait and WEOF is done with wait. Otherwise, the action taken depends on word 1.

DCB address is the address of the associated DCB.

Word 1 (optional)

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_2 is the abnormal address parameter presence indicator (0 means absent; 1 means present).

P_9 is the end-action parameter presence indicator (0 means absent; 1 means present).

P_{10} is the completion code parameter presence indicator (0 means absent; 1 means present).

P_{13} is the time interval parameter presence indicator (0 means absent; 1 means present).

$F_0 = 1$ specifies long wait.

$F_1 = 1$ specifies that I/O is to be allowed only if the device has been deactivated (i.e., marked "down"); this bit is for diagnostic-program usage.

$= 0$ specifies that I/O is to be allowed only if the device has not been deactivated; normal program usage.

F_3 is the wait indicator (0 means no-wait; 1 means wait for I/O completion).

Word Options

Error address is the address of the entry to the user routine that will handle error conditions for requests specifying wait.

Abnormal address is the address of the entry to the user routine that will handle abnormal conditions for requests specifying wait.

I, End-action address/no. I indicates the contents of the End-action address/number field. End-action is allowed only for foreground.

Value 0 indicates an end-action address.

Value 1 indicates an interrupt number.

Value 2 indicates an interrupt operational label.

Value 3 indicates a completion signal address.

End-action is taken only in the case of an actual I/O operation involving a peripheral device.

Completion status is the word wherein the I/O system posts the completion parameters for the request. Presence of this word indicates that the I/O request should be of Type II (see P_{10}). The I/O system initializes the completion code to zero before starting the operation. At completion of the request the completion code is posted if the wait indicator was set. Otherwise, these parameters are posted when the service is CHECKed.

BUSY is the FPT busy indication that is set and maintained by the I/O system to indicate that a Type II request using the FPT is in progress.

Time interval is the maximum number of seconds that the caller will wait for the CALL to complete. If not specified the CALL will not be timed-out. Note that the I/O system may time out a device but this is independent from the use of the "Time Interval" parameter.

FILE AND RECORD POSITIONING FUNCTIONS

These functions are used to alter position within a data file on magnetic tape or disk.

PFIL,PREC A Position File (PFIL) call causes a magnetic tape to be positioned at the beginning or end of the current file if a backward or forward direction respectively is specified and no skip is requested. If skip is requested, the tape is positioned as above except that the file mark is skipped over in the specified direction. Position File forward without skip positions the tape at the end of the current file (before the EOF). With skip, Position File forward positions the tape at the beginning of the next file.

Position File causes a "rewind" of disk files when "backward" is specified, and positioning after the last record in a disk file when "forward" is specified. The file will be positioned after the last record in the last extent if the file is extensible.

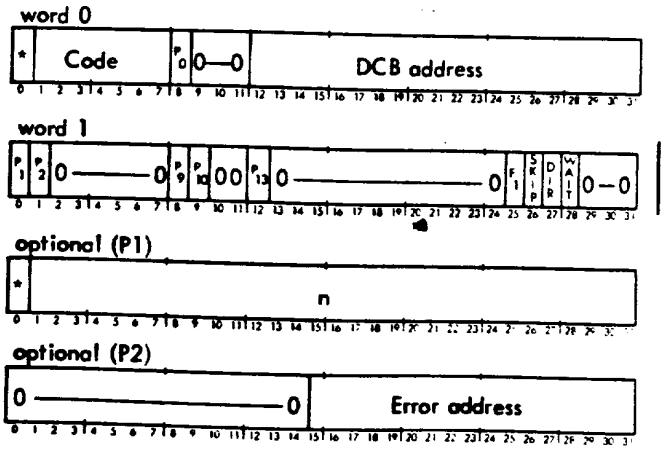
Position Record (PREC) causes a tape or disk file to be moved n records in the specified direction. For extensible disk files, this may require crossing extents.

Position File and Position Record are ignored when the data files are on devices other than magnetic tape or disk file.

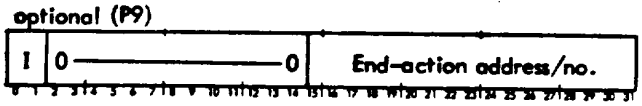
File and Record positioning calls are of the form

CAL1, 1 address

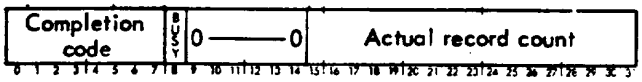
where address points to word 0 of the FPT shown below.



optional (P9)



optional (P10)



optional (P13)

where

Word 0

Code is X'1C' for Position File (PFIL) and X'1D' for Position Record (PREC).

$P_0 = 1$ indicates the explicit use of the P_9, P_{10}, F_1, and WAIT parameters.

$= 0$ P_9, P_{10}, F_1, and WAIT are ignored and the CAL is processed as Type I with wait and no end-action.

DCB address is the address of the associated DCB.

Word 1

P_1 is the record count (n/address parameter presence indicator (0 means absent; 1 means present). P_1 is ignored for Position File. If P_1 is absent, n assumes a value of 1.

P_2 is the error address parameter presence indicator (0 means absent; 1 means present).

P_9 is the end-action parameter presence indicator (0 means absent; 1 means present).

P_{10} is the completion code parameter presence indicator (0 means absent; 1 means present).

P_{13} is the time interval parameter presence indicator (0 means absent; 1 means present).

$F_1 = 1$ specifies that I/O is to be allowed only if the device has been deactivated (i.e., marked "down"); this bit is for diagnostic-program usage.

$= 0$ specifies that I/O is to be allowed only if the device has not been deactivated; normal program usage.

SKIP indicates whether the EOF is to be skipped over in positioning magnetic tape. This parameter has significance only for Position File and on magnetic tape (0 means no skip; 1 means skip).

DIR is the direction indicator (0 means forward positioning; 1 means backward positioning).

WAIT is the wait indicator (0 means no-wait; 1 means wait for I/O completion).

Word Options

n is the number of records to position.

Error address is the address of the entry to the user's routine that will handle error and abnormal conditions (EOT, BOT, etc.), for this I/O operation. EOF is considered an abnormal condition for PREC.

I, End-action address/no. I indicates the contents of the End-action address/number field. End-action is allowed only for foreground with the exception that no end-action will take place for blocked disk file I/O.

Value 0 indicates an end-action address.

Value 1 indicates an interrupt number.

Value 2 indicates an interrupt operational label.

Value 3 indicates a completion signal address.

Completion status is the word wherein the I/O system posts the completion parameters for the request (presence of this parameter indicates that the request is of Type II). The I/O system initializes the completion code and actual record count before starting the operation. At completion of the request the actual record count and the completion code are posted in the word if the wait indicator is set. Otherwise, these parameters are posted when the service is CHECKed. BUSY is the FPT busy indicator that is set and maintained by the I/O system to indicate that a Type II I/O request using the FPT is in progress.

Time interval is the maximum number of seconds that the caller will wait for the CALL to complete. If not specified the CALL will not be timed-out. Note that the I/O system may time out a device but this is independent from the use of the "Time Interval" parameter.

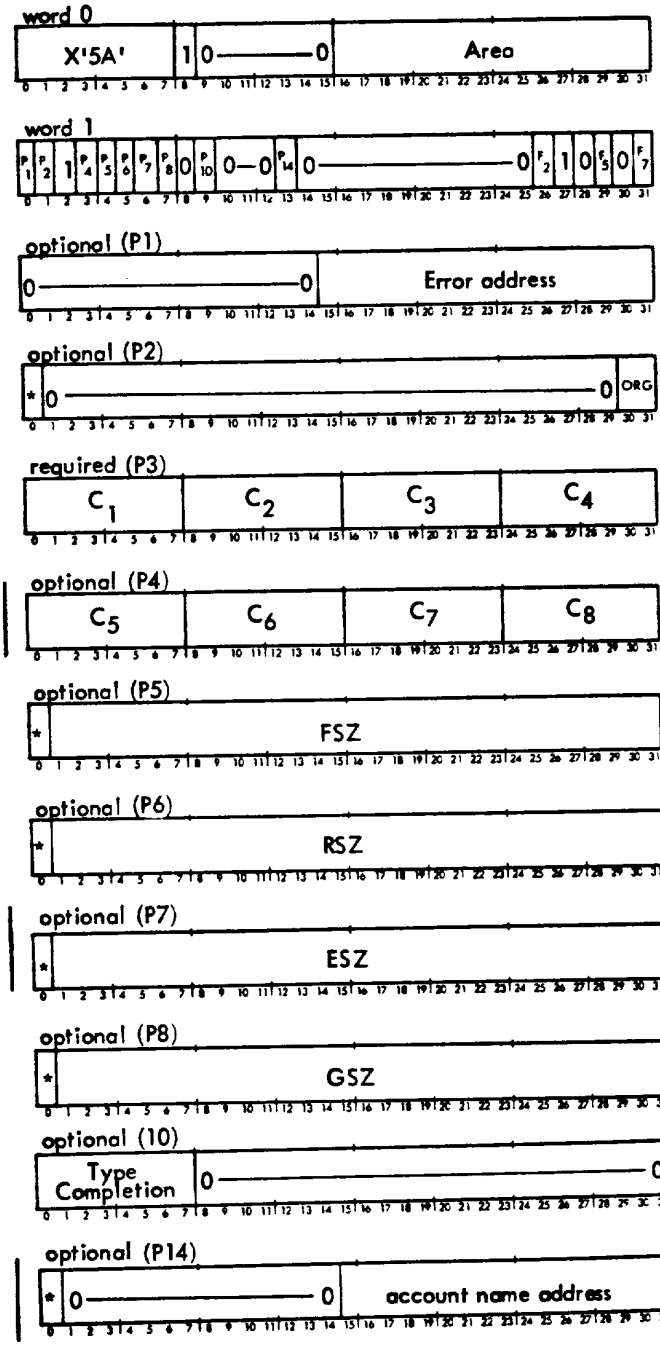
ALLOT, DELETE, AND TRUNCATE FUNCTIONS

ALLOT The ALLOT function is used to define a new file in a permanent disk area. This is accomplished by adding a new entry to the file directory in the specified area. The defined file will receive the specified size and format.

This call has the form

CAL1,7 address

where address points to word 0 of the FPT shown below



where

Word 0

X'5A' specifies the code for the ALLOT call.

Area is the name of the permanent area in which the file is being defined. If the disk area name value is zero, the disk area is unspecified.

Word 1

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_2 is the file organization parameter presence indicator (0 means absent; 1 means present).

P_4 is the file name second word parameter presence indicator (0 means absent; 1 means present).

P_5 is the file size parameter presence indicator (0 means absent; 1 means present).

P_6 is the record size parameter presence indicator (0 means absent; 1 means present).

P_7 is the extent size parameter presence indicator (0 means absent; 1 means present).

P_8 is the granule size parameter presence indicator (0 means absent; 1 means present).

P_{10} is the type-of-completion parameter presence indicator (0 means do not post; 1 means post type of completion).

P_{14} is the account name address parameter presence indicator (0 means absent; 1 means present).

F_2 is the resident foreground load module indicator and is applicable only if the area is FP. A zero means the file is not to contain a resident foreground load module.

F_5 is meaningful only if ESZ option is present and is used to determine whether or not the file should be combined with its extents if the area is squeezed (1 means do not combine extents; 0 means combine extents).

$F_7 = 1$ is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

Word Options

Error address is the return address for all errors. The error code will be in byte 0 of register 10 and the address of the location following the call will be in register 8.

C_i are the EBCDIC characters naming the disk file being defined. If C_5-C_8 are blanks, the word containing them may be omitted.

ORG is the file organization type:

00 for unblocked
01 for blocked
10 for compressed

The default is unblocked.

FSZ is the file size in logical records. The default is 1000 records.

RSZ is the number of words per record. The logical size is used in sequentially accessing a file. For a compressed file, record size is omitted and the monitor blocks compressed files into 256-word records. Blocked files have a default value equal to 128 words per record. If the record size is greater than 128 words, unblocked organization will be given. Unblocked files have a default record size equal to the granule size. The maximum record size is 16,383 words.

ESZ is the size of file extents, in logical records, to be allocated and attached to the file anytime the file is out of space. A value of zero means that file extents will be the same size as the original file (FSZ). The default is no automatic file extension.

GSZ is the granule size in words and is used for direct access only. The default size will be equal to the disk sector size. The maximum granule size is 16,383 words.

Type completion is the byte in which a type-of-completion code will be posted by the service function.

account name address is the address of a two-word data block that contains the account name in EBCDIC, filled to eight characters with trailing blanks. If the account name is all blanks or numeric zero, it is assumed to be unspecified.

Software write protection is provided which allows foreground load modules to ALLOT files in any data area. Background load modules may only ALLOT files in background data areas. Software write protection may be overridden with an SY key-in.

The following defaults for area and account names apply:

1. If neither account name nor area name is specified, the calling task's account is used, and the area may be any public area. This provides the simplest specification, and the user need not be concerned with the possibility of name conflicts other than within his own account.

2. If the account name is specified, but the area name is not, the file may be in any public area. This provides the area-independent file specification.

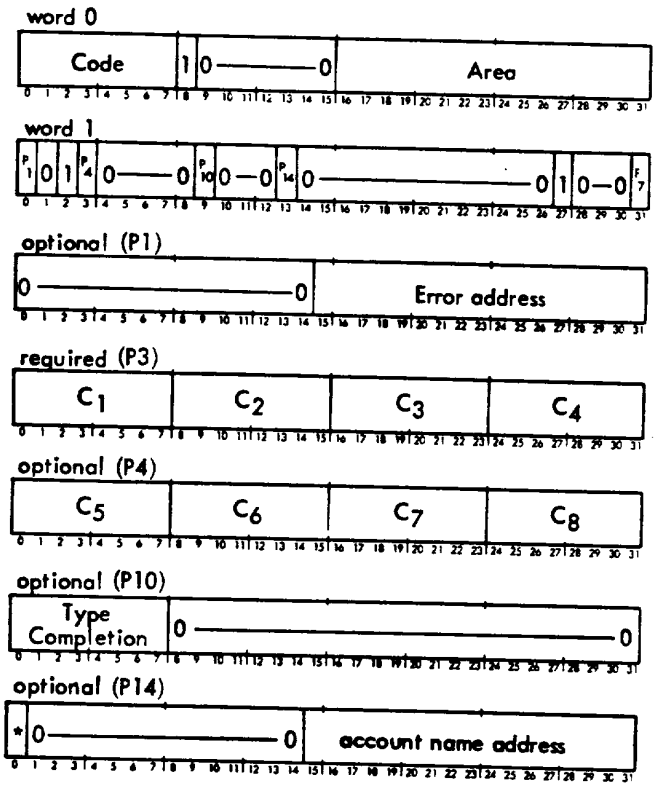
3. If the area name is specified but the account name is not, the system account will be used. This case provides compatibility for code written before the addition of file account names.

DELETE,TRUNCATE The DELETE system call is used to delete a disk file in a permanent area by removing the entry from the directory in the specified area. The space that was formerly used is available for a new file. If the file is extensible, all extents in the file are deleted. The TRUNCATE system call is used to truncate empty space from the end of the specified file in a permanent area. The allocated space will be set equal to the actual length of the file. If the file is extensible, the last extent will be truncated to the last record in that extent. Prior extents are unaffected. Note that if the file size is zero, the file will not be truncated.

Calls for these functions are of the form

CAL1,7 address

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



Word 0

Code X'5B' specifies the DELETE call. X'5C' specifies the TRUNCATE call.

Area is the name of the permanent area that contains the specified file. If the disk area name value is zero, the disk area is unspecified.

Word 1

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_4 is the file name second word parameter presence indicator (0 means absent; 1 means present).

P_{10} is the type-of-completion parameter presence indicator (0 means do not post; 1 means post type of completion.

P_{14} is the account name address parameter presence indicator (0 means absent; 1 means present).

F_7 is an abort override indicator (0 means abort if errors are detected and no error address is provided; 1 means do not abort if errors are detected but no error address is provided).

Word Options

Error address is the return address for all errors. The error code will be in byte 0 of register 10 and the address of the location following the call will be in register 8.

C_i are the EBCDIC characters defining the disk file to be deleted or truncated. If C_5-C_8 are all blanks they may be omitted.

Type completion is the byte in which a type-of-completion code will be posted by the service function.

account name address is the address of a two-word data block that contains the account name in EBCDIC, filled to eight characters with trailing blanks. If the account name is all blanks or numeric zero, it is assumed to be unspecified.

Software write protection is provided that allows foreground load modules to DELETE/TRUNCATE files in any data area. Background load modules may only DELETE/TRUNCATE files in background data areas. Software write protection may be overridden with an SY key-in.

The following defaults for area and account names apply:

1. If neither account name nor area name is specified, the calling task's account is used, and the area may be any public area. This provides the simplest specification, and the user need not be concerned with the possibility of name conflicts other than within his own account.

2. If the account name is specified, but the area name is not, the file may be in any public area. This provides for area-independent file specification.

3. If the area name is specified but the account name is not, the system account will be used. This case provides compatibility for code written before the addition of file account names.

ALLOT OR DELETE SYMBIONT FILE

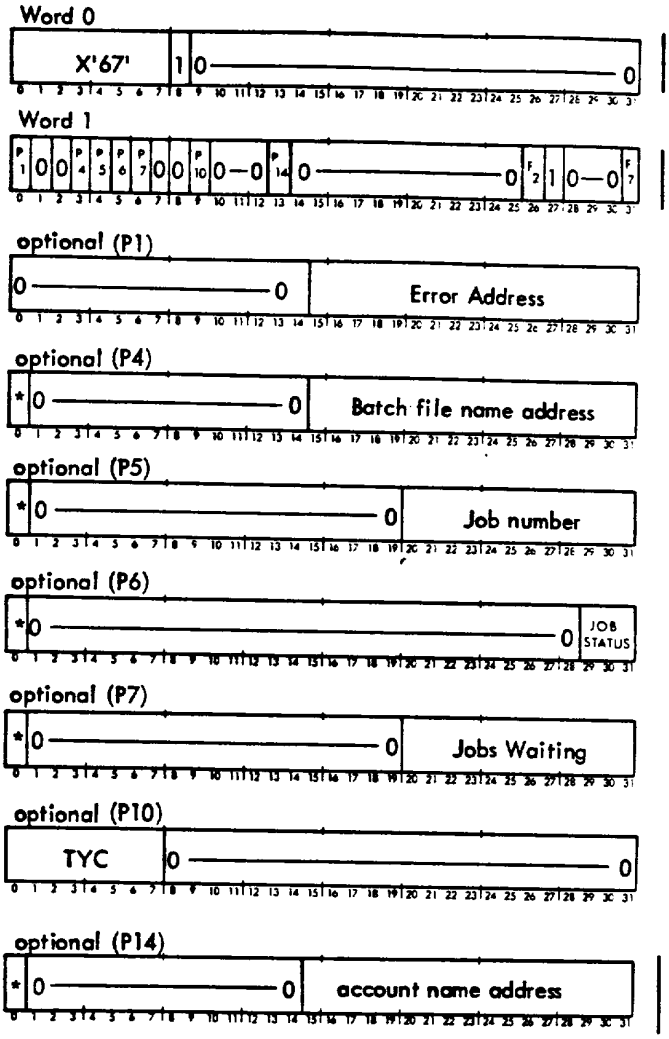
JOB The JOB call is used to maintain symbiont files by performing the following functions.

- Allocate a job file in the input symbiont area.
- Check status of a previously allocated job file.
- Delete a job waiting in the symbiont area.

The JOB call has the following format:

```
CAL1,7  address
```

where address points to word 0 of the FPT below.



where

Word 0

X'67' specifies the code for the JOB call.

Word 1

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_4 is the batch file name address parameter presence indicator (0 means absent; 1 means present).

P_5 is the job number parameter presence indicator (0 means absent; 1 means present).

P_6 is the job status parameter presence indicator (0 means absent; 1 means present).

P_7 is the jobs waiting parameter presence indicator (0 means absent; 1 means present).

P_{10} is the type completion parameter presence indicator (0 means absent; 1 means present).

P_{14} is the account name address parameter presence indicator (0 means absent; 1 means present).

F_2 1 means to delete all symbiont files associated with the specified job number. 0 means no files are to be deleted. Note that when F2=1, the job number parameter (P5) must be present.

F_7 is the abort override indicator (0 means abort if errors are detected and no error address is present; 1 means do not abort if errors are detected and no error address is present).

Word Options

Error address is the address of the entry to the user's routine that will handle error conditions.

Batch file name address is the address of a three-word data block that contains an area name in the first word in bits 16-31 in EBCDIC and a file name, in EBCDIC, left-justified with training blanks in the second and third words. Presence of this parameter indicates that an input symbiont file is being defined and the symbiont file will redirect symbiont accesses to the program deck in the specified disk file.

Job number is the word wherein the system will post the assigned job number when an input symbiont file is defined.

When the status option is specified, job number identifies the job for which status is being requested.

Job status is the word wherein the system will post the status of the file whose identification is specified in the "job number" parameter. Job status may also be requested when defining an input symbiont file. Job status may be one of the following:

0 = completed

1 = running

2 = waiting for execution

3 = does not exist

Jobs Waiting is the word wherein the system will post the number of jobs ahead of the specified job number. The jobs waiting parameter may be requested on a status request or when an input symbiont file is being defined.

TYC indicates the type of completion of the requested service.

account name address is the address of a two word data block that contains the account name in EBCDIC, extended to eight characters with trailing blanks. If the account name is all blanks or numeric zero, it is assumed to be unspecified.

The following defaults for area and account names apply:

1. If neither account name nor area name is specified, the calling task's account is used, and the area may be any public area. This provides the simplest specification, and the user need not be concerned with the possibility of name conflicts other than within his own account.

2. If the account name is specified, but the area name is not, the file may be in any public area. This provides for area-independent file specification.

3. If the area name is specified but the account name is not, the system account will be used. This case provides compatibility for code written before the addition of file account names.

PRINT AND TYPE FUNCTIONS

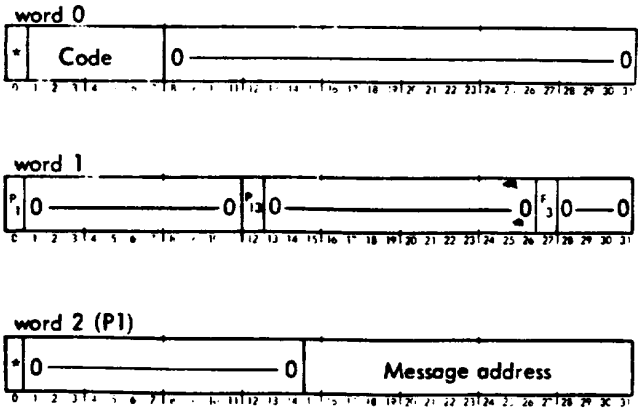
PRINT,TYPE The PRINT function causes the system to list the user's message on the listing log device (operational label LL). The TYPE function causes the system to list the user's message on the operator console device (operational label OC). These functions are reentrant and available to foreground programs. Error and abnormal conditions resulting from these functions are ignored.

Print and Type may be performed without a wait for completion, but the user is warned that changing the output buffer after return from such a request may result in the output message being modified.

Calls for these functions are of the form

CAL1,2 address

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



optional (P13)

where

Word 0

Code is X'01' for Print and X'02' for Type.

Word 1

P_1 is the message address parameter presence indicator (P_1 = 1).

P_{13} is the time interval parameter presence indicator (0 means absent; 1 means present).

F_3 is the wait indicator (0 means no-wait; 1 means wait for I/O completion).

Word 2

Message address is the address of the first word of the message. Note that the first byte of the first word of the message must specify the number of characters to be listed, up to a maximum of 132 characters for a line printer and 85 characters for a typewriter. The address may be indirect and/or in a register. If the message is in registers, the wait bit must be set.

Time interval is the maximum number of seconds that the caller will wait for the CALL to complete. If not specified the CALL will not be timed-out. Note that the I/O system may time out a device but this is independent from the use of the "Time Interval" parameter.

DEVICE

The DEVICE/file Mode function is used to set the following parameters:

Modes (MOD, ASC, D/P) in the addressed DCB.

Record Size (RSZ) in the addressed DCB and in the RFT entry if the DCB is assigned to an output file.

File Organization in the assigned file's RFT entry if the DCB is assigned to an output file.

Granule Size in the assigned file's RFT entry if the DCB is assigned to an output file.

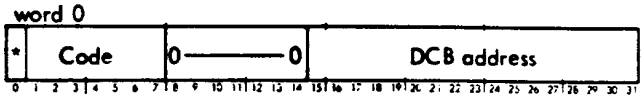
The parameters set in the RFT entries for permanent files will be written into the file directory entry for the file when the file is closed. Thus, this function defines the parameters for permanent disk files.

This function is of the form

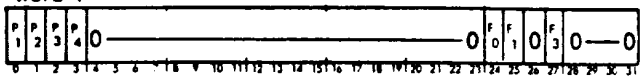
CAL1, 1 address

where address points to word 0 of the FPT below.

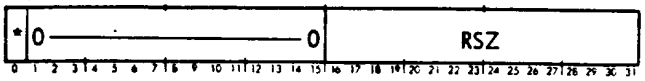
word 0



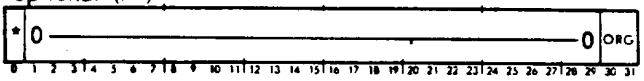
word 1



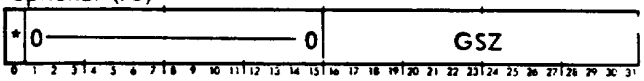
optional (P1)



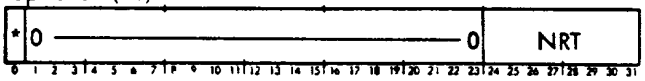
optional (P2)



optional (P3)



optional (P4)



where

Word 0

Code is X'22' for Device/File Mode.

DCB address is the address of the associated DCB.

Word 1

P_1 is the record size parameter presence indicator (0 means absent; 1 means present).

P_2 is the file organization parameter presence indicator (0 means absent; 1 means present).

P_3 is the granule size parameter presence indicator (0 means absent; 1 means present).

P_4 is the number of retries parameter presence indicator (0 means absent; 1 means present).

F_0 1 means ASCII translation, 0 means EBCDIC.

F_1 1 means unpacked format (7T) or 800 bpi (9T selectable density); 0 means packed format or 1600 bpi.

F_3 1 means BIN; 0 means BCD.

Word Options

RSZ is the maximum record size specification, in bytes.

Word Options (cont.)

ORG is the file organization type:

- 00 for unblocked
- 01 for blocked
- 10 for compressed

GSZ is the granule size in bytes.

NRT is the number of error recovery tries.

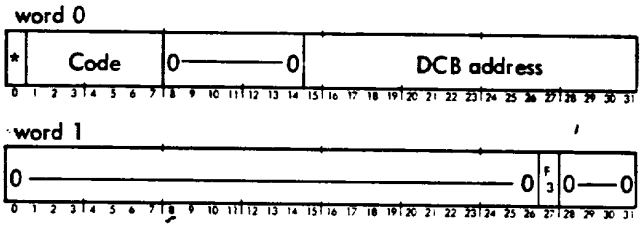
DEVICE CONTROL FUNCTIONS

The DEVICE Control functions cause the system to set the vertical-format-control or direct record control indicator of the specified DCB to 1 or to 0.

This call has the form

CAL1,1 address

where address points to word 0 of the FPT below.



where

Word 0

Code is X'05' for DEVICE/Format Control.
X'0B' for DEVICE/Direct Record Control.

DCB address is the address of the associated DCB.

Word 1

F3 is the control specification (0 means no format control or no direct record control; 1 means format control or direct record control).

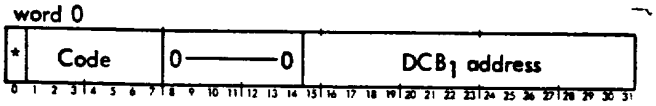
CHECK CORRESPONDENCE OF DCB ASSIGNMENTS

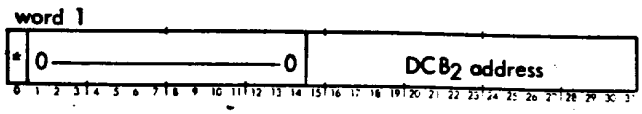
CORRES The CORRES system call provides the user's program with a means of determining if two open DCBs have been assigned to the same physical device or file. If the assignment of the two DCBs are equal, register 8 is returned with a value of 1. If any error is found in the DCBs while processing the CAL, if the assignments are found to be unequal, or if either DCB is not OPEN, a zero is returned.

The CORRES function call is of the form

CAL1,1 address

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





where

Word 0

Code is X'2B' for the CORRES function.

DCB_1 address is the address of the first DCB that is to be checked for assignment correspondence with DCB_2.

Word 1

DCB_2 address is the address of the second DCB that is to be checked for assignment correspondence with DCB_1.

ASSIGN (Formerly Set Device/File/Oplb Index)

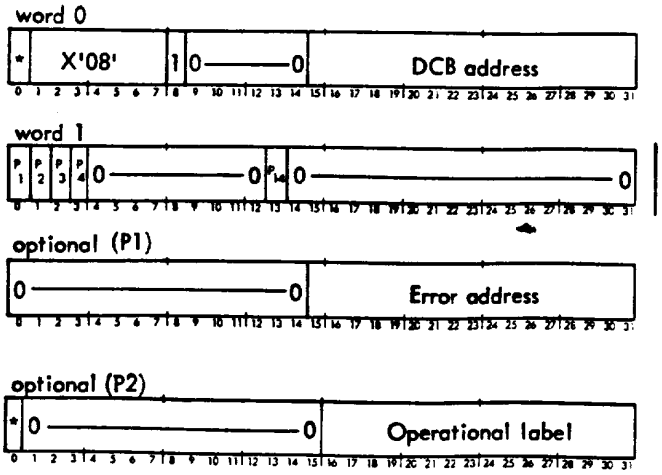
This service call is used to modify the I/O medium in the specified DCB. The DCB being modified must be closed. No action will be taken if the DCB is open. When an I/O medium may be identified by either EBCDIC name or system table index, the EBCDIC name is used if the size of the DCB permits. If the size of the DCB is too small to allow the specified assignment by either name or index, the service returns an error, without changing the DCB.

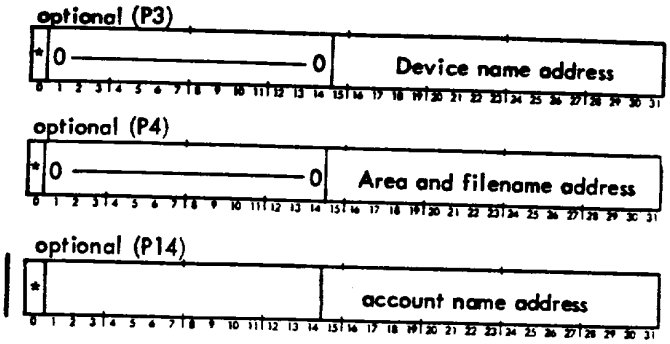
When used in conjunction with the DEVICE (File Mode and Format Control) service call, this function provides the ability to dynamically assign a DCB to a file, device, or operational label.

Calls for this function are of the form

CAL1,1 address

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





where

Word 0

X'08' specifies the ASSIGN call.

DCB address is the address of the associated DCB.

Word 1

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_2 is the operational label parameter presence indicator (0 means absent; 1 means present and the specified DCB will be assigned to an operational label).

P_3 is the device name address parameter presence indicator (0 means absent; 1 means present and the specified DCB will be assigned directly to a device).

P_4 is the area and file name address parameter presence indicator (0 means absent; 1 means present and the specified DCB will be assigned to a RAD file).

P_{14} is the account name address parameter presence indicator (0 means absent; 1 means present).

Parameter bits (P2, P3, and P4) are scanned in left to right order and the first nonzero bit terminates the scan (i.e., other nonzero bits are ignored).

Word Options

Error address is the address of the entry to the user's routine that will handle error and abnormal conditions.

Operational label is the operational label to which the DCB will be assigned in bits 16-31 in EBCDIC.

Device name address is the address of a two-word data block that contains the name of the device to which the DCB will be assigned. The device name is in EBCDIC, left-justified (i.e., CRA03).

Area and filename address is the address of a three-word data block that contains the area and file name to which the DCB will be assigned. The first word contains the area name in bits 16-31 in EBCDIC. The second and third words contain the file name in EBCDIC, left-justified with trailing blanks.

account name address is the address of a two-word data block that contains the account name in EBCDIC, extended to eight characters with trailing blanks.

The ASSIGN service does not provide defaults for area and account names. It inserts only what is requested.

GETASN (formerly Get Device/File/Oplb Name)

This service call is used to obtain the device, file, or operational label name to which the specified DCB is assigned and information about the device/file.

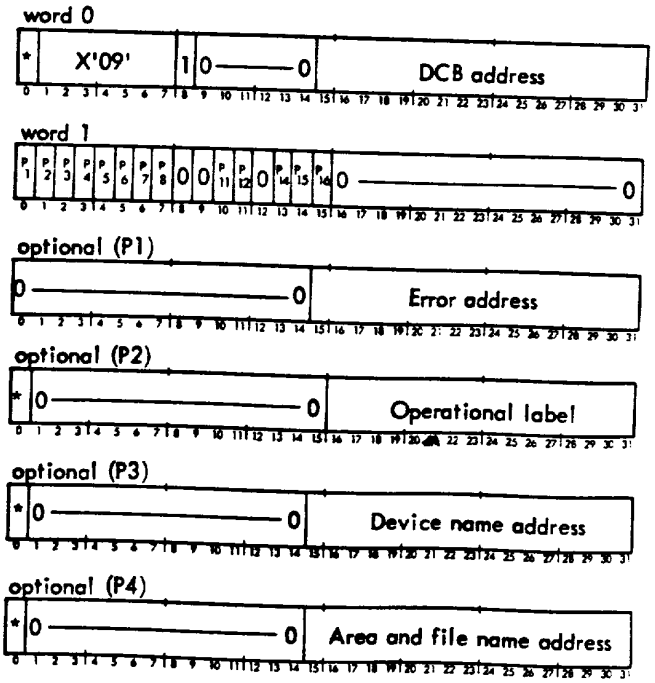
The call for this function is of the form

CAL, 1 address

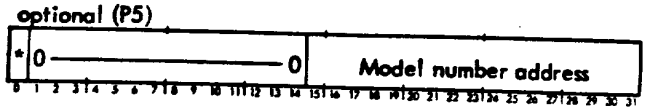
where address points to word 0 of the FPT shown below.

Note that if the requested optional parameter is not applicable, the post word(s) will not change. For example, if the call requests a file name to be posted but the DCB is assigned to a device, then the area and file name data block will not be modified.

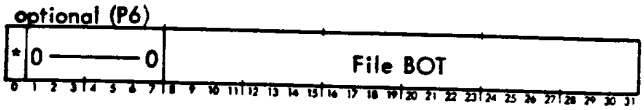
The DCB need not already be open, but will be left open on completion.



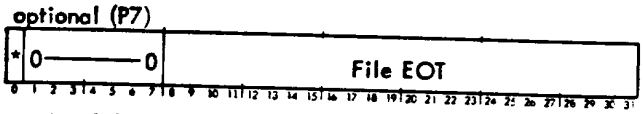
optional (P5)



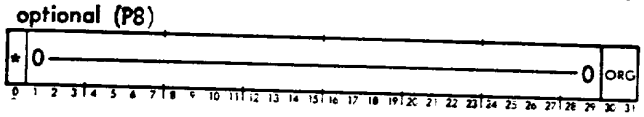
optional (P6)



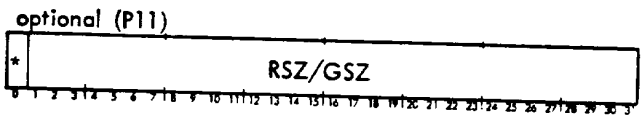
optional (P7)



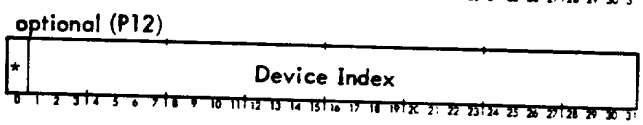
optional (P8)



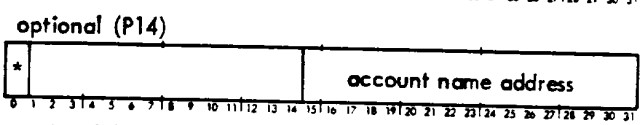
optional (P11)



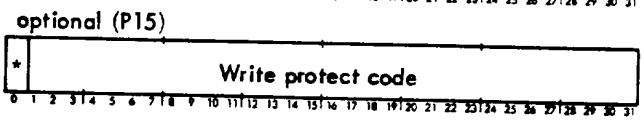
optional (P12)



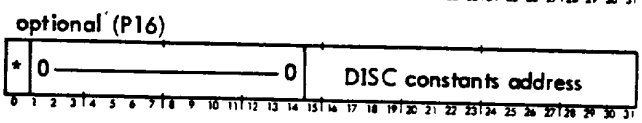
optional (P14)



optional (P15)



optional (P16)



where

Word 0

X'09' is the code that specifies GETASN call.

DCB address is the address of the associated DCB.

Word 1

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_2 is the operational label parameter presence indicator (0 means absent; 1 means present).

P_3 is the device name address parameter presence indicator (0 means absent; 1 means present).

P_4 is the area and file name address parameter presence indicator (0 means absent; 1 means present).

P_5 is the model number address parameter (0 means absent; 1 means present).

P_6 is the file BOT parameter presence indicator (0 means absent; 1 means present).

P_7 is the file EOT parameter presence indicator (0 means absent; 1 means present).

P_8 is the file organization parameter presence indicator (0 means absent; 1 means present).

P_{11} is the record size/granule size parameter presence indicator (0 means absent; 1 means present).

P_{12} is the device index parameter presence indicator (0 means absent; 1 means present).

P_{14} is the account name address parameter presence indicator (0 means absent; 1 means present).

P_{15} is the write protect code parameter presence indicator (0 means absent; 1 means present).

P_{16} is the DISC constants address parameter presence indicator (0 means absent; 1 means present).

Word Options

Error address is the address of the entry to the user's routine that will handle error and abnormal conditions.

Operational label is the word wherein the system will post the name of the operational label to which the DCB is assigned. The operational label name will be posted in bits 16-31 in EBCDIC.

Device name address is the address of a two-word data block wherein the system will post the device name in EBCDIC, left-justified. If the specified DCB is assigned to a disk file or to an operational label which, in turn, is assigned to a disk file, the address of the disk containing the file will be posted.

Area and file name address is the address of a three-word data block wherein the system will post the area and file name. The area name is posted in the first word of the data block in bits 16-31 in EBCDIC. The file name is posted in the second and third words of the data block, left-adjusted with trailing blanks.

Model number address is the address wherein the system will post the device model number in EBCDIC (i.e., 7446). If the DCB is assigned either directly or indirectly to a disk file, the model number of the disk containing the file will be posted.

File BOT is the word wherein the system will post the beginning sector number of the file to which the specified DCB is assigned. The sector number is relative to the start of the disk. If the DCB is assigned to an area (file name zero), the number of the first sector in the area will be posted in the BOT word.

File EOT is the word wherein the system will post the last sector number of the file to which the specified DCB is assigned. The sector number is relative to the start of the disk. If the DCB is assigned to an area, the number of the last sector in the area will be posted.

ORG is the word wherein the system will post the file organization type:

00 for unblocked

01 for blocked

10 for compressed

RSZ/GSZ is the word wherein the system will post the record size or granule size of a disk file. Granule size will be posted if the file is a random access file; otherwise the record size will be posted.

Device Index is the word wherein the system will post the device index.

account name address is the address of a two word data block to contain the account name in EBCDIC, extended to eight characters with trailing blanks.

Write Protect Code is the word wherein the system will post the write protection code for the file's area.

where

0 is Public

1 is Background

2 is Foreground

3 is System

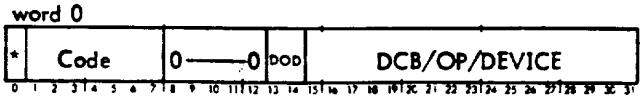
5 is IOEX

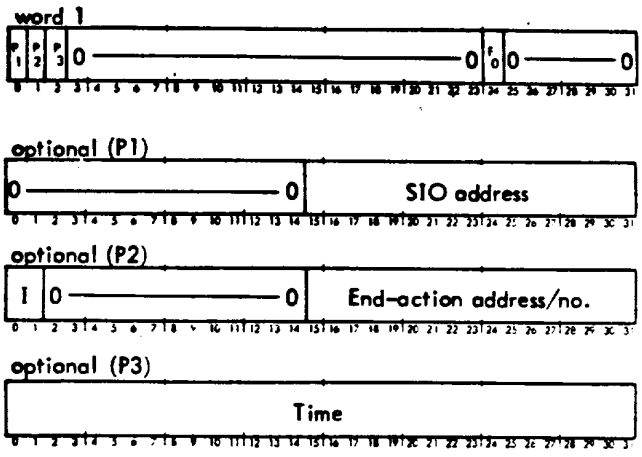
DISC Constants is the address of a three-word data block wherein the system will post the Tracks per Cylinder, Sectors per Track, and Words per Sector of the disk on which the file is allocated.

IOEX IOEX calls (available to primary tasks only) are of the form

CAL1,5 address

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





where

Word 0

Code = X'12' for SIO.
= X'13' for TIO.
= X'14' for TDV.
= X'15' for HIO.

DOD = 00 if DCB address is given.
= 01 if an operational label index is given.
= 10 if a device index is given.

DCB/OP/DEVICE contains the DCB address, operational label index, or device index as specified.

Word 1

P_1 is the SIO address parameter presence bit (0 means absent; 1 means present).

P_2 is the end-action parameter presence bit (0 means absent; 1 means present).

P_3 is the time-out presence bit (0 means absent; 1 means present).

$F_0 = 1$ specifies that I/O is to be allowed only if the device has been deactivated (i.e., marked "down"); this bit is for diagnostic-program use.

= 0 specifies that I/O is to be allowed only if the device has not been deactivated; normal program usage.

Word Option

SIO address is the address of the IOP command doubleword and need be present only when the call is an SIO request.

I, End-action address/no. I indicates the contents of the End-action address number field.

Value 0 indicates end-action address.

Value 1 indicates interrupt number.

Value 2 indicates interrupt label.

Value 3 indicates completion signal address.

Time is the number of seconds before the SIO is considered timed out and is meaningful only for queued requests. An HIO will be performed followed by end-action if requested. If the time-out increment is zero or not present the SIO will not be timed out.

Condition code 3 is set and condition codes 1, 2, and 4 are reset if the IOEX code is assembled out or if the caller is a secondary task. If an error is detected during the CAL processing, no change is made to the user's condition codes. CAL processing will be terminated and control will be transferred to the BADCAL (TRAP50) processor.

TIO and TDV instructions are executed immediately and the condition codes and status are returned as shown in Table 14.

SIO and HIO instructions may or may not be executed immediately, depending on whether or not the device has been preempted. The DEVICE PREEMPTION and DIRECT I/O EXECUTION sections in this chapter describe device preemption and the effect it has on SIO and HIO requests.

For IOEX function status returns on a device that is an IOEX device (preempted device and can be used by IOEX only), it is necessary to refer to the appropriate computer and device reference manuals. This is required because, for preempted devices, true hardware status and condition codes are returned and these may vary from machine to machine. The condition codes and status returned to the user are shown in Table 14.

Table 14. IOEX Function Status Returns

Operation	Major Status	Condition Codes		Register 8	Register 9
		Queued IOEX	Preempted Device		
SIO	IOEX not present or caller is background.	0010	0010	Unchanged	Unchanged
	Device is down (SIO not accepted).	1000	1000	Unchanged	Unchanged
	Request accepted.	0000	00--[t]	Current command address	X'10000000'
	Other	None	----	----	----
HIO	IOEX not present or caller is background.	0010	0010	Unchanged	Unchanged
	Request accepted.	0000	00--	Current command address	X'10000000'
	Other	None	----	----	----
TIO and TDV	IOEX not present or caller is background.	0010	0010	Unchanged	Unchanged
	Other	None	----	----	----

[t] Dashes indicate true hardware status and condition codes returned. The appropriate computer and device reference manuals should be consulted to determine the meanings of the condition code settings and the contents of the status register.

The command pairs for an SIO operation are not checked for validity and therefore must be coded very carefully. The flags in the command pairs may be set according to the needs of the user, however, if device preemption is not being used and the IOEX request goes through the I/O queue, then the flags must always be set in such a way that only one interrupt occurs for the request. A recommended set of flags for an SIO request with one command pair is:

IOP Command Doubleword Bit	Function	Bit State
32 (DC)	Data Chain	0
33 (IZC)	Interrupt at Zero Byte Count	0
34 (CC)	Command Chain	0
35 (ICE)	Interrupt at Channel End	1
36 (HTE)	Halt on Transmission Error	1
37 (IUE)	Interrupt on Unusual End	1
38 (SIL)	Suppress Incorrect Length	1
39 (S)	Skip	0

If command chaining is to be used, bit 34 (CC) will be set and bit 35 (ICE) must be reset in all but the last command pair.

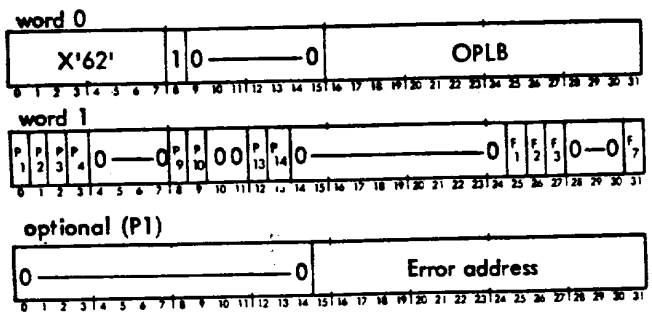
When I/O interrupts occur as a result of IOEX (SIO only), end-action is initiated as requested in the FPT (see I/O END ACTION section in this chapter).

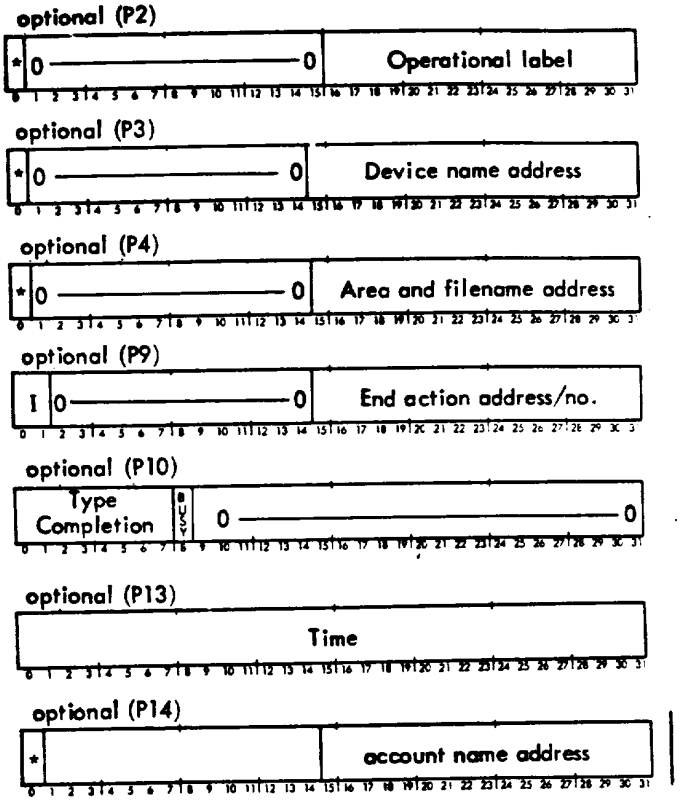
STDLB This function call is used to acquire or release the specified resource for a job and/or change the assignment of an operational label. The call is available to both foreground and background users. It is an asynchronous service when the enqueue option is specified; it must be followed by a CHECK function if the no-wait option is requested. Primary users may not use the enqueue option with wait.

The STDLB call is of the form

CAL1,7 address

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





where

Word 0

X'62' is the code to call the STDLB function.

OPLB is the name of the associated operational label in EBCDIC.

Word 1

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_2 is the operational label parameter presence indicator (0 means absent; 1 means present).

P_3 is the device name address parameter presence indicator (0 means absent; 1 means present).

P_4 is the area and file name address parameter presence indicator (0 means absent; 1 means present).

P_9 is the end-action parameter presence indicator (0 means absent; 1 means present).

P_{10} is the completion code parameter presence indicator (0 means absent; 1 means present).

P_{13} is the timeout value parameter presence indicator (0 means absent and no timeout will be done; 1 means present and the CAL will be timed out).

P_{14} is the account name address parameter presence indicator (0 means absent; 1 means present).

F_1 is the enqueue option indicator (0 means do not enqueue on resource; 1 means enqueue on resource).

F_2 is the release option indicator (0 means do not release a previously acquired resource; 1 means release a previously acquired resource).

F_3 indicates to wait for service to be completed prior to returning from the CAL.

$F_7 = 1$ is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

Word Options

Error address is the location to return to if immediate errors are detected.

Operational label is the operational label to which the OPLB will be assigned in bits 16-31 in EBCDIC. If bits 16-31 are zero, the OPLB is assigned to the null device.

Device name address is the address of a two-word data block that contains the name of the device to which the OPLB will be assigned. The device name is in EBCDIC, left-justified (i.e., CRA03). If the two word data block is all zeros, the OPLB is assigned to the null device.

Area and file Name Address is the address of a three-word data block that contains the area and file name to which the OPLB will be assigned. The first word contains the area name in bits 16-31 in EBCDIC. The second and third words contain the file name in EBCDIC, left-justified with trialing blanks. If the three word data block is all zeros, the OPLB is assigned to the null device.

end-action is as follows (applies only if enqueue requested):

I = 00 end-action is an address to BAL from STDLB post.

I = 01 interrupt address is to be triggered by post.

I = 10 interrupt label is to be triggered by post.

Type Completion contains the code describing the disposition of the service, including any error codes.

BUSY busy bit is set to 1 when the service call is initiated and reset to 0 when the service is completed.

Time is the maximum number of seconds that the requestor will wait for STDLB to complete its enqueue for the requestor. If not specified or zero, STDLB will not time out the enqueue.

account name address is the address of a two word data block that contains the account name in EBCDIC, extended to eight characters with trailing blanks. If the account name is numeric zero or all blanks, it is assumed to be unspecified.

Parameter presence indicators (P2, P3, and P4) are scanned in left to right order and the first nonzero indicator is the one used. If all three parameter presence indicators are zero, the operational label is assigned to the null device.

The following defaults for area and account names apply:

1. If neither account name nor area name is specified, the calling task's account is used, and the area may be any public area. This provides the simplest specification, and the user need not be concerned with the possibility of name conflicts other than within his own account.

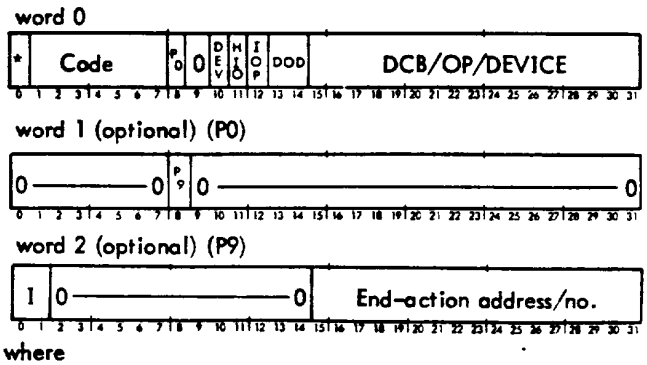
2. If the account name is specified, but the area name is not, the file may be in any public area. This provides for area-independent file specification.

3. If the area name is specified but the account name is not, the system account will be used. This case provides compatibility for code written before the addition of file account names.

STOPIO,STARTIO These calls are of the form

CAL1,5 address

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



where

Word 0

code = X'10' for stop all system I/O.

= X'11' for start all system I/O.

= X'0E' for stop all background I/O.

= X'0F' for start all background I/O.

= X'16' for deactivate (dedicate to diagnostic use) and abort all I/O.

= X'17' for activate and allow new I/O.

P0 = 1 if more parameters follow and indicates the presence of word 1.

HIO = 0 for no HIO; = 1 for HIO (abort active request).

IOP, DEV = 0, 0 reserve entire controller(s).

= 0, 1 reserve device.

= 1, 0 reserve entire IOP.

DOD = 00 DCB address is given.

= 01 operational label index is given.

= 10 device index is given.

DCB/OP/DEVICE contains the DCB address, operational label index, or device index as specified.

Word 1

P_9 is the preemption address (end-action) parameter presence indicator (0 means absent; 1 means present). A stop all system I/O call with P_9= 1 means the device will be dedicated to IOEX. P_9 has no meaning except in a stop all system I/O call.

Word 2

I, End-action address/no. I indicates the contents of the End-action address number field.

Value 0 indicates end-action address.

Value 1 indicates interrupt number.

Value 2 indicates interrupt label.

Value 3 indicates completion signal address.

Condition code 3 is set and condition codes 1, 2, and 4 are reset if any of these calls are made by a secondary task.

All condition codes are reset if a normal return is made from the call but no change is made to the condition code if an error is encountered during the CAL processing.

5. USER-TASK SCHEDULING AND OPERATION

SCHEDULING AND LOADING PROGRAMS

The Overlay Loader links one or more relocatable object modules to form a load module, which is an absolute representation of the program. The load module is created as a disk file and consists of a header and an absolute memory image of the various program segments. The load module header contains the program parameters used by the Root Loader for loading the program root.

LOADING AND TERMINATING FOREGROUND SECONDARY TASKS

Foreground secondary tasks are loaded into memory and prepared for execution through the use of the INIT system call or the INIT key-in. This process involves the following operations:

1. The INIT service acquires the necessary control table entries to establish the new task in the system. The task is then placed in a CP-R dispatch queue and will begin execution in the task initialization run-time loader.

2. When the task is scheduled for execution (in normal priority sequence) the run-time loader performs the following steps.

   - Acquires temporary work space from the Task Reserved Pages segment.
   - Opens the file containing the absolute load module and reads the load module header.
   - Acquires memory for the root segments and reads them into memory.
   - Tests for segments that are to be loaded with the root and loads them into memory.
   - Satisfies any Public Library requirements.
   - When the task is completely initialized in memory the temporary work space is released and the original INIT service is posted as complete.
   - Control is then transferred to the task's normal entry point unless the original INIT service requested the STOP option or the DEBUG option.
   - If the STOP option was requested the task is placed in a 'stopped' state awaiting a START service call at which time the tasks normal entry point will get control.
   - If the DEBUG option was requested the CP-R debug service will be given control.

The entire task initialization sequence is executed in the context of the task that is being loaded. Thus, multiple simultaneous task-loading operations may be ongoing in which case the tasks involved compete for system resources (CPU time, memory, etc.) based strictly on their priority.

Secondary task termination may be effected by any of the following:

- TERM service call (from the task itself).
- EXTM service call (from any task).
- EXTM key-in.
- KJOB service call.
- KJOB key-in.
- Failure of the task to properly process an error or abnormal condition from a service call or trap condition.

Task termination takes place in the context of the terminating task. The process involves the following steps.

- Waiting for any ongoing I/O operation to complete.
- Closing any open data files.
- Releasing memory resources.
- Releasing all internal system-task controls which has the effect of deleting the task from the system.
- If the terminated task was the last task in a job, the job will be deleted from the system (except for CP-R and background jobs).

QUEUEING PRIMARY FOREGROUND PROGRAM RUN REQUESTS

If the monitor contains the "run queuing" option, all RUN or INIT requests for primary load modules are placed in the Load Module Inventory Table and are given a priority. The priority can be supplied by the user; if absent, CP-R assigns the default (lowest) priority to the request. In addition, a sequence number is assigned to the request so that priority conflicts can be resolved. The priority and sequence determines the order in which the Foreground Root Loader, which is part of the CP-R Control Task, attempts to load the primary foreground programs. Such queued requests remain in the queue until loading is accomplished or until a release is issued.

LOADING AND RELEASING PRIMARY FOREGROUND PROGRAMS

Loading and initializing of a primary foreground program root is performed by the Foreground Root Loader, and involves the following steps:

1. If the "run queuing" option has been assembled into the monitor, the program to be loaded is selected by priority, where priority conflicts are resolved on a first queued, first to be loaded basis.

2. Opening the file containing the absolute load module.

3. Building a Load Module Inventory Table entry that contains the program name, core memory to be used by the program (root and all segments), and the public libraries required by the program.

4. Testing for required memory availability. If the "run queuing" option has not been assembled into the monitor and some portion of required foreground memory is busy, a message is typed, the Load Module Inventory Table entry is purged, and appropriate status is posted. If the "run queuing" option is in the monitor and some portion of required foreground memory is busy, a message is typed, but the entry remains in the table. If the foreground memory is not busy, the load process performs the following:

   - Loads the program root.

   - Transfers control to the program start address, taken from the load module header, where the user program initializes itself (connects tasks to interrupts, conditions interrupts, etc.). When initialization is completed, the user program performs an EXIT function call. The EXIT function will recognize that initialization of a primary foreground program has been completed and will transfer control back to the CP-R Control Task (the EXIT call does not cause an exit from the Control Task).

Steps 1 to 4 above are then repeated for each program to be loaded.

A primary foreground program root can be loaded by any of the following:

IRUN control command
IROV control command
IINIT control command
RUN key-in
RUN system call from a foreground task
INIT system call
INIT key-in

Since the root loading occurs at the level of the CP-R Control Task, primary foreground programs making RUN calls must give up the CPU (EXIT) before the load can be accomplished. Primary tasks can request the triggering of an interrupt at conclusion of the root load and initialization.

Release of a primary foreground program is also performed at the CP-R Control Task priority through the following steps:

1. Disarming all interrupts connected to tasks within the program.

2. Closing any open DCBs within the program to cause I/O run-down in addition to closing data files.

3. Purging the Load Module Inventory entry, which has the effect of marking the memory as not busy.

Release of a foreground program occurs as a result of either an RLS key-in, RLS system call, EXTM key-in, EXTM system call, KJOB key-in, or KJOB system call.

LOADING AND EXECUTING BACKGROUND PROGRAMS

The Task Initialization Run-time Loader also loads background programs as specified by control commands in the background job stream at the background priority level. Upon completion of the load, control is transferred to a background program at its start address. The background program terminates execution with an EXIT or ABORT service call. After terminating a background program, CP-R resumes processing the control commands from the background job stream.

TASK CONTROL BLOCK (TCB)

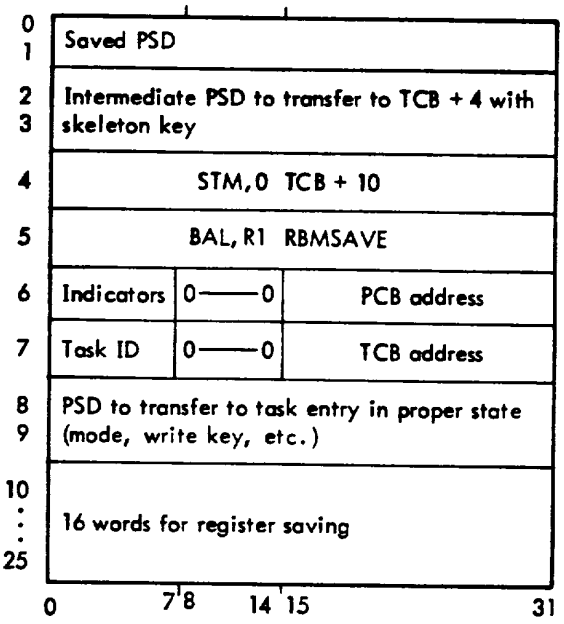
A Task Control Block must be associated with each centrally-connected primary task, and is used by the system to save the context of the interrupted task upon occurrence of the given task's interrupt. The TCB is in the user program (assembly language users must allocate and define their TCBs in the source code of their program). The FORTRAN compiler generates implicitly the TCBs needed for a real-time FORTRAN program.

Secondary tasks need not allocate TCB space as CP-R automatically provides the table for them.

PRIMARY TASK CONTROL BLOCK FORMAT

The assembly language user must allocate a TCB in the source code for each centrally-connected primary task in the program. Each TCB begins on a doubleword boundary and has a length of 26 words.

The CP-R CONNECT function fills in the TCB. When completed, a TCB has the following form:



where RBMSAVE is a system routine that saves the interrupted task's context in the TCB of the interrupting task and transfers control to the start address of the interrupting task.

Users must never alter any portion of a TCB.

PROGRAM CONTROL BLOCK (PCB)

The Program Control Block contains the program-associated parameters used by the CP-R system to provide service functions for the program. Every program, background and foreground, contains a PCB that is allocated and constructed by the Overlay Loader.

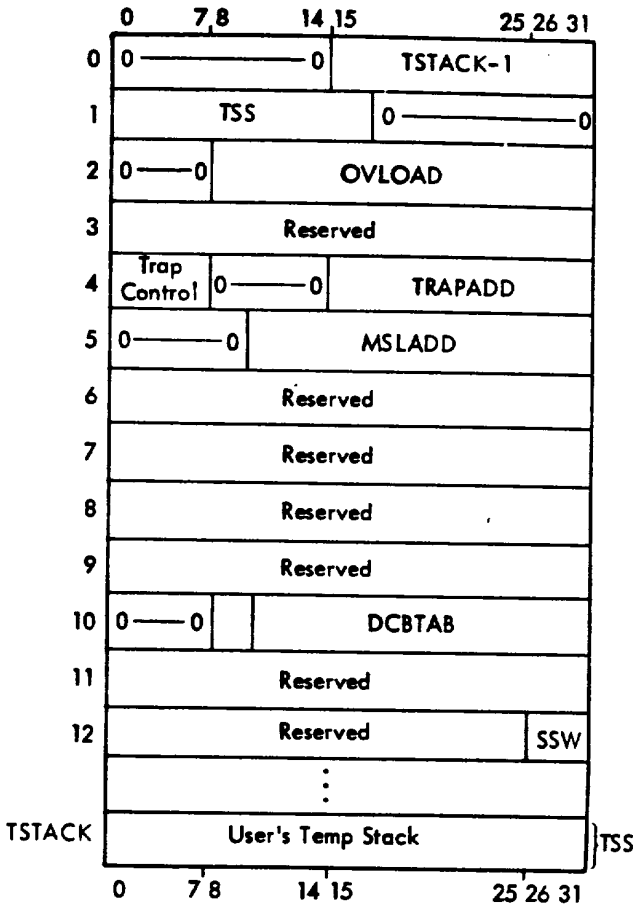
The Program Control Block defines the user Temp Stack to be used by a program. It also contains a pointer to a list of the DCBs associated with a program.

Since the Temp Stack is associated with the program rather than individual tasks, different tasks within a program should not use these stacks for data communication. Common storage can be used for communication between tasks or between occurrences of a given task.

PCB FORMAT

The PCB is built by the Overlay Loader from parameters specified on the IOLOAD control command.

The PCB is of the form



where

TSTACK is the address of the current top of the user's Temp Stack.

TSS indicates the size, in words, of the user's Temp Stack.

OVLOAD is the address of the table used by the segment loader to manage the program overlays.

Trap Control (bits 0-7) specifies how the various traps are to be handled. An explanation of these bits is given in the TRAP function description later in this chapter.

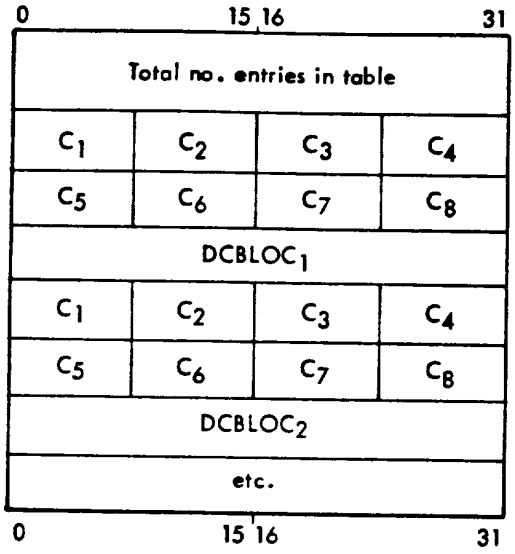
TRAPADD is the address of the user's routine that processes the various traps.

MSLADD is the address of the M:SL DCB, which is used to load overlay segments.

DCBTAB is the address of a table of names and addresses of all of the user's DCBs. This table has the form shown below.

SSW contains the user's sense switch settings. Bit 26 contains the setting of switch 1, etc. These switches can be set and reset by the user via the FORTRAN Library routines.

DCBTAB Format



where

C_1-C_8 indicates the EBCDIC name of the DCB, left-justified, with trailing blanks.

DCBLOC is the absolute address of the first word location of the DCB.

USER TEMP STACK

The user Temp Stack is a "push-down/pull-up" stack of memory locations that have been allocated by the Overlay Loader. It is required for user-coded trap control and subroutines that use Temp Stack storage (i.e., reentrant FORTRAN IV Library routines).

The user can manipulate the Temp Stack by push/pull stack instructions (PSW, PLW, PSM, MSP) that symbolically reference the external symbol U:PCB within their program (the Overlay Loader will satisfy the REF to U:PCB when the user's program is linked). The first doubleword of the Program Control Block is the stack pointer doubleword used in allocating (pushing) and releasing (pulling) blocks within the user Temp Stack.

The "push-down/pull-up" functions operate on a last-in, first-out basis, and these operations must be symmetrical in number and size. An attempt to push a block that is greater than the remaining stack space results in overflow. Similarly, an attempt to pull more out of the Temp Stack than had been previously pushed down would result in underflow. These conditions result in traps that may be handled by the user (see TRAP system call).

The size of the Temp Stack must be equal to or greater than the total number of temp cells required by the maximum number of nested routines using temporary storage; (i.e., if a FORTRAN routine needs 16 temp cells and it calls a routine that needs 19 cells, the total number of cells required would be 35). The number of cells required for trap handling CALs is 22; the reentrant FORTRAN IV Library subroutines require 148 temp cells for each task.

Primary tasks at different interrupt levels within the same program share the program's Temp Stack, and allocation must be sufficient to accommodate the maximum number of tasks that could be enabled at one time. When an executing task exits, it must restore the Temp Stack pointer to its original condition. This is particularly important in a primary foreground program where a Temp Stack is allocated for each program and not each task. Thus, if several tasks in the same program share the program's Temp Stack, the housekeeping of the Temp Stack pointer (e.g., symmetrical pushes and pulls) must be meticulous.

CP-R TEMP STACK

The CP-R Temp Stack is a "push-down/pull-up" stack of memory locations that have been allocated by the Overlay Loader. It is required for all service functions. The CP-R Temp Stack is reserved for the exclusive use of the monitor; users may not manipulate the stack since the stack pointer doubleword resides within the monitor in protected memory.

The size requirement of the CP-R Temp Stack is a function of the particular monitor service being utilized. The worst-case requirement is approximately 150 words. The default size is 150 words.

MASTER AND SLAVE MODES

Both primary and secondary foreground tasks can change their execution mode through MASTER and SLAVE system function calls. At entry to a primary task, the mode is set as specified by the function call that caused the connection.

For secondary tasks the operating mode at entry is slave mode and mapped. Secondary tasks are given master-protected Mode, Mapped, by the MASTER system call.

Note: Serious consequences can result from improper operation in master mode. Secondary tasks must never operate in unmapped mode.

OVERLAY SEGMENT OPERATIONS

Primary tasks control the loading of overlay segments through the SEGLOAD system call by explicitly naming the desired segment, by number. Background programs that have been linked with the Simplified Memory Management (SMM) option may also use the SEGLOAD call.

Secondary foreground tasks control their segment activity through the use of CP-R memory management service calls described in a later chapter.

TRAP HANDLING

CP-R provides standard processing of trap conditions. A user can either take advantage of the system processing or request that he himself handle certain trap conditions. Also, certain traps can be ignored. System trap handling involves aborting the task that is active at the occurrence of a trap, with the following exceptions:

1. An unimplemented instruction trap occurrence will result in the instruction being simulated if the simulation package is in the system. If simulation is impossible, the task will be aborted.
2. The user can mask out fixed-point arithmetic and decimal arithmetic traps either through system call (TRAP or JTRAP function) or at primary-task connection time (CONNECT, ARM, DISARM functions).
3. The user can mask out some floating-point trap occurrences through use of the LCF and LCFI instruction.
4. Any unmasked trap can be received by the active program. This is set up through the TRAP or JTRAP function call, wherein the user can specify the address of a routine to handle the various trap conditions. This address is kept on a per program or per job basis as opposed to a per task basis.

When the user trap routine receives control, the following items are stored in sequence in a 19-word block of the program's Temp Stack, starting on a doubleword boundary: the PSD and the 16 registers saved when the trap occurred, and a word containing the trap location (right-justified). Register 1 points to word 0 (first word of the PSD) in this block. If 19 words are not available in the user Temp Stack, the task is aborted.

The address of the user trap routine and the control bit for each trap is kept in the PCB and the JCB (Job Control Block, see Appendix H).

User trap handling or system trap handling is also invoked if an invalid parameter exists in an FPT for a system call that is unable to post the error condition. In this case, a code of X'50' will be posted in the last word of the user's Temp Stack if a user trap address is present in the PCB or JCB.

If the user has requested break control via the INT service call, the user Temp Stack will be set as described above and an X'51' code will be posted in the last word of the users Temp Stack before control is transferred to the user's break control routine.

Return from the user trap routine to the interrupted program is accomplished by the TRTN and TRTY function, which restores the context from the Temp Stack and returns to the location following the trapped instruction or to the trapped instruction itself, respectively.

CAL HANDLING

Memory locations X'48', X'49', X'4A', and X'4B' are the respective trap locations for the CAL1, CAL2, CAL3, and CAL4 instructions. CAL1 instructions are reserved for monitor services, but the CAL2, CAL3, and CAL4 trap locations may be connected (centrally or directly) to a user-defined routine by means of the CONNECT function call. User CAL service routines must reside in primary (unmapped) load modules, even when they are intended for use by secondary tasks.

Centrally connected CAL routines will be entered in the caller's map mode. The calling task's context (registers and PSD) is saved in the calling task's user Temp Stack. In addition, register 0 will contain the address of the caller's user stack control doubleword. Since centrally connected CAL service routines are entered in the map mode of the caller, but reside in unmapped memory, they must be in a one-to-one mapped area of any secondary task which uses them. Possibilities are the memory between the end of the monitor and the start of secondary task memory (X'6000') or an active fixed segment of the secondary task. Centrally connected CAL routines return to the calling task via the CALRTN function call.

Directly connected CAL service routines are responsible for saving and restoring the calling task's context. They are entered unmapped. If they do not need access to the memory of the calling task, they need not be concerned with map mode or memory residence. If they need to access the memory of a secondary caller, however, they must not only reside in one-to-one mapped memory for the caller, they must also determine that they were called from a mapped context and set map mode for their own execution.

When a CAL; is executed, the effect is immediate; that is, traps function more like BALs than interrupts, except that the environment is saved for centrally connected traps, but is not saved for traps that are directly connected.

RETURN FUNCTIONS

CP-R provides users with the following return functions:

EXIT is used by background programs at the normal completion of a background program. EXIT is used by foreground programs when a centrally-connected primary task has concluded the processing of an interrupt. An Exit call from a primary task causes the system to restore the context of the interrupted task, clear and arm the highest priority active interrupt, and return to the point of interrupt. Secondary tasks should not use EXIT but should use STOP instead.

ABORT is used by background programs to cause the system to abort the job containing the program. The system will abort the background program and type a message on the operator console (OC) that the job was aborted and give the address of the ABORT call. Unless an !ATTEND command was present in the background job, the system will read and ignore all records from the C device until the next !JOB card is encountered. If an ABORT call is made from a foreground program, the program will be terminated.

STOP is used by both foreground and background secondary tasks when they wish to cease execution and wait for a START call from another task.

TERM may also be used by background programs at the normal completion of a background program. TERM is used by foreground programs to release and exit.

INTERRUPT CONTROL

CONNECTING AND DISCONNECTING PRIMARY TASKS TO INTERRUPTS

Interrupt connection may be accomplished through use of the CONNECT function call and disconnection through the DISCONNECT function call. While these calls are usually made during foreground program initialization (performed at the Control Task priority level), this is not a requirement. Connections may be made to any interrupt in the system except the clock pulse levels, the counter 4 equals zero level, the I/O level and any levels defined at SYSGEN for dispatchers, I/O deferral or remote communications handling. However, tasks connected at a higher priority than the I/O level can not request any I/O or any services which require I/O. A table of interrupts connected within a program is kept by the system. The table enables the foreground program release function to disarm and disable all interrupts associated with a foreground program. In calling for connection the user may specify the mode (master, slave) and the interrupt inhibit conditions that are to exist at entry to the specified task. Two types of connections are available, direct and central:

1. Direct connection results in immediate entry to the task upon occurrence of the interrupt. The task must ensure that the context is saved, as necessary, and restored upon exit. Directly connected tasks may not use any CP-R services.

2. Central connection results in entry to the task after the interrupted task context has been saved. The CONNECT function constructs the TCB so that the context save will occur. Exit from a centrally connected task is by EXIT, which will restore the interrupted task status, arm and clear the highest priority active interrupt, and return to the interrupted task.

   Central connection causes only register block zero to be saved, regardless of the block used in either the new or interrupted context. The user must provide register management if he uses other register blocks.

   To perform the connection, the system fills in the TCB previously shown. The PSD is constructed (TCB +8) to transfer control to the user in the proper mode and with the proper write key. An XPSD TCB instruction is placed in the proper interrupt location to complete the connection.

Details of the system calls concerned with connection are given later in this chapter under "System Function Call Formats".

ARMING, DISARMING, ENABLING, AND DISABLING

These functions can be performed by the ARM, DISARM, ENABLE, and DISABLE system function calls, which specify an interrupt by number or label. As options on ARM and DISARM, connection of the interrupt to a task can be performed, and/or a clock can be set to interrupt at specific intervals.

ARM, DISARM, ENABLE, and DISABLE functions can also be performed by operator request through the CINT key-in.

TRIGGERING OF INTERRUPTS

An interrupt can be triggered through a TRIGGER system function call. The interrupt to be triggered is specified by number or by label. TRIGGER calls may be made from any foreground task. An interrupt can also be triggered by operator request through the CINT key-in.

END-ACTION

Primary tasks may use end-action. Two types of end-action are possible.

1. The user provides an end-action address in the FPT. A transfer to this address will be made following the completion of the service. This end-action transfer is made by executing.

   BAL, 11 end-action address

   with the CPU in master mode, unmapped, and the posting level active.

   Return from the end-action routine must be made by

   B *11

   It should be noted that since end-action may be performed with the posting level high, any task whose priority is lower than that of the posting level is effectively disabled for the duration of the end-action.

   Since the end-action user can seriously degrade interrupt response for lower priority tasks, it is strongly recommended that this type of end-action not be used for applications where other techniques are satisfactory.

2. The user FPT contains either an interrupt number or interrupt label specifying a system interrupt. The system interrupt is triggered upon completion of the request. The task connected with the specified interrupt then performs the end-action function at the proper priority level. The user is responsible for connecting the interrupt and ensuring that it is armed and enabled.

The triggering service sets a flag in the TCB to indicate that the trigger has been performed. The EXIT routine interrogates this flag before performing the EXIT for centrally connected tasks. If the flag is set, the occurrence of the interrupt (previously lost by the triggering of an active interrupt) will be simulated. If more than one service completion can occur while a task is active (or prior to such a task becoming active), the user is responsible for checking all possible reasons for which an interrupt may have occured. (The flag bit in the TCB indicates one or more interrupt attempts.) Directly connected tasks using this type of end-action must assume the responsibility for solving problems of this type since there is no TCB in which a flag bit can be set.

SYSTEM FUNCTION CALL FORMATS

The following system function calls should not be attempted unless the user has been given control as follows:

1. The background or foreground program loader has transferred control to the usjer' start address.
2. The user's centrally-connected primary task has been given control upon the occurrence of the associated interrupt.

In all of the FPT formats which follow, an asterisk in bit 0 indicates that indirect addressing is permitted. That is, if bit 0 contains a 1, bits 15 through 31 contain the address of the parameter rather than the parameter itself.

Whenever a return is made to an abnormal or error address, the error or abnormal code will be in byte 0 of register 10 and the address of the location following the call will be in register 8.

Calls for which F3 (wait indicator) is not available are either immediate or synchronous. That is, control will never be returned to the caller until the service has been completed (See Appendix I for more detailed information). Primary task users should recognize that during any wait for completion of the service all lower priority tasks are blocked.

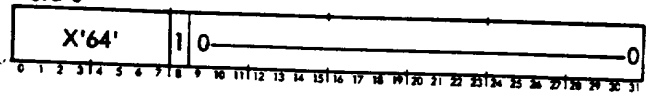
If an error is detected in the call and no error address is available, the situation will be handled similar to a trap and the user will be aborted unless he has elected to do his own trap handling.

KJOB This function call is available only to foreground and terminates a job by terminating every task in the job and deleting the job controls. KJOB has the format

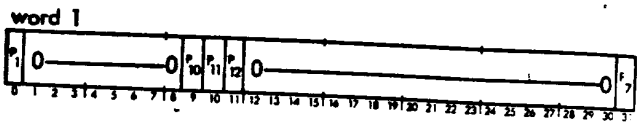
CAL1,7 address

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

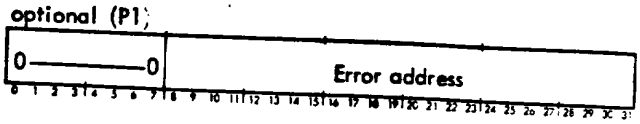
word 0



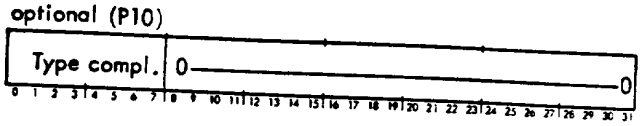
word 1



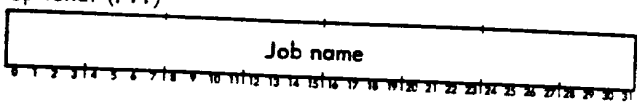
optional (P1)



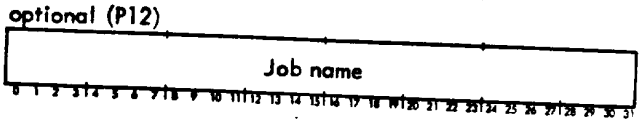
optional (P10)



optional (P11)



optional (P12)



where

Word 0

X'64' is the code for the KJOB call.

Word 1

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_{10} is the type completion parameter presence indicator (0 means absent; 1 means present).

P_{11}/P_{12} are the optional jobname-parameter presence indicators and are set to ones if a full eight-character job name is supplied.

F_7 is the abort override indicator. If set, control will be returned to calling instruction +1 if errors are detected and no error address parameter has been provided.

Word options

Error address (optional) is the location to return to if errors are detected (see word 3 description for completion codes).

Type compl. is set by CAL processing.

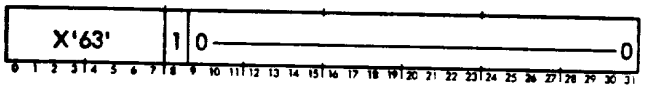
Job name is a four- or eight-character EBCDIC job name, left-justified and blank filled. If omitted, the caller's job is terminated.

SJOB This function call is available only to foreground and creates a foreground job by setting up job controls and building a job control block. SJOB is an immediate service that uses a standard FPT. It does not initiate any task within the job; instead, RUN or INIT must be used following SJOB to start a task. It has the format

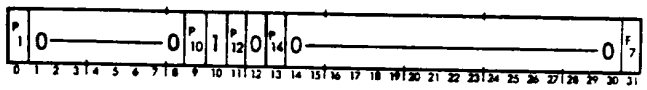
CAL1,7 address

where address points to word 0 of the FPT shown below

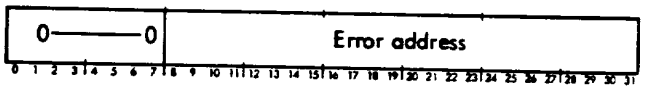
word 0



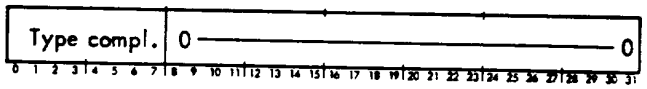
word 1



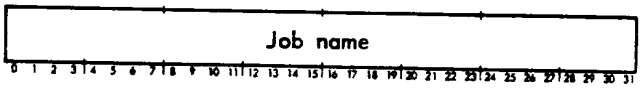
optional (P1)



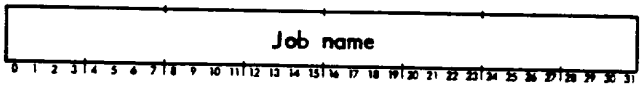
optional (P10)



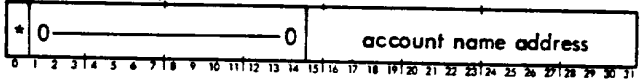
required (P11)



optional (P12)



optional (P14)



where

Word 0

X'63' is the code for an SJOB call.

Word 1

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_{10} is the type completion parameter presence indicator (0 means absent; 1 means present).

P_{12} is the optional jobname-parameter presence indicator and is set to one if a full eight-character job name is supplied.

P_{14} is the optional account and name parameter presence indicator (0 means absent; 1 means present) if omitted the callers account and name are used

F_7 is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

Word options

Error address (optional) is the location to return to if errors are detected (see word 3 description for completion codes).

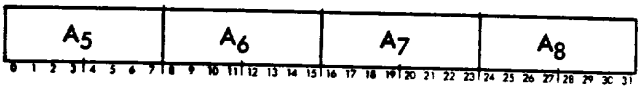
Type Compl. is set by CAL processing.

Job name is a four- or eight-character EBCDIC job name, left-justified and blank filled. This parameter is required.

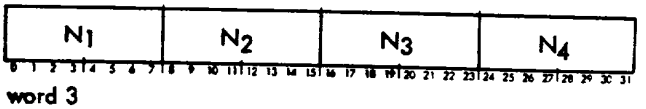
Account and name address is the address of a five word data block that contains the account and name to be associated with the JOB. The format of the data block is shown below.

word 0

word 1

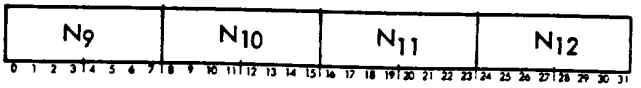


word 2



word 3

word 4



where

A_i are the EBCDIC characters naming the account.

N_i are the EBCDIC characters in the user name.

If the account and/or the name information is zero or blanks the callers account and/or name will be used instead.

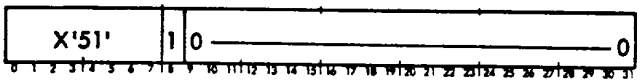
The SJOB CAL processor will validate the account and name against the AI file and if they are not found, will return an error code of X'7B' for illegal or invalid account or name.

SETNAME This function call allows the user to give the name of a load module to be used for execution of a task. SETNAME has the format

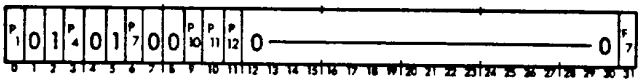
CAL1,7 address

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

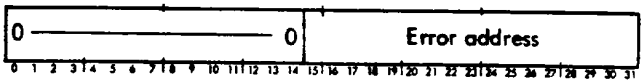
word 0



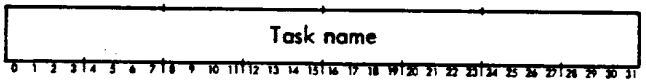
word 1



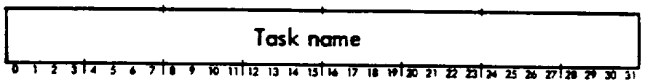
optional (P1)



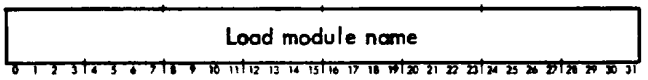
required (P3)



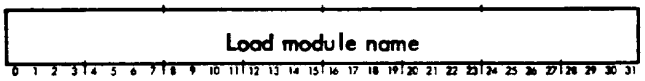
optional (P4)



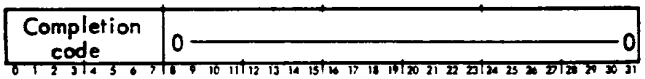
required (P6)



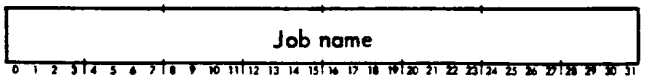
optional (P7)



optional (P10)



optional (P11)



optional (P12)

where

Word 0

X'51' is the code for the SETNAME call.

Word 1

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_4 is the task name parameter indicator (0 means absent; 1 means present). Only used if eight-character task names are being given.

P_7 is the load module name parameter indicator (0 means absent; 1 means present). Only used if eight-character load module names are given.

P_{10} is the completion code parameter presence indicator (0 means absent; 1 means present).

P_{11}-P_{12} are the job name parameter presence indicators (0 means absent; 1 means present). If four-character job names are given P_{11} must be set. If eight-character job names are given both P_{11} and P_{12} must be set.

$F_7 = 1$ is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

Word options

Error address is the return address to process any detected errors.

Task name is one or two words containing the 4 or 8 character (EBCDIC) task name whose load module equivalence is to be established. This parameter is required.

Load module name is one or two words containing the 4 or 8 character (EBCDIC) load module name that will be used to load the task whose task name is being specified. If load module name is all blanks, then task name specifies a prior equivalence that will be deleted.

Completion code indicates completion status of the function call.

Job name is one or two words containing the four- or eight-character (EBCDIC) job name in which the specified task name/load module name equivalence is to be made. If Job name is not present the equivalence will take place in the caller's job.

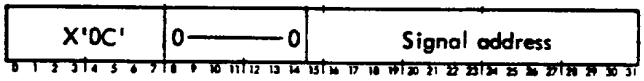
Tasks in progress are not affected by load-module name changes.

RUN This function call is available only to the foreground and initiates a primary task in the CP-R job. It has the format

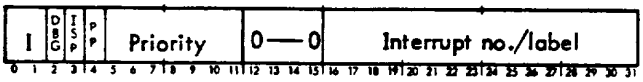
CAL1,5 address

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

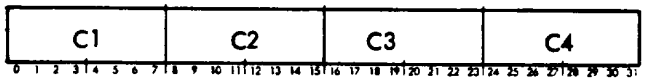
word 0



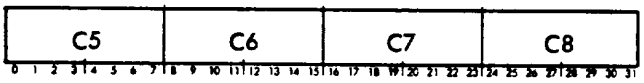
word 1



word 2



word 3



where

Word 0

X'0C' is the code for the RUN call.

Signal address is the address of a status word into which the system posts one of the following signals:

0 if the program was successfully loaded.

1 if the space was not available in the foreground-private memory area or if there was insufficient space in the Load Module Inventory (LMI) table to make entries for the public libraries needed by the program.

2 if the requested program did not exist in the FP area of the disk or if an I/O error occurred attempting to load the program.

3 if the program is already loaded.

4 if a previous request has been made to load the same program but the program is not yet loaded. In this case, the Foreground Root Loader is able to notify only the first requester when the program is loaded.

5 if the space was not available in the Load Module Inventory (LMI) table for the requested program.

6 if invalid attempt has been made to load a public library. Since public libraries are automatically loaded and released by the Foreground Root Loader, they cannot be loaded via a RJN call.

7 if no table space was available to load the load module header. The caller may reissue the call if desired.

The signal address cannot be a register (addresses 0 through F) and must be in the calling program's portion of memory. An invalid signal address results in control being transferred to the System or User Trap Handler.

The user should inspect the signal word upon return from the CAL, since some of the signals (3,4, and 5) are posted immediately by the RUN processor.

Note that an interrupt is triggered only if the RUN request is passed on to the Foreground Root Loader. That is, signals 3, 4, and 5 are returned immediately by the RUN processor and, in this case, no interrupt is triggered.

Alarms are output by the Foreground Root Loader for error conditions 1, 2, and 6.

Word 1

I indicates the contents of the Interrupt number/label field (0 for no interrupts; 1 for interrupt number; 2 for interrupt label). This interrupt is triggered by the Root Loader at the conclusion (successful or unsuccessful) of the root load and initialization.

DBG causes the program to run under control of DEBUG.

ISP inhibits all status posting after the attempt to load the program.

PP specifies whether or not the priority of the RUN request is present in bit positions 5 through 11. If PP = 1, priority is present; if PP = 0, it is not present.

Priority specifies the order in which the RUN request will be processed. The highest priority is 0; the lowest priority is X'7F'.

Words 2-3

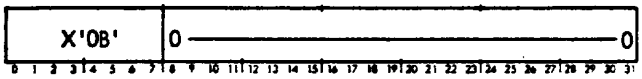
C_i are the characters in the name of the load module. The name is left-justified with trailing blanks.

RLS This function call (Release Primary Foreground Program) is available only to the foreground and releases a primary task in the calling job. It has the format

CAL1,5 address

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

word 0



word 1

word 2

where

Word 0

X'0B' is the code for the RLS call.

Words 1-2

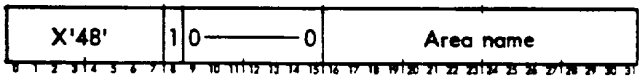
C_i are the characters specifying the name of the program. The name is left-justified and filled with trailing blanks. An invalid name results in a return with no action taken by the system.

INIT This function (initiate a new task) causes the named task to be read into memory and initiated. It has the format

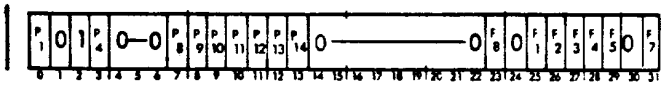
CAL1,7 address

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

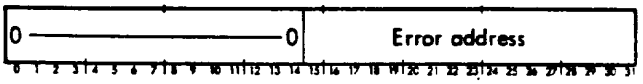
word 0



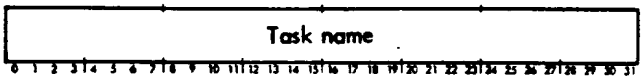
word 1



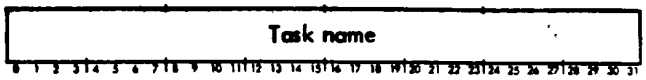
optional (P1)



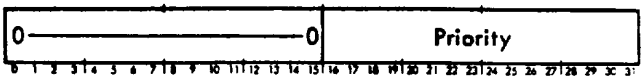
required (P3)



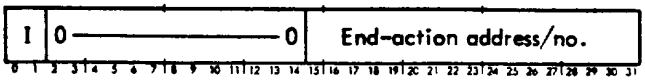
optional (P4)



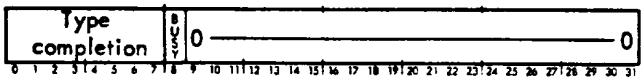
optional (P8)



optional (P9)

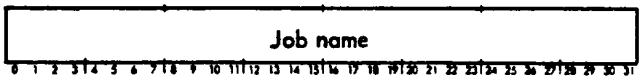


optional (P10)

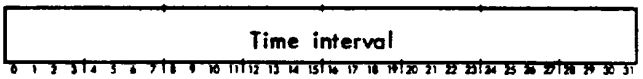


optional (P11)

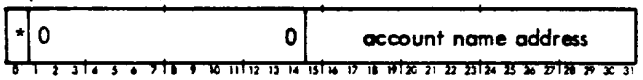
optional (P12)



optional (P13)



optional (P14)



where

Word 0

X'48' is the code to initiate the call.

Area name is the optional EBCDIC name of a disk file area for the load module. If Area name is all zeros, the disk area is unspecified.

Word 1

P_1 is the error address parameter presence indicator (0 indicates absent; 1 indicates present).

P_4 is the optional task name parameter presence indicator (0 indicates absent; 1 indicates present).

P_8 is the priority parameter presence indicator (0 means absent; 1 means present).

P_9 is the end-action address parameter presence indicator (0 indicates absent; 1 indicates present).

P_{10} is the type completion code parameter presence indicator (0 means absent; 1 means present).

P_{11}-P_{12} are the job name parameter presence indicators (0 means absent; 1 means present).

P_{13} is the time interval parameter presence indicator (0 means absent; 1 means present).

P_{14} is the account name address parameter presence indicator (0 means absent; 1 means present).

F_8 is the delete on post indicator and signifies that no check will be performed.

F_1=0 execute after load.

=1 leave in suspended (STOP) state after load.

F_2=0 task being initiated is primary.

=1 task being initiated is secondary.

F_3 wait for the task to be initiated before returning from CAL. Wait may be specified from secondary tasks only.

F_4=0 normal initiation.

=1 initiate under DEBUG control.

F_5=0 normal initiation.

=1 task is to be time-sliced.

F_7=1 is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

Word options

Error address is the return address for errors.

Task name is the name of the task to be loaded and initialized. It will be converted to a load module name by lookup in the Job Program Table (JPT). A nonmatch defaults to the task name being used for the load module name (required).

Priority indicates the priority that is to be given to the task being initiated. Priority is a four-digit hexadecimal number the leftmost two digits represent the CP-R dispatcher interrupt number minus X'4F', the rightmost two digits represent a CP-R software priority in the range X'0' - X'EF'. The default priority is the caller's priority.

I, End-action address/no. I indicates the contents of the End-action address/number field. End-action is allowed only for primary foreground tasks.

I=0 indicates an end-action address.

=1 indicates an interrupt number.

=2 indicates an interrupt operational label.

Type completion and BUSY indicate status of an initiate service. An initiate service is complete when the load module has been loaded and task controls established.

BUSY=0 indicates service is complete.

=1 indicates service is incomplete.

Job name is the four- or eight-character name of a job in which the task should be initiated. The caller's job is the default name.

Time interval is the maximum number of timer units allowed for the task initiation and loading. If no time interval is specified the service will not be timed out.

account name address is the address of a two-word data block that contains the account name in EBCDIC, filled to eight characters with trailing blanks. If the account name is all blanks or numeric zero, it is assumed to be unspecified.

The default area and account names for INIT differ from those of the other services, to provide compatibility to previous behavior.

1. If neither account nor area name is specified, area FP with a null account name is assumed, to provide backward compatibility.

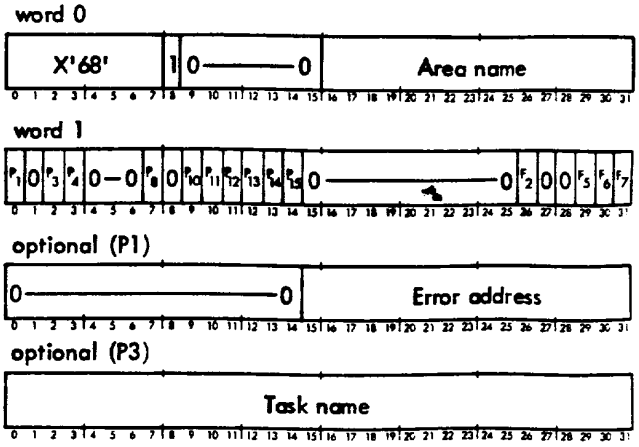
2. If the account name is specified but the area name is not, the file may be in any public area. (This is the same default as for other services.)

3. If the area name is specified but the account name is not, the system account name will be used. (This is the same default as for other services.)

SCHED This function (periodically schedule a task) causes an INIT call to be issued to a named task. It has the format:

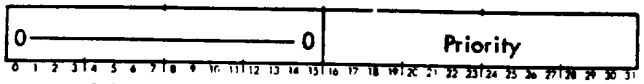
CAL1,7 address

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

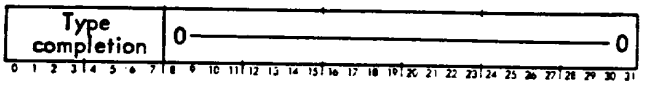


optional (P4)

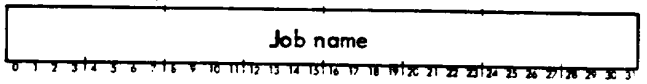
optional (P8)



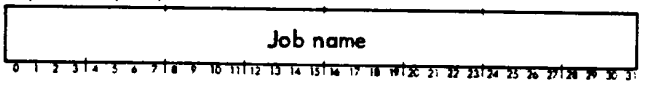
optional (P10)



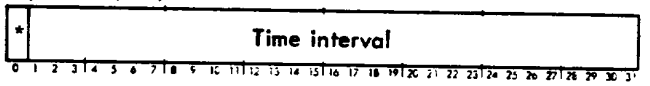
optional (P11)



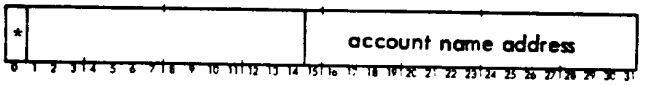
optional (P12)



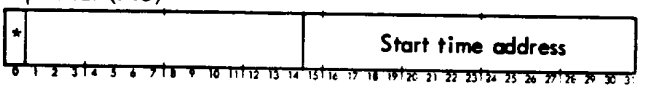
optional (P13)



optional (P14)



optional (P15)



where

Word 0

X'68' is the code to initiate the call.

Area name is the optional EBCDIC name of a disk file area for the load module. See "account name address" description for default values.

Word 1

P_1 is the error address parameter presence indicator (0 indicates absent; 1 indicates present).

P_3–P_4 are the task name parameter presence indicator (0 indicates absent; 1 indicates present).

P_8 is the priority parameter presence indicator (0 means absent; 1 means present).

P_{10} is the type completion code parameter presence indicator (0 means absent; 1 means present).

P_{11}–P_{12} are the job name parameter presence indicators (0 means absent; 1 means present).

P_{13} is the time interval parameter presence indicator (0 means absent; 1 means present).

P_{14} is the account name address parameter presence indicator (0 means absent; 1 means present).

P_{15} is the start time address parameter presence indicator (0 means absent; 1 means present).

$F_2 = 0$ task being scheduled is primary.
$= 1$ task being scheduled is secondary.

$F_5 = 0$ normal initiation.
$= 1$ task is to be time-sliced.

$F_6 = 0$ schedule the task.
$= 1$ abort (delete) the scheduling.

$F_7 = 1$ is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

Word options

Error address is the return address for errors.

Task name is the name of the task to be loaded and initialized. It will be converted to a load module name by lookup in the Job Program Table (JPT). A nonmatch defaults to the task name being used for the load module name. A nonexistent field defaults to the calling task.

Priority indicates the priority that is to be given to the task being initiated. Priority is a four-digit hexadecimal number, the leftmost two digits represent the CP-R dispatcher interrupt number minus X'4F', the rightmost two digits represent a CP-R software priority in the range X'0' – X'EF'. The default priority is the caller's priority.

Type completion indicates the type completion code returned by the SCHED call.

Job name is the four- or eight-character name of a job in which the task should be initiated. The caller's job is the default name.

Time interval is the period, in seconds, between successive INIT calls for the task. If the value is not an integral multiple of five, it is rounded up to the next higher integral multiple of five. If this parameter is zero or absent, the task will be INITed once, at the specified start time.

account name address is the address of a two-word data block that contains the account name in EBCDIC, filled to eight characters with trailing blanks. If the account name is all blanks or numeric zero, it is assumed to be unspecified.

The following defaults pertain to the combinations of disk area name and file account name:

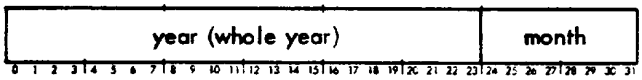
1. If neither account nor area name is specified, area FP with a null account name is assumed, to provide backward compatibility.

2. If the account name is specified but the area name is not, the file may be in any public area. (This is the same default as for other services.)

3. If the area name is specified but the account name is not, the system account name will be used. (This is the same default as for other services.)

Start time address is the address of a two-word block which contains the desired time of the first issue of the INIT call. The start time block has the following format:

word 0



word 1

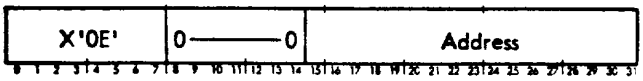
Note that all fields are binary values in terms of calendar date/time; hour is military (0-23), and year is whole year, i.e., X'7C1' (1985). A special TIME call is provided, which returns date/time in the above format to facilitate date/time calculations. Any field set to a -1 (all hexadecimal F's) will default to the present time (at issuance of CAL) for that field.

If the start time parameter is absent or zero but the time interval is present, the task will be INITed periodically beginning at one period from the occurrence of the SCHED CAL.

INT This function is used to establish BREAK control in the calling task's job. The call has the following format:

CAL1,8 address

where address points to word 0 of the FPT shown below:



where

X'0E' specifies the INT function.

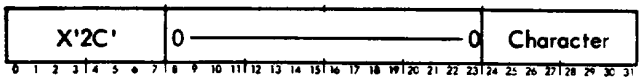
address is the address to which control will be transferred when a BREAK function is received for the caller's job. If this field is zero, any previous BREAK control will be removed.

Note that the user-stack arrangement of registers and PSD is identical to that used for the TRAP service calls. The Trap Retry (TRTY) call is used to return from the BREAK control processing routine, since the interrupted instruction is not executed before the interrupt occurs.

PC This function is used to set the prompt character for the job. The default is no prompt. The call has the following format:

CAL1,1 address

where address points to word 0 of the FPT shown below:



where

X'2C' is the code for the PC call.

character is the prompt character to be output when an input request is received for the terminal. Illegal EBCDIC characters and lower case ANSCII characters are not allowed.

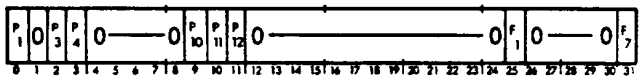
EXTM This external task termination function allows one task to terminate another. It has the format

CAL1,7 address

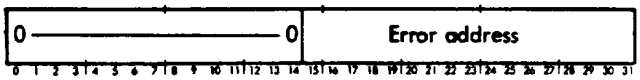
where address points to word 0 of the FPT shown below.

word 0

word 1

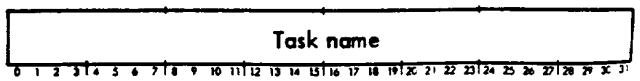


optional (P1)

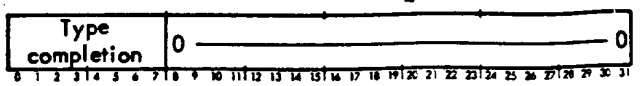


optional (P3)

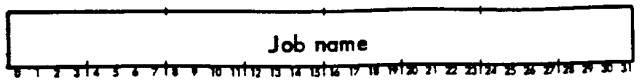
optional (P4)



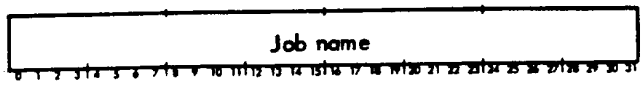
optional (P10)



optional (P11)



optional (P12)



where

Word 0

X'49' is the code for the EXTM call.

Word 1

P_1 is the presence indicator for the error address parameter (1 indicates present; 0 indicates absent).

P_3-P_4 are the presence indicators for the task name parameters (1 indicates present; 0 indicates absent).

P_{10} is the presence indicator for the type completion code parameter (1 indicates the type completion code is to be posted; 0 indicates no posting).

P_{11}-P_{12} are the presence indicators for the job name parameters (1 indicates a job name is specified; 0 indicates a job name is not specified).

F_1 is the abnormal termination request (0 means do not abort the task; 1 means abort the task).

F_7=1 is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

Options

Error address is the return location if errors are found.

Task name is the name of the load module for the task to be terminated. The caller's load module name is the default name. All tasks in the same load module are terminated.

Type completion set by CAL process table.

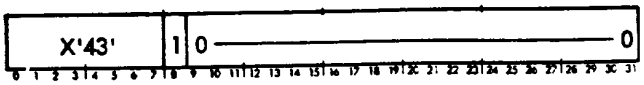
Job name is the name of the job in which the named task is being executed. The caller's job is the default name.

SIGNAL Send data to another task. This function call allows a task to communicate and post data to another task. It has the format

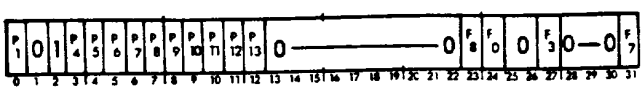
CAL1,7 address

where address points to word 0 of the FPT shown below:

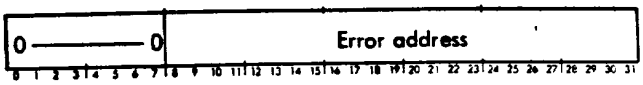
word 0



word 1

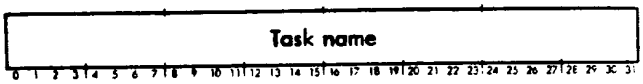


optional (P1)

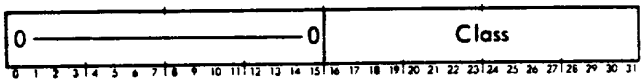


required (P3)

optional (P4)



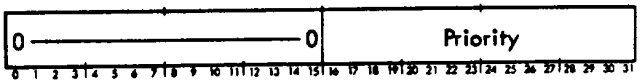
optional (P5)



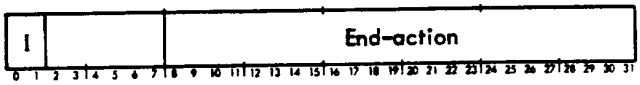
optional (P6)

optional (P7)

optional (P8)

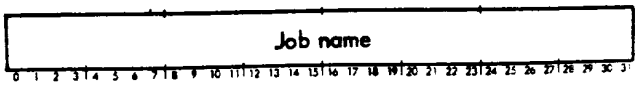


optional (P9)

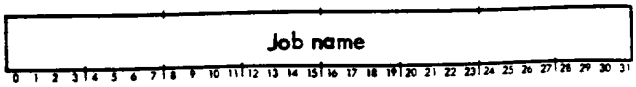


optional (P10)

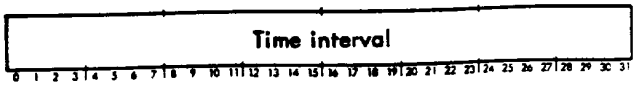
optional (P11)



optional (P12)



optional (P13)



where

Word 0

X'43' specifies the SIGNAL call.

Word 1

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_4 is the task name (second word) parameter presence indicator (0 means absent; 1 means present).

P_5 is the class parameter presence indicator (0 means absent; 1 means present).

P_6 is the data area address parameter presenct indicator (0 means absent; 1 means present).

P_7 is the feedback data area address parameter presence indicator.

P_8 is the priority parameter presence indicator (0 means absent; 1 means present).

P_9 is the end-action parameter presence indicator (0 means absent; 1 means present).

P_{10} is the type completion parameter presence (0 means absent; 1 means present).

P_{11}-P_{12} are the job name parameter presence indicators (0 means absent; 1 means present).

P_{13} is the time interval parameter presence indicator (0 means absent; 1 means present).

F_8 is the delete on post indicator. Indicates that no check will be performed.

F_0=1 is the long wait indicator. If set, the calling task will be made a prime candidate for rollout.

F_3 is the wait indicator. If set to 1, the calling task will wait for a post of the SIGNAL service.

F_7=1 is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

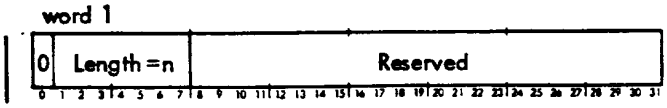
Word options

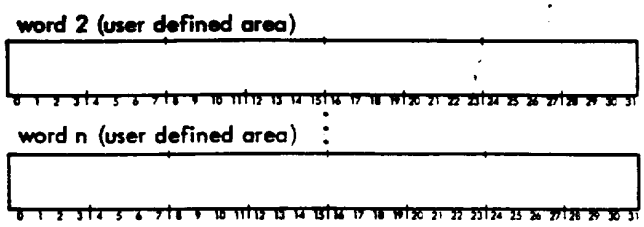
Error address on calls with wait, error address is the return for all error completions. On calls without wait error is the return address for immediate errors.

Task name is the name of the task being signaled. Default is the caller's task.

Class is a class bit mask. Default is X'FFFF'. Class is used to separate different types of signals to a signaled task. It is used in conjunction with a similar class bit mask supplied with the POLL service. The signal is only accepted if one or more one bits in the two class masks compare.

Signal data area is the address of word 0 of a data area to be made available to the signaled task. The data area format is as follows:





where length n is the number of words (from 1-127) in the data area. Short data areas are recommended.

Length and feedback data area address the number of words and first word address of an area available for receipt of any data sent by the signaled task at POST time. The area will be in the same format as the Signal data area above, with the number of data words returned given in the first byte. If the feedback area is smaller than the passed data area only the number of words specified in this word of the SIGNAL FPT will be moved. The remainder of the data is lost.

The feedback data is stored on return from a SIGNAL with wait (F3=1) or upon return from a CHECK of a completed SIGNAL.

If no feedback option is used, any passed data is discarded.

Priority is the priority of the SIGNAL. Used on POLL's to control the order of input to the signaled task. Default is the caller's priority.

I and End-action is the primary task end action option, as follows:

I=00 end action is an address for POST to BAL to.

=01 interrupt address to be triggered by POST.

=10 interrupt label to be triggered by POST.

Type completion, BUSY, and Completion status:

BUSY=1 if service is in process.

=0 when the service is complete.

Type completion and completion status is set by the signaled task. If the completion word is not provided, the type completion and completion status are not available to the signaling task.

Job name is the four- or eight-character name of the job to which the signaled task belongs. Default is the caller's job.

Time is the maximum number of seconds to be allowed between the SIGNAL and completion of the service. If a time interval is not provided, the service will never be timed-out.

The signaled task must exist at SIGNAL time. Standard CHECK calls are used with SIGNAL without wait.

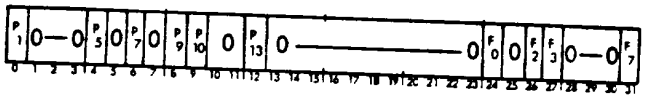
POLL Request receipt of data from another task. This function call requests the input of the highest priority SIGNAL. It has the format:

CAL1,7 address

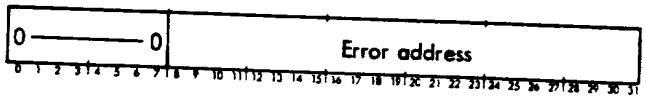
where address points to word 0 of the FPT shown below.

Word 0

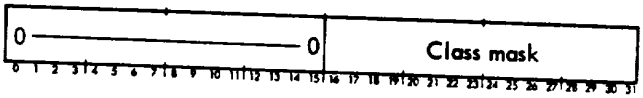
word 1



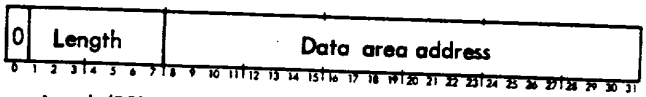
optional (P1)



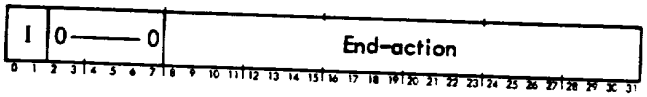
optional (P5)



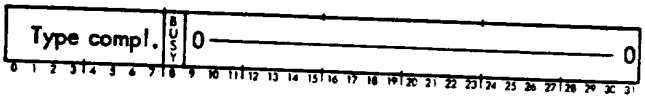
optional (P7)



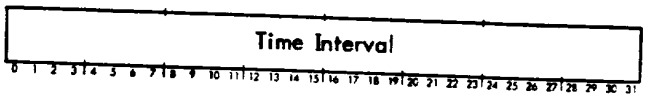
optional (P9)



optional (P10)



optional (P13)



where

Word 0

X'45' specifies a POLL function.

SIGNAL identification stored by CP-R for subsequent POST.

Word 1

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_5 is the class parameter presence indicator (0 means absent; 1 means present).

P_7 is the data area address and length parameter indicator (0 means absent; 1 means present).

P_9 is the end action parameter presence indicator (0 means absent; 1 means present).

P_{10} is the type completion parameter indicator (0 means absent; 1 means present).

P_{13} is the time interval parameter presence indicator (0 means absent; 1 means present).

F_0=1 is the long wait indicator; makes secondary task a prime candidate for roll-out.

F_2=1 immediate POLL only. If set, an attempt is made to satisfy the POLL with a matching SIGNAL. If no match is found, an error code is returned and the service is completed.

F_3=1 wait for SIGNAL.

F_7=1 Is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

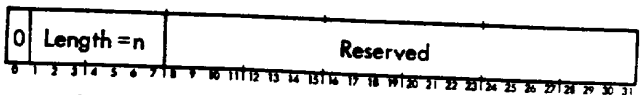
Word options

Error address is the location to return to if immediate errors are detected. Used for all errors on POLL calls with wait.

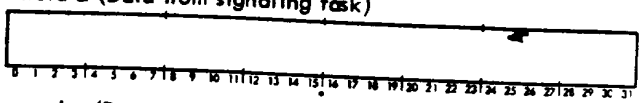
Class mask is a bit pattern for selective polling. The mask is logically ANDed with the class field in the signal. If the result contains all zero, the signal is not selected. The default is X'FFFF'.

Data area address and length is the length and location of an area where any data attached to the signal will be stored. A maximum of 'length' words will be moved, the format of which follows:

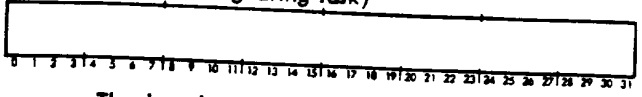
word 1



word 2 (Data from signaling task)



word n (Data from signaling task)



The length in the data area is the actual length attached to the signal. (Length is from 1-127.)

I and end-action is the primary task end action as follows:

I=00 end action is a BAL address to use when a SIGNAL is received.

=01 interrupt address to be triggered when a SIGNAL is received.

=10 interrupt label to be triggered when a SIGNAL is received.

Type completion and BUSY is as follows:

BUSY=1 if service is in process.

=0 if service is complete.

Time interval is the maximum number of seconds allowed to satisfy the POLL. If no time is provided, the POLL will not be timed out.

Standard CHECK calls are used with a POLL. The SIGNAL identifier in word 0 is used by CP-R on the POST to locate the SIGNAL to be posted, and must be provided on the POST call by the signaled task. Subsequent POLLs will not cause the same SIGNAL to be delivered.

When a SIGNAL occurs, the POLL call will be posted (but not the SIGNAL). If the level of the polling task is equal to or greater than the signaling task priority, the signaling task will be interrupted and the polling task dispatched.

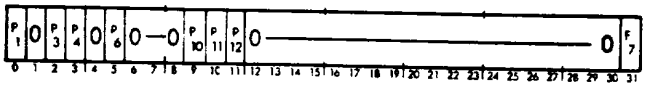
POST Posts the completion of a SIGNAL service. The function has the format

CAL1,7 address

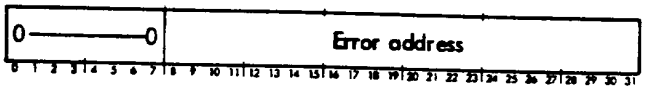
where address points to word 0 of the FPT shown below:

word 0

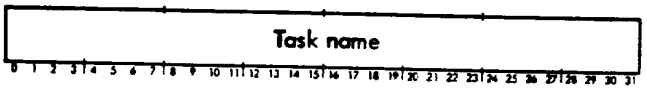
word 1



optional (P1)

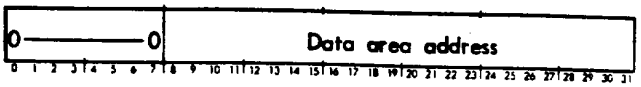


optional (P3)

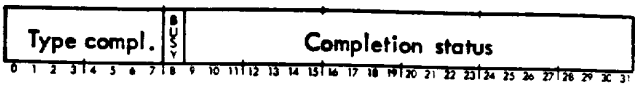


optional (P4)

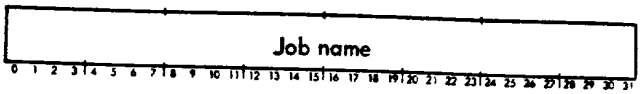
optional (P6)



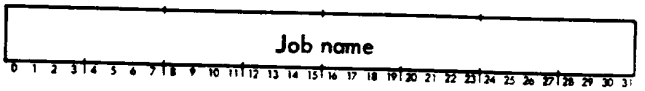
optional (P10)



optional (P11)



optional (P12)



where

Word 0

X'46' specifies the POST function.

SIGNAL identification specifies the SIGNAL to be posted and must be provided by the caller from the POLL service.

Word 1

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_3-P_4 are the task name parameter presence indicators (0 means absent; 1 means present).

P_6 is the data area address parameter presence indicator (0 means absent; 1 means present).

P_{10} is the type completion parameter presence indicator (0 means absent; 1 means present).

P_{11}-P_{12} are the job name parameter presence indicators (0 means absent; 1 means present).

F_7=1 is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

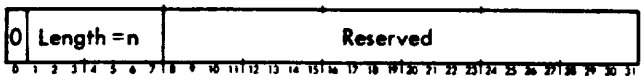
Word options

Error address is the location to return on errors.

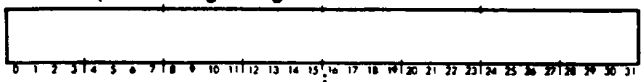
Task name is the four- or eight-character name of the task to which SIGNAL was sent. Default is the caller's task. This option allows the user to SIGNAL task A, poll from task B, but post from another task, e.g., task C.

Data area address is the location of feedback data. Data area format is as follows:

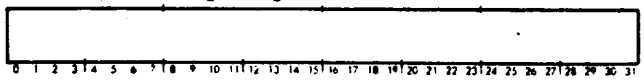
word 1



word 2 (Data to signaling task



word n (Data to signaling task



Length, in words, is from 1-127.

Type completion, BUSY and Completion status is as follows:

On entry BUSY must be 0, meaning service complete.

Type completion is the code to be posted in the signal and eventually back to the signaling tasks FPT type completion word. If the Type completion word is absent, a code of X'01' will be posted in the signal.

Job name is the four- or eight-character name of the job in which the task in P3, 4 resides. Default is the caller's job. Job name is not used if P3, P4 are zero.

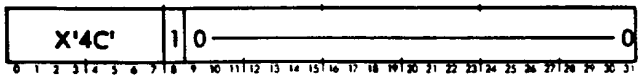
POST is an immediate service and does not use CHECK calls. Users should avoid completion codes F0-FF which are used for special purposes by CP-R.

ENQ Obtain the right to use a controlled item. The ENQ function allows tasks to gain exclusive or shared access to a controlled item. The item is user defined and could be a table, a piece of equipment, a file, or anything which for logical or physical reasons cannot be used by two tasks at the same time. The call has the format

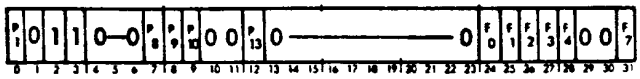
CAL1,7 address

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

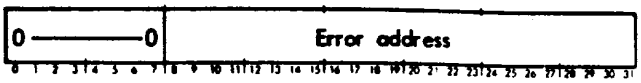
word 0



word 1

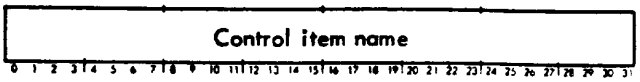


optional (P1)

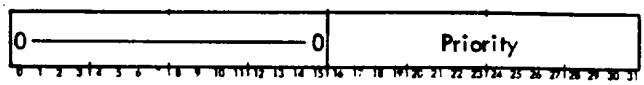


required (P3)

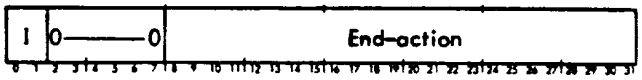
required (P4)



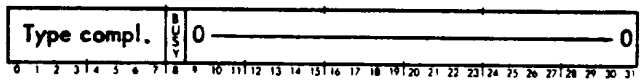
optional (P8)



optional (P9)



optional (P10)



optional (P13)

where

Word 0

X'4C' specifies the ENQ function.

Word 1

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_8 is the priority parameter presence indicator (0 means absent; 1 means present).

P_9 is the end-action parameter presence indicator (0 means absent; 1 means present).

P_{10} is the type completion parameter indicator (0 means absent; 1 means present).

P_{13} is the time interval parameter presence indicator (0 means absent; 1 means present).

$F_0 = 1$ is a long wait option making the task a prime candidate for roll-out.

$F_1 = 1$ is a system level controlled item.

$= 0$ is a job-level controlled item.

$F_2 = 1$ is an immediate acquisition option. If the resource is not available the service returns to caller with an error.

$F_3 = 1$ is the wait indicator for completion.

$F_4 = 1$ is a shared acquisition.

$= 0$ is an exclusive acquisition.

$F_7 = 1$ is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

Word options

Error address is the location to return to if immediate errors are detected. Also used on all errors on ENQ calls with wait option.

Priority is the relative priority of the request, as a hardware and software priority. Default is the caller's priority level. Priority is used only when multiple ENQs are outstanding on a controlled item when it is DEQed.

I and end action is the primary task end action as follows:

I = 00 end action is a BAL address to use when the ENQ is posted.

= 01 end action is an address to be triggered when the ENQ is posted.

= 10 end action is a label to be triggered when the ENQ is posted.

Type completion and BUSY is as follows:

BUSY = 1 if service is in process.

= 0 if service is completed.

Time interval is the maximum number of seconds that the caller will wait for the ENQ to be satisfied. If not specified, the ENQ will not be timed out.

After a caller is given a normal type completion on an ENQ, he must DEQ or his acquisition will remain until his task terminates.

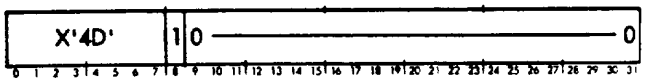
Once a shared acquisition has been made, additional shared acquisitions will be allowed, while exclusive acquisitions will be queued awaiting a DEQ by all shared users. Once an exclusive ENQ has been queued, only higher priority shared ENQs will be allowed to acquire the device. Exclusive acquisitions always cause all following ENQs to be queued for posting when the item is DEQed. Standard CHECK calls are used with ENQ.

DEQ Release the right to use a controlled item. This function call allows the caller to release control of a previously acquired controlled item, and has the format

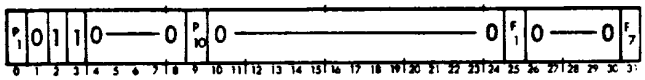
```
CAL1,7    address
```

where address points to word 0 of the FPT shown below:

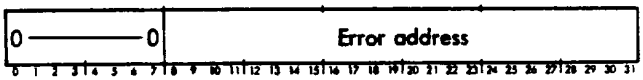
word 0



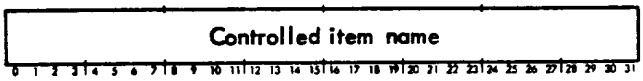
word 1



optional (P1)

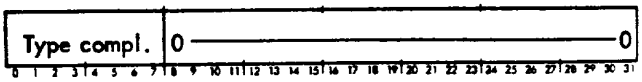


required (P3)



required (P4)

optional (P10)



where

Word 0

X'4D' specifies a DEQ call.

Word 1

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_{10} is the type completion parameter presence indicator (0 means absent; 1 means present).

$F_1 = 1$ is a system level controlled item.

$= 0$ is a job level controlled item.

$F_7 = 1$ is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

Word options

Error address is the address to return to if errors are detected on the service call.

Controlled item name is the eight-character name of the item being released. It must previously have been ENQed successfully.

Type completion contains the code describing the disposition of the service, including any error codes.

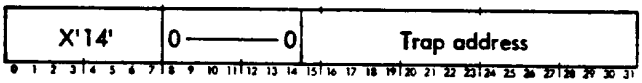
DEQ is an immediate service and is not subject to CHECK. DEQ automatically posts the highest priority waiting ENQs which can be satisfied.

TRAP This function call has the form

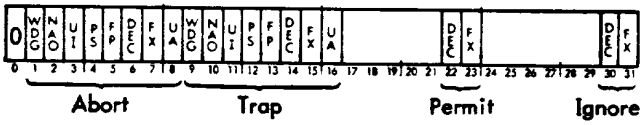
CAL1,8 address

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

word 0



word 1



where

Word 0

X'14' is the code for the TRAP call.

Trap address is the address in the user program that receives the requested traps. The address is optional unless it is the initial call and one of the trap bits is set. The address must lie in the calling program's portion of memory.

Word 1

Bits 1-8 are the Abort flags specifying which traps are to be handled by the system.

Bits 9-16 are the Trap flags specifying which traps are to be handled by the user's trap handler.

Bits 22-23 are the Permit flags specifying that the decimal or arithmetic mask in the PSD is to be set so that these traps are permitted.

Bits 30-31 are the Ignore flags specifying that the decimal or arithmetic mask in the PSD is to be set so that these traps are ignored.

The Abort, Trap, Permit, and Ignore fields specify the changes to be made in the disposition of trap occurrences.

The bits in these fields have the following significance:

UA	User Task Abort (X'50')
WDG	Watchdog Timer
NAO	Nonallowed operation
UI	Unimplemented Instruction
PS	Pushdown stack limit
FP	Floating-point arithmetic
DEC	Decimal arithmetic
FX	Fixed-point arithmetic

If a control bit has value 1, the trap is to be handled as specified. A value of zero specifies that no change is to be made in the handling of that trap. The fields are processed from left to right (Abort, Trap, Permit, Ignore), with the last-processed code overriding any previously processed code.

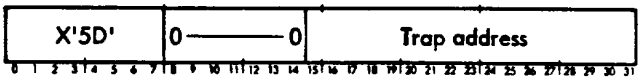
If a given trap condition has a control bit value of 1 in both the Abort and Trap fields, the Trap bit will override the Abort bit and the user will receive the trap, since the Trap bit is the last one processed.

JTRAP This function call allows the user to specify job level trap controls. The call has the format

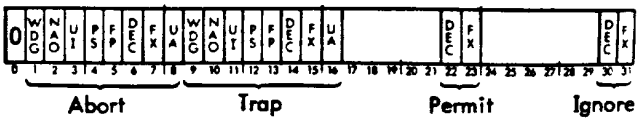
CAL1,7 address

where address points to word 0 of the FPT shown below

word 0



word 1



where

Word 0

X'5D' is the code for the JTRAP call.

Trap address is the address of the user's trap handler. The address is optional unless it is the initial call and one of the trap bits is set. The address must lie in the calling load module portion of memory and will be used for all traps from within the job. Separate trap addresses are maintained for primary and secondary tasks within a job.

Word 1

Bits 1-8 are the Abort flags specifying which traps are to be handled by the system.

Bits 9-16 are the Trap flags specifying which traps are to be handled by the user's trap handler.

Bits 22-23 are the Permit flags specifying that the decimal or arithmetic mask in the PSD is to be set so that these traps are permitted. Note that the callers PSD is the only one affected.

Bits 30-31 are the Ignore flags specifying that the decimal or arithmetic mask in the PSD is to be set so that these traps are ignored. Note that the callers PSD is the only one affected.

The Abort, Trap, Permit, and Ignore fields specify the changes to be made in the disposition of trap occurrences.

The bits in these fields have the following significance:

UA	User abort
WDG	Watchdog Timer
NAO	Nonallowed operation
UI	Unimplemented Instruction
PS	Pushdown stack limit
FP	Floating-point arithmetic
DEC	Decimal arithmetic
FX	Fixed-point arithmetic

If a control bit has value 1, the trap is to be handled as specified. A value of zero specifies that no change is to be made in the handling of that trap. The fields are processed from left to right (Abort, Trap, Permit, Ignore), with the last-processed code overriding any previously processed code.

If a given trap condition has a control bit value of 1 in both the Abort and Trap fields, the Trap bit will override the Abort bit and the user will receive the trap, since the Trap bit is the last one processed.

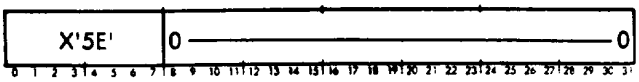
The user abort bit UA is used in conjunction with the other bits. If a condition occurs that causes an abort (such as a trap handled by the system) the user will get control after the error messages but before the abort sequence. This will allow him to try a recovery procedure. The trap address (X'40', X'41', etc.) is placed in the top word of the user's task. The user abort will occur for all traps, incorrect calls (trap X'50'), and other major problems. If the user's recovery has been successful, he can continue execution; if not, he can abort the trapped tasks. The user shall take care to determine that his recovery has been successful since CP-R will allow a return on an unsuccessful recovery.

TRTY The trap retry function call is of the form

```
CAL1,7    address
```

where address points to word 0 of the FPT shown below

word 0



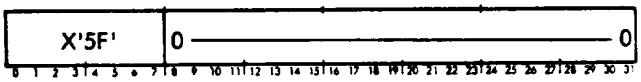
This function returns to the instruction that trapped for retry. This can be used if the user feels he has corrected the problem in his trap handler.

TEXIT The trap exit function call is of the form

```
CAL1,7    address
```

where address points to word 0 of the FPT shown below

word 0



This function performs an EXIT from within a trap handler routine. For secondary tasks, this will result in a STOP function.

TRTN This function call (trap return) is of the form

```
CAL1,9    5
```

No FPT is used. Return is to the trapped instruction + 1.

EXIT This function call is of the form

```
CAL1,9    1
```

No FPT is used.

For secondary tasks this call causes termination, and for primaries it exits and clears the interrupt level.

EXDA The Exit Disarmed function call, available to primary tasks only, has the form

```
CAL1,9    X'A'
```

No FPT is used.

EXDA performs like an EXIT except that the primary task's interrupt is left in the disarmed state.

TERM This function call is of the form

```
CAL1,9    8
```

No FPT is used. If the call is given by a background program, control is returned to the Job Control Processor.

If the call is given by a foreground program, the program is terminated.

ABORT This function call is of the form

```
CAL1,9    3
```

No FPT is used.

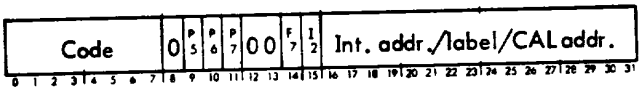
CONNECT,ARM,DISARM,DISCONNECT (Primary Task Creation, Deletion, Start, and Stop)

These functions connect a task to an interrupt level or a processing routine to a CAL2, 3, or 4; arm an interrupt level; disarm an interrupt level; or disconnect a task from an interrupt level or CAL processing routine. The task associated with the affected interrupt need not be in the same load module or job as the task executing the CAL. The calls have the format

```
CAL1,5    address
```

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

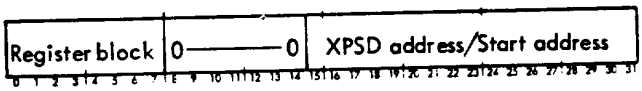
word 0



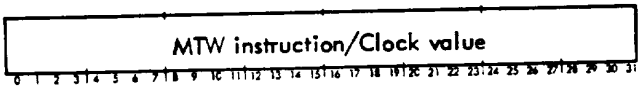
word 1

Note: Bit 3 of word 1 can be either AI or DE.

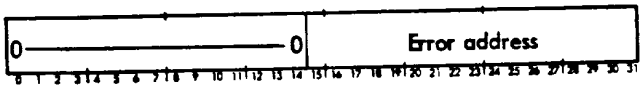
optional (P1)



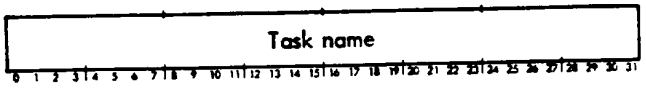
optional (P2)

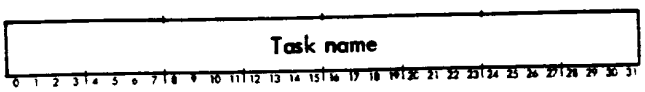


optional (P3)

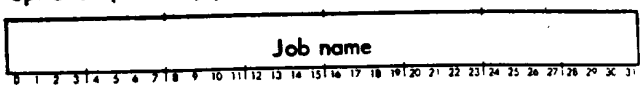


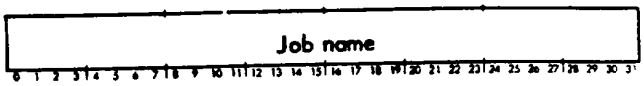
optional (2 words) (P5 in word 0)



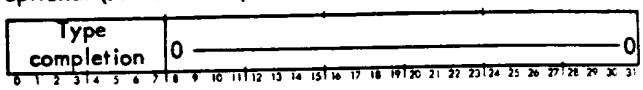


optional (2 words) (P6 in word 0)





optional (P7 in word 0)



where

Word 0

Code X'03' specifies DISCONNECT or DISARM; X'04' specifies CONNECT or ARM.

P_5 is the Task name option presence indicator.

P_6 is the Job name option presence indicator.

P_7 is the Type completion presence indicator.

F_7=1 is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

I_2 indicates that the address (I_2 = 0), or the label (I_2 = 1), of the interrupt is specified in Interrupt address/label.

Int. addr./label/CAL addr. is the hexadecimal address for the interrupt or CAL.

Label is the two-character EBCDIC name for a label.

Word 1

P_1 is the CONNECT and DISCONNECT indicator. If the service is CONNECT, P1 also is the XPSD/Start address parameter presence indicator. If P_1 = 1, a connection or disconnection is performed; if P_1 = 0, an ARM or DISARM is performed. P_1 must = 1 if the address is that of a CAL2, 3, or 4.

P_2 is the MTW/clock value parameter presence indicator (0 means absent; 1 means present).

P_3 is the error address parameter presence indicator (0 means absent; 1 means present).

AI† when connecting a CAL, indicates that the address increment option on a CAL is desired. In this case, the entry address will be the contents of the

†Bit 3 of word 1 can be either AI or DE.

R field from the CAL added to the user's start address. Therefore, the user must provide a 16-word table of entry points at the start address on the basis of one per R value.

DE[†] when connecting an interrupt, specifies that the interrupt is to be disabled (DE = 1), or enabled (DE = 0). This parameter is only used on ARM and CONNECT to an interrupt.

DI specifies that the connection is to be direct (DI = 1) or central (DI = 0).

CI specifies that the task is to be entered with the clock group inhibit set (CI = 1) or reset (CI = 0).

II specifies that the task is to be entered with the I/O group inhibit set (II = 1) or reset (II = 0).

EI specifies that the task is to be entered with the external group inhibit set (EI = 1) or reset (EI = 0).

MS specifies that the task is to be entered in master mode (MS = 0) or slave mode (MS = 1).

DM specifies that the task is to be entered with the decimal mask set (DM = 1) or reset (DM = 0).

AM specifies that the task is to be entered with the arithmetic mask set (AM = 1) or reset (AM = 0).

NR is the number of registers to be saved upon occurrence of the interrupt (if connection is central). Value 0 is used to denote that 16 registers are to be saved. Registers are saved beginning with register 0 and at least four registers must be saved. If a task uses CALIs, registers through R10 should be saved in case error exits cause R8 and R10 to be altered.

TCB address contains the TCB address for central connection. For direct connection, this portion of word 1 is unused. For interrupt, the TCB should be 26 words long. Central CAL connections need an eight-word TCB. The TCB address must be a doubleword address.

Word Options

Register block is the register block pointer to be used by the task when a central connection is specified. The specified register block pointer is in the PSD used to enter the task. The effect is the same as if the task executed LRP instructions at entry and exit. The context saving and switching prior to task entry and after the task EXITs is always executed using register block zero.

XPSD/Start address if the task is directly connected, this word is the address of the XPSD instruction to be used to enter the task. The XPSD instruction furnished will be stored in the interrupt location by the CONNECT function. If the task is centrally connected, this word contains the start address of the primary task.

MTW/Clock value is the value (in units of the clock's resolution) to which the clock is to be set in the case of a centrally connected counter-equals-zero interrupt. An MTW -1 instruction will be used to decrement the value furnished.

For directly connected, counter-equals-zero interrupts, this word should contain the actual Modify and Test Word instruction which is to be stored in the corresponding count pulse location. For both types of connections, a clock parameter is required unless the interrupt is being reconnected (i.e., is connected when the CAL is initiated). In this case, no change will be made to the clock pulse interrupt location if $P_2 = 0$.

Error address is the entry address to the user's error handling routine if the monitor encounters any error in processing the call and is unable to complete the requested function. If no error address is provided, 'trap 50' is simulated. Error address will be used for all returns but normal. A type completion option can be used to determine the exact error source.

Task name is the name with which the task is to be associated, and must occupy two words. The task will use the PCB associated with the load module having this task name and will be terminated in a group with the other tasks using the task name and PCB (i.e., the other task connected into the load module). The caller's task is the default name. The connection will not be performed if the load module with the specified task name has not been loaded or is in termination.

Job name is the name of the job in which the task resides and must occupy two words. The caller's job is the default name.

Type completion is the code describing the status of the service call.

[†] Bit 3 of word 1 can be either AI or DE.

Clock interrupt connections are always to the counter-equals-zero interrupt. Interrupt address/label cannot contain a clock pulse level. Use of clock 4, CT, and HI interrupts is restricted and is to be used only by the monitor.

On clock counter-equals-zero connects, the MTW is moved automatically to the clock pulse, or an MTW -1 is stored in the clock pulse and the clock value is stored in a fixed real memory location for the clock level (K:CLOCK1, K:CLOCK2, and K:CLOCK3). The values in K:CLOCK1-3 will be moved to the actual counter location (CLK1-3) on the central connection each time the task is entered, which will be via CLK1SAVE-CLK3SAVE. If a direct connection is used, the user must reset his counter. If a central connection with a clock value is used, the user need not perform any resetting.

Successful processing will return to the instruction following connect. If an error address is specified, all other completions will exit to the error address.

DISCONNECT frees the interrupt location, making the level available for any task.

Parameters DI to TCB, XPSD/Start address, and MTW/Clock value are only used on CONNECT. DE is used on ARM and CONNECT. Interrupt label, Code and P_1 - P_6 flags must always be provided. Error address, Task name, and Job name are options used by all services. These services are only available to foreground tasks.

If the interrupt or CAL is connected when a CONNECT CAL is performed, the current connection must be to the same task as the new one being requested.

Once a CONNECT or DISCONNECT is in progress from one task, a CONNECT or DISCONNECT from a higher priority task to the same interrupt is not allowed until the original one is complete. An error code will be given to the second caller chronologically.

Once a CAL routine is connected, a call by any task will cause entry to the connected code. CAL processing will be done using the PCB and TCB of the calling task. If the CAL processor does CP-R service calls (CAL1s), CP-R will perform the service for the original calling task.

For example, if a CAL2 processing routine does a TERM, the calling task is terminated, not the CAL processing routine.

EXIT will cause the caller to become idle. The user's CAL processor should clear the temp stack of the PSD and registers saved by the central CAL connection prior to the EXIT.

The CALRTN function should be used to exit from centrally connected CAL2, 3, and 4 processing. CALRTN will clean the user's Temp Stack of CAL linkage words.

In direct connections, the CAL processing routine is responsible for all reentrancy and returning control to the caller.

The format of the stack upon entry to centrally connected CAL processing routines is shown in Figure 6. TCBs for centrally connected CALs are shown in Figure 7. The condition codes at entry to centrally connected CALs contain the 'R' field from the CAL instruction.

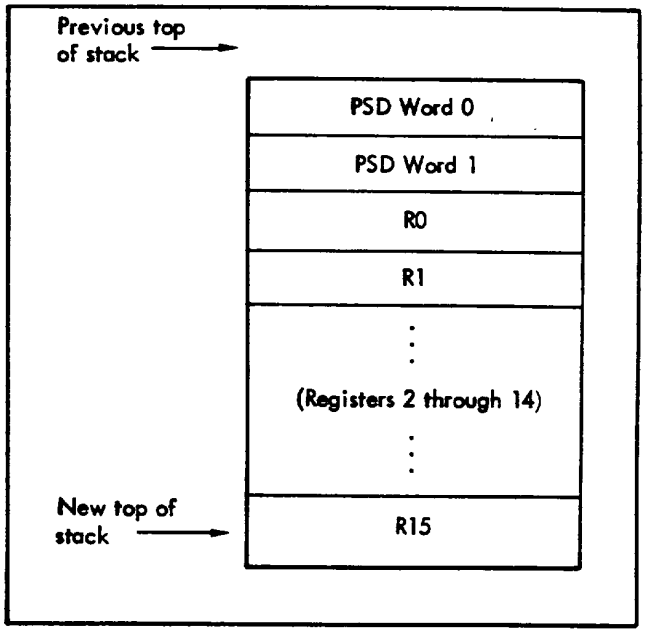


Figure 6. User Temp Stack Format at CAL Processor Entry

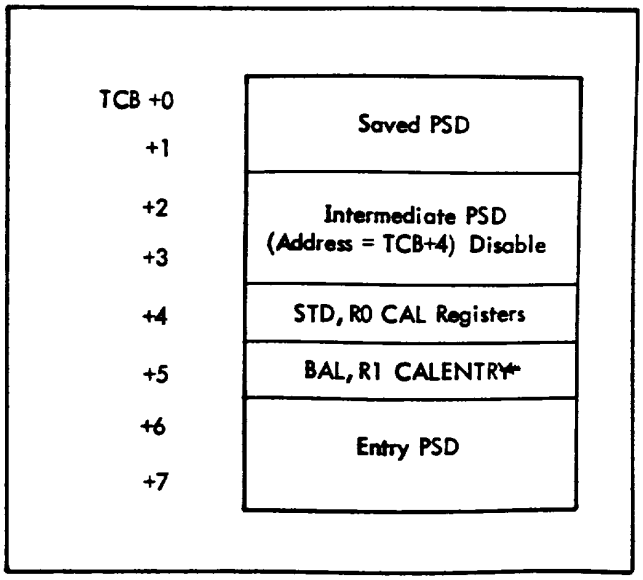


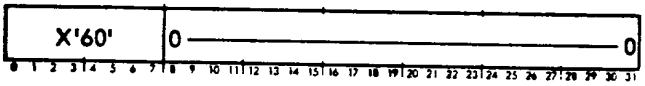
Figure 7. TCB for Centrally Connected CALs

CALRTN The CALRTN function call is used by centrally connected CAL2, 3, and 4 processing routines to return to the calling task at the instruction following the call. CALRTN has the format

CAL1,7 address

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

word 0



where X'60' is the code for the CALRTN call.

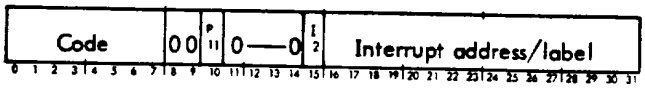
ENABLE,DISABLE,TRIGGER (Interrupt Controlling Commands)

These function calls (available only to the foreground) are of the form

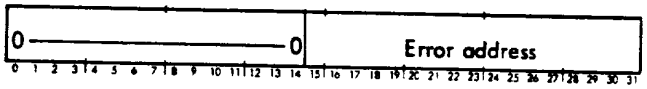
CAL1,5 address

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

word 0



optional (P11)



where

Word 0

Code = X'00' specifies trigger.

= X'01' specifies disable.

= X'02' specifies enable.

P_{11} is the Error address parameter presence indicator (0 means absent; 1 means present).

I_2 specifies that either the address (I_2 = 0), or the label (I_2 = 1), of the interrupt is specified in Interrupt address/label.

Interrupt address is the hexadecimal address of an interrupt.

Label is a two-character EBCDIC name for a label.

Word Option

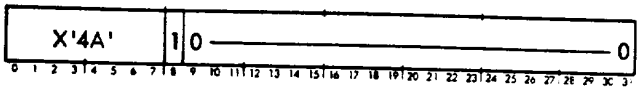
Error address is the entry address to the user's error handling routine if the monitor encounters any error in processing the call and is unable to complete the requested function. If no error address is provided, the monitor will process an ABORT return.

START Begin executing a secondary task. This function call removes a secondary task from idle state following a STOP call. The call has the form

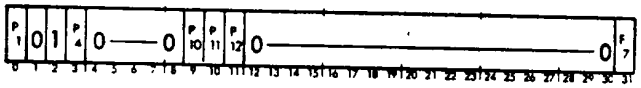
CAL1,7 address

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

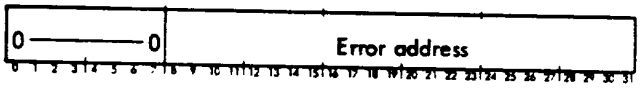
word 0



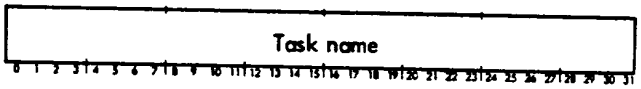
word 1



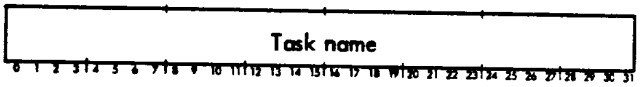
optional (P1)



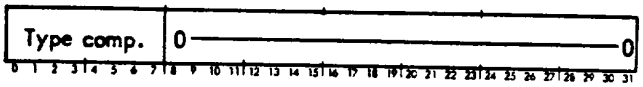
required (P3)



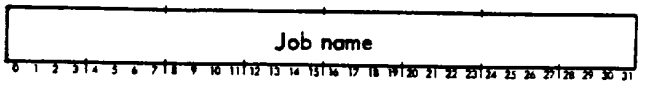
optional (P4)



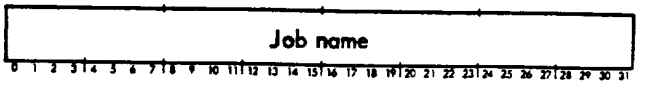
optional (P10)



optional (P11)



optional (P12)



where

Word 0

X'4A' specifies the START function.

Word 1

P_1 is the error address parameter presence indicator (0 means absent, 1 means present).

P_4 is the task name parameter presence indicator and is one if an eight-character task name is provided.

P_{10} is the type completion code parameter indicator (0 means absent; 1 means present).

P_{11}-P_{12} are the job name parameter presence indicators (0 means absent; 1 means present).

F_7=1 is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

Word Options

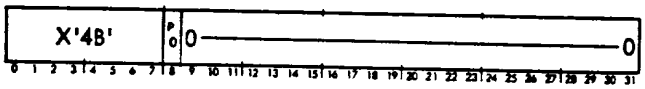
Word options are the same as for STOP below.

STOP Stop executing a secondary task. This function allows a secondary task to place itself or another task in a stopped state. It has the format

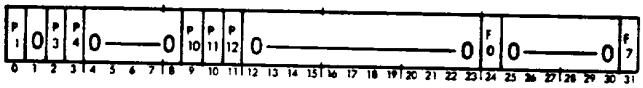
CAL1,7 address

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

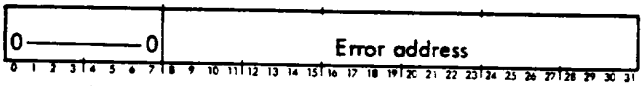
word 0



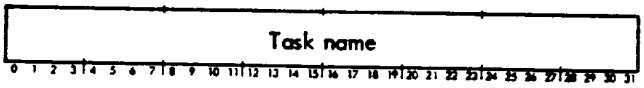
optional (P_0 in word 0)



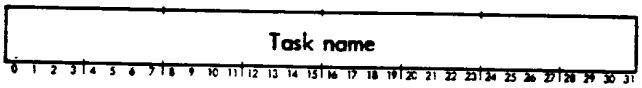
optional (P1)



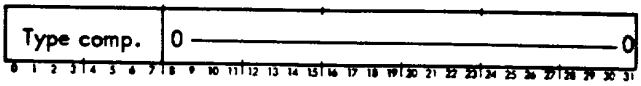
optional (P3)



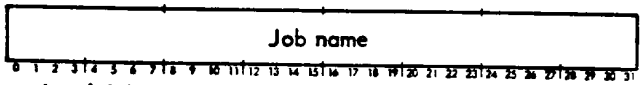
optional (P4)



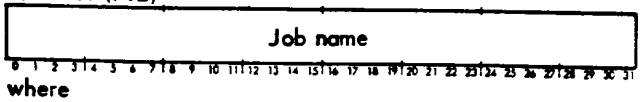
optional (P10)



optional (P11)



optional (P12)



where

Word 0

X'4B' specifies the STOP function.

P_0 is the presence indicator for word 1 (0 means absent; 1 means present). If P_0=0 then the STOP applies to the calling task.

Word 1

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_3-P_4 are the task name parameters presence indicators (0 means absent; 1 means present).

P_{10} is the type completion parameter presence indicator (0 means absent; 1 means present).

P_{11}-P_{12} are the job name parameter presence indicators (0 means absent; 1 means present).

F_0=1 is the long wait indicator making the normal secondary task a prime candidate for roll-out.

F_7=1 is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

Word Options

Error address is the return address if errors are detected in service processing.

Task name is the four- or eight-character name of the secondary task to be stopped. Default is the caller's task.

Type completion is the completion code for the service including the error code, if any.

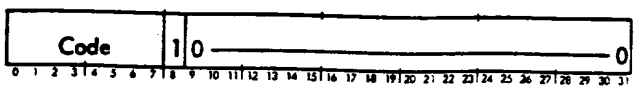
Job name is the four- or eight-character name of the job in which the task resides. Default is the caller's job.

STATUS,MODIFY Obtain and modify task controls. STATUS is a function call that allows the calling task to obtain task control data. MODIFY is a function call that allows certain secondary task controls to be modified. The call has the form

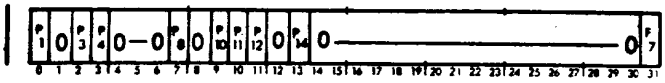
CAL1,7 address

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

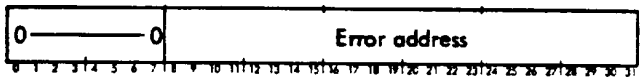
word 0



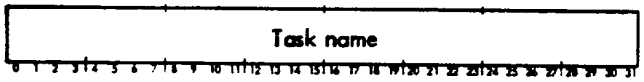
word 1



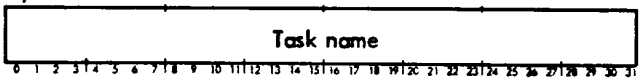
optional (P1)



optional (P3)

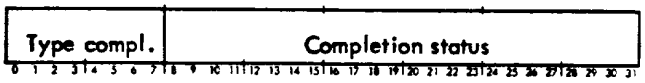


optional (P4)

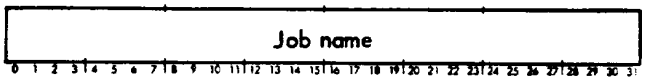


optional (P8)

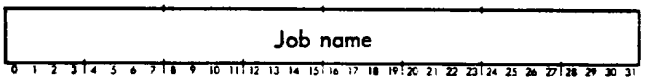
optional (P10)



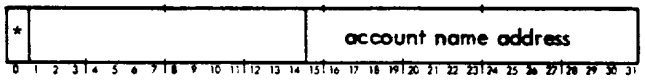
optional (P11)



optional (P12)



optional (P14)



where

Word 0

Code = X'4E' specifies STATUS function.

= X'4F' specifies MODIFY function.

Word 1

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_3-P_4 are the task name parameter presence indicators (0 means absent; 1 means present).

P_8 is the flags and priority parameter presence indicator (0 means absent; 1 means present).

P_{10} is the type completion parameter presence indicator (0 means absent; 1 means present).

P_{11}-P_{12} are the job name parameter presence indicators (0 means absent; 1 means present).

P_{14} is the optional account name parameter presence indicator (0 means absent; 1 means present).

F_7=1 is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

Word Options

Error address is the location to return to if errors are detected in the service.

Task name is the name of the task whose controls are being fetched or modified. Default is the caller's task.

Flags are as follows:

Bits	Value	Meaning
0	1	Task in final termination.
1	1	Task connected to CAL2.
2	1	Task connected to CAL3.
3	1	Task connected to CAL4.
4	1	Background task.
5	1	Secondary task.
6	1	Task being aborted.
7	0	Task initiated via RUN.
	1	Task initiated via INIT.
8	1	Load to be performed.
9	1	Public Library used by primary tasks.
10	1	Public Library used by secondary tasks.
11	1	Release to be performed.
12	1	Control sequence requested.
13	1	Task is loaded.
14	1	Task is run queued.
15	1	Break control requested.

Priority is the current task priority of the task as output from STATUS. On a MODIFY call, it is the priority to which the caller wishes to change the task.

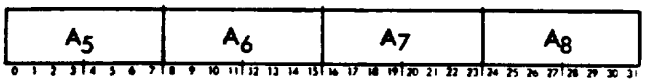
Type completion is the completion code for the service, including error codes.

Job name is the four- or eight-character name of the job to which the secondary task belongs. Default is the caller's job.

account name address is the address of a five word data block that will receive the account and name associated with the JOB. The format of this block is shown below.

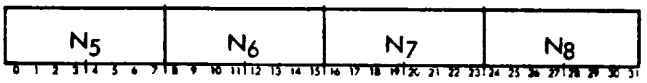
word 0

word 1

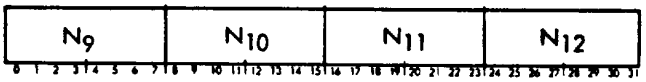


word 2

word 3



word 4



where

A_i are the EBCDIC characters naming the account.

N_i are the EBCDIC characters in the user name.

A task whose priority is modified to a higher level may not be capable of being dispatched at that level until the hardware interrupts drop back to the tasks previous level. This is the case if it was in execution when some higher priority interrupt caused it to be interrupted, and the modify call was done at this higher level.

If a STATUS call has P_{11}-P_{12} but not P_3-P_4, the current task's status will be acquired, including the job name which will be stored in the Job name parameter.

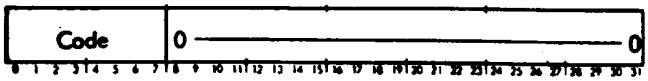
For STATUS calls only, if P14 is set, the account and user names for the current JOB will be stored in the five word data block. This parameter is not used by the MODIFY call.

MASTER, SLAVE These function calls (available only to the foreground) are of the form

CAL1,5 address

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

word 0



where

Code = X'07' specifies a slave-mode request.

= X'08' specifies a master-mode request.

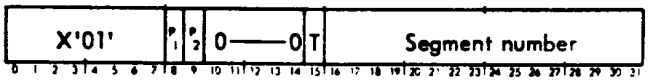
When a secondary task requests master mode (X'08'), it is placed in master-mode/protected.

SEGLOAD This function call is available to primary tasks and background tasks loaded with the SMM option. The call is of the form

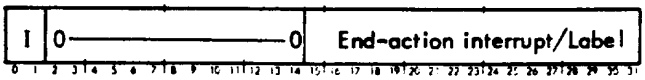
CAL1,8 address

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

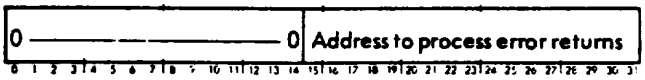
word 0



optional (P1)



optional (P2)



where

Word 0

X'01' specifies the SEGLOAD function.

P_1 indicates the presence or absence of word 1; 0 means absent, 1 means present. (P_1 is processed only if T = 0.)

P_2 indicates the presence or absence of the error address word.

T indicates whether control is to be returned following the call or transferred to the starting location of the segment at the conclusion of the segment load (0 = return to calling program; 1 = transfer to new segment.)

Word Options

I indicates the contents of the end-action interrupt field (primary tasks only).

I = 0 indicates no end-action.

I = 1 indicates an interrupt address.

I = 2 indicates an interrupt label.

If end-action is specified, the request to load the segment will be queued and control will be returned immediately to the calling program. The calling program can then exit and release control while the segment is being loaded. If end-action is not specified (I = 0), control will not be returned until the segment is loaded. The user is responsible for checking the status of the load if end-action is selected.

Address to process error returns is the address of the user's routine for processing any error or abnormal returns received while attempting to load the overlay segment. The codes returned will be identical to those of the Type II READ CAL since a READ CAL is used by SEGLOAD to load the segment. If this address is not present and an error occurs, a foreground program will be exited or a background program aborted. If an error is detected in the user's PCB or OVLOAD table, the User or System Trap Handler will be entered.

Warning: Do not issue SEGLOAD calls from tasks of higher priority than the I/O interrupt.

WAIT A background program will enter the "wait" state through this function call if an !ATTEND card was included in the control commands for the job. Normally, a background program would use WAIT after typing an alarm to the operator that requires an operator response. While in this state, the Control Task waits for a key-in from the operator specifying the disposition of the background program. The operator may specify continue (C), continue from OC (COC), or abort (X).

This function call is of the form

CAL1,9 9

No FPT is used.

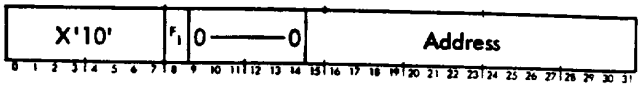
TIME Programs may interrogate the monitor to determine the time of day and date.

This function call is of the form

CAL1,8 address

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

word 0



where

X'10' is the code for the TIME call.

F_1 indicates the format of the returned values.

Address is the address of the first word of the block to receive the time and date. If $F_1 = 0$, a four word block is returned; if $F_1 = 1$, a two word block is returned.

For $F_1 = 0$, the block contains EBCDIC characters as shown below:

word 0

h	h	:	m

word 1

m	b	m	o

word 2

n	b	d	d

word 3

,	b	y	y

where

hh is the hour ($0 \leq hh \leq 23$).

mm is the minute ($0 \leq mm \leq 59$).

mon is the month (3-letter abbreviation).

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

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

For $F_1 = 1$, the block contains packed binary values as shown below:

	byte0	byte1	byte2	byte3
Word 0	whole year (i.e., 1976)			mo
Word 1	dd	hh	mm	ss

where the values are equivalent to those stated above except for:

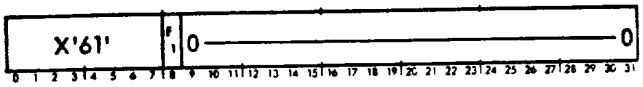
whole year is the full date.
mo is the number of the month.
ss is the seconds ($0 \leq ss \leq 59_{10}$).

GETTIME This function call is used to return elapsed time to the calling task. It has the form

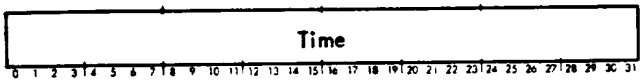
CAL1,7 address

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

word 0



word 1



where

Word 0

X'61' is the code for the GETTIME call.

F_1 is the Timer option (1 is for time since midnight; 0 is for time since system initialization).

Word 1

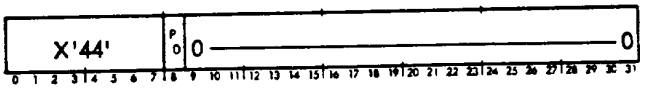
Time is the current value of the timer. For F_1=1, this is a binary number representing the number of seconds since midnight (this is the time set by the operator). For F_1=0, this is a binary number representing the number of elapsed seconds since system initialization.

STIMER (Request a Clock Interval Posting Service) This function call allows a task to sense when a specified time has been reached, either by subsequently checking the service or by waiting. The call is of the form

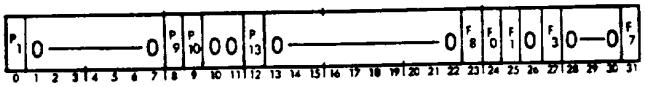
CAL1,7 address

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

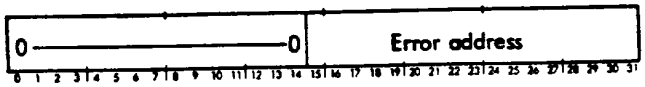
word 0



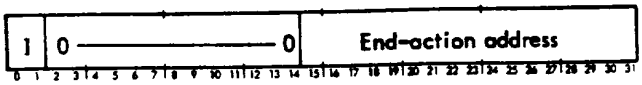
word 1



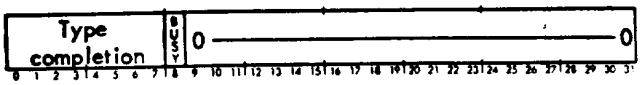
optional (P1)



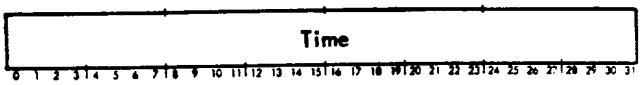
optional (P9)



optional (P10)



optional (P13)



where

Word 0

X'44' is the code identifying the STIMER function.

P_0 is the presence indicator for word 1, (1 indicates presence). If P_0 = 0, STIMER functions as though the time parameter was absent.

Word 1

P_1 is the presence indicator for the error address parameter (1 indicates present).

P_9 is the presence indicator for end-action parameter (1 indicates present).

P_{10} is the presence indicator for the type completion code parameter (1 indicates present).

P_{13} is the presence indicator for the time parameter (0 indicates absent and no timeout will be done; 1 indicates present and the CAL will be timed out).

F_8 is a delete on post indicator (1 means delete the event when it is posted – no CHECK will be performed and the user's FPT will not be posted; 0 means a CHECK will be performed). F_8 has meaning only when F_3 = 0.

F_0=1 is a long-wait indicator, making a secondary task a prime candidate for rollout.

F_1=1 specifies that the time word contains a threshold setting versus an interval value.

F_3=1 wait for time elapse before returning, versus returning to the next instruction as soon as the service request has been established.

F_7=1 is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

Word Options

Error address is the return for immediate errors.

I,end-action address I indicates the contents of the end-action address number field. (End-action is allowed only for foreground.)

I = 0 indicates an end-action BAL address to use when the time has elapsed.

= 1 indicates an interrupt address to be triggered when the time has elapsed.

= 2 indicates an interrupt operational label to be triggered when the time has elapsed.

Type completion,BUSY indicates the status of a timer service.

BUSY = 1 indicates timer has been initiated.

= 0 indicates timer is completed and checked.

Type completion = 1 indicates that the timer has elapsed.

Time if $F_1 = 0$ specifies a time interval in seconds.

$F_1 = 1$ specifies the threshold time (since system initialization) when posting should occur.

If the Time parameter is omitted control returns to the caller immediately. The user may obtain the current time in seconds via the GETTIME call (specifying time since system initialization) to establish a base time for time threshold setting.

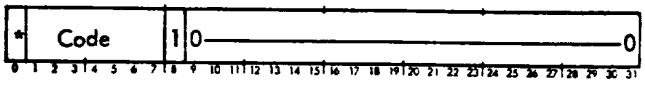
ERRSEND (generate an error log message)

A text message can be placed in the error log through use of the ERRSEND function call. The ERRSEND CAL has the form

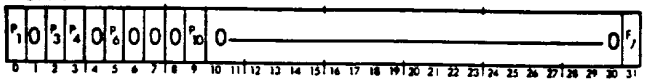
```
CAL1,7    address
```

where address is the location of word 0 of an FPT. The FPT has the form:

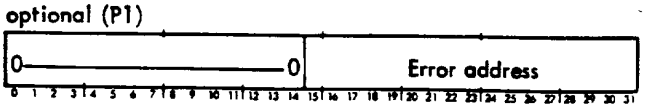
word 0



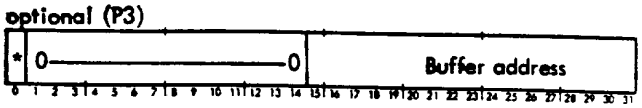
word1



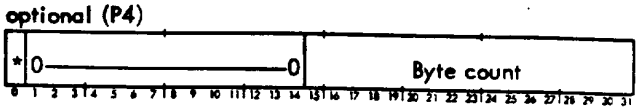
optional (P1)



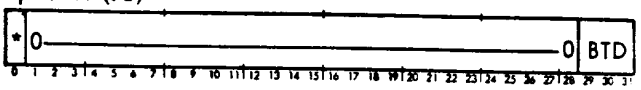
optional (P3)



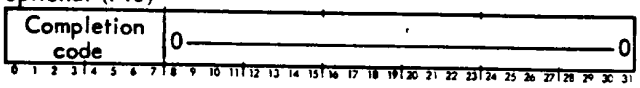
optional (P4)



optional (P6)



optional (P10)



where

Word 0

Code is X'66' and bit 8 must be set.

Word 1

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_3 is the buffer address parameter presence indicator (0 means absent; 1 means present).

P_4 is the byte count parameter presence indicator (0 means absent; 1 means present).

P_6 is the byte displacement parameter presence indicator (0 means absent; 1 means present).

P_{10} is the completion type parameter presence indicator (1 means present; 9 means absent).

F_7 is an abort override indicator: 1 means do not abort if errors are detected but no error address is provided; 0 means abort if errors are detected and no error address is provided.

Word Options

Error address is the address of the entry to the user routine that will handle error conditions.

Buffer address is the word address of the user buffer to be used in the I/O operation. Data is written from this buffer.

Byte count is the size in bytes of the data record (55 bytes maximum).

BTD is the byte displacement (0–3) from the word boundary of the beginning of the data record. If this parameter is omitted and the buffer address parameter is included in the FPT, value 0 is assumed for BTD.

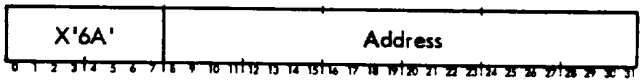
Completion status is the word wherein the system posts the completion parameters for the request.

ALARM (initiates a system alarm shutdown)

The ALARM call causes the system to perform an orderly shutdown. It has the form

CAL1,7 address

where address points to the FPT shown below.



where address points to a TEXTC message which will be logged on the operator's console as the alarm message.

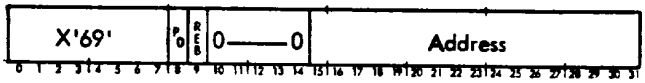
RECALARM (set post-shutdown routine entry)

The ALARM Receiver call specifies a post-shutdown action. It has the form

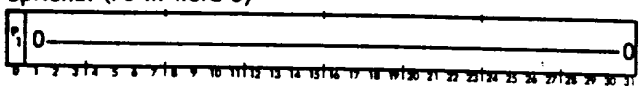
CAL1,7 address

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

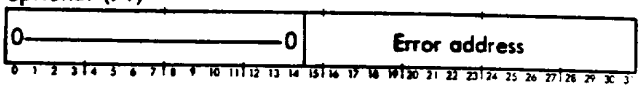
word 0



optional (P0 in word 0)



optional (P1)



where

Word 0

X'69' is the code for the RECALARM call.

REB = 1 will cause the system to automatically reboot after alarm processing is completed. In this case, address is ignored.

Address is the address to which control will be transferred when an alarm occurs. This transfer occurs after CP-R alarm processing is completed. A zero address will cause reset of the alarm address. Note that only one alarm address is active per system and the alarm address must reside in a foreground private memory area.

P_0 is the parameter presence indicator (0 means absent; 1 means present).

Word Options

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

Error address Is the location to return to if errors are detected in the request.

6. CP-R MEMORY MANAGEMENT

REAL MEMORY ALLOCATION

Real memory for CP-R is allocated at SYSGEN into a number of partitions that fall into four distinct classes as follows:

- CP-R System Memory
- Foreground Private Memory
- Foreground Preferred Memory
- Secondary Task Memory

Each class of memory consists of one or more partitions whose boundaries and extent are established during the SYSGEN process.

Memory partitions within a given class have certain common attributes:

1. They must begin and end on a page boundary.
2. Within themselves they occupy contiguous real pages.
3. They cannot be larger than 64 pages (32K).
4. They may not overlap another partition.

Note that the Secondary Task Memory Partition consists of all of the real memory that is not explicitly allocated to another partition and therefore the 64-page restriction does not apply to this class of memory.

An example of real memory allocation for a 64K system is shown in Figure 8.

CP-R SYSTEM MEMORY

CP-R system memory is comprised of one partition starting at memory location 16 (X'10') occupying a variable number of real memory pages sufficient to contain the following items:

- Space required for trap and interrupt locations.
- Space required for resident CP-R system pointers.
- Space required for resident CP-R system tables.
- Space required for resident CP-R routines including resident overlays.
- Space required for system patch area.
- Space required for CP-R temporary space pool.
- Space required for CP-R overlay area.

The layout of CP-R system memory is shown graphically in Figure 9.

CP-R system memory is given a write lock of 11. Primary and secondary tasks execute with a write key of 01 and 10, respectively, and thus are prevented from modifying CP-R system memory.

FOREGROUND PRIVATE MEMORY

This CP-R class of memory is used exclusively by foreground primary tasks. Foreground blocking buffers used by primary tasks may exist as a separate partition of foreground private memory or they may exist as an area within the first defined foreground private memory partition.

If foreground blocking buffers are specified as a separate partition then the restrictions governing partitions are in effect. If they are defined as an area then these restrictions are not in effect.

The foreground mailbox is a special area within the foreground private memory class. The foreground mailbox may exist as a separate partition of foreground private memory or it may exist as an area within the first defined foreground private memory partition.

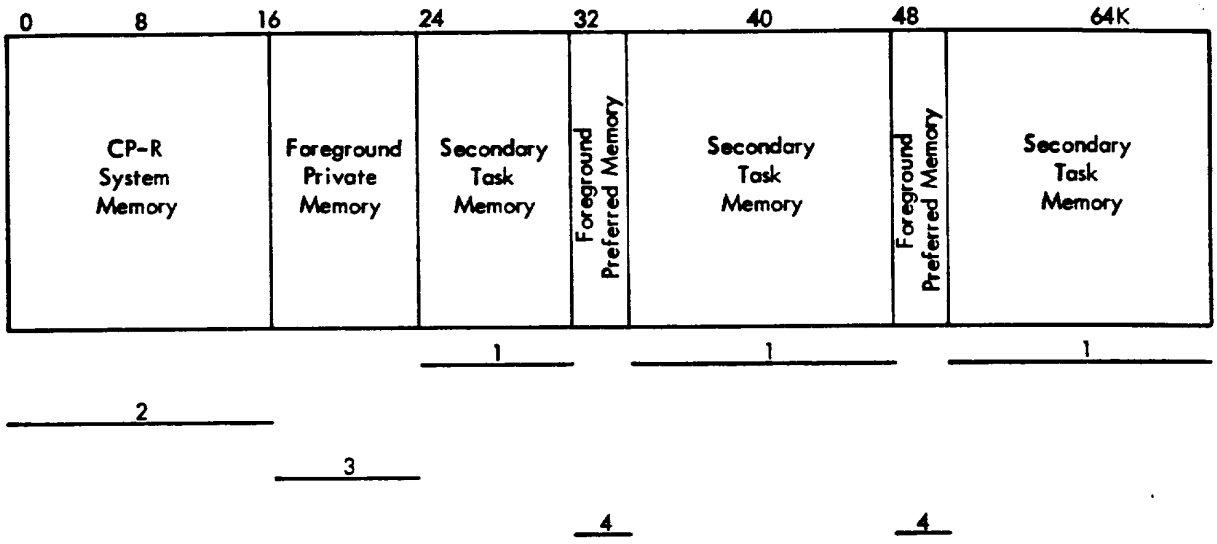
If the mailbox is a separate partition, it is subject to the restrictions placed on memory class partitions. If it is an area within the first partition of foreground private memory, then these restrictions are not in effect.

Foreground private memory is given a write lock of 01.

FOREGROUND PREFERRED MEMORY

This class of memory may be allocated to either primary or secondary tasks. Primary tasks may acquire memory pages in this memory class only through explicit calls to CP-R memory management routines giving the real memory address of the requested page(s)

Secondary tasks may acquire memory pages in this memory class by requesting virtual to real address correspondence



1. Secondary Task Memory is used for Foreground or Background Secondary Tasks.
2. CP-R System Memory for use by CP-R only.
3. Foreground Private Memory used by Foreground Primary Tasks only.
4. Foreground Preferred Memory may be used by either Foreground Primary Tasks or Foreground Secondary Tasks.

Figure 8. Example of Memory Organization for 64K System

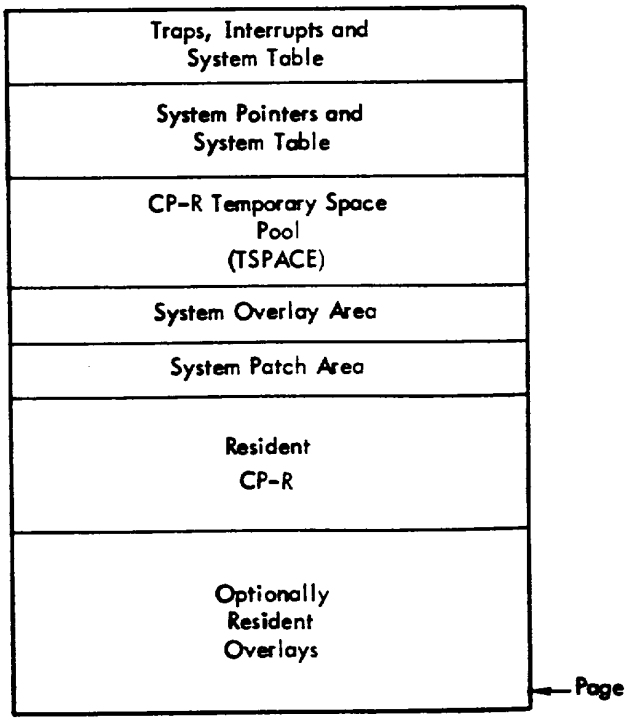


Figure 9. CP-R System Memory

for a particular segment via the FIX option on the :SEG or :PUBLIB commands when the task is linked with IOLOAD. In this case the real page that has address correspondence to the requested virtual page must be defined as foreground preferred memory.

Foreground preferred memory is given a write lock of 00 making it accessible from both foreground primary and secondary tasks.

SECONDARY TASK MEMORY

All real memory not explicitly defined as belonging to another memory class is called secondary task memory. All memory greater than 128K is secondary task memory and there may be pages of memory in the first 128K that fall into this class

This memory is used exclusively for secondary tasks and Public Libraries that are not used by primary tasks.

PRIMARY TASK MEMORY ALLOCATION

Primary tasks are allocated contiguous real memory at !RUN or !INIT time from information contained in the task load module. If Public Libraries are to be loaded with the task, the memory that they require must be available at !RUN or !INIT time or the task load cannot be completed.

More than one primary task load module may be loaded into a given foreground private partition, by issuing multiple RUN or INIT calls, providing that memory residency (as defined in the load module) conflicts do not occur.

Primary tasks may acquire and release memory pages in foreground-preferred partitions by using the GETPAGE and RELPAGE function calls. The requested memory must be available at the time of the call or the request is not completed. The memory acquired in this fashion does not belong to any named segment as such and, therefore, cannot be shared with any secondary task. It is considered private to the requesting primary task.

The user must "remember" the pages he has acquired so that he may release them when they are no longer needed.

The memory occupied by all tasks in a load module is released when any task in the load module terminates.

VIRTUAL MEMORY ALLOCATION

Each secondary task in CP-R has 128K of contiguous virtual memory available, regardless of the size of the real memory or of other users in the system. Some of this memory is available directly to the task and some is used for services that the task requests. Figure 10 shows the fundamental allocation of virtual memory.

SYSTEM VIRTUAL MEMORY

System Virtual Memory (SVM) occupies virtual page 0 through virtual page 47 and is divided into two separate areas, the first of which (CP-R) is common to all tasks, and the remaining area (CP-R overlays) is task dependent. SVM is given an access protection code of Read and Execute to prevent accidental destruction of critical system data.

Virtual pages 0 through 47 are reserved for resident CP-R routines. This area is mapped one-for-one with real memory; that is, there is address correspondence between real and virtual addresses.

One page of SVM is reserved for CP-R overlays. These overlays are part of secondary task context and are used by CP-R in providing certain system services to the user.

Note that the end of CP-R system memory does not necessarily end at the start of task virtual memory. Typically there is space left over in system virtual memory (which is always mapped one for one with real) that the user may define partitions in at his discretion. This allows user code to exist in both real and virtual memory if desired.

TASK VIRTUAL MEMORY

Task Virtual Memory (TVM) starts at page 48 (24K) and ends at page 239 (120K) providing 192 pages (96K) of virtual memory directly available to each secondary task. TVM is allocated, under user control, on a segment basis, where a segment consists of one or more pages. Control is exercised via a set of memory management operations which provide for activation, deactivation, overlay, and erasure of segments.

RESERVED PAGES

Sixteen pages (page 240 through 255) are reserved for system use on behalf of the user. These pages are used for such items as I/O blocking/deblocking buffers, Loader tables,

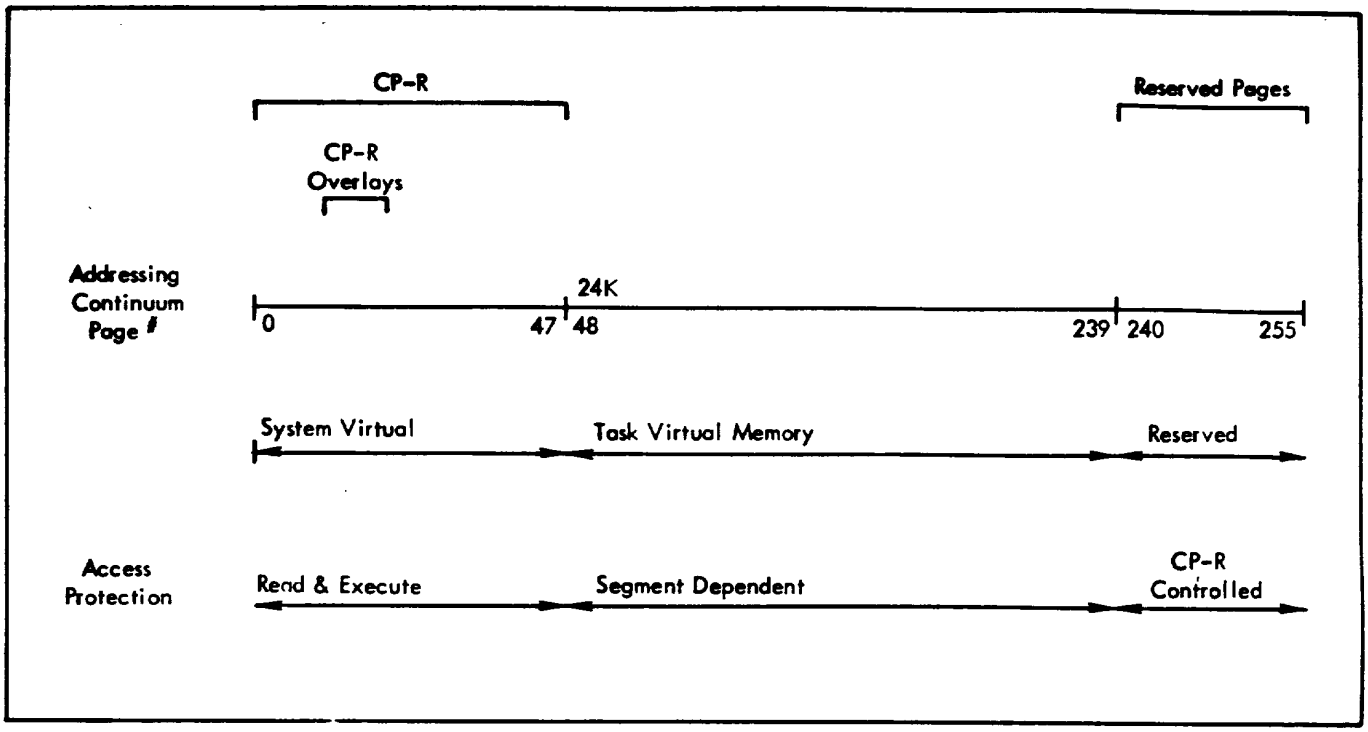


Figure 10. Virtual Memory Organization

debug tables, symbol tables and other system information as needed. The first 12 of these pages (240-251) are job-level pages and are shared by all tasks in that job. The remaining four pages are private to each task.

ACCESS PROTECTION

System virtual memory and overlay area is always marked read and execute. Task virtual memory access protection is controlled by the user on a segment basis. The reserved pages access protection is controlled by the system consistent with the type of data involved.

Initially, all pages in TVM that are undefined or not activated are marked no-access, an attempt to reference these pages results in trap conditions being returned to the user.

The user can control the access codes in TVM between 24K and 120K but cannot change access to SVM or the reserved pages. On dispatching every task, the access codes for all TVM is initialized to assure that all memory is properly protected.

SOFTWARE SEGMENTATION

The method of controlling the location of data or instructions in virtual memory is by means of a software segmentation scheme as follows:

1. The active memory at any time is composed of a series of segments under software control.
2. Each segment consists of an integral number of pages, from one to a maximum of 32 (the 32-page limit does not apply to the root segment). These pages within a segment are contiguous but there may be gaps in virtual memory between segments.
3. Each segment is built by the Overlay Loader from one or more relocatable object modules (ROMs) or library routines.
4. Each segment is given a number and a virtual memory starting location as well as length at linkage edit time. This number, virtual location, and length remain with the segment permanently.
5. Each segment must begin on a page boundary. Segments may contain data, instructions or reserved space, or a combination.

The segment is the smallest sharable unit of memory. Several segments may begin at the same virtual memory address if desired, and if so, result in their being overlay segments for each other. Only one of these may be active at any one time, and the active one is represented in the actual hardware map at execution time. Because each segment is independent of other segments, any sort of overlay structure may be built; the user is not limited to a conventional tree structure.

SEGMENT ACTIVITY

The segments of a task may be in one of two states relative to that task:

ACTIVE — The segment is in real memory and in the map if the task is active.

INACTIVE — The segment is currently not available to the using task. If the task is active, the segment is not on the map.

The above states reflect segment activity relative to a given task. They do not, however, reflect the actual state of a segment from a memory management standpoint. This comes about because of sharability, initialization, and roll-in/roll-out considerations.

For purposes of memory management, a segment has five distinct states:

ERASED — Segment contents do not exist in real memory but may be found in the tasks' Load Module (LM). This is the initial state of a segment before any segment activation has occurred. This state may also be reached if all tasks using a segment have ERASED it.

DEFINED — This is a substate of ERASED. This state represents an empty segment, one whose contents do not exist in the Load Module.

ACTIVE — One or more tasks have ACTIVATEd the segment. The segment contents exist in real memory and in the map of any active using task in which the segment is active.

INACTIVE — The segment contents exist in real memory; the segment is not active to any using tasks; and all using tasks have not erased the segment.

ROLLED-OUT — The segment contents exist in the roll-out file.

Table 15 shows the transition operators and their effect on a segment relative to a task. Figure 11 shows the possible state transitions of a segment.

Table 15. Segment States Relative to a Task

Operation \ Segment State	Segment Inactive	Segment Active
ACTIVATE	Activates requested segment from Load Module if segment state is ERASED, from memory if segment state is INACTIVE, from roll-out file if segment state is ROLLED OUT. ERROR if overlapping segments in virtual memory. If ACTIVATE is to DEFINED segment, real memory for the entire segment is acquired.	ERROR
DEACTIVATE	ERROR	Causes segment to be made inactive to this task. Actual segment state may or may not change depending on sharability. CAL or FPT may not be in segment that is being deactivated.
ERASE	Segment contents may be lost (real memory released) depending on sharability. Segment remains inactive to calling task but segment state may change to ERASED or DEFINED.	
GETPAGE	ERROR if segment state is not DEFINED. If page request to DEFINED segment acquires real pages and makes the segment ACTIVE to the calling task.	Acquires real memory.
RELPAGE	ERROR	Releases real pages. If all real pages in the segment are released the segment is marked inactive to the task and the segment state is either ERASED or DEFINED based on initial segment state when loaded.

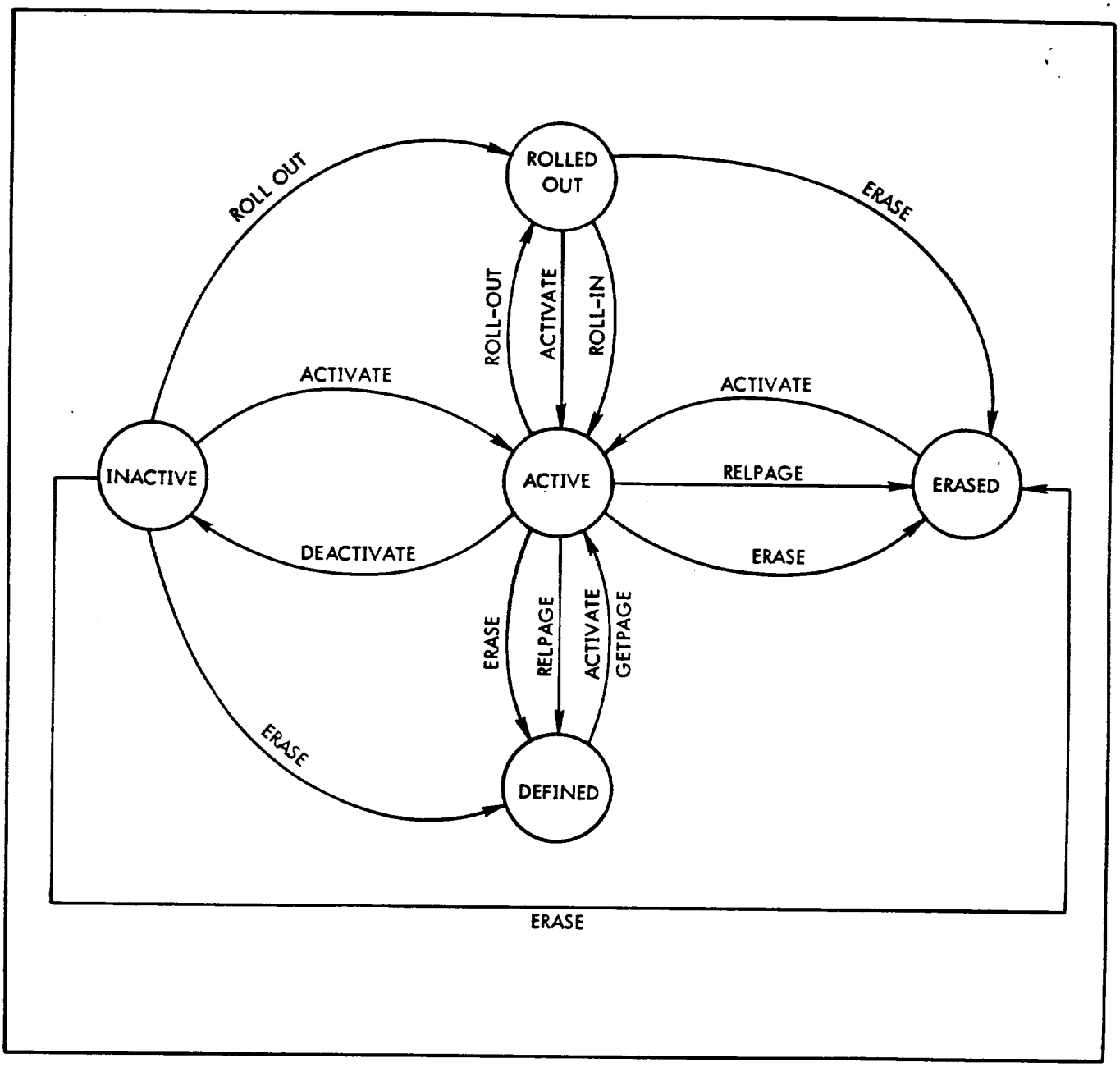


Figure 11. Segment States

SHARING SEGMENTS

The user may specify to the Overlay Loader that certain segments are to be shared between using tasks. A segment is system level if it is shared by two or more tasks within different jobs. A segment is job level if it is shared by two or more tasks in a job. A segment is task level if used exclusively by a single task. Segments are given numbers (by the user) that are used as parameters on CP-R service requests to manage the segment. Numbers need not be unique between jobs if a segment is defined by the user as job level. System level segments numbers must be unique. Public Libraries are system level segments that are managed by CP-R on behalf of the user.

Segments that are job level may only be used by tasks within a given job. System level segments may be used by any task in the system. An example of system level sharing are Public Library segments.

If all tasks sharing a segment terminate or erase it, it is erased. Therefore, the user must never allow all using tasks to erase a segment if the contents are to be permanently maintained. In this case, it is rolled-out (if real memory is needed) when it is not active to any task, and is rolled in if it becomes active again.

A system level segment is also erased if all using tasks within the using jobs terminate, or erase it.

SEGMENT ACCESS PROTECTION

Access protection for each segment is specified by the user through controls provided by the Overlay Loader. Segments may be given any one of the following types of protection.

- No Access
- Read Only
- Read and Execute
- Write, Read, or Execute

The given protection type applies to all pages in that segment on a permanent basis, that is, CP-R does not provide services to dynamically change the segments access protection.

SECONDARY TASK STRUCTURES

A mapped secondary task in CP-R is synonymous with a file which contains a header and a load module consisting of one or more segments. Load modules, when read into core memory and executed, become secondary tasks. The load module is written by the Overlay Loader (OLOAD) which inputs control cards, ROMs, PUBLIB linkage controls, etc.; performs all address resolution; develops absolute core images; and writes them on disk. It also writes the load module header which contains all control data required at INIT time to load and execute the task.

In order to understand the input options to OLOAD, the user must understand Memory Management in CP-R.

The availability of a memory map on Sigma 9 and use of virtual memory allows more flexibility in structuring overlaid programs.

Figure 12 illustrates a program with a conventional tree structure and an alternate structure, non-overlaid in virtual memory. The latter structure, if loaded in real memory, preempts large areas of real space which may be required by other tasks when activated. Such a structure for real memory programs is therefore not desirable.

Use of the linear tree structure is ideal for reentrant programs in virtual memory. Those segments that contain reentrant code may exist in core only once, and may be shared by multiple tasks that are simultaneously active. The ACTIVATE and DEACTIVATE calls affect only the status of a segment with respect to the calling task. Therefore, each task has a unique list of segments that must be core resident and CP-R assures that it is only given execution control when all its active segments are in real memory. The total real memory required for the combination of tasks at any instant may be greater than available. In this case,

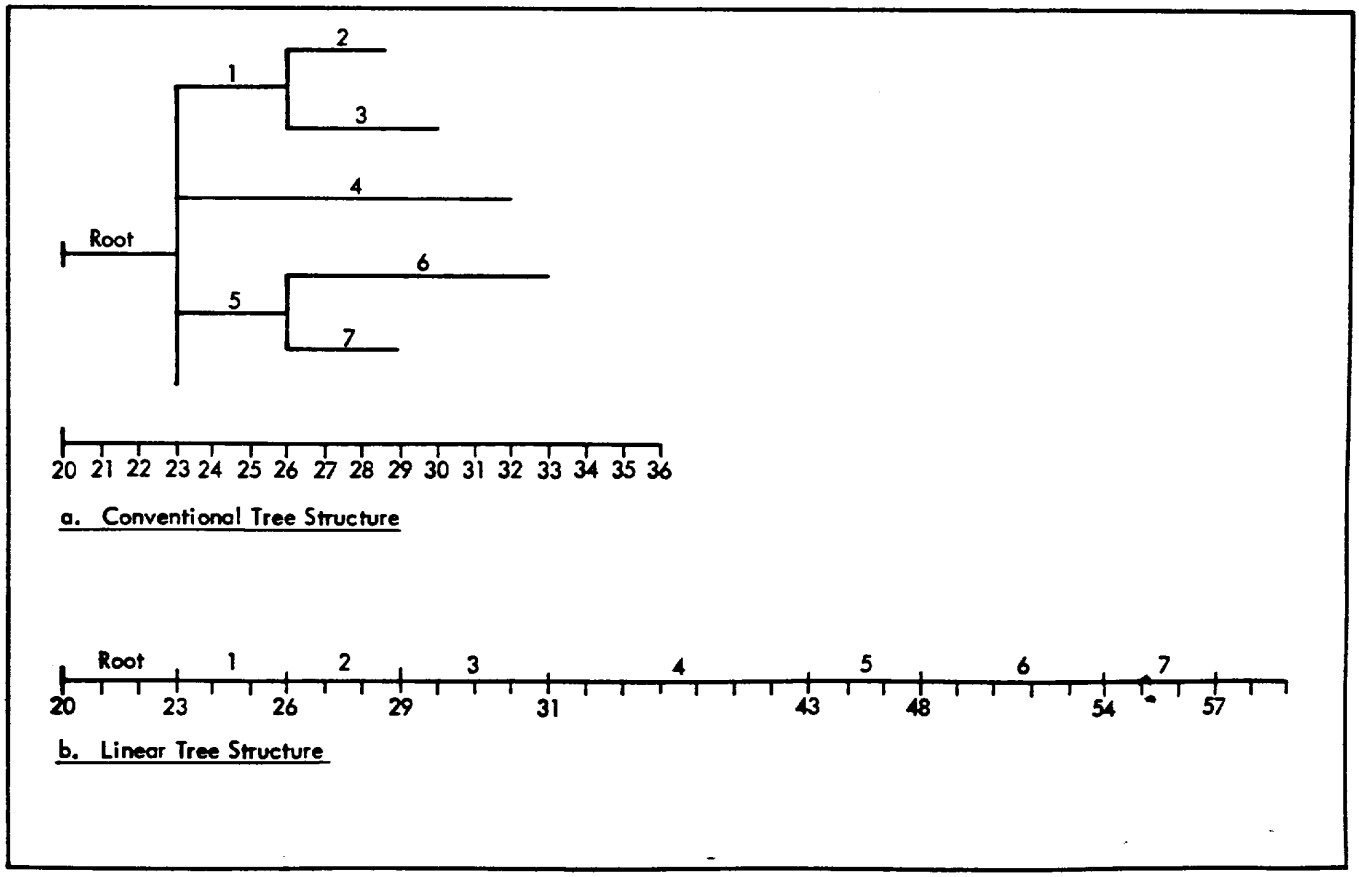


Figure 12. Tree Structure Options

the lowest priority task is made inactive and its segments are rolled out to free real memory pages. These pages are then available to satisfy the higher priority tasks' requirements. When enough core is available for the rolled-out segments to be returned, the inactive tasks are dispatched and allowed to complete.

LINKING A PROGRAM IN REAL VERSUS VIRTUAL MEMORY

The utilization of memory addresses by a program is largely identical in real and virtual memory with the characteristics shown in Table 16.

Table 16. Real and Virtual Memory Program Linkage

Real	Virtual
Range is 0 to last real memory address.	Range is 0–128K.
Access beyond 128K requires real extended addressing.	Simultaneous access of over 128K words is not possible.
CP-R monitor and services use some portion of real memory which is only available to the user as "read" or "read and execute".	CP-R monitor and services use some portion of virtual memory which is only available to the user as 'read' or 'read and execute'.
Real Public Libraries load into fixed real addresses as established when PUBLIB is OLOADed. User must not use this real space for any other purpose if he ever executes with the PUBLIB loaded, even if he doesn't call it.	Virtual Public Libraries load into fixed virtual addresses as established when the PUBLIB is OLOADed. User must not use this virtual space for any other purpose if he uses the PUBLIB. If he doesn't use the PUBLIB, he may use the virtual space.
Segments that are loaded into the same area in real memory cannot be simultaneously resident. The user is responsible for explicitly loading segments (by use of SEGLOAD) prior to using them and assuring that his logical flow and overlay structure are consistent.	Segments that are loaded in the same area of virtual memory can not be simultaneously in the map. The user must continue to do explicit ACTIVATES and properly structure logical flow and physical layout.
Segments that are loaded into unique real areas of real memory can be simultaneously resident and are always available for use once loaded via SEGLOAD.	Segments that are loaded into unique areas of virtual memory can be simultaneously resident, i.e., will use unique real pages. Availability in real memory and presence of the pages in the map are controlled via explicit system calls ACTIVATE, DEACTIVATE and ERASE.
The memory required for a segment is always available since it is acquired at task initiation as a part of the contiguous space in the private partition.	The core required for the segment must be acquired when the segment is activated, and is released when the segment is erased or rolled-out.
Segments are reloaded from the load module on SEGLOAD.	Segments are rolled-out and rolled-in from the disk as they are DEACTIVATED and ACTIVATED; therefore, can be modified during execution. They are reloaded from the load module on ACTIVATES after ERASES.
User tasks linked such that they overlap in real memory cannot execute simultaneously. The user must carefully plan his real memory utilization within the private partition.	User tasks linked to overlapping virtual addresses can be executed simultaneously on different tasks. That is, each task is completely independent of other tasks, except for PUBLIB or shared segments.

LINKING A PROGRAM USING SIMPLIFIED MEMORY MANAGEMENT

Secondary tasks may not use the SEGLOAD service, and may not access memory addresses which are not in an active segment. However, the FORTRAN SEGLOAD subroutine uses the SEGLOAD service, as do many programs written for use under Xerox RBM, from which CP-R developed. RBM programs also could use memory beyond their active segments. To support these as secondary tasks, a "Simplified Memory Management" (SMM) program structure has been provided. In this structure, the program, although secondary, is allowed to use the SEGLOAD function, and may access any address in a range determined when it is initiated; but it may not use any memory management services.

For a background program, the accessible address range starts at the beginning of the program and continues for the number of pages specified by the BMEM value. This value defaults to 32, but may be set both at SYSGEN and by operator key-in. Certain low-core locations are set to point at interesting locations in this range: K:BACKBG (X'140') points to the start of the range, K:BCKEND (X'141') points to the end of the range, and K:BPEND points at the first word past the end of the program as linked. This means that K:BPEND and K:BCKEND define an area which may be used as scratch storage.

For a foreground program, the range starts at X'6000' and extends for a number of pages defined in byte zero of cell K:FSMM (X'217'). This value may be set only by reassembling the SYSGEN processor.

A program with SMM structure is treated by memory management as only two segments, which are always active. (The user, however, deals with the segment structure defined when the program was linked.) One of these segments covers the entire program as it was linked. The other covers the virtual addresses between the end of the program and the end of the accessible range described above. Since these segments are always active, the program will be rolled out whenever there is not enough secondary task memory for its entire accessible address range available at its priority.

MEMORY MANAGEMENT SYSTEM CALLS

Memory management system calls may be made only when a user has been given control as follows:

1. The background or foreground loader has transferred control to the user's start address.

2. The user's centrally connected primary task has been given control upon the occurrence of the associated interrupt.

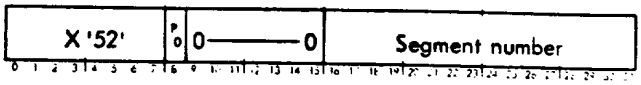
The function calls are as follows.

ACTIVATE This function call activates the specified segment. The call has the format:

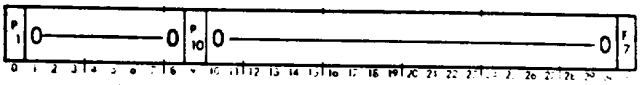
```
CAL1,7   address
```

where address points to word 0 of the FPT shown below:

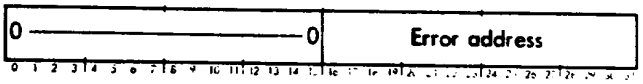
word 0



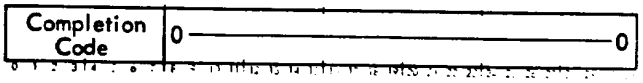
optional (P0 in word 0)



optional (P1)



optional (P10)



where

Word 0

X'52' is the code for the ACTIVATE call.

P_0 is the parameters presence indicator (0 means absent; 1 means present).

Segment number identification of the user's segment.

Word Options

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_{10} is the completion code parameter presence indicator (0 means absent; 1 means present).

F_7 is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

Error address is the location to return to if errors are detected.

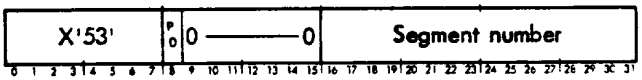
Completion code will be posted by the system indicating the outcome of the service.

DEACTIVATE This function call deactivates the specified segment. It has the format:

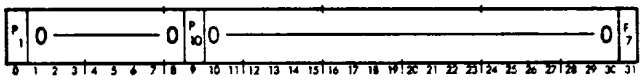
CAL1,7 address

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

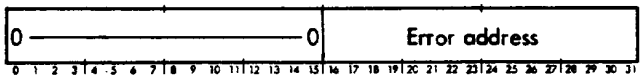
word 0



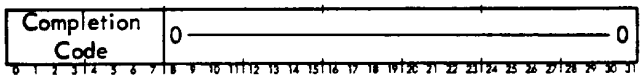
optional (P0 in word 0)



optional (P1)



optional (P10)



where

Word 0

X'53' is the code for the DEACTIVATE call.

P_0 is the parameters presence indicator (0 means absent; 1 means present).

Segment number identification of the user's segment.

Word Options

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_{10} is the completion code parameter presence indicator (0 means absent; 1 means present).

F_7 is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

Error address is the location to return to if errors are detected.

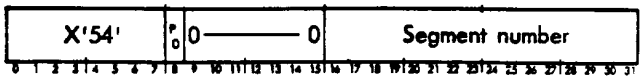
Completion code will be posted by the system indicating the outcome of the service.

ERASE This function call erases the specified segment. It has the format:

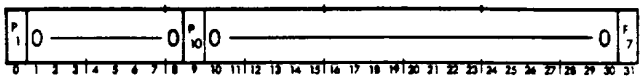
CAL1,7 address

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

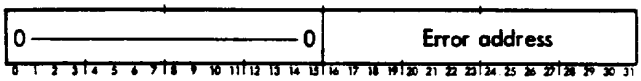
word 0



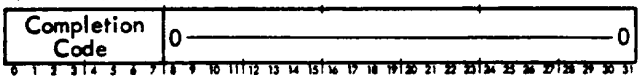
optional (P0 in word 0)



optional (P1)



optional (P10)



where

Word 0

X'54' is the code for the ERASE call.

P_0 is the parameters presence indicator (0 means absent; 1 means present).

Segment number identification of the user's segment.

Word Options

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_{10} is the completion code parameter presence indicator (0 means absent; 1 means present).

F_7 is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

Error address is the location to return to if errors are detected.

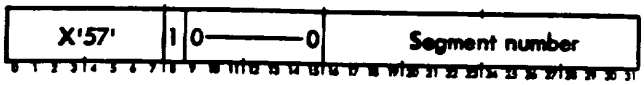
Completion code will be posted by the system indicating the outcome of the service.

GETPAGE This function call gets pages of memory for the specified segment. If the segment is not active, it is activated. The call has the format:

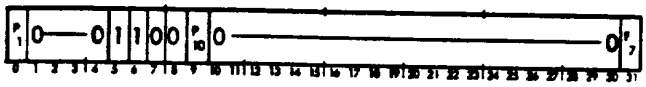
CAL1,7 address

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

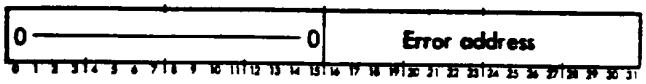
word 0



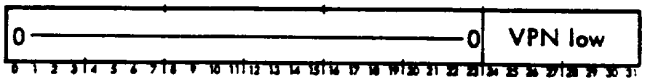
word 1



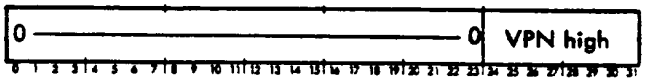
optional (P1)



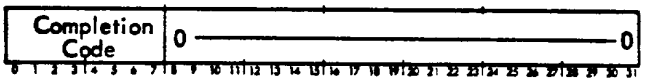
required (P6)



required (P7)



optional (P10)



where

Word 0

X'57' is the code for the GETPAGE call.

Segment number identification of the user's segment.

Word 1

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_{10} is the completion code parameter presence indicator (0 means absent; 1 means present).

F_7 is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

Word Options

Error address is the location to return to if errors are detected.

VPN low is the virtual page number of the lowest page requested. This is the high order 8 bits of the 17 bit virtual memory address of the page. VPN low is set equal to VPN high + 1 on completion of the service.

VPN high is the virtual page number of the highest page requested. This is the high order 8 bits of the 17 bit virtual memory address of the page. VPN high is set equal to VPN high + 1 on completion of the service.

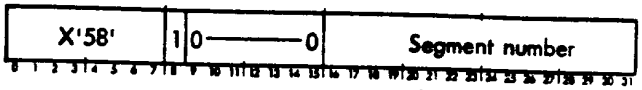
Completion code will be posted by the system indicating the outcome of the service.

RELPAGE This function call releases pages of memory from the specified segment. If all pages are released, the segment is erased. The call has the format:

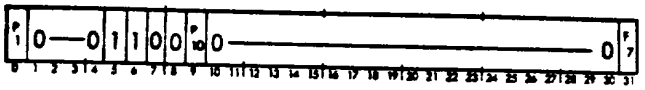
```
CAL1,7  address
```

where address points to word 0 of the FPT shown below:

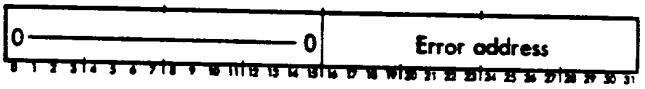
word 0



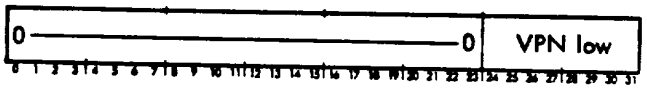
word 1



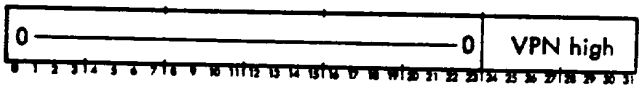
optional (P1)



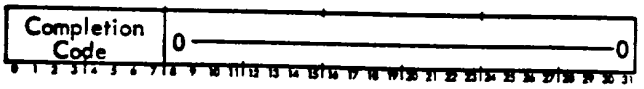
required (P6)



required (P7)



optional (P10)



where

Word 0

X'58' is the code for the RELPAGE call.

P_0 is the parameters presence indicator (0 means absent; 1 means present).

Segment number identification of the user's segment.

Word 1

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_{10} is the completion code parameter presence indicator (0 means absent; 1 means present).

F_7 is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

Word Options

Error address is the location to return to if errors are detected.

VPN low is the virtual page number, plus one, of the lowest page to be released. This is the high order 8-bits of the 17-bit virtual address of the page. VPN low is set to VPN low, minus 1, on completion of the service.

VPN high is the virtual page number, plus one, of the highest page to be released. This is the high-order 8-bits of the 17-bit virtual address of the page. VPN high is set to VPN low, minus 1, on completion of the service.

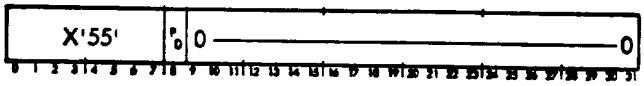
Completion code will be posted by the system indicating the outcome of the service.

LOCK This function call locks all active segments in the task in core memory inhibiting roll-out for the calling task. It has the format:

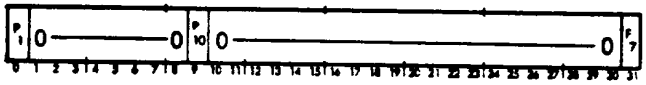
CAL1,7 address

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

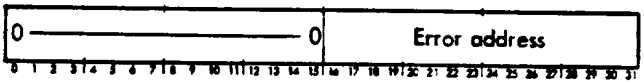
word 0



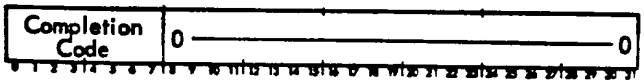
optional (P0 in word 0)



optional (P1)



optional (P10)



where

Word 0

X'55' is the code for the LOCK call.

P_0 is the parameters presence indicator (0 means absent; 1 means present).

Word Options

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_{10} is the completion code parameter presence indicator (0 means absent; 1 means present).

F_7 is the abort override indicator. If set, control will be returned to calling instruction + 1, if errors are detected and no error address parameter has been provided.

Error address is the location to return to if errors are detected.

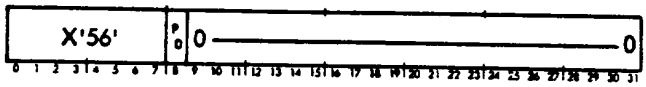
Completion code will be posted by the system indicating the outcome of the service.

UNLOCK This function call unlocks all segments in the task and makes them candidates for roll out. It has the format:

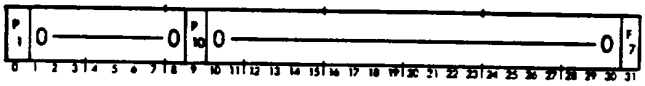
CAL1,7 address

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

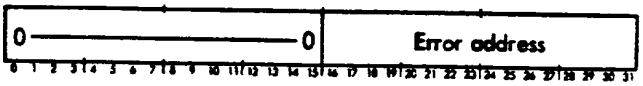
word 0



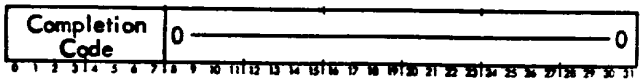
optional (P0 in word 0)



optional (P1)



optional (P10)



where

Word 0

X'56' is the code for the UNLOCK call.

P_0 is the parameters presence indicator (0 means absent; 1 means present).

Word Options

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_{10} is the completion code parameter presence indicator (0 means absent; 1 means present).

F_7 is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

Error address is the location to return to if errors are detected.

Completion code will be posted by the system indicating the outcome of the service.

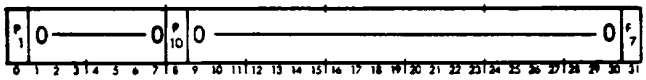
PREFMODE This function call, available only to foreground tasks, is used to control the availability of foreground preferred partitions. When the function call is executed, all pages in the specified preferred partition that have not been acquired by a task will be released (release mode) to the free page pool (STM memory) or acquired from the free page pool (recover mode). The call has the format:

CAL1,7 address

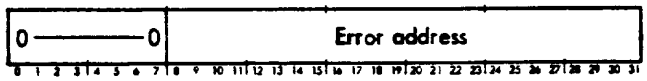
where address points to word 0 of the FPT shown below:

word 0

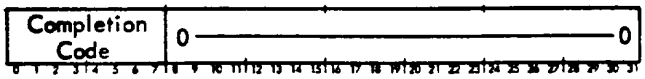
optional (P0 in word 0)



optional (P1)



optional (P10)



where

Word 0

X'50' is the code for the PREFMODE call.

P_0 is the parameters presence indicator (0 means absent; 1 means present).

MODE is 0 if operation is the recover STM pages and is 1 if operation is to release the partition to STM.

Partition Base Address is the starting address of the preferred partition that the operation is to affect. If this field is zero the operation will take place on all preferred partitions.

Word Options

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_{10} is the completion code parameter presence indicator (0 means absent; 1 means present).

F_7 is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

Error address is the location to return to if errors are detected.

Completion code will be posted by the system indicating the outcome of the service.

ROLL-OUT/ROLL-IN

Roll-out and roll-in of mapped secondary tasks is performed automatically by CP-R as demand for memory resources varies. The roll-out process is controlled on a segment basis with selection of segments to be rolled-out based on segment state, number of tasks sharing the segment, and the priority of the using tasks.

The selection algorithm is constructed such that there are nine distinct levels of roll-out. Within each roll-out level, selection of candidate segments is based strictly on the using task priority; that is, low priority segments will be selected before higher priority segments.

The advancement of roll-out level is controlled by two factors:

1. Roll-out will be stopped when a sufficient number of memory pages have been acquired to satisfy the highest priority memory request.

2. Roll-out will be stopped when the highest priority memory request would require that active segments of a task of equal or higher priority be rolled out (with the exception of tasks on long-wait).

The roll-out levels are presented in Table 17. When level 9 is reached the message "MEM SATURATED" is output on the OC device and the system will wait for a change in memory status.

Roll-in of segments that were active when rolled out is performed in priority order. That is, highest priority segments will be rolled in first. Roll-in of segments that were inactive when rolled-out is done when an ACTIVATE call is issued for the segment. In this case the roll-in will be done at the priority of the requesting task.

Table 17. Roll-Out Levels

Level	Segment Selection
1	Background inactive segments.
2	Long-wait task's inactive segments.
3	Long-wait task's active segments.
4	Requesting task inactive segments.

Table 17. Roll-Out Levels (cont.)

Level	Segment Selection
5	Normal task's inactive segments.
6	Overlay pages.
7	Background active segments.
8	Normal task's active segments.
9	Memory saturated.

7. ASYNCHRONOUS OPERATION CONTROL

Asynchronous operations consist of independent events that can take place concurrently with task execution. The available asynchronous services and their interface with an executing task are more fully defined in Appendixes H and I, "Job Management" and "Task Management". The monitor functions that may operate on these asynchronous services are described below.

CHECK COMPLETION

CHECK The CHECK function tests the type of completion of an asynchronous operation initiated by a no-wait request. The user specifies addresses, which are entries to his routines, that handle error and abnormal conditions. At entry, register 10 contains the error or abnormal code as detailed in Appendix N. Background users may take advantage of the standard system handling of the error and abnormal address in the FPT. The action taken by the system in this case is also detailed in Appendix N. Foreground users must provide both error and abnormal addresses when checking (CHECK, no-wait) requests.

Users may specify a CHECK with no-wait by including a busy address in the FPT. This address is taken (with the address of the location following the CHECK CAL1 in register 8), if the CHECKed operation is not complete. If no busy address is included in FPT, the CHECK function will wait for completion before taking the appropriate action.

The CHECK function (through its own FPT) addresses a DCB or an FPT, depending upon whether the request was Type I or Type II. The FPT associated with a request is addressed if the request was Type II and the completion parameters were posted in the FPT by the I/O system. A DCB is addressed if the request was Type I and the completion parameters were posted in the DCB.

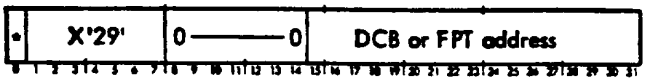
For non-I/O operations, the CHECK request always addresses an FPT.

The CHECK function call is of the form

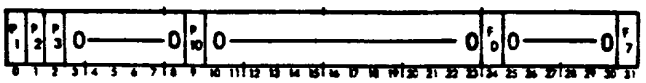
CAL1, 1 address

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

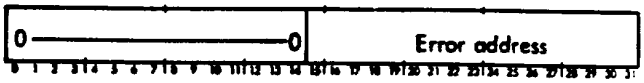
word 0



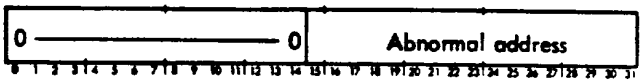
word 1



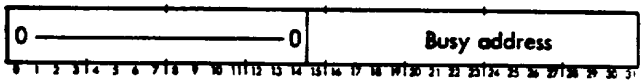
optional (P1)



optional (P2)



optional (P3)



where

Word 0

X'29' is the code for the CHECK function.

DCB or FPT address is the address of the DCB of FPT where the completion status is posted. P_{10} determines whether this field contains a DCB or FPT address.

Word 1

P_1 is the error address parameter presence indicator (1 means present; 0 means absent).

P_2 is the abnormal address parameter presence indicator (1 means present; 0 means absent).

P_3 is the busy address parameter presence indicator (1 means present; 0 means absent).

P_{10} is a bit indicating whether a DCB or FPT is addressed (0 means DCB; 1 means FPT).

F0 is the long wait indicator; makes secondary task a prime candidate for roll-out.

F7 = 1 specifies an abort override (return to next instruction) if an error occurs.

Word Options

Error address is the address of the entry to the user's routine that will handle error conditions.

Abnormal address is the address of the entry to the user's routine that will handle abnormal conditions.

Busy address is the address of the entry to the user's routine that will handle the "request busy" conditions.

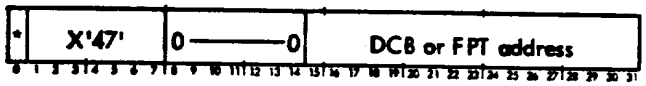
DELFPT (Delete a Requested Service)

This function forces premature completion and/or aborts the request to the greatest possible extent. This function is of the form

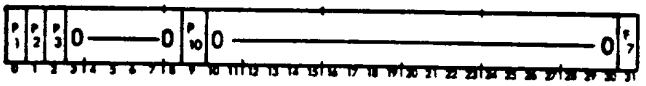
CAL1,7 address

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

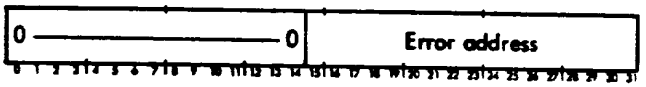
word 0



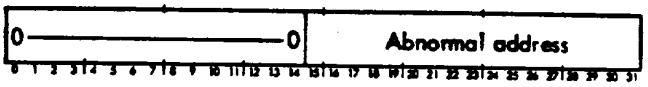
word 1



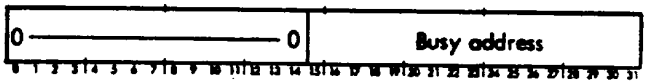
optional (P1)



optional (P2)



optional (P3)



where

Word 0

X'47' is the code to call DELFPT.

DCB/FPT address is the location of the FPT (DCB in Type 1 I/O only) with which service was originally requested.

Word 1

P_1 Indicates the error address is present.

P_2 Indicates the abnormal address is present.

P_3 Is the busy address parameter presence indicator (1 means present; 0 means absent).

P_{10} Indicates either the FPT or DCB address in word 0 (1 means FPT; 0 means DCB).

F_7 Indicates that if an optional error and abnormal address is reached but was not provided, return is to the next instruction versus abort.

Word options

Error address is the address where all non-I/O errors exit, plus any I/O errors of the FPT error address class.

Abnormal address is the return location for FPT abnormal address I/O errors.

Busy address is the address to return to if the service being checked is not complete.

If the FPT or DCB address provided does not have a service outstanding, the FPT or DCB is not altered in any way and the service normal, error, or abnormal exit is taken, based on the last completion in the FPT/DCB whose service is being deleted.

If the service is still in process, it will be processed until a wait condition occurs.

If $P_3 = 1$, a busy address is provided and will be taken. The user must issue another CHECK or DELFPT later.

If $P_3 = 0$ and the service being checked is not one that can legally be waited for, the task is aborted. Otherwise, the task will wait for the service to be completed, at which time the exits described below will apply.

If the completion was normal (completion code = 01), the status and any feedback data will be stored per the original FPT, and the next instruction exit will be taken.

If the completion was in error (on I/O services, those conditions taking the FPT error exit; on all other services, any nonzero completion code), the status and feedback data will be stored per the original request FPT. R8 will be set to the DELFPT CAL+1. Byte 0 of R10 will be set to the error code. Bytes 1-3 of R10 will be set to an address; to the DCB address if the request was for I/O; to the request FPT address if the request was another service.

If an error address was provided on the DELFPT FPT, control will be returned to it.

If no error address was provided but $F_7 = 1$, the normal exit will be taken.

If no error address was provided and $F_7 = 0$, the callers task will be trapped to the BADCAL trap (X'50').

If an abnormal I/O completion occurred, the logic is as above using the abnormal address parameter. That is, the original FPT is completed, R8 and R10 are set and the users abnormal address is given control. If the abnormal address is not present, return is to the instruction following the DELFPT call with R8, R10 unchanged.

Tasks must provide a busy address, on DELFPT of services for which wait is illegal. Requests without a busy address by a task will cause the caller to be trapped.

Fields in the original request FPT which will be stored on completion are the completion code (optional) and data from the data area (optional).

Service requests should be deleted only once after they are completed. If a busy address is provided and CP-R exits to it the service is still in process. Once a normal, error, or abnormal exit is taken, the service has been deleted.

Services can only be deleted by the original requesting task or another task in the same load module. The quiescing of the service and removal from the system will vary as a function of the activities in process. Waits may occur in order to bring the request to a controlled termination. The busy address will be used if provided when a wait is encountered.

Multiple, simultaneous deletes of a service by more than one task cannot be allowed. This has the following significance:

- A CHECK or DELFPT on a service which was requested with wait will cause a busy return (if provided) or trap (no busy return is provided).

- A CHECK or DELFPT on a service which is in CHECK, DELFPT, or TEST processing by another task will cause a busy return (if provided) or a trap (no busy return provided).

These conditions can be avoided by proper coding of CHECKs and TEST calls, or by only allowing one of the tasks to do all checks and/or tests.

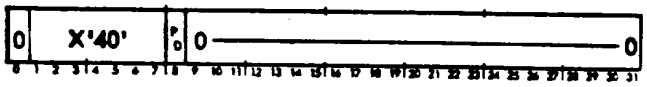
If a task does a service request without wait using an FPT in registers, the register content must be the same at check time as it was when the service was requested.

WAITALL This function call allows a task to wait until all previously requested no-wait services have completed. It has the form

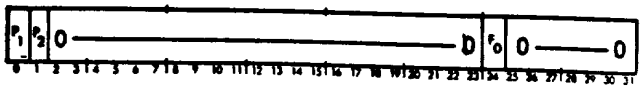
CALl,7 address

where address points to word 0 of the FPT shown below:

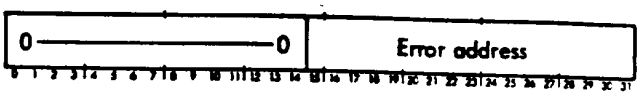
Word 0



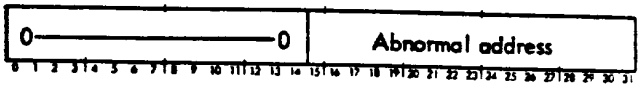
optional (P0 in word 0)



optional (P1)



optional (P2)



where

Word 0

X'40' is the code identifying the WAITALL function.

P0 is the word-1 parameter presence indicator (1 indicates present).

Word options

P_1 indicates the error address is present.

P_2 indicates the abnormal address is present.

F0 is a long-wait indicator, making a secondary task a prime candidate for roll-out.

Error address is the address where all non-I/O errors exit, plus any I/O errors of the FPT error address class.

Abnormal address is the return location for FPT abnormal address I/O errors.

If any previously requested service is complete, a CHECK will be performed; that is, the completion code and data will be moved to the originally requested FPT. WAITALL will return only if it has checked and deleted all outstanding services.

Tasks cannot use WAITALL if any previously requested service cannot legally be waited upon. WAITALL will wait for all services requested by any task in the caller's load module.

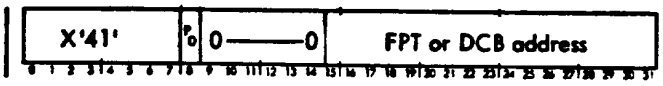
If any of the "checked" services completed in error, registers 8 and 10 will be set as shown in Appendix N and the return will be to the specified error or abnormal address in the WAITALL FPT. If any of the services completed in error and the error address is not present, the task will be aborted.

WAITANY This function call allows a task to wait for any previously requested service to complete. WAITANY has the format

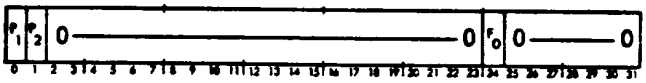
CAL1,7 address

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

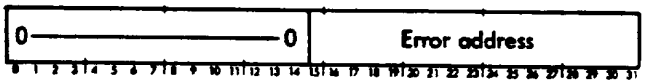
Word 0



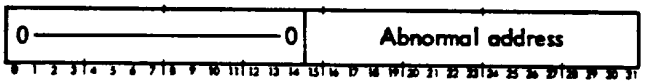
optional (P0 in word 0)



optional (P1)



optional (P2)



where

Word 0

X'41' is the code identifying the WAITANY function.

P_0 is the word-1 parameter presence indicator.

FPT or DCB address is the field where WAITANY returns the address of an original service-request FPT or DCB (for type 1 I/O) which has been completed and checked. If FPT or DCB address is zero on return from WAITANY call, no outstanding requests were found.

Word options

P_1 Indicates the error address is present.

P_2 Indicates the abnormal address is present.

F_0 is the long-wait indicator, making a secondary task a prime candidate for roll-out.

Error address is the address where all non-I/O errors exit, plus any I/O errors of the FPT error address class.

Abnormal address is the return location for FPT abnormal address I/O errors.

If any service is complete, a CHECK will be performed; that is, the completion code and data will be moved to the original request FPT. WAITANY will then return to the caller. The address of the FPT that was checked is stored in the WAITANY FPT.

If the "checked" service completed in error, registers 8 and 10 will be set as shown in Appendix N and the return will be to the specified error or abnormal address in the WAITANY FPT.

If no services are complete, WAITANY will not return control until at least one service has completed. WAITANY will return control immediately if no service requests are outstanding.

Tasks may not use WAITANY if all previously requested services cannot legally be waited upon. WAITANY will wait for any service requested by any task in the caller's load module.

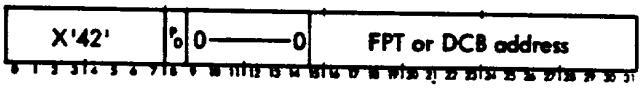
Services originally requested using FPTs in registers will give unpredictable results on WAITANY.

TEST This function call allows a task to test if any previously requested services have completed. TEST is like a WAITANY which never WAITS but always returns control to the user. This function call has the format

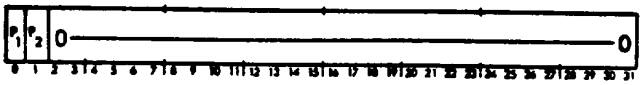
CAL1,7 address

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

Word 0

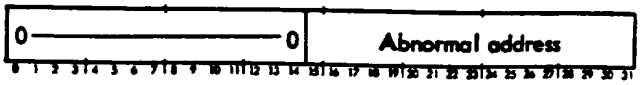


optional (P0 in word 0)



optional (P1)

optional (P2)



where

Word 0

X'42' is the code to call the TEST function.

P_0 is the word-1 parameter presence indicator (1 indicates present).

FPT or DCB address is the field where TEST returns the address of an original service-request FPT or DCB (for Type I I/O) that has been completed and checked. If FPT or DCB address is zero on return from a TEST call, none of the outstanding requests are completed.

Word options

P_1 Indicates the error address is present.

P_2 Indicates the abnormal address is present.

Error address is the address where all non-I/O errors exit, plus any I/O errors of the FPT error address class.

Abnormal address is the return location for FPT abnormal address I/O errors.

If a service is complete, a CHECK will be performed; that is, the completion code and data will be moved to the original request FPT and the address of the FPT that was checked is stored in the TEST FPT. If no service is complete, TEST will return control immediately and the FPT address field in word 0 of TEST FPT will be zero.

If the "checked service completed in error, registers 8 and 10 will be set as shown in Appendix N and the return will be to the specified error or abnormal address in the TEST FPT.

TEST will check any service requested by any task in the caller's load module.

Services originally requested using FPTs in registers will give unpredictable results on TEST.

8. CP-R DEBUG SERVICE

The CP-R Debug monitor service provides the foreground and background (batch stream) user with a versatile set of commands that allow the user to modify, examine and control the execution of a job. Both primary and secondary tasks may use the debug service; however, only one task per job can have debug control at any given time.

The Debug monitor service consists of a set of CP-R overlays that are called to perform the requested Debug functions. Debug control of a task in a given job can be established by any of the following:

- Debug system call.
- Debug Key-in.
- Debug option on INIT system call, key-in or control command.
- Debug option on RUN system call.
- Debug option on !RUN control command.

Control of Debug is through a set of operational labels assigned to the devices that will be used to input commands and output dumps, snapshots, etc.

These operational labels are given default assignments at SYSGEN time but the user is responsible for establishing the appropriate assignments in the JOB to be debugged.

Debug has provisions for accepting an unsolicited interrupt or break function that allows the user to interrupt or break out of the task being debugged and to give control to debug. Two methods are provided for this feature. At SYSGEN time the user may associate an external interrupt with a keyboard printer (TYxxx device type). When one of these interrupts is received, Debug will associate the interrupt with a JOB using the specific keyboard printer and will cause any task that has Debug control to return to Debug for input. For primary tasks this control break function only occurs when the task is returning from a monitor service call. For secondary tasks, control can be returned to Debug at any time.

The second method of causing this break function is from the operators console by using the BREAK key-in.

DEBUG CALL

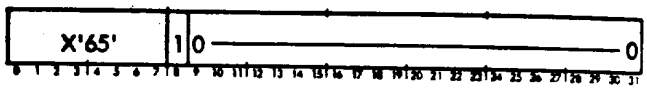
The Debug monitor service may be invoked by execution of the following monitor service call.

DEBUG This call will establish Debug control in the named task. The call has the following format

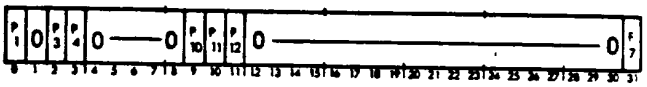
```
CAL1,7    address
```

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

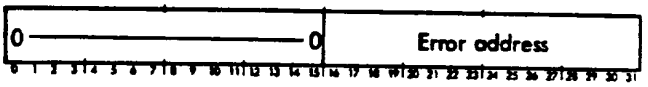
word 0



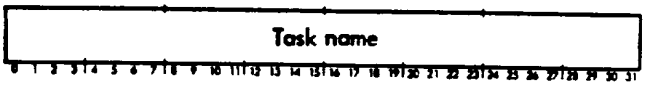
word 1



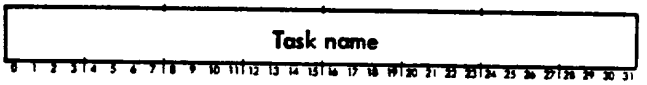
optional (P1)



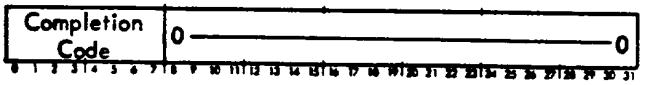
optional (P3)



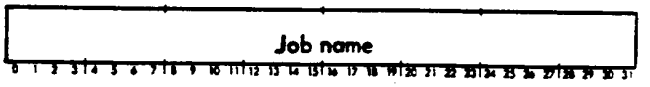
optional (P4)



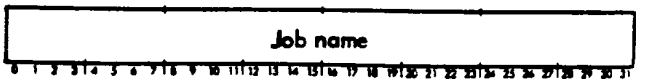
optional (P10)



optional (P11)



optional (P12)



where

Word 0

X'65' is the code to initiate the call.

Word 1

P_1 is the error address parameter presence indicator (0 means absent; 1 means present).

P_3-P_4 are the task name parameter presence indicator (0 means absent; 1 means present).

P_{10} is the completion code parameter presence indicator (0 means absent; 1 means present).

P_{11}-P_{12} are the job name parameter presence indicators (0 means absent; 1 means present).

F7=1 is the abort override indicator. If set, control will be returned to calling instruction + 1 if errors are detected and no error address parameter has been provided.

Word Options

Error address is the return address if errors are found.

Task name is the name of the task to be run with DEBUG. The caller's task is the default name.

Job name is the name of the JOB in which the named task is being executed. The caller's job is the default name.

METHOD OF OPERATION

To make the most effective use of the provided Debug facilities, the user should assemble, load, and run his job with a knowledge of Debug functions.

The Debug service allows the user to reference locations by using symbols. These symbols may be defined by the user at debug time and need not be related to the symbols in the user's programs. If the user wants Debug to intercept traps he should not make his own trap CALs.

After the user has established the Job/console/operational label assignments, any INIT of a task with Debug will transfer control to Debug at the console assigned to that job. This will cause the message TASK IS XYZ to be output where XYZ is the name of the task being initiated. Debug then prompts with a ":". At this time, the user can change the prompt if desired, using the NP command. The user can modify the task or start execution of the task.

Debug uses the following oplabels for communication:

DL is the oplabel for Debug log messages. These include messages describing trap conditions, messages from snapshots and other control output. If not otherwise specified, DL will be defaulted to the user's debug console.

DI is the oplabel for Debug input. All on-line commands are read through DI.

DP is the oplabel used for reading patches. The input read on DP is assumed to be in the format described in the "Debug Commands Used On Line and From Batch" section.

DO is one of the oplabels used for Debug output. This is normally assigned to the line printer and may be used by Debug for longer outputs.

P1, P2 are the other oplabels used for Debug output. Each of these can be assigned to a device and used for output from dump or snapshot commands.

DEBUG INPUT

Background jobs using Debug services from the batch stream can assign DI to the control command input. When a job is to be debugged on-line, the DI oplabel is assigned to a keyboard printer. There is only one keyboard printer allowed per job and this is designated the control console. Although input and output may use other devices, all on-line control comes from the control console. The control console may be a debug console or the operators console. Input may also come from the DP (Debug Patch) oplabel via a P command.

When a user is debugging on-line, all unsolicited messages and requests for input are routed to the control console. An interrupt is provided at each control console to allow the user to initiate unsolicited input.

Debug is normally used from a debug console. This is a standard keyboard printer, but it is distinct from the operators console. It has an external interrupt level available for unsolicited input.

The operators console can be used as a control console. In this case, the keyboard printer and the console interrupt will be shared between Debug and CP-R. All Debug output and requests for input are easily discernable from CP-R output, since CP-R messages start with !! and Debug messages start with the user-specified prompt character. The user will immediately know whether to respond to CP-R or Debug. If the user wishes to do an unsolicited input to Debug he should do a key-in to CP-R as follows: CINT XX, where XX is the hexadecimal number of the interrupt allocated to Debug or use the BREAK key-in. Output messages from Debug and CP-R will be queued and will appear on the OC device.

DEBUG OUTPUT

Although a user can only have one control device for each job, he can use multiple output devices. Each separate dump can be routed to a different output device.

If no output device is specified, the output will be done on the control console (DL) device.

A user can also output to any standard peripheral. The A command is used to associate devices with oplabels. If multiple users select the same output device from Debug, their output will be intermixed. The users must do their own device allocation to avoid mixing output.

DEBUG TRAP CONTROL

When a program is initiated with Debug, it connects all traps (X'40'-X'50') to an entry in Debug. When a trap occurs, Debug prints the following message:

:TRAP X/Y AT SYMBOL + VALUE

where

: is the trapped task user prompt character.

X is the hexadecimal value of the trap.

Y is the CC set by the trap logic. For traps 40 and 42, CC identifies the type of trap. Debug will then prompt and request input. If a trap occurs in DEBUG the message

:TRAP X/Y at DEBUG + VALUE

will be printed. In this case the user's registers and the value of $I will not be modified. Traps will occur in Debug if the user attempts to dump no-access memory or modify write protected memory. Any attempt by the user to connect traps will override DEBUG trap management.

DEBUG COMMANDS

Debug commands are provided to allow the user to modify, examine, and monitor the execution of his task. The commands and a description of their functions follow.

- A Assign device to DEBUG oplabel
- C Control segments
- M Modify memory
- I Insert code logically
- N Give a value to a name (symbol)
- P Read patch cards
- B Branch to the users code – move snapshot
- D Dump selected parts of core
- E Execution control (task)
- R Remove snapshot, insertion, or names
- L Look at memory for a match
- Q Quit task(s)
- S Snapshot definition

DEBUG COMMAND SYNTAX

The required debug command formats are the same whether commands are read from a user's debug console or from the patch file. All patch cards must have a command identifier as the first character. When Debug is used in the patch mode, all commands are read and interpreted by Debug immediately. Debug uses monitor CALs to do its input/output. Each command must be contained on one line. Multiple blanks are treated as a single blank and blanks are used as field delimiters where shown in the commands.

Each debug control card will be of the form:

Command Identifier options

The Command Identifier must be delimited by one or more blanks.

Each control console command will be of the form:

Command Identifier options (NL)

For debug commands the following definitions and conventions will be used:

symbol From one to eight EBCDIC characters as described in the AP Assembler character set. One character must be an alphabetic character. All Sigma machine mnemonics are reserved to Debug. $ is the value of the current location counter (first location patched) for modifies and inserts. $C is the current value of the condition codes. $F is the current value of the floating controls. $I is the value of the instruction counter (the next location to execute).

value From one to ten numerals or hexadecimal values. If the number is preceded by a ".", the number will be treated as hexadecimal. A value may be preceded by a + or - sign.

register
1. A positive value ≤ 15.
2. A symbol whose value is ≤ 15.

loc
1. A positive value.
2. NAME, where NAME is a symbol defined in Debug.

loc (cont.)

3. Sums or differences of either of the above two forms.

4. One of the above forms preceded by an "*". "*1" refers to the location pointed to by register 1.

word

1. A signed (plus sign (+) optional) hexadecimal or decimal value.

   Examples: -6, 100,.2A, -.AF.

2. A name plus or minus an optional offset. The offset can be either a hexadecimal or decimal value. Address resolution for the name can be specified by using the AP Assembler character set notation: rr(name)±offset where:

   rr = BA, HA, WA, or DA

   Word resolution is assumed by default. Note that both BA(ALPHA) + 3 and BA(ALPHA + 3) are legal. If the name specified has not been defined somewhere in the task, it will be flagged as undefined.

3. An EBCDIC constant (limited to four characters contained in quotes). Trailing blanks will be provided.

   Examples: 'ABC', 'A'.

4. A symbolic instruction. The mnemonic field of the instruction must be an EBCDIC operation code. The register and index fields can be any location in the proper range. The address field can be any location in the proper range.

The following commands are provided by Debug:

A The assign command is similar in function to the STDLB cal; it changes the assignment of a Debug oplabel on a job basis.

The form of the A command is

```
A oplabel [,area] ,name
```

where

oplabel is a Debug oplabel.

area is a RAD area if the label is being assigned to a RAD file.

name specifies a physical device name, a numeric zero, the name of a RAD file (if area is specified), or another oplabel.

C The form of the control segments command is

```
{CA}
{CD} seg number
{CE}
```

where

CA activates the desired segment.

CD deactivates the desired segment.

CE erases the desired segment.

seg number if the unique number of any valid segment in the task. A segment must be active before any patching in a segment can be done.

M The modify command replaces one set of locations by another set.

The form of the M command is

```
M loc/word0[/word1/. . ./wordn]
```

where

loc is the first memory location to be modified.

word$_i$ is the content to be inserted in the designated location. The i th word is inserted into loc + i.

I The insert command designates insertion of one or more instructions logically before (IB), after (IA), or replacing (IR) the instruction at the designated location.

The form of the I command is

```
{IB}
{IA} loc/word0[/word1. . ./wordn]
{IR}
```

where

IB designates Insert Before.

IA designates Insert After.

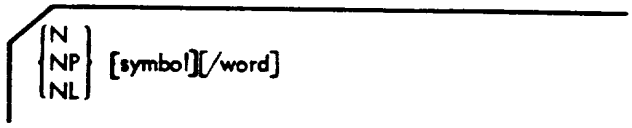
IR designates Insert Replace.

loc is the location that insertion is done with respect to is the i th change that is to be logically inserted.

The information for the insert is maintained so that an insertion may be removed if desired.

N The name command associates a word (value) with a symbol in the currently open task or evaluates an expression and returns a hexadecimal value. This command can also change the Debug prompt character for a task.

The form of the N command is



where

N specifies associate a symbol with a word value.

NP specifies the first character of the symbol as the new prompt character.

NL specifies evaluate the word.

symbol is any AP symbol.

word is any valid expression as described previously.

P The patch command causes Debug to read patch cards from the DP oplabel. The form of the P command is

where

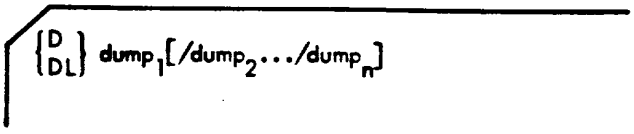
P causes a read from the DP device.

PE returns control to the DI device.

Debug will continue to read from the DP oplabel until it encounters a PE command or an EOD.

D The dump command allows selected areas of core to be output.

The form of the D command is



where each dump after the first one must be preceded by a "/".

$dump_n$ is of the form

where

loc_1 is the start location of the dump in the currently open task.

loc_2 is the last location to be dumped if the command is DL. If loc_2 is not specified, only loc_1 will be dumped.

size is the number of locations to be dumped in a D command. If not specified, the default size is 1.

format is one of the following:

- E for an EBCDIC format dump.
- D for a signed decimal format dump.
- B for a binary format dump.
- F for a hexadecimal format dump.

If no format character is present, the dump will be defaulted to hexadecimal.

oplabel is one of the defined Debug oplabels.

If no oplabel is present, the dumps will be output on the DL oplabel.

If the D command has no arguments, the next sequential cell (following the last dumped cell) will be dumped.

E The execution control command allows the user to stop and start a task in his job.

The form of the command is

where

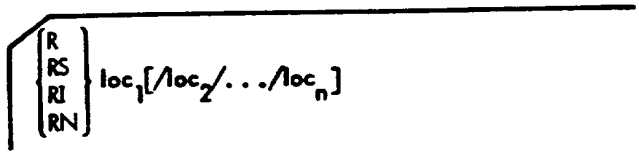
EH designates execution halt (stop).

This command causes a STOP CAL to be done to the task. This will inhibit the named task from further execution until a START is done.

EB designates execution begin (start). This will start a secondary task via a START CAL.

R The remove function will remove either snapshots or insertions and replace the original contents of the location(s).

ıe form of the R command is



$$\left\{\begin{matrix}R\\RS\\RI\\RN\end{matrix}\right\} loc_1[/loc_2/.../loc_n]$$

where

R specifies removal of any snapshots or insertions in loc_1 through loc_n.

RS specifies removal of all snapshots.

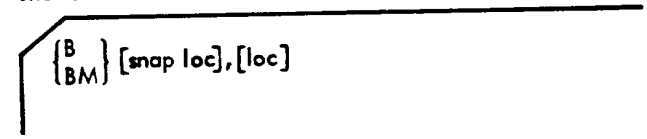
RI specifies removal of all insertions.

RN specifies removal of all names.

loc_i are the locations where removal is to be done. When the removal is done, the location is returned to its original state and the patch is removed from the Debug work space. No locations are allowed for either RS or RI commands. Inserts may not be placed over a location already accommodating an insert or snap.

B The branch command is used to start execution at the specified location in the currently open task and to move a snapshot if desired.

The form of the B command is



$$\left\{\begin{matrix}B\\BM\end{matrix}\right\} [snap\ loc],[loc]$$

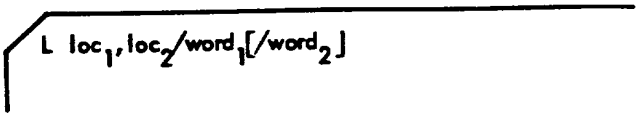
where

B designates a branch to loc. If no arguments are present, the task will continue from the current active snapshot or program start ($I). This option allows the user to continue from a snapshot or start at the entry of a newly initialized task. The snap loc parameter will be ignored if present.

BM designates branching to loc and moving the snapshot location. This command should be used only when control has been transferred via a snapshot and breakpoint. The BM will move the snapshot from its original loc (L) to the specified snap loc and execution will resume at the specified loc. If loc is not specified it defaults to L. If neither argument is specified, loc defaults to the original loc (L) and snap loc defaults to L + 1. This gives the user a stepping function which allows him to execute instructions line by line. This command with no arguments will not allow the user to step a branch instruction. If it is desired to continue execution from the current snapshot and move the snapshot the command "BM snap loc" can be used.

L The look command allows the user to search memory for a certain pattern.

The form of the L command is

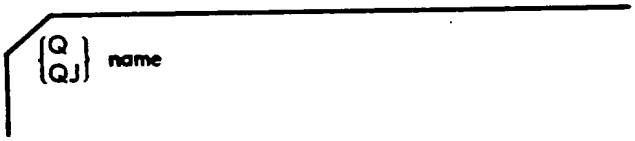


$$L\ loc_1,loc_2/word_1[/word_2]$$

The search will be conducted between loc_1 and loc_2 for a match to $word_1$. Each location will be masked (logical and) by $word_2$ before being compared $word_1$. If $word_2$ is not present the default will be X'FFFFFFFF'. All locations that match and their contents will be printed on the control console.

Q The quit function allows the user to terminate a single task in his job or to terminate the whole job.

The form of the Q command is



$$\left\{\begin{matrix}Q\\QJ\end{matrix}\right\} name$$

where

Q terminates the named task.

QJ terminates the entire job.

W The W command allows the user to compress debug workspace and make returned space available for extensions by name, insert, and snap commands. Should there be snaps or inserts associated with currently nonexistent or protected space, the squeezing will be incomplete and will return an error report.

DEBUG SNAPSHOT

There are two types of snapshots that can be specified. The snapshot and continue does the dumps if required, and continues execution after the snapshot. The snapshot and breakpoint executes the snapshot as above and then goes to the control console for input. Either type of snapshot can be specified in patch commands.

S The S command inserts (in the same manner as the command IB) a snapshot at the designated location, so that when

control passes through the location, the following transpires prior to executing the instruction that was at loc:

The following is output on the DL oplabel.

:SNP AT loc

where

: is the prompt character for the task where the snap is.

loc is the location specified in a previous S command.

The form of the S command is

{S / SC} loc /dump requests

where

S is a request to snapshot and transfer control (snapshot and breakpoint) to the control console for Debug input.

SC is a request to snapshot and resume execution.

loc is the location in the task where the snapshot is to be placed. The location cannot already be associated with an insert or snap.

Dump requests are the same format specified in the D (Dump) command. Each dump specification must be preceded by a "/". Only five such requests may be accommodated.

DEBUG ERROR MESSAGES

A message is output for all Debug errors. Debug will then request input from the Debug control console. The following message is output to the control console.

ERR N IN FIELD M

where

N is a hexadecimal value between 1 and FFF. All errors in the range FF0 to FFF are system errors and indicate a system malfunction. Other errors are user errors.

M is a decimal value that indicates the field where the error was detected. The first error found will abort the command. In reporting the field count, Debug considers the command identifier to be field 1. Each subsequent delimiter, including blank/+ - ().,' * terminates a field.

All errors are detected before the command has been processed except those indicated below by an *. The indicated commands are processed up to the point of the error. The command is then aborted and further input is requested.

The Debug error numbers and their meaning are itemized in Table 18.

Table 18. Debug Error Messages

Error Number	Description of Error
11	An empty field after a delimiter is invalid.
12	A. followed by a nonhexadecimal value is illegal.
13	The indicated field should be a number or a defined symbol.
14	Invalid address resolution has been used.
15	An illegal left parenthesis has been used.
16	A right parenthesis is followed by illegal syntax.
17	An illegal EBCDIC constant has been specified.
22	Illegal hexadecimal value has been specified.
27	Undefined symbol.
28	Illegal character.
31	Register and address field missing.
32	Register missing.
33	An * is illegal in the register field.
34	A register designation must be followed by a blank.
35	An empty address field is illegal.
36	The index field is missing.
37	An * is illegal in the index field.
38	The specified index is too large.

Table 18. Debug Error Messages (cont.)

Error Number	Description
39	The specified register value is illegal.
41	No location has been specified.
42	No words have been specified.
43	Too many modifies or inserts have been specified.
51	Illegal format specified.
52	Illegal size in a D command.
53	Illegal oplabel specified.
54	Illegal use of indirection.
61	More than two continuation records have been input.
62	The command identifier is undefined.
64	The input command is illegal from the Patch (DP) device. Delete snaps, or inserts to release space.
71	No more work space available.
81	The named location contains neither an insert nor a snapshot.
82	RS and R1 commands cannot have arguments specified.
83	R commands must have arguments specified.
89	Workspace already compact.
8A	Workspace has unidentified block.
8B	Snap or insert linkage inaccessible.
8C	Program link to snap or insert is inappropriate.
91	Invalid leading delimiter in the Dump command.
92	Invalid oplabel specified in a dump command.
A1	No segment specified.
A2	Illegal segment.
A3	Illegal completion status from the CAL.
B1	An insert to a register is invalid.
B2	An insert or snapshot is already present in the specified location.
C1	A location must be specified for a snapshot.
C2	A snapshot to a register is invalid.
C3	The delimiter after the location is invalid.
C4	A snapshot or insert is already present.
C5	Too many dump requests.
CE	The Message option is not allowed.
CF	The condition option is not allowed.
D1	The delimiter after the location is invalid in a BM command.
D2	No snapshot is active.
E1	Start, end, or word 1 are not present.
E2	The delimiter after start, end, or word 1 is illegal.
F1	Illegal character in the named symbol.
F2	A value must be specified in an N command.
F3	The delimiter after the symbol name is illegal.
F4	An NL command must have an expression.
101	Illegal character in Task Name.
102	STOP/START error.
111	Illegal character in Task Name.
112	SIGNAL Error.
121	Illegal character in Task Name.
122	Error in TERM call.
131	Illegal Oplabel.
132	Illegal Delimiter.
133	Illegal File Name.
134	Error in STDLB call.
1F1	Write Error (DL).
1F2	Read Error (DI).
1F3	Write Error.
FF1	A patch element type is too large.
FF2	A patch element size is too large.

9. CP-R MEDIA SERVICE

The CP-R MEDIA service provides the operator facilities for performing media conversions concurrent with the operation of foreground and background jobs, and foreground tasks the ability to submit permanent disk files for later printing on a line printer.

KEY-IN CONVERSIONS

The MEDIA key-in allows the operator to request conversions from

{file, tape, cards} to {file, tape, printer, punch}

or from file to keyboard printer.

Multiple key-in requests may be made. Each request will be queued for later processing.

FOREGROUND CONVERSIONS

For the foreground tasks, the MEDIA service call provides a delayed printing mechanism. The task can output printer destined information to a permanent disk file, and request the MEDIA task to copy the file to the printer. Multiple requests can be made by one or more foreground tasks causing the requests to be queued for later processing.

Requests from a foreground task's service call do not explicitly state the destination of the printed output. The destination is implicitly the MO operational label in the CP-R job. The default assignment of the oplabel is determined during System Generation (SYSGEN), but may be changed by the STDLB key-in.

MEDIA PROCESSING

MEDIA handles oplabels in a slightly nonstandard manner. Oplabels are converted to the device or file to which they are assigned. This is done by the key-in when the request is accepted and for a foreground service request when it is ready to be processed. This means that changing the assignment of an oplabel during a MEDIA conversion has no effect on that conversion. This is done to protect the integrity of a MEDIA output.

MEDIA performs its conversions as an independent task in the CP-R job concurrently with other tasks in the system.

Each conversion request, as it is entered into the queue, is assigned an identification number. This number is typed on the operator's console (OC) device for key-in requests, and is used to identify output for the request. Printer output will be identified by a break page containing the name of the requesting task (this will be "OPERATOR" for key-in requests) and the identification number.

Queued requests are processed in the following order:

1. Key-in requests;
2. Service requests, by priority.

When the Symbiont processor is in use for Background jobs, MEDIA will coordinate with the Symbionts for usage of devices common to both. Any requests to use a device already in use by the Symbionts causes that request to be delayed until the device is free. All devices used by either processor are freed at the end of each job or request to allow the other access. This insures that a job's or conversion's output will be contiguous and not intermixed with other output. Outputs may alternate between MEDIA and Symbionts.

MEDIA SERVICE CALL

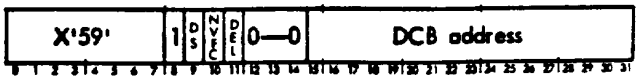
MEDIA The Media service call allows foreground tasks to submit permanent disk files to the CP-R Media task for later processing. The Media task, when processing the request, will copy the file to the MO operational label as defined in the CP-R job.

The call for this function has the following format:

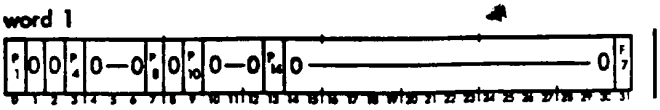
CAL1,7 address

where address points to word 0 of the FPT shown below

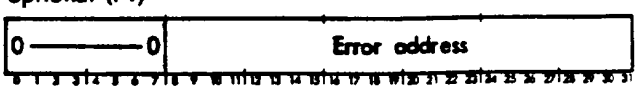
word 0

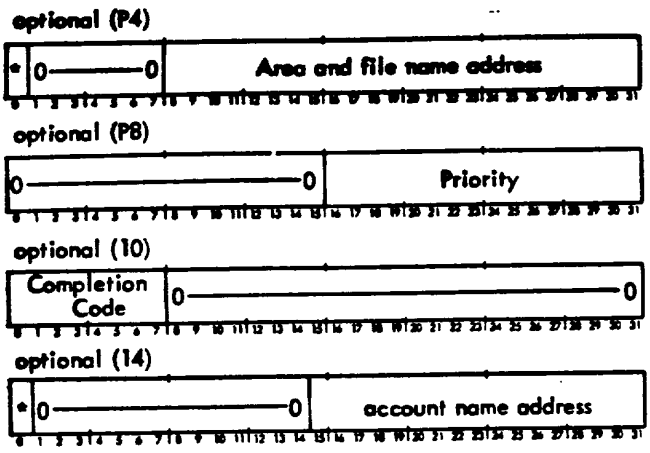


word 1



optional (P1)





where

Word 0

X'59' is the code that specifies the media call.

DS is the double space option that is meaningful only when NVFC = 1. If present (DS = 1) the printed output will be double spaced between each line of output. If DS is zero (0), then single spacing will occur.

NVFC is the vertical format control option. If present (NVFC = 1) then printer output will take place as specified by the DS parameter. If NVFC is zero (0), then the first byte of each line image is interpreted as a format control character.

DEL is the delete file after printing option. If present (DEL = 1) the RAD file specified to be printed will be deleted after the printing operation has successfully been completed. If DEL is zero (0), the RAD file will not be deleted.

DCB address if parameter P4 in word 1 is not present (= 0), then this field is interpreted as the address of a DCB that is assigned to the RAD file that is to be printed. If the DCB is not assigned to a RAD file, then an error condition is generated.

Word 1

P1 is the optional error address parameter presence indicator (0 means absent, 1 means present).

P4 is the parameter presence indicator for the optional file name and area address parameter (0 means absent; 1 means present).

P8 is the parameter presence indicator for the optional priority parameter (0 means absent; 1 means present).

P10 is the parameter presence indicator for the optional type completion code parameter (0 indicates no posting; 1 indicates the completion code is to be posted).

P14 is the account name address parameter presence indicator (0 means absent; 1 means present).

F7 is the abort override indicator.

Word Options

Error Address is the return address to process any detected errors.

Area and File Name Address is the address of a three word data block that contains the area and file name of the file to be listed. The first word contains the area name in bits 16-31 in EBCDIC. The second and third words contain the file name in EBCDIC.

Priority is the 16-bit priority which is to be assigned to the printing operation. This priority is used to control the selection of files to be printed. If not specified, priority defaults to the callers priority.

Completion Code is the word where the system will post the results of the service call.

account name address is the address of a two word data block that contains the account name in EBCDIC, extended to eight characters with trailing blanks. If the account name is all blanks or numeric zero, it is treated as unspecified.

a. If neither account name nor area name is specified, the calling task's account is used, and the area may be any public area. This provides the simplest specification, and the user need not be concerned with the possibility of name conflicts other than within his own account.

b. If the account name is specified, but the area name is not, the file may be in any public area. This provides for area-independent file specification.

c. If the area name is specified but the account name is not, the system account will be used. This case provides compatibility for code written before the addition of file account names.

Note: If a DCB address is supplied in lieu of the area and file name data block, the DCB will be closed upon successful processing of the MEDIA service call.

MEDIA KEY-INS

The MEDIA key-in has two functions; it allows the operator to submit requests for a Media conversion and it gives him control over the operation of the Media task.

The MEDIA key-in has the following general format:

```
MEDIA (input specification)[,(input option),
└─(input option),...],(output specification
└─[,(output option),(output option),...]
```

or

```
MEDIA (control specification[,(control
└─specification),...]
```

where

Input specification may take any of the following forms:

IN,device
IN,oplabel
FILE,fid

Device may be a card reader (CRxxx) or a magnetic tape (9Txxx or 7Txxx). Oplabel must be associated with one of these devices or a file.

Input option may be one of the following:

DEL	SFILE,n	UNLOAD
ALL	REW	

The DEL input option is applicable only to a file input specification and indicates the file is to be deleted after the copy is successfully completed.

The ALL option is applicable to tape and card input specifications and indicates that all files from the starting file to the end-of-file (volume) or two consecutive IEODs are to be copied.

The other options are applicable only to magnetic tape devices and specify processing as follows:

SFILE,n space forward from the starting (current) position "n" files before starting the copy.

REW rewind the input tape after the copy operation is completed.

UNLOAD rewind the input tape "off-line" after the copy operation is completed.

Output specifications may be of the form:

OUT,device
OUT,oplabel
FILE,fid

Device may be a tape (9Txxx or 7Txxx), a printer (LPxxx), or a card punch (CPxxx). Oplabel must be associated with one of these devices or a file. If a file is specified as input, then a keyboard printer may be specified as an output device (TYxxx).

Output option may be one of the following:

NVFC	SFILE,n	REW
SPACE,n	WEOF,n	UNLOAD
ADD		

The NVFC and SPACE,n options are applicable to printer destined files and have the following meanings:

NVFC the first byte of each record is not to be interpreted as a VFC byte; instead, it is to be printed as data. The default is the first byte is used as a VFC byte and the data starts in the second byte.

SPACE,n this has meaning only when NVFC is specified and indicates there are to be "n" blank lines between each output line, $1 \leq n \leq 15$.

The ADD option specifies that the output file(s) is either to be an extension to an existing disk file or to follow the last file currently on a magnetic tape.

The remaining options are applicable to magnetic tape and specify processing as follows:

SFILE,n space forward from the starting (current) position "n" files before starting the copy.

WEOF,n write "n" end-of-files after the copy. If WEOF is not specified, "n" is defaulted to 2.

REW rewind the output tape after the copy.

UNLOAD rewind the output tape "off-line" after the copy.

Options which do not have meaning for their associated input or output specifications are accepted and ignored without an error notification.

The control specification may be one of the following:

S
L
I
X

where

S causes an immediate suspension of the current copy operation.

L will inhibit the initiation of any new copy operations but will allow the current copy to continue.

I continue the current copy if stopped via an "S", remove the previous "L" request, or search for a new copy to start.

X terminate the current copy.

Multiple control options may be selected in any order in one MEDIA key-in. However, if an "S" and/or "L" are selected in combination with an "I", processing is as follows: The "S" and/or "L" action requests are set (combined with the current status of the MEDIA task); then, if "S" is set, "I" resets it only; if "S" is not set and "L" is, "I" will reset "L".

CONVENTIONS

Input files are delimited by a IEOD from a card reader, an end-of-file from a tape, or the end-of-file from a disk. When multiple files are being copied in response to the ALL option, double IEODs or end-of-files are required to terminate the input. Tapes being processed in response to ADD or ALL options must be terminated by double end-of-files.

Whenever one of the devices involved in a MEDIA copy requires operator intervention, a message indicating this condition will be output on the control console and the MEDIA task will enter the suspended operation state by simulating an "S" control key-in. This message may be:

!!MEDIA: MOUNT TAPE(S) FOR dddd

to indicate the tape or tapes for conversion request "dddd" are to be mounted; or

!!yyndd MANUAL

to indicate that the device is in the manual state; or

!!yyndd WRT PROT

to indicate the output device is write-protected.

The copy is continued after the condition is corrected by entering the MEDIA "I" control key-in or aborted if the condition cannot be corrected by entering both the "X" and "I" control key-in. Note that if the "X" and "I" are not entered in one key-in, the "X" should be entered first to prevent a reoccurrence of the condition.

Multiple positioning options may be requested for magnetic tapes that might cause conflicts. If both SFILE and ADD are specified for an output tape, the SFILE positioning is performed first and then the ADD positioning. Whenever both REW and UNLOAD are specified, only the UNLOAD is performed.

Specifying an SFILE,n count greater than the number of files remaining on a tape will cause the output file(s) to be appended as if an ADD had been specified.

MESSAGES

The acceptance of a valid key-in request and its entry into the MEDIA task's queue is indicated by the message,

!!QUEUED AS ID NUM dddd

A key-in request that is rejected for any reason is indicated by the message,

!!KEY-ERR

and can be corrected by re-entering the correct key-in.

The message,

!!MEDIA: MOUNT TAPE(S) FOR dddd

is typed when the MEDIA task has begun processing request "dddd" and will wait until the operator indicates he has mounted the correct tape(s) on the correct drives.

The message,

!!MEDIA: ABORTED REQ dddd:yyyy[, xx]

is output whenever a copy request cannot be completed normally and is aborted. The "dddd" gives the request's ID number, "yyyy" gives the reason for the abort, and the optional "xx" gives the error code returned in R10 by the service call reporting the fatal error. The meaning of the various "yyyy" codes may be one of the following:

yyyy	Meaning
BUFS	unable to get memory for buffers.
DEV	unrecoverable error during the copy.†
NOMO	the MO operational label was not defined by SYSGEN time.
OPER	the operator keyed in the "X" control command.
OPNI	unable to open the input file.†
OPNO	unable to open the output file.†
PREP	fatal error during SFILE or ADD pre-copy positioning.
SPEC	an error in the request specification that was not detected when request was accepted.

†These errors will have the optional "xx" displayed.

10. OVERLAY LOADER

OVERVIEW

The Overlay Loader is a two-pass processor that creates programs in overlay form. Modules in standard object language format are converted to overlays in absolute core image form in accordance with the Loader control commands. The Loader creates programs for execution in either foreground or background, prepares standard processors for execution under the Job Control Processor, and creates Public Libraries.

The Overlay Loader permits the user to assemble, load to background or foreground and execute programs with minimal control information. The default cases documented in this section for each control command will handle most normal situations.

The control command structure permits the user to tailor the loading procedure for a wide variety of situations, and the control commands add control and flexibility by overriding default cases and adding options.

The size of the program that can be loaded is a function of the size of the symbol table and available core storage at load time, rather than the amount of core memory that the program occupies at execution time. Therefore, the Overlay Loader may load user programs equal in size to the maximum available area in core at execution time, even though this area is not available at load time.

The loading of mixed media is allowed, and all library loading will be from library files on disk. There are no constraints on the ordering of modules within a library.

FUNCTIONAL FLOW

The options specified on the !OLOAD control command are scanned and those not specified are assigned their default values. A :ROOT or :SEG control command is scanned to determine the source of the binary object modules from which the segment will be created, and to define its linkage.

The Loader makes the first pass over the binary object modules, allocating the segment's labelled COMMON blocks (dummy sections) and control sections. It concurrently builds a symbol table of DEFs and unsatisfied REFs. Object modules input from non-disk devices are saved on a temporary disk file (X1).

After the last object module for a segment has been input, the libraries are searched. Pointers to the selected library object modules are saved and their DEFs and REFs are added to the symbol table. At the end of a path, segment symbol tables are written on temporary disk files. At this point all the Loader control commands except :ASSIGN and :LMHDR have been input.

During the second pass, each segment's binary object modules and selected library modules are loaded. The absolute core image of each segment is created and written on the program file. Part two of the ROOT (the DCBTAB, OVLOAD table, the temp stacks and any DCBs created by the Loader) is built at the end of the second pass. If a MAP has been specified, it is output. If an output file used by the Loader overflows, an attempt is made to output all possible MAP information. The Loader returns to the system by calling either the EXIT or ABORT function.

LIMITATIONS

There are certain limitations in the use of the Overlay Loader due to the total system considerations or because the efficiency of the Loader could otherwise be degraded.

1. No discontinuous segments will be output by the Overlay Loader. The SEGLOAD and ACTIVATE service functions read only contiguous core images. Since each discontinuity would result in at least one additional disk access, considerable degradation of the run-load process for the foreground would result.

2. The contents of reserve areas within a program will not be predictable at execution time unless initialized in some manner (e.g., by DATA statements). Labeled COMMON will be unpredictable unless initialized by a DATA statement. Blank COMMON is not written to disk and is not loaded as part of the program.

3. Allocation of program and labeled COMMON within a program area is generally determined by the Loader.

4. Only relocatable modules or those containing absolute origins falling within the limits of the segment currently being loaded will be allowed.

5. No implicit loading of segments will take place at execution time. Only explicit calls for Memory Management services will read in overlay segments. Thus, the overlay structure must be accurately defined at load time to coincide with explicit calls in the user's program. The user may, however, specify a set of segments to be loaded initially.

OVERLAY LOAD MODULES

An overlay load module is a file containing the absolute core images of the segments of a program, along with a body of information to control the preparation of the program for execution. The absolute core image of a segment

in the load module is a single continuous byte string, beginning at a sector boundary. Its length is such that it includes all areas of the segment generated from ROMs and all locations altered by :MODIFY commands. It will stop short of any areas at the end of a segment that result from a :RES command if these areas do not have specific initial contents (i.e., if there are no :MODIFY commands affecting them).

OVERLAY STRUCTURES

An overlay program is generally composed of two root segments, a Blank COMMON segment and several overlay segments; however, it can consist of as little as a single root with no other segments. The root and common segments are always resident during execution of the program. Residence of overlay segments depends on the explicit use of Memory Management calls by the program.

Each segment is created from one or more binary object modules and associated library routines. The segments are assigned arbitrary identification numbers (except for the root, which is always segment 0) that must be unique within the overlay program. Segment numbers are used by the Overlay Loader and the Memory Management fuctions.

The overlay structure of the program is communicated to the Loader with the :ROOT, :SEG, and :COMMON control commands. This structure defines all sets of segments that logically may be in memory together. Any such set of segments will be called a "path" of the program. The overlay structure of a program is commonly represented by a tree diagram, where the root (both segments) is the origin branch and each overlay segment is a branch. The diagram is drawn so that the paths of the program are the same as the sets of segments defined by the paths through the diagram.

The overlay program example given in Figure 13 consists of a root (segment 0) and overlay segments 1 through 15. The segments (horizontal lines) are numbered in the order in which they were built by the Loader. There are nine paths:

1. 0, 1, 2
2. 0, 1, 3
3. 0, 4
4. 0, 5, 6, 7
5. 0, 5, 6, 8
6. 0, 5, 9, 10, 11, 12
7. 0, 5, 9, 10, 11, 13
8. 0, 5, 9, 10, 14
9. 0, 5, 9, 15

Normally, the allocation of memory to the segments of the program would correspond to the structure diagram above. Each segment would begin in memory at the end of the segment at which it is based in the diagram. This allocation is provided by the Overlay Loader when segment origins are not specified explicitly.

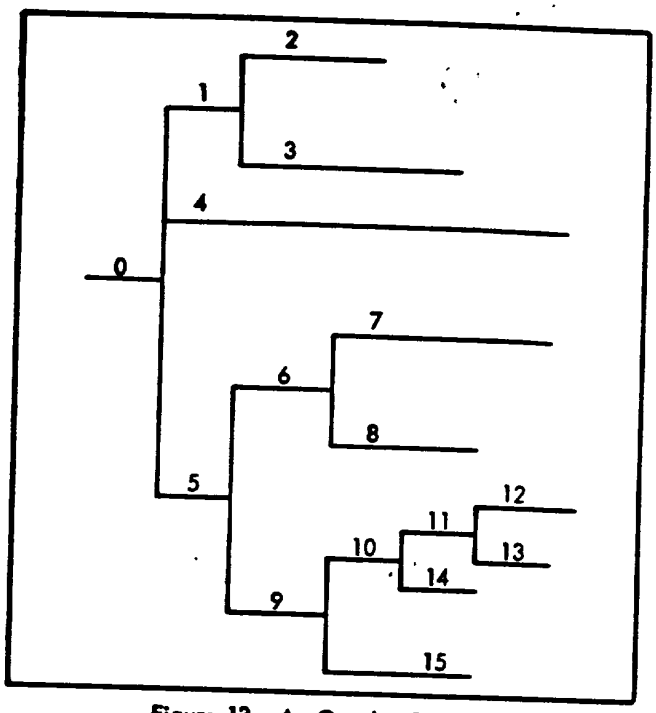


Figure 13. An Overlay Program

OVERLAY RESTRICTIONS

Communication between segments by external DEF/REF linkages is permitted with the following restrictions:

1. The Loader will satisfy a DEF/REF linkage only within a path.

2. A segment in one path cannot reference a segment in another path. For example, segment 2 must not reference any of segments 3-15.

3. The user must ensure that any segments that intercommunicate are in core. For example, if segment 5 references segments 6 and 8, then segments 6 and 8 must have been explicitly loaded. If segment 8 references segments 5 or 6, these segments must have been explicitly loaded since the loading of segment 8 does not cause the implicit loading of segments 5 and 6.

4. Identical definitions cannot be used in segments that are in the same path. For example, segments 5 and 13 cannot have identical definitions because they are both in path (0, 5, 9, 10, 11, 13).

5. Identical definitions and references may be used in segments of different paths that do not involve a common segment. For example, if segments 7 and 15 reference identical definitions in segments 6 and 9, the Loader will link the reference in 7 with the definition in 6 and the reference in 15 with the definition in 9.

6. Identical references in segments of different paths may be made to a definition in a segment common to both paths. For example, segments 6 and 9 can each reference a definition in segment 5 because 5 is a common segment in the two paths (0, 5, 6, 7) and (0, 5, 9, 10, 14).

7. A segment that is common to two paths cannot reference identical definitions in the different paths. For example, segment 10 cannot reference identical definitions in segment 12 and 13, even though segments 12 and 13 are in different paths.

Where possible, the Loader will warn the user about errors in overlay structure and segment communication; however, it is the user's responsibility to attempt a reasonable, workable overlay construction.

OVERLAY CONTROL COMMANDS

The prime Overlay Loader command, !OLOAD, is read by the Job Control Processor (JCP) and causes the Overlay Loader processor to be read into the background and executed. All Loader subcommands are identified by a leading colon (e.g., :SEG). They are read from M:C and logged onto M:LL. Blank cards are passed over without comment. When a CP-R control command is encountered, the Loader completes the load process and exits to the JCP.

Note that !EOD must occur only as a terminator for object module input; its use is illegal for terminating the Loader control command stack.

SYNTAX

The syntax for Overlay Loader control commands is identical to that defined for job-control commands (except for MODIFY).

ORDER OF CONTROL COMMANDS

The control command stack is divided into major divisions or substacks, which must occur in the following order:

```
                                    !OLOAD
   :ROOT
:SEG      or  :COMMON
:
:
:SEG      or  :COMMON        or  { :PUBLIB
:ASSIGN   or  :LMHDR              { :SEG
```

The :LIB, :INCLUDE, :RES, and :MODIFY commands may occur in any :ROOT or :SEG substack. :EXCLUDE and :LCOMMON may occur in a :ROOT substack, or in a :SEG substack unless it follows :PUBLIB. The :ASSIGN and :LMHDR commands must follow all other commands. A :COMMON command may occur (only once in the command stack) wherever a :SEG substack is legal, except following a :PUBLIB substack. Only a single :SEG substack may follow a :PUBLIB substack.

A ROOT or SEG substack has the following order:

```
:ROOT or :SEG
:INCLUDE
:EXCLUDE†
:LIB                    } These commands may occur
:LCOMMON†               } in any order.
:RES

Binary Object Module1   }
:                       } Binary object modules are in-
:                       } cluded at this point in the
Binary Object Modulen   } substack only if the input
                        } device specified on the
:MODIFY                 } preceding :ROOT or :SEG
:                       } command is the same as the
:                       } "C" device.
:MODIFY
```

The PUBLIB substack has the following order:

```
:PUBLIB
:INCLUDE
:EXCLUDE                } These commands may occur
:LIB                    } in any order.

Binary Object Module1   }
:                       } Binary object modules are
:                       } included in the substack
Binary Object Modulen   } only if the input device
                        } specified on the PUBLIB
:MODIFY                 } command is the same as
:                       } the "C" device.
:
:MODIFY
```

!OLOAD The !OLOAD control command signifies that the Overlay Loader Processor is to be executed in the background area. Any error on the !OLOAD control command causes the Loader to abort. Recovery consists of correcting the error and reloading the entire job.

If an !OLOAD control command is continued to another card, the continuation command must have a colon (:) in column one instead of an exclamation (!) character.

---

†These commands are not permitted in a :SEG substack if it follows a :PUBLIB substack.

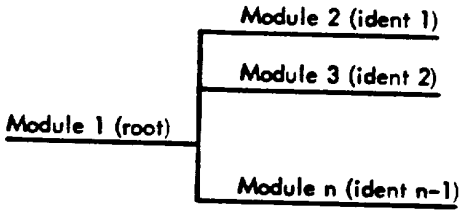
The form of the command is

```
!OLOAD [(option1)(option2)...(optionn)]
```

where the options are

GO specifies that the Loader is to input all object modules from the GO file and form a root. The only other control commands recognized in this mode are :RES, :INCLUDE, :MODIFY, and :ASSIGN. All other commands are considered illegal. This is considered the default input option unless the first command following the OLOAD command is either :ROOT or :PUBLIB, which provide their own input specifications.

GO,LINKS specifies that the Loader is to form a link type overlay structure from GO in the following manner: module 1 is identified as the root (segment 0); module 2 is identified as segment 1 and is linked to the root, ...; module n is identified as segment n-1 and is linked to the root.



Libraries are searched at the end of each segment. Only :MODIFY and :ASSIGN commands are honored. The user must make explicit calls to load segments 1, 2, ...n. No implicit calls are built by the Loader.

PUBLIB,name$_1$[,name$_2$,name$_3$,name$_4$] specifies that the named Public Libraries are to be resident when the loaded program executes, and that the Loader is to establish the appropriate linkage. Name$_i$ is the file name of a Public Library in the Foreground Programs area of the disk. The PUBLIB keyword may not be used when a Public Library is being created. (i.e., one Public Library cannot reference another Public Library.) The class of the task(s) constituted by the program to be loaded must match that of the libraries referenced (i.e., primary or secondary).

LIB[,USER,SYSTEM] specify the Libraries to be searched following each segment. The order of the keywords USER,SYSTEM defines the order of the search. If the USER or SYSTEM keywords are omitted, and only the LIB keyword is specified, the Library search is suppressed. If the LIB option is omitted, the Loader searches the System Library after each segment.

Note: The :LIB control command overrides this option for the segment in which it appears (see below).

{FORE / BACK}[[,REL],fwa,lwa] specifies whether the program being loaded is to execute in foreground or background. If the option is omitted, the program will execute in the background. The "fwa" and "lwa" parameters are hexadecimal values denoting the first word address (on a doubleword boundary) and last word address of the area within which the program will execute. If REL appears, "fwa" and "lwa" denote word displacements of the program limits from the default first word address described below.

For primary programs, the default fwa and lwa values are the FWA and LWA of the first Foreground Private partition (defined at SYSGEN).

For secondary programs, the default limits are the limits of task virtual memory. Note that "lwa" is an indicator of upper limit. If the program exceeds this limit, the user is warned but loading is not inhibited (except when a Public Library is being created). If the program loads in less space, the shorter area will be output in the header.

TASKS,value has no effect but is accepted so as to retain compatibility with RBM-system loader command decks.

TEMP[,value$_1$[,value$_2$]] specifies the decimal number of words to be allocated for the user temp stack (value$_1$) and the CP-R temp stack (value$_2$). These stacks will be located in part two of the root. The user temp stack size defaults to 200. The CP-R temp stack size defaults to 200 (225 if the symbionts are assembled into the system). (Public Libraries do not have temp stacks. Therefore, this option may not be specified when a Public Library is being created.) The "value" parameter is a decimal number.

FILE,fid specifies the CP-R file identifier of the output file to which the loaded program is to be written (hereafter referred to as the Program File). The default assignment of the program file is OV in the Background Temp area. If the Background Temp area (BT) is specified, the file name must be OV. When a Public Library is being created, the Foreground Programs area (FP) must be specified.

MAP[,{PROGRAM / ALL}][,{ALPHA / ADDRESS}] specifies that a map of the program is to be output to M:LO. If no keyword follows MAP, a short map consisting of information about program allocation and overlays is output. If the PROGRAM keyword is given, external definitions and control section designations for each segment are listed without library definitions. For the ALL keyword, both program and library

definitions are listed. In default, no MAP is output. (Diagnostics and unsatisfied references are still listed on M:LL.)

If a PROGRAM or ALL map is specified, the user may further use the keyword ALPHA or ADDRESS to request that symbol tables be sorted alphabetically or numerically. If neither sort is requested, symbols are listed in the order of encounter.

BOUND,value sets the loading (and execution) origin for each object module to the next higher multiple of the bound value (e.g., if BOUND = 100, then an origin would change from 3EF to 400). The "value" parameter must be a hexadecimal number less than or equal to 1000 and a power of 2. Suggested values are 10, 100, or 1000. The BOUND does not apply to Library modules. If BOUND is not specified, the Loader begins each module on a doubleword boundary.

UDCB,value specifies the number of unnamed DCBs to be allocated by the Loader. (See "Loader-Generated Items" for details.) The "value" parameter is a decimal number.

STEP specifies a "WAIT" after loading each module from paper tape. Used in CP-R ATTEND mode.

{PRI[MARY] / SEC[ONDARY]} specifies task type, and affects segment definitions. Segment origins are forced to page boundaries for secondary-task segments, but not for primary-task segments. If not specified, SECONDARY is assumed. For background programs, only SECONDARY is valid. Either option may be specified for Public Library creation.

SMM specifies the Simplified Memory Management option. This option provides for compatibility of programs designed for the memory organization of background in Xerox RBM (unmapped) systems, and is described in detail in the chapter on CP-R Memory Management. (Note that certain of the standard CP-R processors, currently including the service processors, must be loaded with SMM if they are loaded by the Overlay Loader.)

Example: Form Root From GO File Modules

```
!OLOAD GO,(TEMP,300),(MAP,PROG),(UDCB,3)
```

This example specifies that the Loader is to form the root from object modules located on the GO file, allocate 300 words for the user Temp Stack, output a PROGRAM map, and allocate three unnamed DCBs.

:ROOT The ROOT control command is used to specify the object modules from which the root segment is to be created. The ROOT command must precede all SEG commands.

The form of the command is

```
:ROOT [(ENTRY,def),(EXLOC[,REL],hexaddr)]
      [('input option1),...('input optionn)]
```

where

ENTRY,def specifies the location at which execution will commence after the root is loaded at execution time. The def parameter must be an external definition (1-8 EBCDIC characters), not necessarily in the root segment. This entry point overrides all subsequent entry addresses encountered in loading. The default entry address is the last transfer address encountered in the object modules of the root.

EXLOC[,REL],hexaddr specifies (subject to bounding) the origin of root part two. The value hexaddr is a hexadecimal value. If REL appears, hexaddr is treated as a word address relative to the default FWA for the program (see option FORE for the OLOAD command). If REL does not appear, hexaddr is an absolute word address. If the EXLOC option is not used, the default address for root part two will be the first bounded address past the last address used by any other segment.

Input options are of the form

```
{DEVICE,type[,PACK]  }
{FILE,fid            }[,value]
{OPLB,label          }
{NONE                }
```

where

DEVICE,type specifies the input device in the format yyndd.

where

yy is a device type code.

n is the IOP to which the device is connected.

dd is the hardware device number of the device (e.g., CRA03, 9TA81).

PACK specifies that the input is from 7-track magnetic tape with packed binary format.

FILE,fid specifies the CP-R file identifier for an input file. If the Background Temp area (BT) is specified, the file name must be GO. Note that a file may be used as input to more than one segment (in different paths). A named file is rewound each time it is specified; the GO file is not.

OPLB,label specifies the operational label from which the object module(s) will be input. The "label" parameter must be a 2-character standard system operational label.

value either a decimal number ($1 \leq value \leq 8191$) that specifies the number of object modules to input from the specified device/file; or the text string, EOD, which means to input from the specified device/file until an IEOD is encountered. If value is omitted, one object module will be input from the specified device/file.

NONE indicates that no ROMS are explicitly requested. The segment will be composed of any areas allocated in response to :RES or :LCOMMON commands, and any library ROMs necessary to satisfy references in :INCLUDE command and the ENTRY option. If only :RES areas are included, a "defined" segment is produced.

If there are no input options on the ROOT control command, one object module will be input from the GO file. Note that the order of the subfields determines the order in which the object modules are loaded.

Example: Form Root From Input File

```
:ROOT (FILE,D1,BETA,3),(ENTRY,START)
```

This example specifies that the root is to be formed from the three object modules in a file called BETA located in the D1 disk area. After loading, execution is to commence at the location defined by the external definition START.

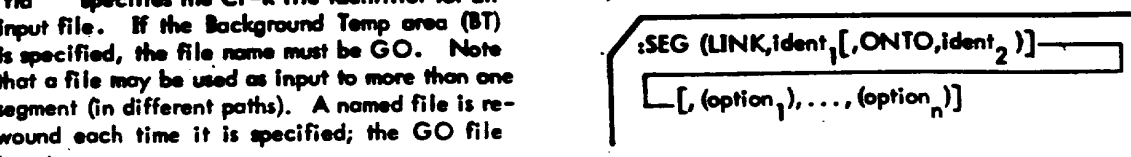
Example: Form Root From Library Modules

```
:ROOT NONE,(ENTRY,LIBROOT)
```

This example specifies that the root is to be formed from only the library modules necessary to satisfy the reference LIBROOT in the ENTRY option.

:SEG The SEG control command is used to define a segment's overlay linkage and to specify the object modules from which the segment is to be created.

The form of the command is



where

LINK,ident$_1$ specifies the identification number of the segment being loaded. The ident$_1$ must be specified and must be the same number used within the overlay program to reference the segment at execution time via memory-management services. The "ident$_1$" parameter must be a decimal number between 1 and 32,767.

ONTO,ident$_2$ specifies the identification number of the segment (which must have been previously loaded when this control command is interpreted) to which this segment is linked as an overlay. If ident$_2$ is absent, ident$_1$ is linked onto the root. The "ident$_2$" parameter must be a decimal number between 0 and 32,767 (0 denotes the root).

option$_i$ is either an input option (as described for the :ROOT command) or one of the following options.

EXLOC[,REL],address specifies an optional execution address for loading of this segment. The "address" parameter is a hexadecimal value. If REL appears, the "address" value is interpreted as a word displacement relative to the default lower limit for the program (see OLOAD command option FORE/BACK). If REL is omitted, the value is treated as an absolute word address. The absolute address obtained in either case will be bounded by either the specified or default BOUND. If the EXLOC option is omitted, the segment will be located at the first bounded address following the segment that this segment is linked to.

ENTRY,def specifies an entry point for the segment. The "def" parameter must be an external definition of the program; not necessarily in the segment itself. The value of the "def" overrides any transfer addresses encountered in loading of the segment. The default entry address is the last transfer address encountered in loading nonlibrary ROMs.

SHARE,{SYSTEM / JOB}[,PRELOAD] specifies the group of tasks that can share the segment being linked: either all tasks in the system (SYSTEM) or only the other tasks in the same job (JOB). If omitted, no segment sharing will be allowed.

The presence of the PRELOAD option causes the Overlay Loader to omit the load module image of the segment and set a flag which indicates that the segment must have been previously defined when the load module is initiated. This can save considerable file space when a large segment is shared by several load modules.

For primary tasks, CP-R memory management does not record segment activity, so the SHARE option acts somewhat differently. Primary task shared segments are not considered in determining the program bounds, so primary programs may share a data area by specifying a shared segment at the same address. (However, no program may be loaded with any segment preceding Root part 1.) For primary tasks, there is no distinction between SYSTEM and JOB shareability. PRELOAD causes omission of the load module image of the segment but the user, not the system, must insure that the segment is actually loaded.

ACCESS, {NO / R / RX / ALL} specifies the memory-access control to be assigned to the segment. If omitted, ALL is assumed. For primary-task (unmapped) segments, this option is accepted but will have no execution-time effect. The meaning of the sub-options are

- NO - No access permitted.
- R - Read only.
- RX - Read and execute only.
- ALL - Read, write and execute.

FIX specifies that the segment must be assigned real memory such that its virtual and real addresses correspond 1:1. If omitted, real memory is assigned wherever it is available. A segment defined with the FIX option must reside in a Foreground-Preferred partition of real memory.

ILOAD specifies that the segment is to be activated with the root segment when the task is initiated. (After initial loading, the segment is subject to the normal memory-residence controls allowed by its other characteristics.) If omitted, the segment must be explicitly activated. This option is invalid for primary-task segments.

The input options are the same as for the ROOT control commands. If there are no input options on the SEG control command, a single object module from the GO file will be input.

The ROOT and SEG control commands must be input in an order determined by the overlay structure of the program. The segments in the example given in Figure 13 have been numbered to illustrate this order. Basically, segments are input one path at a time, with the restriction that segments common to more than one path are input only once.

Example 1.

The following control commands define the overlay structure of Figure 14. This example specifies that one object module for each segment will be input from the GO file.

```
:ROOT
:SEG (LINK, 1, ONTO, 0)
:SEG (LINK, 2, ONTO, 1)
:SEG (LINK, 3, ONTO, 1)
:SEG (LINK, 4, ONTO, 0)
:SEG (LINK, 5, ONTO, 0)
:SEG (LINK, 6, ONTO, 5)
:SEG (LINK, 7, ONTO, 6)
:SEG (LINK, 8, ONTO, 6)
:SEG (LINK, 9, ONTO, 5)
:SEG (LINK, 10, ONTO, 9)
:SEG (LINK, 11, ONTO, 10)
:SEG (LINK, 12, ONTO, 11)
:SEG (LINK, 13, ONTO, 11)
:SEG (LINK, 14, ONTO, 10)
:SEG (LINK, 15, ONTO, 9)
```

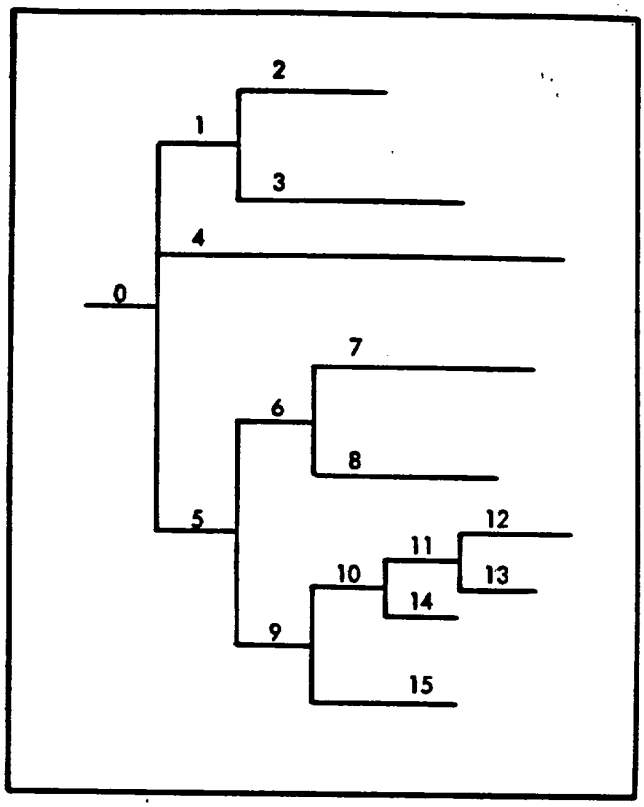


Figure 14. Overlay Example

Example 2.

The following control commands define the overlay structure illustrated in Figure 15. This example specifies that one object module for each segment will be input from the GO file.

```
:ROOT
:SEG (LINK, 10, ONTO, 0)
:SEG (LINK, 5, ONTO, 10)
:SEG (LINK, 25, ONTO, 10)
:SEG (LINK, 103, ONTO, 0)
```

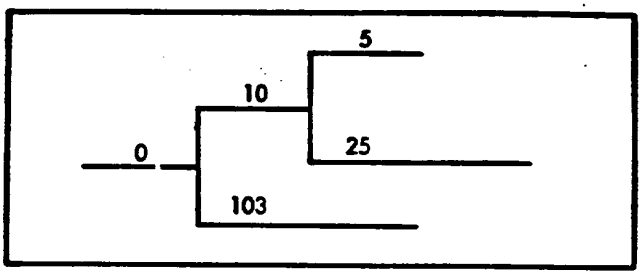


Figure 15. Object Module from GO File

BINARY OBJECT MODULES

The Loader inputs binary object modules from mixed media according to the input files and devices specified on the ROOT, SEG and PUBLIB commands. Files may be blocked or unblocked. Non-disk input is written to a temporary disk file for Pass 2. Binary modules are read sequentially from each disk file. Each disk file, with the exception of GO, is rewound each time that it is named as input on a control command. Therefore, multiple inputs from a file (other than GO) result in the file being reread from the beginning.

In this example,

```
:SEG (LINK,204,ONTO,0),(FILE,FP,PROG1,2);
:(FILE,BT,GO,4)
.
.
.
:SEG (LINK,205,ONTO,0),(FILE,FP,PROG1,5);
:(FILE,BT,GO,2)
```

the first access to the PROG1 file (in SEG 204) would result in the first two modules being loaded from the file. The second access (in SEG 205) would result in the first five modules of the file being loaded (not modules 3-7). The GO file is read contiguously throughout a pass, no matter how many accesses are made. In segment 204, the first four object modules from GO would be loaded. In segment 205, the next two modules (5 and 6) from GO would be loaded.

:LIB The LIB control command specifies the library search for one segment only (i.e., the segment identified by the preceding ROOT or SEG command). It overrides the library search specified by the LIB option on the OLOAD control command.

The form of the command is

```
:LIB [(USER,SYSTEM)]
```

where option keywords USER and SYSTEM are used to denote the libraries and order of search; e.g., :LIB (USER) would cause only the USER library to be searched for that segment. If neither USER nor SYSTEM is specified, library search (except for Public Library) is suppressed for that segment.

Example:

In Figure 16, assume an overlay program with four ROMs the GO file; with segments 0, 1, and 2 coded in assembl language, and segment 3 coded in FORTRAN.

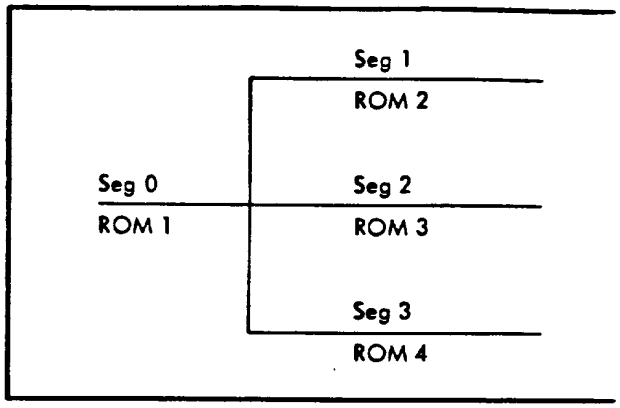


Figure 16. :LIB Command Usage

To speed up the load process, the :LIB command in the stack of commands given below would be used to specify a search of the System Library in segment 3, and the no-library search option would be specified on the !OLO command.

```
!OLOAD (MAP,ALL),LIB
:ROOT
:SEG (LINK,1)
:SEG (LINK,2)
:SEG (LINK,3)
:LIB SYSTEM
```

:INCLUDE The INCLUDE control command allows routines to be loaded from libraries when no reference to the routine has been made in any module of the segment.

The form of the command is

```
:INCLUDE (def1 [,def2,...,defn])
```

where def_i is the EBCDIC symbol of a definition contained in the library routine to be loaded. The symbol may be one to eight EBCDIC characters. The def_i must be available in a library specified in a preceding :LIB command or the LIB option on the !OLOAD command; any unfound def results in an error diagnostic. INCLUDE also cancels a prior EXCLUDE for a given symbol.

Example: Load Two Routines From Library

```
:INCLUDE (9SETUP,7SET)
```

In this example, the routines 9SETUP and 7SET are to be included in the load from a library previously specified in the search criteria.

:EXCLUDE The EXCLUDE control command inhibits library search and linkage for the named definition(s) even though an external reference occurs in a module of the segment.

The form of the command is

```
:EXCLUDE (def1[,def2,...,defn])
```

where def_i is the EBCDIC symbol of an external reference contained in a module of the segment. However, def_i must not occur as an external definition in a lower level segment of the path. The symbol may be one to eight EBCDIC characters. Note that EXCLUDE also inhibits linkage with the specified Public Library for the given symbols.

Example: Exclude Search for Named Routine

In this example, the tree structure illustrated in Figure 17 shows a routine called SIN in segment 1 that has the same name as a library routine, and is referenced in an earlier segment (Seg 0). The command

```
:EXCLUDE (SIN)
```

inhibits library search and linkage for the named routine only so that the TAN routine would be included in the root but the SIN routine would not.

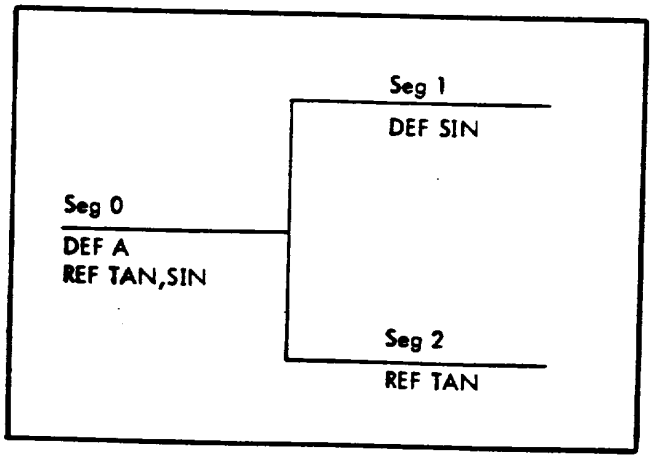


Figure 17. :EXCLUDE Command Usage

:COMMON The COMMON control command specifies that the Loader is to set the base of Blank COMMON at the end of the segment identified by the preceding ROOT or SEG control command, or at a location indicated in the :COMMON control command. It also allows specification of the size of blank COMMON. If this control command is not included, Blank COMMON is set at the end of the longest path. Only one COMMON control command may be used in a control command stack.

The form of the command is

```
:COMMON [(option1),...,(optionn)]
```

where $option_i$ is one of the following:

EXLOC,hexloc specifies the base of Blank COMMON to be the address "hexloc", which is a hexadecimal number. If the option is omitted, the base of Blank COMMON will be set at the end of the segment (or root) defined by the previous SEG (or ROOT) command.

SIZE,hexwords specifies the word size of Blank COMMON to be "hexwords", which is a hexadecimal number. This size will override any size declared in a later ROM, and any smaller size in an earlier ROM, but will be ignored if a larger Blank COMMON size has already been declared. If the option is omitted, Blank COMMON size is set to the largest size declared in a previously loaded ROM. Whether the size is set by the option or by an earlier declaration, if a later declaration requires a larger size, a warning is issued and the larger size is not allocated.

SHARE, {SYSTEM / JOB} specifies the range of shareability of the Blank COMMON segment, as described for the :SEG command, for both primary and secondary load modules. PRELOAD is not allowed however, since Blank COMMON is never included in a load module image.

FIX specifies 1:1 virtual-to-real address correspondance for the segment, as described for the :SEG command.

:RES The RES control command allows the user to reserve and name one or more areas at the end of the segment for load-time or run-time debug purposes (see "MODIFY" control command for further comment). The areas are defined as DSECTs and may be referenced by ROMs.

The form of the command is

```
:RES (def,size)1 [,(def,size)2,...,(def,size)n]
```

where

def creates an external definition whose value is the FWA of the reserve area. The definition must be unique within the path.

size is a decimal value specifying the number of words in the reserve area.

Example: Reserve Two Areas at Segment End for Debug Purposes

```
:RES (PATCH1,5),(PATCH2,10)
```

In this example, two areas are being reserved at the end of a given segment; the first (PATCH1) comprising an area of five words and the second (PATCH2) an area of ten words.

:LCOMMON The LCOMMON control command allows the user to determine the allocation of Labeled COMMON blocks (DSECTs) within the root and overlay segments of the program. (See "FORTRAN Interface-Labeled COMMON" for a discussion of restrictions concerning Labeled COMMON allocation and initialization.)

The form of the command is

```
:LCOMMON (dsect0)[,(dsect1),...,(dsectn)]
```

where

$dsect_i$ is either blockname,size or blockname,DEFER.

blockname is the one- to eight-character EBCDIC name of the Labeled COMMON block or DSECT.

size is a decimal value specifying the largest word size needed for the allocated block. If 'size' is omitted, zero is used.

DEFER specifies that the Labeled COMMON block will not be allocated until it is mentioned again in either an :LCOMMON command without DEFER or a :RES command. The size allocated will be the largest specified in either that command or any declarations encountered in ROMs by that time. If a Labeled COMMON block is deferred, it is the user's responsibility to insure that the following rules are observed:

- The block may be deferred only once before it is declared in a ROM being loaded.
- The block must be allocated by one of the methods described, or an unsatisfied-reference error will result.
- If the block is deferred in a segment common to several paths which declare it in ROMs, it must be allocated in a segment which is common to those paths. Otherwise, it will remain unallocated in some paths, and will result in unsatisfied-reference errors in those paths.
- If the content of the block is initialized by assembled code, that code must be in the segment where the block is allocated.

Examples: Specify Allocation of a Labeled COMMON Block

In the overlay structure given in Figure 18, seg 0 references a Labeled COMMON block A of 50 words; seg 1 references a Labeled COMMON block A of 100 words; seg 2 references a Labeled COMMON block A of 60 words.

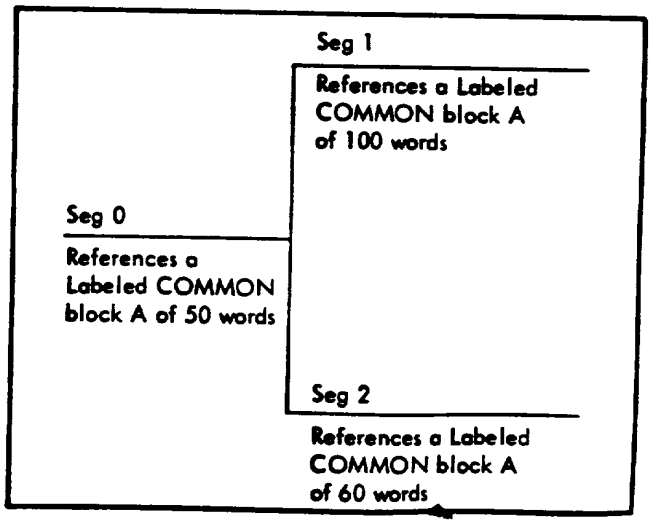


Figure 18. DSECT Allocation Example

Normally, the Loader would allocate block A with the size first encountered (50 words), and would output a diagnostic alarm when the second block of 100 words in seg 1 is encountered.

However, the :LCOMMON command inserted in the deck structure

```
!OLOAD (MAP,ALL)
:ROOT (DEV,CRA03)
:LCOM (A,100)
:SEG (LINK,1),(DEV,CRA03)
:SEG (LINK,2),(DEV,CRA03)
```

would set the size of block A to 100 words.

:MODIFY The MODIFY control command modifies core locations of relocatable programs at load time. Core locations in either root or overlay segments can be modified. Since the reserved area at the end of a segment (allocated with the RES command) is output to disk as part of the segment, that area can be used for "patches" that will be read in with the segment at execution time. The MODIFY commands must be input at the end of the ROOT, SEG, or PUBLIB substack for the segment being modified. If the GO option is specified, the MODIFY commands must follow any RES or INCLUDE commands and precede any :ASSIGN commands. If the (GO, LINKS) option is specified, the MODIFY commands must be ordered by segment number and follow the OLOAD command.

Normally, the image of a segment in a load module does not include :RES areas, since these areas need not be initialized. If a :MODIFY command affects such an area however, the load module image of the segment is extended to include the last modified word.

The form of the command is

```
:MODIFY[(SEG,ident),](LOC,address),word1[,...,wordn]
```

where

SEG,ident specifies the identification number of the segment to be modified. This option is only necessary when the (GO,LINKS) option has been specified. If this option is omitted, the segment identified by the preceding ROOT, SEG, or PUBLIB command will be modified.

LOC,address specifies the relative location of the first 32-bit word to be modified. The address must be expressed as an external definition name plus or minus an optional hexadecimal or decimal offset. Hexadecimal values are distinguished from decimal values by a period preceding the hexadecimal value (i.e., .A9B).

word$_i$ specifies the word to be inserted (right-justified) at the ith location. The word can be expressed as:

1. A signed (plus sign (+) optional) hexadecimal or decimal value that cannot be enclosed in parentheses. Hexadecimal values are preceded by a period.

   Examples

   -6, 100, .2A, -.AF

2. An external name plus or minus an optional offset that cannot be enclosed in parentheses. The offset can be either a hexadecimal or decimal value. Address resolution for the external can be specified by using the SYMBOL notation: rr(name) ± offset

   where

   rr = BA, HA, WA, or DA

   Word resolution is assumed by default. Note that BA(ALPHA) + 3 is legal; BA(ALPHA + 3) is not. If the name specified has not been declared an external somewhere in the overlay segment's path, it will be listed as an unsatisfied REF on the MAP.

   Examples:

   TABL + .F, TABL-1, TABL, HA(TABL)

3. A symbolic instruction that must be enclosed by parentheses. The mnemonic field of the instruction must be an EBCDIC operation code that immediately follows the left parenthesis. (The floating-arithmetic, floating-shift, decimal, and byte string instructions have not been implemented.) The register and index fields can only be signed hexadecimal or decimal values. The address field can be either a signed hexadecimal value, a signed decimal value, or an external name plus or minus an optional offset.

Examples:

```
:MODIFY (LOC,MAP + . F0),(B PATCH + 6)
```

```
:MODIFY (LOC,PATCH + 6),(LI,6 BA(TABL)),
    (LW,9 *WA(VAL) + 9,6),(B MAP + . F1)
```

```
:MODIFY (SEG,0),(LOC,U:PCB+.140),.3F,-.F,250,-1
```

```
:MODIFY (SEG,1),(LOC,CCI + .80),(LI,5 0),-
    (BAL,15 SERCHTAB),(MTW,-1 FLCHG),
    (BLEZ*CC1 + 5),(LB,6 0,5),
    (STB,6 *WA(TABL) + 1,5),(LW,0 .4E)
```

```
:MODIFY (SEG,3),(LOC,FPTLO),M:LO + .11000000,
    .F0400090,ERRIO,ABNIO,BUF1,80,0
```

In reporting MODIFY command errors, any EBCDIC string, decimal number, or hexadecimal number that is separated by a comma, blank, plus sign, or minus sign (ignoring parentheses) is counted as an item. An example of items on a MODIFY command is given in Figure 19.

:ASSIGN The ASSIGN control command is used to create, initialize, or modify DCBs at load time. If the DCB is in the program's DCBTAB table, it will be either initialized or modified. If the DCB is not named in DCBTAB, the Loader will build the DCB from the parameters on the :ASSIGN control command in an unnamed DCB's entry. An error diagnostic is output if an unnamed DCB entry is not available (see "Load Time ASSIGN").

The format and options are identical to the Monitor IASSIGN control command. The :ASSIGN control commands must be the last commands in the control command stack.

Example: Assign a DCB at Load Time

```
:ASSIGN (F:2,LPA02),VFC,(RECL,133)
```

In this example, the DCB F:2 is assigned to the line printer. Vertical format control is specified, so that the first byte in each record controls the spacing on the line printer. Although 133 bytes are output for each record, only the last 132 bytes are printed because VFC is specified.

:PUBLIB The PUBLIB control command is used to specify the object modules from which the Public Library is to be created. The order of the parameters determines the order of loading.

The form of the command is

```
:PUBLIB[(input option),(input option),...,(input option)]
```

where the input options are the same as for :ROOT.

If there are no input options on the PUBLIB control command, the first object module on the GO file will be input.

When the specified object modules have been input, the Loader searches the libraries (specified on the OLOAD control command or the System Library by default) to satisfy any unsatisfied primary references. If a COMMON, labeled COMMON block, or other DSECT is encountered in an object module of the Public Library, the load process is aborted and an error diagnostic is output. If the severity level exceeds zero in the load process, the Public Library is not loaded. If anything was written on the Public Library file, the file is destroyed and an error diagnostic is output.

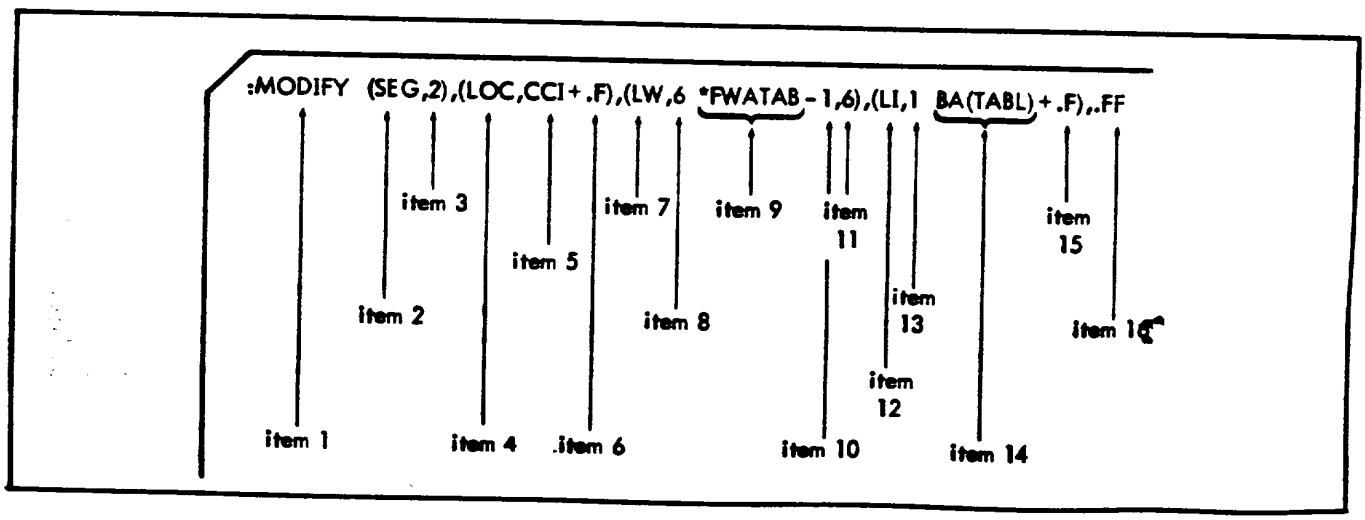


Figure 19. :MODIFY Command Items Example

The following conventions concerning other control commands should be observed when using the PUBLIB command:

1. The FORE option must be specified on IOLOAD to define the area that the Public Library is to occupy at execution time. If the limits of this area are exceeded, the Loader aborts.

2. The FILE option on IOLOAD must either specify the name of the Public Library file being created in the Foreground area or omit the area name so that it defaults to FP.

3. The TEMP, PUBLIB, GO, and TASKS options are illegal, and if used, the Loader will abort with an OLOAD control command error.

4. BOUND should be avoided unless a special debug version of a Public Library is being created.

5. :ROOT, :ASSIGN, :LCOMMON, and :COMMON control commands cannot be used in creating a Public Library. The :RES command cannot be used in the PUBLIB command substack, but may occur in a SEG substack of a Public Library load deck.

6. A single :SEG command substack may follow a :PUBLIB command substack. This is intended primarily for defining a context data area to be used by reentrant Public Libraries. The segment so defined will be acquired in separate real memory for each secondary task requiring the Public Library. It will not be subject to memory management calls. The only parameter groups permitted for this :SEG command will be "EXLOC" and "input option" as described for the :SEG command, earlier. (Note that the LINK parameter group, required for other :SEG commands, is not allowed for a Public Library context :SEG command.) The substack for this command may contain only :RES, :MODIFY, :LIB, and :INCLUDE commands.

Example: Create Public Library From Specified Module

```
:PUBLIB (FILE,SP,MODULE,10)
```

In this example, the Public Library will be created from the first 10 object modules in the System Library.

Example: Create a Public Library with a Context Segment

```
:RES (CONTEXT,100)
:SEG NONE,(EXLOC,9000)
:PUBLIB (FILE,SP,MODULE,10)
```

In this example, the Public Library will be created from the first ten ROMs of the System Library. The Public Library will include a segment of 100 (decimal) words at location 9000 (hexadecimal).

:LMHDR This command allows the user to specify values for certain task control parameters when the load module for the task is linked. These parameters do not affect the linking operation in any way; they are mainly saved in the load module header for reference by the task initiation service. The command or any of its options may be omitted, in which case CP-R task initiation will provide values for the parameters not specified. For most uses, the default values provided are appropriate, so the command may be omitted. When it is used, the :LMHDR command must appear past all commands other than :ASSIGN commands. It may not appear during linking of a PUBLIB.

The options which may appear on a :LMHDR command are described below.

```
:LMHDR option[,option,...,option]
```

where option is one of the following:

(SECB,m) This option specifies that the task will be allowed no more than m asynchronous service calls. If the option is omitted, the system default value pertains.

(RECB,m) This option specifies that the task will be allowed to service no more than m signals or other events. If the option is omitted, the system default value pertains.

(ENQ,m) This option specifies that the task will be allowed no more than m simultaneous enqueues for resource availability. If the option is omitted, the system default value pertains.

PROGRAM FILE

The Program File contains the root and overlay segments in core image format and a one-granule header. The program header is located at granule 0 and contains information necessary to run-load the program.

ROOT SEGMENTS

The root is divided into two parts (see "Core Layout at Execution Time" later in this chapter). Part one of the root always beings in granule 1 of the Program File, and contains the PCB, root code, library code, labeled COMMON, and RES area for the root. Part two contains the DCBTAB, OVLOAD Table, Loader-created DCBs, and the Temp Stacks.

The Temp Stacks are not output on the Program File. Each part of the root is written as a continuous string of bytes.

OVERLAY SEGMENTS

Each overlay segment begins on a granule boundary and is written on the Program File as a continuous string of bytes. The order of segments on the file is unimportant, since the granule displacement pointer (in the OVLOAD table) for each segment specifically determines its position. Load module images of segments cannot be longer than 16K words (64K bytes); however, since RES areas are omitted from the load module, :RES commands may extend the size of a segment beyond 16K words.

TEMPORARY DISK FILES

The Loader uses six scratch files in the Background Temp area of the disk (X1, X2, . . . , X6). If one of these files overflows, the Loader completes the pass over the object modules even though the load will be aborted. The Loader calculates the number of records (for sequential files) or granules (for direct-access files) required for all scratch files and lists this information on the Map. With this information, the user can then allocate the Background Temp files with an !ALLOBT command and reload the program.

LOADER-GENERATED ITEMS

All items discussed in the following paragraphs are generated by the Loader and located in the root segment of the overlay program (see Figure 22 in this chapter for a diagram of the core allocation).

PROGRAM CONTROL BLOCK

The PCB is built by the Loader and located at the FWA of the overlay program area.

DATA CONTROL BLOCK

The Loader automatically includes a copy of the M:SL DCB in any program that has overlay segments. (M:SL is used by the memory management functions to read in overlay segments at execution time.)

Any external DEF/REF that begins with an M: or F: is defined to be either a system (M:) DCB or user (F:) DCB. DCBs referenced by the program that are not satisfied at the conclusion of the load process are either created or allocated by the Loader. Copies of system and FORTRAN DCBs are created with their standard system parameters and operational label assignments. Space for user DCBs is allocated at the rate of seven words per DCB.

Parameters for user or system DCBs may be defined by either !ASSIGN control commands at execution time (for background programs only) or :ASSIGN control commands at load time.

The user can create his own DCBs within the source code and locate them in any segment of his overlay program. However, if the user wishes to change parameters in a DCB at execution time via an !ASSIGN command, he must declare the DCB as an external definition (with a name that begins with an F:) and locate the DCB in the root segment. To utilize the FORTRAN capability of performing I/O by using variables as operational labels, the user can specify (on the OLOAD control command) a number of unnamed DCBs to be allocated by the Loader. The user must name and define these DCBs before the program executes; either at load time (with :ASSIGN), or execution time (with !ASSIGN).

DCBTAB

The DCBTAB table is created by the Loader, and contains the EBCDIC name (if any) and location of each Loader-recognized or created DCB in the root of the overlay program. The EBCDIC name of an unnamed DCB is inserted when the DCB is given a name by either the !ASSIGN or :ASSIGN control command.

OVLOAD TABLE

The OVLOAD table is built by the Loader and contains the information necessary for memory-management functions to access overlay segments at execution time. The OVLOAD table consists of one entry for each overlay segment, with a total of eleven words per entry.

TEMP STACKS

The Loader allocates space for the overlay program's user and CP-R Temp Stacks either according to the number of words specified on the OLOAD control command, or by default. The Temp Stacks are located at the end of part two of the root segment and are not output on the Program File (see "Core Layout of User Program at Execution" later in this chapter).

EXTERNAL DEFINITIONS

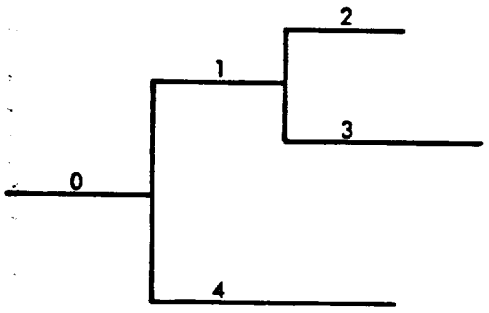
The Loader adds the external DEFs F4:COM, U:PCB, and P:END to all programs except for Public Libraries. F4:COM is the name of FORTRAN's blank COMMON. The initial size is set to zero and changed to the largest size encountered during the load process. If there are no references to F4:COM, blank COMMON is allocated with a size of zero. The Loader indicates the LWA + 1 (including blank COMMON) of the loaded overlay program by an external definition, P:END. External references to P:END within the overlay program will be linked to this definition. The external definition U:PCB has as its value the FWA of the PCB.

The external DEF FP:MBOX is added to primary foreground overlay programs by the Loader only if an area was allocated at SYSGEN time. FP:MBOX is the name of the primary program's mailbox. External references to FP:MBOX will be linked to this definition. Secondary tasks cannot use the mailbox for intertask communication.

LIBRARIES

The Overlay Loader supplies the capability to search the System Library or the User Library in any order. The default condition is for the Loader to search and load only from the System Library. Control commands and keywords enable the user to control more specifically the search and load options. Note that an attempt will first be made to satisfy all REFs with DEFs from the Public Library, if a Public Library has been specified on the OLOAD control command.

If any unsatisfied primary references exist after loading the specified modules for a root or an overlay segment, the Loader searches the library or libraries in the specified order to satisfy those references. Thus, if an external REF is made to a higher level segment, the name should not be the same as a library definition. Consider the following:



If segment 1 contains a primary reference, 9SIN, it will normally be satisfied by loading a Library at the completion of segment 1. Thus, if the definition 9SIN occurred in segment 2, it would be in error (a duplicate definition). The loading of 9SIN from the library can be suppressed by using the EXCLUDE command. In this case, the forward REF would be linked and no duplicate DEF would occur. However, if the definition 9SIN occurred in the root, or in the library loaded in the root, no search for 9SIN would be made in segment 1, and the occurrence of the definition 9SIN in segment 2 would be in error. Primary references can occur in two ways: As external references in a module, or by listing the primary references on the INCLUDE control command.

SYSTEM AND USER LIBRARY LIBRARIES

Cross-references between System and User Libraries are allowed. However, since each library is searched only once per segment, the order of search is important.

If Library A contains references to be resolved by Library B, the search criteria :LIB (A, B) must be specified to guarantee cross-reference resolution. If B also contained references to A they would not be resolved. (Note that these remarks do not apply to cross-references within any single library).

Generally, the System Library should contain the FORTRAN Math and Run-Time Routines and should be independent. The User Library is a repository for user subroutines and alternate Math and Run-Time routines that supersede the same routines in the System Library.

The typical search order would be

```
:LIB (USER,SYSTEM)
```

where both libraries are referenced. In this case, all unsatisfied REFs from the User Library would be satisfied (where possible) from the System Library.

ASSEMBLY LANGUAGE

Library routines may be coded in AP or FORTRAN IV.

ENTRY ADDRESS

Entry addresses in library routines are ignored.

SYSTEM AND USER LIBRARIES ON DISK

The System Library and the User Library on disk are structurally identical. Each library consists of four files:

EBCDIC

MODIR

DEFREF

MODULE

The System Library is located in the System Programs (SP) area. The User Library is located in the Foreground Programs (FP) area.

Only the MODULE file contains the actual binary modules of the library. The other files are tables constructed by the RADEDIT to facilitate the rapid search of the library by the Overlay Loader without actually reading the module. The library is structured on the principle that access should be as fast as possible, since it is performed frequently during an overlay loading procedure.

The three files: EBCDIC, MODIR, and DEFREF contain enough information to determine which modules from the actual MODULE File are to be loaded without examining these modules directly. All four library files are constructed and maintained by the RADEDIT. These short files contain coded information about the external definitions and primary references for each module in the library.

CONSTRUCTING AND MAINTAINING LIBRARY

To begin construction of a library, the user allocates the EBCDIC, DEFREF, MODIR, and MODULE files with the RADEDIT, and then copies the library's binary object modules onto the MODULE file. As each module is copied, the DEFs and REFs are scanned, and corresponding entries are built in the other files by the RADEDIT. Library routines may be added or deleted by using the RADEDIT :COPY and :DELETE commands.

PUBLIC LIBRARIES

The Public Library is a file containing a set of reentrant subroutines in core image format that can be shared in common by all secondary or all primary programs. The resultant saving in core can be considerable where library routines are shared. The Public Library is created from input modules or library routines by the Loader (see "Forming a Public Library"). The availability of the Public Library is determined at execution time.

CALLING THE PUBLIC LIBRARIES

When a user indicates by the PUBLIB keyword on the OLOAD control command that Public Libraries are to be used to satisfy references, the names are set in the program header for the Root Loader, and the Public Library Symbol tables are read from the Public Library files and added to the loaded program's Symbol table. The Loader will satisfy primary external references with Public Library definitions at the time the external reference is encountered in the object module, not at the end of the segment (as when the other libraries are searched). When the program is initiated, the header is searched to determine if the program contains the name of one or more Public Libraries. If so, and one of the named Public Libraries is not already in core, the Monitor determines whether Public Library space is available. If available, the Root Loader reads in the named Public Library or Libraries and the program executes. If the space is not available for all Public Libraries referenced, the program will be neither root loaded nor executed.

Each Public Library file is designated at Public Library creation time (see "Forming a Public Library"). All Public Libraries are located in the Foreground Programs area of the disk. Public libraries must be either primary or secondary; they may not be both.

LIBRARY PROTECTION

Secondary Public Library root segments are given read-and-execute protection, as they are shared. Context segments are writable, since they are differently mapped for each task.

RELEASING A PUBLIC LIBRARY

If no currently executing program is utilizing a Public Library it is released.

FORMING A PUBLIC LIBRARY

A Public Library is created by using the :PUBLIB control command in place of the ROOT command, and modules may be input and libraries searched and loaded in the same manner as for standard loading. Because each Public Library has a unique name, more than one Public Library can exist in the system. Although no more than four Public Libraries can be called by any one program, any number can be created.

ROUTINES USED TO FORM A PUBLIC LIBRARY

All routines used to form a Public Library must be reentrant.

If the Public Library is primary, the user Temp Stack may be used directly for context, or the method of moving local storage in routines to the user Temp Stack on reentrance can be employed (e.g., Real-Time FORTRAN subroutines and FORTRAN Run-Time Library). FORTRAN main routines are not reentrant and cannot be used.

Secondary Public Libraries should use a context segment to achieve reentrancy.

Routines assembled in AP are acceptable provided the reentrancy requirements are met.

No references to COMMON, labeled COMMON, or DSECTs are allowed in any Public Library routine.

Since DCBs in the Public Library could not be assigned and might not be reentrant, DCBs will not be allowed in any Public Library routine. Note that it is not possible for the Loader to warn the user about DCBs that are not named according to the conventions and made externals.

The file associated with each Public Library is in the FP area. This file contains the actual core image of the Public Library and the corresponding Symbol table used by the Loader. The name of the file must correspond to the name given with the FILE keyword on the OLOAD control command, and the file must be previously allocated in the FP area by the user. If loading of the requested modules and libraries has been completed and there are no remaining unsatisfied primary references, the Loader writes the core image and the Symbol table to the file in the FP area. If unsatisfied primary references are found, the file in the FP area is destroyed. A file name of a previous Public Library may be used, but at the risk of obliterating the old file if the new one cannot be completed.

MAP

Three types of maps may be output to M:LO following Pass 2 according to the MAP keyword on the !OLOAD control command: a short map, PROGRAM map, or ALL map. If the MAP option is not specified, none is output.

The short map is output when the MAP keyword appears alone. It consists of essential information about the overlay structure.

The PROGRAM Map consists of all elements of the short Map, plus all external definitions and control sections contained in the input modules (excluding those from Library ROMs).

The ALL Map consists of all elements of the PROGRAM map and includes all definitions and control sections from Library ROMs. A typical PROGRAM map is illustrated in Figure 20.

In Figure 20, the header keywords have the following meaning:

1. Program Header Keywords:

FILE: Area and name of the program file.

NUMBER OF SEGMENTS: Decimal number of each type of segment.

LIMITS: FWA and LWA of the Program area.

BOUND: Hexadecimal value on which object module addresses are bounded.

BLANK COMMON BASE: FWA of blank COMMON with the SIZE specified in decimal words.

PUBLIC LIBRARIES: Names of the Public Libraries, if any, referenced by the program.

TOTAL MEMORY SIZE: Sum of the memory sizes in words of all segments of the program (includes Root and COMMON segments; excludes Public Libraries), in hexadecimal/decimal. This would be the memory required by the program if it were not overlaid.

TOTAL FILE SIZE: The hexadecimal/decimal number of granules used in the load module file.

LIBRARY SIZE: Total number of words loaded from the user and/or system libraries.

PROGRAM ERROR SEVERITY: Set to one if any kind of error is encountered; otherwise, set to zero.

2. Loader-Supplied Structure Keywords: (All values are hexadecimal.)

OVLOAD: FWA of the OVLOAD table.

PCB: FWA of the Program Control Block (PCB).

ENTRY: Entry word address for the root.

UTSFWA: FWA of the User Temp Stack.

SIZE (first occurrence): Size in words of the User Temp Stack.

RTSFWA: FWA of the CP-R Temp Stack.

SIZE (second occurrence): Size in words of the CP-R Temp Stack.

DCBTAB: FWA of the DCBTAB.

3. Segment Header Keyword:

INPUT: Total number of hexadecimal words in the segment loaded from the ROMs, RES, and LCOMMON control commands.

LIBRARY: Total number of hexadecimal words in the segment loaded from User and/or System Libraries.

MEM WL: Total hexadecimal length in words occupied by the segment in memory.

LM BL: Total hexadecimal length in bytes of the segment image in the load module file.

FWA: First word address of the segment.

LWA: Last word address of the segment.

ENTRY: Entry address of the segment.

ROMERR: Set to one if the error severity level is set on any ROM input for the segment; otherwise, it is set to zero.

LDERR: Set to one if any loading errors are encountered for the segment; otherwise, it is set to zero.

GRAN: The granule number (decimal) on the program file where the segment's core image beings.

4. Control Sections:

Control sections input from the program ROMs are listed with the following information:

ROM	ROM number	address (hex.)	size (dec.)

Control sections input from user and/or system libraries are listed with the following information:

{ULIB / SLIB}	Record displacement in the MODULE file	address (hex.)	size (dec.)

5. DCBs

The user and system DCBs are listed after the control section of the ROOT with the following information:

{SDCB / UDCB}	{name / UNNAMED}	address (hex.)

```
!OLOAD (MAP,PRO,ALP),(FOR,6000,7000),(LIB,SYS),(UDCB,5),PRI
:ROOT (FIL,BT,GO,4)
:SEG (LIN,1),(FIL,BT,GO,3)
:RES (ABC,400)
:ASSIGN (F:50,LO)
:ASSIGN (F:51,LO)
:ASSIGN (F:52,LO)

PROGRAM MAP
FOREGROUND PROGRAM

 FILE  BT,OV
 NUMBER OF SEGMENTS:      2  ROOT
                          1  COMMON
                          1  OVERLAY
                          0  PUBLIC LIBRARY
 LIMITS:  FWA:  6000  LWA:  6903
 BOUND      2
 BLANK COMMON  BASE   6686  SIZE    200
 PUBLIC LIBRARIES     NONE
 TOTAL MEMORY SIZE:    904/  2308  WORDS
 TOTAL FILE SIZE:      8/    8  GRANULES
 LIBRARY SIZE    4A9/  1193  WORDS
 PROGRAM ERROR SEVERITY  1

 OVLOAD  PCB    ENTRY  UTSFWA  SIZE   RTSFWA  SIZE   DCBTAB
  674E    6000   600E   67D8      64   683C      C8   67BA

ROOT PART ONE

 INPUT  LIBRARY  MEM WL  LM BL  FWA    LWA    ENTRY  ROMERR  LDERR  GRAN
    35     405      43A    10E4   6000   6439   600E       1      0     1

 CONTROL SECTIONS
    ROM       1      600E       20
    ROM       2      6022        4
    ROM       2      6026        2
    ROM       3      6028        4
    ROM       3      602C        2
    ROM       4      602E        4
    ROM       4      6032        2

    SDCB      F:108              677B
    SDCB      M:DO               6782
    SDCB      M:OC               6789
    SDCB      M:SL               6790
    SDCB      F:50               6797
    SDCB      F:51               679E
    SDCB      F:52               67A5
    SDCB      UNNAMED            67AC
    SDCB      UNNAMED            67B3

     DEF  IM  A                  6022  0
     DEF  IM  B                  6028  0
     DEF  IM  C                  602E  0
U    REF  IM  D
    DSCT  IM  F4:COM             6686  0      200
     DEF  IM  P:END              6904  0
     DEF  IM  U:PCB              6000  0
     DEF  IM  V#A                6026  0
     DEF  IM  V#B                602C  0
```

Figure 20. Typical PROGRAM Map

```
      DEF  IM     V#C              6032  0
      DEF  IM     4MAIN            600E  0

  ROOT PART TWO

   INPUT  LIBRARY  MEM WL  LM BL  FWA    LWA    ENTRY  ROMERR  LDERR  GRAN
    1B6       0     1B6    228   674E   6903      0       0      0     7

    CONTROL SECTIONS

  SEGMENT      1  LINKED TO      0

   INPUT  LIBRARY  MEM WL  LM BL  FWA    LWA    ENTRY  ROMERR  LDERR  GRAN
    1A8      A4     24C    2F0   643A   6685      0       0      0     6

    CONTROL SECTIONS
      ROM       1      643A        4
      ROM       1      643E        2
      ROM       2      6440        8
      ROM       2      6448        4
      ROM       3      644C        4
      ROM       3      6450        2

      DSCT  IM    ABC              64F6  0      400
       DEF  IM    0A               643A  0
       DEF  IM    0B               6440  0
       DEF  IM    0C               644C  0
       DEF  IM    V#0A             643E  0
       DEF  IM    V#0B             6448  0
       DEF  IM    V#0C             6450  0

  BLANK COMMON

   INPUT  LIBRARY  MEM WL  LM BL  FWA    LWA    ENTRY  ROMERR  LDERR  GRAN
      C8       0     C8       0  6686   674D      0       0      0     7

    CONTROL SECTIONS
  LOADING WAS COMPLETED

    FILE  BT,OV          USED     8  GRANULES

    FILE  BT,X1          USED     0  GRANULES

    FILE  BT,X2          USED     2  GRANULES

    FILE  BT,X3          USED     2  GRANULES

    FILE  BT,X4          USED     2  GRANULES

    FILE  BT,X5          USED     2  GRANULES

    FILE  BT,X6          USED     0  GRANULES

  WARNING: UNSATISFIED REF'S

  END OF MAP
  !FIN
```

Figure 20. Typical PROGRAM Map (cont.)

DEFs, REFs, and DSECTs

The externals are listed with the following information:

a. { U } = unsatisfied, undefined, or unallocated
{ D } = doubly defined or referenced

b. { DSCT } = dummy section
{ DEF } = definition
{ REF } = primary reference
{ SREF } = secondary reference

c. { PL } = Public Library
{ UL } = User Library
{ SL } = System Library
{ IM } = Input Module

d. The symbol name in EBCDIC (one to 63 characters).

e. If the definition is an address, it is expressed as a word address and a byte offset. If the definition is a constant, it is expressed as a hexadecimal number followed by the letter 'C'. For undefined symbols, the address field is blank.

f. The DSECT size in words (decimal).

ERROR DIAGNOSTICS

The Overlay Loader outputs diagnostic messages to M:OC and M:LL. Duplication is suppressed if OC and LL are assigned to the same device.

If an operator response is required, the Loader will call the "WAIT" function. The operator should hit the console interrupt and key in one of the following:

C	Continue.
X	Abort.
COC	Read the corrected command from M:OC and continue (used only in response to control command errors).

Note that the "WAIT" routine aborts if an IATTEND control command has not been encountered in the job stack. The diagnostic messages in Table 19 are output by the Overlay Loader.

Table 19. Overlay Loader Diagnostics

Text	Meaning	Action
BACKGROUND TOO SMALL	User's program cannot be loaded in the current size of the background. This is a function of the number of external symbols and forward references that a program has, not a function of the program length.	Abort
BINARY CARD ENCOUNTERED INSTEAD OF CC	A binary record was encountered on the C device instead of a control command.	Wait
BOT ON {yyndd / area, name	Unexpected beginning-of-tape has been encountered on the specified device/file.	Abort
BUF SMALLER THAN DATA RECORD DCB x:xxxxxx	Specified DCB has been assigned to a record size larger than the I/O buffer associated with the Read request. Either the user has assigned incorrectly or the Loader has a program error.	Abort

Table 19. Overlay Loader Diagnostics (cont.)

Text	Meaning	Action
CC ERR: BACKGROUND AND PRIMARY	The option PRI appeared on an OLOAD command for a background program.	Abort. The option FORE must appear if PRI is specified on the OLOAD command.
CC ERR: DUP NAME IN ITEM xx	Item number xx on the command is a duplicate of a name in the symbol table.	Wait
CC ERR: DUP SEG IDENT	Ident on :SEG command is a duplicate of a previous segment's ident.	Wait
CC ERR: FOLLOWING ITEM xx	There is an error following item xx on the command (e.g., a parameter has been omitted, an extra parameter has been included, etc.).	Abort if IOLOAD CC. Wait if any other CC.
CC ERR: ILLEGAL CC SEQUENCE	Loader commands have been ordered incorrectly.	Wait
CC ERR: ILLEGAL OPTION FOR PUBLIB LOAD (PUBL, name)	Option (PUBLIB, name) was specified on IOLOAD and a :PUBLIB command has been encountered.	Abort
CC ERR: ILLEGAL OPTION FOR PUBLIB LOAD (TASKS, value)	Option (TASKS, value) was specified on IOLOAD and a :PUBLIB command has been encountered.	Abort
CC ERR: ILLEGAL OPTION FOR PUBLIB LOAD (TEMP, value)	Option (TEMP, value) was specified on IOLOAD and a :PUBLIB command has been encountered.	Abort
CC ERR: ILL OPCODE IN ITEM xx	Specified operation code is either illegal or unimplemented.	Wait
CC ERR: ILL SEG IDENT	Seg ident on the :MODIFY command does not match the segment being modified.	Wait
CC ERR: ITEM xx	Item number xx on the command is in error.	Abort if IOLOAD CC. Wait if any other CC.
CC ERR: MODIFY OUTSIDE SEG LIMITS	One of the locations on the :MODIFY command is outside the limits of the segment.	Wait
CC ERR: NEED (FORE, fwa, lwa) OPTION FOR PUBLIB LOAD	Option (FORE, fwa, lwa) was not specified on the IOLOAD command and a :PUBLIB command has been encountered.	Abort

Table 19. Overlay Loader Diagnostics (cont.)

Text	Meaning	Action
CC ERR: SEG IDENT NOT 1ST OPTION	Segment ident (LINK, ident$_1$) is not the first option on the :SEG command.	Wait
CC ERR: SEGMENTS ORDERED INCORRECTLY	:SEG commands have been input in the wrong order.	Wait
CC ERR: SPECIFIED AREA FOR PUBLIB LOAD NOT 'FP'	Option (FILE, area, name) on !OLOAD did not specify the Foreground Programs area (FP) and a :PUBLIB command has been encountered.	Abort
CC ERR: STEP OPTION ILLEGAL WITHOUT ATTEND	An !ATTEND command must be included in the job when the STEP option is specified.	Abort
CC ERR: UNDEFINED FILE area, name	Area and file specified in the option (FILE, area, name) has not been defined by the RAD Editor.	Abort if !OLOAD CC. Wait if any other CC.
CC ERR: UNDEFINED SYMBOL IN ITEM xx	Symbol name in item xx on the :MODIFY command has not been defined.	Wait
DCB CANNOT BE A DSECT SEGxxxxx {ULIB / ROM / SLIB} xxx	A DCB was encountered in the named segment that was assembled as a dummy section instead of being part of the control section.	Abort
DCB HAS BAD PARAMETERS DCB x:xxxxxx	Specified DCB has bad parameters. Either the user has assigned incorrectly or the Overlay Loader has a program error.	Abort
DCB HAS INSUFFICIENT INFO DCB x:xxxxxx	Specified DCB contains insufficient information to open a Read or Write operation. Either the user has assigned incorrectly or the Loader has a program error.	Abort
DCB NOT ASSIGNED DCB x:xxxxxx	Specified DCB has been assigned to the "null" device. Either the user has assigned incorrectly, or the Overlay Loader has a program error.	Abort
DEFAULT ENTRY ADDRxxxxx SUPPLIED FOR ROOT	A transfer address was not encountered on any ROM in the Root and an entry address was not specified on the CC; therefore, a default has been supplied.	Continue

Table 19. Overlay Loader Diagnostics (cont.)

Text	Meaning	Action
DSECT'S IN PUBLIB LOAD SEGxxxxx {ULIB / ROM / SLIB} xxx	Labeled COMMON blocks (DSECTs) are illegal in the Public Libraries.	Abort
EOT ON {yyndd / area, name}	End-of-tape has been encountered on the specified device/file.	Wait
EXLOC TOO LARGE SEGxxxxx	The execution locations of the specified segment will exceed 131K at the given EXLOC.	Abort
FILE DESTROYED area, name	Overlay Loader is aborting past the point where data has been written on the specified program file. The first sector of the file has been zeroed out.	Abort
FILE UNCHANGED area, name	Overlay Loader is aborting at a point where the program file is unchanged.	Abort
WARNING: ILLEGAL LOAD LOCATION xxxxx SEGxxxxx {ULIB / ROM / SLIB} xxx	Specified "load location" origin has been defined with a value that is either not an address or that lies outside the address limits of the specified segment. (A labeled COMMON block must be initialized in the segment where the block is allocated.)	Continue with COMMON non-initialized.
ILL SEG IDENT TERMINATED MODIFY'S	The :MODIFY commands have been ordered incorrectly for the (GO, LINKS) option. The MODIFY mode has been terminated at this point. If the user wishes to continue, all :MODIFY commands that follow will be ignored.	Wait
LIB ROM'S EXCEED MAX SEGxxxxx	Maximum number of library ROMs that can be loaded is 2000.	Abort
MEMORY CONFLICT INVOLVING PUBLIBS USED	There is an overlap in the memory areas used by a requested Public Library and either another requested Public Library or a user segment or root.	Abort
MONITOR CC ENCOUNTERED INSTEAD OF :ROOT or :PUBLIB	Monitor control command was encountered on the C device instead of a :ROOT or :PUBLIB command.	Abort

Table 19. Overlay Loader Diagnostics (cont.)

Text	Meaning	Action
MOUNT PAPER TAPE ROM	STEP option was specified on IOLOAD and the next relocatable object module (ROM) is to be input from the paper tape reader.	Wait. Operator should load the paper tape, interrupt, and key in "C".
NOT ENUF UDCBS	An ASSIGN command requires that a DCB be built, and there are no unassigned DCBs (UDCBs) remaining.	Continue. The DCB on the erring ASSIGN command is not built.
PROGRAM ERR: {CCI / ONE / TWO / MAP / LIB} ADDR xxxx	Loader has a program error in the named overlay at the specified address.	Abort. Operator should get a core dump.
SEGxxxxx {ULIB / ROM / SLIB} xxx		
PROGRAM ERR: {CCI / ONE / TWO / MAP / LIB} SB=xx, ADDR xxxx DCB x:xxxxxx	Specified error status has been returned from an Overlay Loader call (in the named overlay) to a Monitor I/O routine. The address of the CAL and the name of the DCB are specified.	Abort
PROGRAM ERR: UNALLOCATED CSECT SEGxxxxx {ULIB / ROM / SLIB} xxx	Loader has encountered a control section that has not been allocated; either a Loader, compiler, or assembler error	Abort
PROGRAM FILE HEADER EXCEEDS ONE SECTOR	Size of program file header was found to be greater than one sector.	Abort
RAD FILE TABLE FULL	RAD File Table size that was allocated at SYSGEN is insufficient.	Abort
READING AN OUTPUT DEVICE DCB x:xxxxxx	A DCB that the Overlay Loader reads with has been assigned to an OUT device. Either the user has assigned incorrectly or the Loader has a program error.	Abort
ROM ERR: BAD SEQ SEGxxxxx {ULIB / ROM / SLIB} xxx SEQNOxxx	Sequence number of the binary record does not equal xxx.	Wait

Table 19. Overlay Loader Diagnostics (cont.)

Text	Meaning	Action
ROM ERR: EXPRESSION SIZE EXCEEDS MAX SEGxxxxx {ULIB / ROM / SLIB} xxx SEQNOxxx	An object language expression on the specified binary record exceeds 120 bytes.	Abort
ROM ERR: ILLEGAL LOAD ITEM SEGxxxxx {ULIB / ROM / SLIB} xxx SEQNOxxx	Object language on specified binary record cannot be translated (assembler or compiler error).	Abort
ROM ERR: NO MODULE END SEGxxxxx {ULIB / ROM / SLIB} xxx SEQNOxxx	Module end was not encountered on the last binary record of the relocatable object module.	Abort
ROM ERR: NOT OBJECT LANGUAGE SEGxxxxx {ULIB / ROM / SLIB} xxx SEQNOxxx	Specified binary record is not in object language format.	Wait
ROM ERR: NOT STANDARD BIN SEGxxxxx {ULIB / ROM / SLIB} xxx SEQNOxxx	Specified record has a non-standard binary format.	Wait
SMM OPTION ILLEGAL WITH FGD OR PUBLIBS	Simplified Memory Management is legal only for background programs.	Abort
SYMBOL DECLARED BOTH AS DEF AND DSECT: xxxx	The symbol xxxx occurred in the same path with the two usages indicated.	Abort
TOO MANY ROM FILES	The maximum number of ROM files that can be loaded is 127 per segment.	Abort
UNDEFINED FILE area, name DCB x:xxxxxx	Specified DCB has been assigned to a disk file that has not been defined by the RAD Editor.	Abort
UNDEFINED ORIGIN SEGxxxxx {ULIB / ROM / SLIB} xxx	Loader has encountered a "load location" origin with an expression that cannot be resolved.	Abort
UNEXPECTED EOD ON {yyndd / area, name	Unexpected IEOD was encountered on the specified device/file.	Wait if the IEOD was encountered instead of a ROM; otherwise, Abort.
UNEXPECTED MONITOR CC ON {yyndd / area, name	Unexpected Monitor control command was encountered while reading ROMs from the C device.	Abort

Table 19. Overlay Loader Diagnostics (cont.)

Text	Meaning	Action
UNRECOVERABLE RD ERR ON {yyndd / area, name	Transmission error has occurred while reading from the specified device/file.	Abort
UNRECOVERABLE WR ERR ON {yyndd / area, name	Transmission error has occurred while writing on the specified device/file.	Abort
WARNING: DCB IN OVERLAY SEGMENT SEGxxxxx {ULIB / ROM / SLIB} xxx SEQNOxxx DCB x:xxxxxx	Specified DCB was declared an external DEF in a segment other than the Root. The DCB will not be included in DCBTAB.	Continue
WARNING: DEF'D DCB NOT DEFINED DCB x:xxxxxx	Specified DCB was declared on external DEF and the DEF was never defined.	Continue
WARNING: DUPLICATE DEF'S	User's program contains duplicate external DEFs. Map will indicate the name(s) of the DEFs.	Continue
WARNING: DUPLICATE REF'S	User's program contins duplicate external REFs. Map will indicate the name(s) of the REFs. Occurs when identical DEFs in different segments of different paths are referenced by the same REF (in a segment common to both paths).	Continue
WARNING: ENTRY ADDRxxxxx OUTSIDE SEGMENT SEGxxxxx	Entry address for the specified segment is outside the segment's address limits.	Continue
WARNING: ILLEGAL DCB ADDR DCB x:xxxxxx	Specified DCB was declared an external DEF and the DEF has been defined with either a negative address or a constant.	Continue
WARNING: ILLEGAL DCB NAME SEGxxxxx {ULIB / ROM / SLIB} xxx DCB x:xxxxxx	Specified DCB name is illegal and will not be included in DCBTAB. Monitor DCBs(M:) must have standard OPLB names. User DCBs(F:) must not exceed eight EBCDIC characters in length.	Continue
WARNING: LCOM name OF SIZE xxxx GREATER THAN ALLOCATED SEGxxxxx {ULIB / ROM / SLIB} xxx SEQNOxxx	The named labeled COMMON block (DSECT) with the size specified in words is greater than the size allocated.	Continue

Table 19. Overlay Loader Diagnostics (cont.)

Text	Meaning	Action
WARNING: NO ENTRY ADDRESS FOR ROOT	Root does not have an entry address.	Continue
WARNING: OVERLAY SEG GREATER THAN 16K	Specified overlay segment exceeds the maximum size record that can be loaded by the Monitor SEGLOAD function.	Continue
WARNING: PROGRAM EXCEEDS SPECIFIED ADDR LIMITS	User's program exceeds the address limits, either specified on !OLOAD or the defaults for background/foreground programs.	Continue
WARNING: PROGRAM FILE RSIZE NOT EQUAL TO GSIZE	Record size of the program file is different from the granule size.	Continue
WARNING: UNDEFINED DEF'S	User's program contains external DEFs that either have not been defined or have been defined with an expression the Loader cannot resolve. Map will indicate the name(s) of the undefined DEFs.	Continue
WARNING: UNDEFINED ENTRY ADDR SEGxxxxx	Expression defining the entry address for the specified segment cannot be resolved by the Loader.	Continue
WARNING: UNSATISFIED REF'S	User's program contains unsatisfied external REFs or unallocated DSECTs. Map will indicate the name(s) of the REFs and DSECTs.	Continue
WRITING ON INPUT DEVICE DCB x:xxxxxx	A DCB that the Overlay Loader writes with has been assigned to an IN device. Either the user has assigned incorrectly or the Loader has a program error.	Abort
yyndd WRITE PROT	Specified RAD is write-protected. Condition is brought about by system write protection or hardware write protect violation.	Wait and 1. Reset RAD protection switches or 2. Interrupt and key in "SYC", or 3. Interrupt and key in "X" if the job is not allowed to write on protected areas of the RAD.

USER LOAD-TIME ASSIGNS

M:DCB AND F:DCB

DCBs identified by external definitions must exist in the root for each unique reference to an M:DCB or F:DCB. These are either inserted explicitly by the user or built implicitly by the Loader. A user can change DCB assignments in several ways:

1. By modifying the DCB at execution time.
2. By using a load-time :ASSIGN (foreground and background).
3. By using a run-time !ASSIGN (used by background jobs).

RUN-TIME ASSIGNS

Run-time !ASSIGNs (by Job Control Processor) apply only to the background job step in which they are inserted. Change of assignment for foreground programs is permitted only through STDLB operations, load-time :ASSIGNs, or DEVICE (set device/file/oplabel index) CALs.

LOAD-TIME ASSIGNS

Load-time :ASSIGNs are changes to the respective DCB at load-time, so that the given assignment remains as a part of the program. This effectively allows assignments for foreground programs, and assignment of DCBs with nondefault cases.

FORTRAN INTERFACE

System interface between FORTRAN-produced programs and CP-R is the shared responsibility of the FORTRAN compiler-Loader-CP-R complex. This complex enables the user to program real-time programs for foreground operation using Real-Time FORTRAN language without having to use symbolic coding to create the system interface (see FORTRAN job examples in Chapter 12).

Symbolic code and control information can be used to give the FORTRAN user added versatility in cases where compatibility with other FORTRAN configurations is not a factor. However, such coding is not required. That is, the user can write and execute a program to service real-time interrupts without any symbolic embellishment of the FORTRAN language and without destroying the real-time response required.

COMMON ALLOCATION

BLANK COMMON

By default, blank COMMON is allocated beginning at the end of the longest path as illustrated in Figure 21.

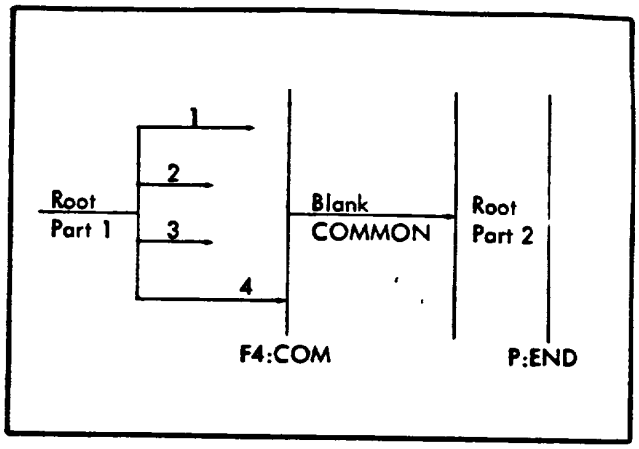


Figure 21. Blank COMMON Allocation by Default

The default size of blank COMMON is determined by the size of the largest blank COMMON encountered during the loading of all segments.

An optional COMMON control command allows the user to specify the size and/or location of blank COMMON, or to force it to follow the segment defined most recently when the COMMON command appeared. In the following example, the COMMON command is used with no parameters. The size then defaults to that in the first ROM that references blank COMMON, and the location is the first boundary (doubleword or specified) past the previous end-of-segment. Figure 22 diagrams the resulting memory layout.

```
:ROOT...

:SEG (LINK, 1)...

:COMMON

:SEG (LINK, 2)...

:SEG (LINK, 3)...

:SEG (LINK, 4)...
```

Note that in Figure 22, segment 1 sets the COMMON base so that segments 1, 2, and 3, share all COMMON, but segment 4 overlays a portion of COMMON. Thus, segments 1, 2, and 3, might operate on a large array, leaving the results in upper COMMON for segment 4, which can reclaim the remainder of the COMMON storage. However, a carefully determined COMMON allocation in segment 4 would be necessary to align references to the upper portion of COMMON.

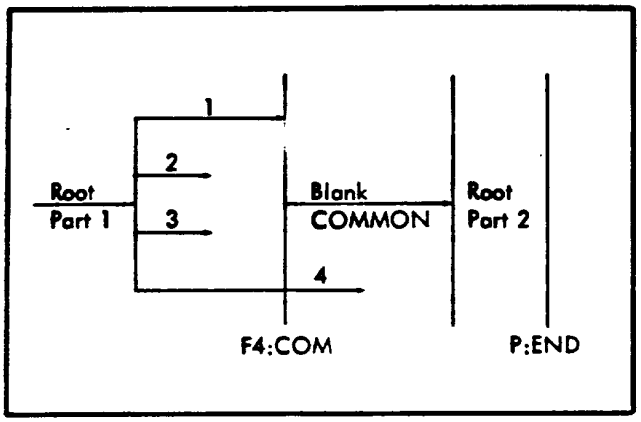


Figure 22. Blank COMMON Option

LABELED COMMON

Labeled COMMON is allocated by the Loader either by default in the segment in which the block is first encountered, or specifically, by the parameters on the LCOMMON control command. All references to a labeled COMMON block must be in the same path as the definition. Note that labeled COMMON in the root is available to all segments. A labeled COMMON block must be initialized in the segment that is allocated.

The :LCOMMON control command will allocate labeled COMMON in the segment specified by the preceding :SEG command, or will cause its allocation to be deferred until a subsequent :LCOMMON or :RES command. The example

```
:LCOMMON (A, 100), (B, 101), (XRAY, 50)

:SEG (LINK, 201, ONTO, 0)
```

will allocate a labeled COMMON block /A/ of 100 words, a block /B/ of 101 words, and a block /XRAY/ of 50 words in segment number 201.

CONNECT

The Loader does not provide any facility for generating code to connect foreground programs to interrupts or to trigger interrupts. The CONNECT statement in FORTRAN plus the Monitor CONNECT call provides the necessary interface.

CALLING OVERLAY SEGMENTS

The Overlay Loader generates no implicit calls for loading overlay segments, and generates no explicit code for such calls. FORTRAN programs to be run in overlay form must call the FORTRAN run-time routine SEGLOD (or its optional alternate name, SEGLOAD), which calls the SEGLOAD function of the Monitor. The identification numbers in the argument list must correspond to the identification number on the SEG control command. Programs that use the SEGLOAD function must be either background programs loaded with the SMM option on the OLOAD command, or primary foreground programs.

The SEGLOAD function calls in the overlay segment and returns, e.g.,

```
CALL SEGLOD (I)

CALL SUBROUTINE
```

where I is the segment ident.

MAIN PROGRAM NAME AND ENTRY

The entry point of a FORTRAN main program is not necessarily the first location of the program. The compiler will output an external definition to identify it as a FORTRAN main program. The entry point for that program is either the transfer address of the main program, or the value specified with the ENTRY keyword on the :ROOT command.

LABELED COMMON NAMES

Labeled COMMON blocks are identified as DSECTs, labeled with an external definition the same as the block name.

BLANK COMMON NAMES

Blank COMMON references are identified as DSECTs with the unique external definition name F4:COM.

CORE LAYOUT AT EXECUTION TIME

The core storage area allocations for a typical segmented program are illustrated in Figure 23.

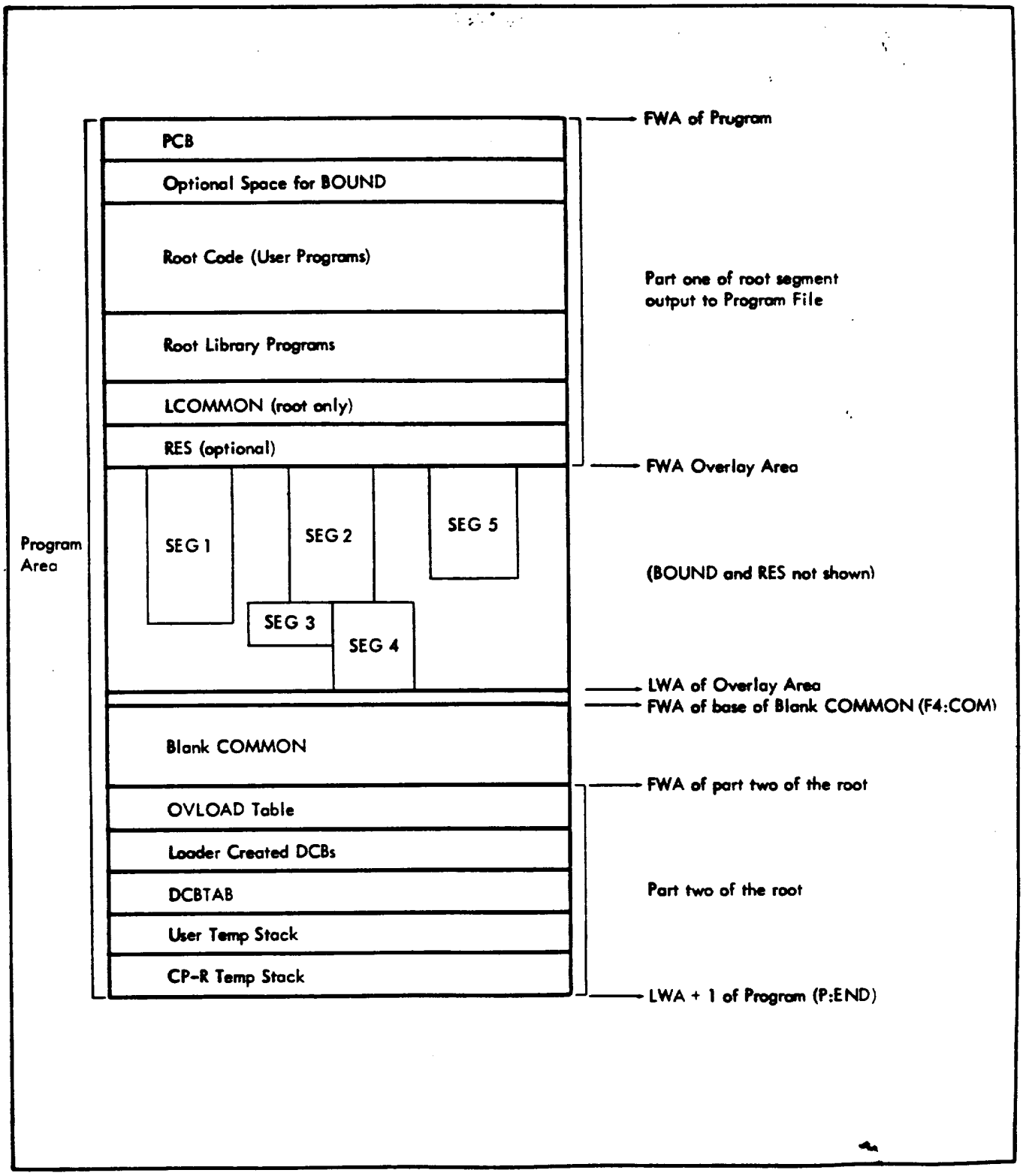


Figure 23. Standard Core Layout of a Program

11. RADEDIT

RADEDIT is a background processor that performs allocation of disk areas by generating and maintaining directories for all permanent files. Through commands input by the user, RADEDIT performs the following functions:

- Adds or deletes entries to the permanent file directories that, in turn, allocate and release permanent space within a disk area.
- Copies data files to and from disk.
- Copies data files from device to device.
- Appends records to the end of an existing disk file.
- Compacts permanent file directories and permanent disk areas.
- Truncates empty space from the end of disk files.
- Maps permanent disk file allocation.
- Maps library module allocation.
- Dumps the contents of disk files, areas, or tapes.
- Copies permanent disk files.
- Copies object modules contained in the libraries.
- Saves the contents of disk areas on a magnetic tape device in a self-reloadable form.
- Restores previously saved disk areas to their disk location.
- Maintains library files on disk for use by the Overlay Loader.
- Zeros out (clears) complete disk areas.
- Temporarily inhibits the use of bad disk sectors.

OPERATING CHARACTERISTICS

FILE ALLOCATION

RADEDIT performs disk allocation for all permanent files. The name, size, and location of each permanent disk area are indicated through the user of a Master Directory that is set up at system generation in the resident portion of CP-R. The permanent disk areas maintained by RADEDIT are

Background programs (BP).

Data (D1-DF) and user defined areas.

Foreground programs (FP) contain User Library.

System programs (SP) contain System Library.

RADEDIT controls file allocation by generating and maintaining a directory entry for each file within the above permanent areas. Every permanent area has a directory that begins in the first sector of its own area. A directory consists of entries with the following information:

- File name (maximum length of eight alphanumeric characters).
- Resident foreground program flag.
- File type; blocked; unblocked, or compressed.
- Granule size in bytes (used for direct access).
- File size (current number of records in file).
- Record size (bytes per logical record).
- Relative disk address of the first sector defined for the file.
- Relative disk address of the last sector defined for the file.
- Extension allocation size.

Before any permanent disk file can be written, space must be allocated for the file by adding a new entry to the designated directory. Directory entries may be added or deleted by using RADEDIT commands. The following method is used to allocate files:

1. Permanent disk files are allocated sequentially, beginning in the second sector of the area, with every file beginning and ending on a sector boundary.

2. A new directory entry is added as the last entry to the existing directory and the corresponding space for the file is allocated.

3. When all available space in an area is exhausted, a complete search of the file directory is made for unallocated areas made available through file deletions. The smallest area containing a sufficient amount of space to allocate for the file is selected. If sufficient space is not found upon searching the directory, the operation is aborted. To overcome this problem, disk squeezing may be requested to recover the unused storage within a permanent area by compressing the directory entries and files (see Figures 24 and 25).

4. File deletion is accomplished by zeroing out the appropriate directory entry.

SKIPPING BAD SECTORS

When a bad disk sector(s) is discovered, it is the user's responsibility to prevent it from being used by deleting the defective file, declaring the bad sector(s) with :BDSECTOR, and reallocating the file if it is to be regenerated.

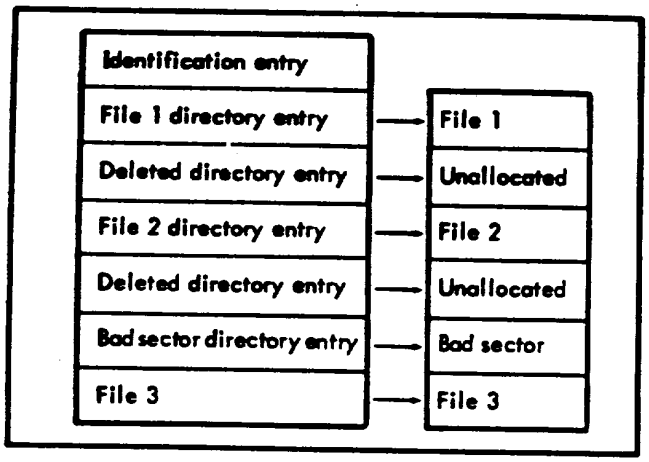


Figure 24. Permanent Disk Area Before Squeezing

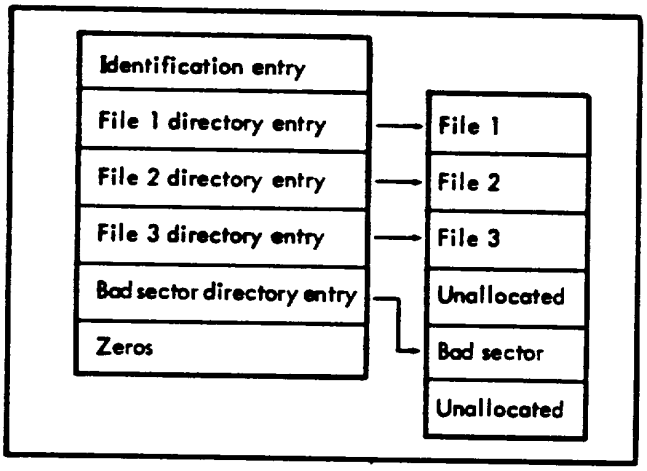


Figure 25. Permanent Disk Area After Squeezing

The method used to handle bad sectors is as follows: the :BDSECTOR command removes the sectors from use by placing a special entry in the file directory and allocating the space as a file. The :GDSECTOR command returns the disk space for use by deleting the file directory entry.

SYSTEM AND USER LIBRARY FILES

System and User Library files are searched by the Overlay Loader to satisfy external references. These files are generated and maintained by RADEDIT in a form that can be rapidly and easily searched by the Overlay Loader. The System Library files must reside in the System Programs (SP) area, and the User Library files must reside in the Foreground Programs (FP) area. Each library consists of three unblocked files: the Module Directory File (MODIR), DEFREF File (DEFREF), and EBCDIC File (EBCDIC), and one blocked file: Module File (MODULE). The user must define and allocate these library files using the file names that appear within parentheses above and defining the files as blocked or unblocked. As an aid in approximating the file sizes, the user can use the algorithms given below.

RADEDIT is the only processor that should write in the library files. The files are generated from information contained in the object modules read in by RADEDIT. Each module is identified within the library files by a DEF. The first DEF encountered in the module is considered the module name, and no other DEF in a program will be so recognized. Any module may be referenced by using the first DEF in a program, and modules may be copied or deleted through its use. This module name may not be the same form as a DCB name (i.e., beginning with "M:" or "F:").

ALGORITHMS FOR COMPUTING LIBRARY FILE SIZES

The following algorithms can be used to determine the approximate sizes of the four files in a library. It is not crucial that the file sizes be exact, since any unused space can be recovered via the :TRUNCATE command. The approximate number of sectors (n_{MODIR}) required in the MODIR file is

$$n_{MODIR} = \frac{3\,(i)}{s}$$

where

- i is the number of modules to be placed in the library.
- s is the sector size in words.
- 3 words is the length of a MODIR file entry.

The approximate number of sectors (n_{EBCDIC}) required in the EBCDIC file is

$$n_{EBCDIC} = \frac{2\,(d)}{s}$$

where

- d is the unique number of DEFs in the library.
- s is the sector size in words.
- 2 words is the average length of an EBCDIC file entry.

The number of records (n_{MODULE}) required in the MODULE file is

$$n_{MODULE} = \sum_{i=1}^{n} C_i$$

where

- n is the total number of modules in the library.
- C_i is the number of card images in the ith library routine.

The number of sectors (n_{DEFREF}) in the DEFREF file is

$$n_{DEFREF} = \frac{\sum_{i=1}^{n} 1 + \frac{d_i + r_i}{2}}{s}$$

where

- n is the total number of routines in the library.
- d is the number of DEFs in the ith library routine.
- r is the number of REFs in the ith library routine.
- s is the sector size in words.

DISK AREA PROTECTION

Updating or squeezing of permanent disk areas containing information for real-time programs (foreground program and foreground data areas) must not occur while the foreground is utilizing these permanent areas. The user must ensure that RADEDIT is not modifying a permanent area at the same time a foreground program is using it.

Software protection of the System and Foreground Data areas of the disk is provided by requiring the operator to key in "SY" before any of these areas are modified by a background processor. The only areas that can be modified that do not require a SY key-in are the Background and Public Data areas.

CALLING RADEDIT

When a !RADEDIT control command is read from the C device, RADEDIT is loaded into core memory from the disk. Control is transferred to the RADEDIT which reads commands from the C device that specify the functions to be performed.

The form of the command is

```
!RADEDIT
```

RADEDIT is terminated when a record with an ! in column one is read from the C device (with the exception of !EOD). An !EOD indicates an end-of-data to RADEDIT when data is input via the :COPY command.

When RADEDIT is called from TJE, the commands SAVE, RESTORE, GDSECTOR, BDSECTOR, and SQUEEZE are not available.

COMMAND FORMATS

All RADEDIT commands are input from the C device and listed on LL. The general form for RADEDIT commands is identical to the control command format described in Chapter 2, with the symbols below being used to aid in describing RADEDIT commands in this chapter.

- aa refers to a disk area and must be one of the following:
  - BP is the Background Programs area.
  - D1 through DF or user defined areas.
  - FP is the Foreground Programs area.
  - SP is the System Programs area.
  - BT is the Background Temp area (can be used with :COPY, :CLEAR, or :DUMP).
- fid is a CP-R file identifier, with standard defaults.
- zz refers to any disk area.
- nnnnnnnn refers to a file name or library module (maximum name length of eight alphanumeric characters).
- yyndd refers to a physical device name, where
  - yy specifies the type of device: CR, CP, etc.
  - n specifies the IOP number: A for IOP0, B for IOP1, etc.
  - dd specifies the device number: 03, B0, etc
- OP refers to an operational label: B1, S1, etc.

RADEDIT COMMANDS

:ALLOT The :ALLOT command adds a new entry to the specified permanent file directory that allocates space for a new file. After space has been allocated, files can be written by either background or foreground programs. The space allocated for the new entry is zeroed out.

The form of the command is

```
:ALLOT (FILE, fid) [,(option)]...[,(option)]
```

where the options are

- FORMAT, type specifies the file format where type is:
  - U for unblocked.
  - B for blocked.
  - C for a compressed file.

The default value is unblocked.

- FSIZE, value specifies the decimal length of the file in logical records. The default value is 1000.

RSIZE,value specifies the decimal number of words per record. The logical record size is used in sequentially accessing a file. For a compressed file, record size is omitted and the Monitor blocks compressed files into 256-word records. Blocked files have a default value equal to 128 words per record. If the record size is greater than 128 words, unblocked organization will be given. Unblocked files have a default record size equal to the granule size.

GSIZE,value specifies granule size in words and is used for direct access only. The default size will be equal to the sector size.

RF indicates that the file contains a resident foreground program and is applicable only if the FP area is specified. If RF is omitted, the file does not contain a resident foreground program. Any program flagged as resident foreground will be automatically loaded into core every time the system is booted from disk.

ESIZE[,value] specifies the decimal length of file extents in logical records. An extent is allocated and attached to the file anytime the file is out of space. If the optional value is omitted, file extents will be the same size as the original file. If the ESIZE parameter is not specified, no automatic file extension will take place.

FIX indicates that during a squeeze of the specified area, the file is not to be combined with its extents. If fix is not specified, the original file will be combined with all of its extents to form one file if the area is squeezed.

The FIX option is applicable only if the ESIZE option is present.

Examples:

1. An unblocked file:

```
:ALLOT (FILE,D3,TEST),(FORMAT,U),(FSIZE,50),-
   -(RSIZE,90)
```

This example allocates space for the unblocked file TEST in the D3 area of the disk, with a file size of 50 records and a record size of 90 words.

2. A blocked file:

```
:ALLOT (FILE,FP,TESTA),(FORMAT,B),-----
   -(FSIZE,50),(RSIZE,256),RF
```

This example allocates space for the blocked file TESTA in the FP area, with a record size of 256 words and a file size of 50 records. This is a resident foreground program.

:COPY The :COPY command copies files of data or modules (EBCDIC, BINARY in standard binary format, or nonstandard binary) from one device to another.† Files are copied until an !EOD or tape mark is encountered, except when the CC option is specified, which is terminated when an :EOD is encountered. Individual records in the file may be as large as 64K bytes if buffer space is available. A logical file mark will be written onto the output file.

On 7-track tape, a copy from a file to a device assigned to an op label will be in packed binary format. If a device is specified, the information (data) will be written unpacked.

When nonstandard binary (BIN) or control commands (CC) are copied from the C device, the C device must be assigned to 0 and reassigned after the copy is completed. The assignment is made when the message

!!KEYIN STDLB C,0

is typed to the operator and reassigned when the message

!!COPY ENDED, REASSIGN C'

appears.

An !ATTEND card must be used to force a pause for operator intervention whenever the BIN and CC options are specified.

The general form of the command is

```
                FROM                    TO
:COPY ( { FILE,fid          } ) , ( { FILE,fid          } )-
        { LIB,aa,nnnnnnnn   }       { LIB,aa            }
        { IN,{op}           }       { OUT,{op}          }
        { IN,{yyndd}        }       { OUT,{yyndd}       }
   -[,VFC][,ADD][,UPD][,BIN][,CC][,FBCD]-----
   -[,(NFIL,{n}) ] [,ASCI][,ASCO]
          {ALL}
```

where

FILE indicates either a disk file or a whole disk area. Areas CK and XA are only allowed as input files.

LIB indicates a library object module(s) in the SP or FP area.

IN indicates an input operation from a non-disk device is to be performed.

OUT indicates an output operation to a non-disk device is to be performed.

VFC indicates vertical format control is desired on printing.

ADD indicates records are to be added to the end of an already existing file.

†File-to-file copies should be between files with the same RSIZE.

UPD indicates records are to be added starting at the current position of the file. Normally, the file would be rewound before the copy started.

BIN specifies that nonstandard binary information is to be copied from the card reader or to the card punch.

CC specifies that control commands are to be copied from the C device.

FBCD specifies that BCD input is to be converted to EBCDIC.

(NFIL,{n / ALL}) specifies that multiple files are to be copied. If n is specified, n files will be copied; if ALL is specified, the copy will continue until a double EOF is read. The default is to copy one file.

ASCI specifies that ASCII input is to be converted to EBCDIC.

ASCO specifies that EBCDIC output is to be converted to ASCII.

The following are examples and explanations of the different types of copies that can be performed.

Examples:

```
:COPY (IN, {op / yyndd}) ,(FILE,fid)
```

This example copies a file of data onto the specified disk file.

```
:COPY (IN, {op / yyndd}) ,(FILE,fid),CC
```

This example copies a file of data containing control commands from the C device onto the specified disk file.

```
:COPY (IN, {op / yyndd}) ,(FILE,fid),ADD
```

This example adds data to the end of an already existing disk file.

```
:COPY (IN, {op / yyndd}) ,(LIB,aa)
```

This example copies the library object modules to the specified library. The library being copied will completely replace an already existing library.

```
:COPY (IN, {op / yyndd}) (LIB,aa),ADD
```

This example adds the library object modules to the specified library.

```
:COPY (FILE,fid),(FILE,fid)
```

This example copies the contents of the first specified disk file to the second specified disk file.

```
:COPY (LIB,aa,nnnnnnnn),(OUT, op / yyndd)
```

This example copies one library object module to the specified output device. The "nnnnnnnn" parameter is the name of the library object module to be copied.

```
:COPY (FILE,fid),(OUT,{op / yyndd}),VFC
```

This example lists the contents of an EBCDIC file with vertical format control.

```
:COPY (FILE,fid),(OUT,{op / yyndd})
```

This example copies the contents of the specified disk file onto the specified device.

```
:COPY (FILE,fid),(OUT,{op / yyndd}),BIN
```

This example copies nonstandard binary from the specified disk file to the card punch.

```
:COPY (FILE,aa),(OUT,{op / yyndd})
```

This example copies the contents of the IOEX access (XA) or Checkpoint (CK) areas to the specified output device.

:DPCOPY The :DPCOPY command copies the contents of one disk pack to another disk pack. The entire contents of the disk is copied (except for the alternate track area) unless otherwise specified.

The :DPCOPY command has the form

```
:DPCOPY (yyndd, yyndd) [,CKRD][,CKWT],
   [,(SSEC,n)][,(NSEC,m)][,(OSEC,p)]
```

where

yyndd are the physical device names of the disk storage devices. The first disk is the input device and will be copied to the second device.

CKRD indicates that read-check will be done when reading the input device.

CKWT indicates that a write-check will be done when writing the output device.

(SSEC,n) specifies to begin the copy with sector n. The default is to start at sector 0. Note that SSEC can apply to the input and output disk.

(NSEC,m) specifies that m sectors should be copied. The default is to copy through the last user accessable sector.

(OSEC,p) specifies that the output is to start at sector p. If this parameter is not given, the SSEC value is used. If it is used, p + m must be less than or equal to the last sector on the disk.

:DELETE The :DELETE command deletes either a file directory entry and file from a permanent disk area, or an object module from the designated library. The space formerly allocated is not used until a :SQUEEZE is executed.

The forms of the :DELETE command are

```
:DELETE {LIB,aa,nnnnnnnn}
        {FILE,fid       }
```

Examples:

1. Delete a file:

```
:DELETE (FILE,BP,TESTA)
```

2. Delete an object module:

```
:DELETE (LIB,SP,CSCN)
```

This example specifies that an object module named CSCN is to be deleted from the Library in the SP area.

:CLEAR The :CLEAR command zeros out the specified disk areas which results in deleting all files and file directories in the area.

The form of the command is

```
:CLEAR zz,zz,. . .
```

where zz is any disk area.

Example:

```
:CLEAR D1,DF
```

This example specifies that permanent areas D1 and DF are to be zeroed out.

:SQUEEZE The :SQUEEZE command regains unused space within permanent disk areas resulting from file deletions and truncations and library module deletions. Unused space is regained by compressing file directory entries and their associated files, or library file entries and their associated library modules. Within the libraries, the Module Directory File (MODIR) and the Module File (MODULE) entries and modules are compressed to regain the unused space. Space is regained in the remaining two files, EBCDIC File (EBCDIC) and DEFREF File (DEFREF), by regenerating them completely from the Module Directory and Module Files. Extreme care should be exercized to ensure there is no file activity going on in the area being squeezed.

The forms of the command are

1.
```
:SQUEEZE aa,aa,aa,. . .
```

2.
```
:SQUEEZE ALL
```

3.
```
:SQUEEZE (LIB,aa)
```

Examples:

1. Regain unused space in specified area:

```
:SQUEEZE SP
```

This example regains unused space between files the SP area only, without affecting the internal structure of the SP Library.

2. Regain space in all permanent disk areas.

```
:SQUEEZE ALL
```

This example regains unused space between all files in all permanent areas. Libraries are not squeezed.

3.

```
:SQUEEZE (LIB,SP)
```

This example regains unused library space in the SP Library.

:TRUNCATE The :TRUNCATE command is used to truncate empty space from the end of specified file(s). If the allocated disk space for a file is greater than the actual length of the file, a considerable amount of space may be left empty. This command will set the allocated space equal to the actual length of the file.

The forms of the command are

1.

```
:TRUNCATE (FILE, fid), (FILE, fid)..., (FILE, fid)
```

2.

```
:TRUNCATE aa, aa, aa, ...
```

Examples:

1. Truncate allocated file:

```
:TRUNCATE (FILE, BP, TEST)
```

This example truncates empty space from the end of the allocated file TEST in the BP area by setting the allocated size equal to the actual size of the file.

2. Truncate all files:

```
:TRUNCATE BP, D2, D3
```

This example truncates all files in the BP, D2, and D3 areas.

:MAP The :MAP command maps the specified permanent disk areas to the LO device (using the M:LO DCB). The map contains

1. Information concerning the area, consisting of the disk address, write protection, area identification, words per sector, sectors per track, and its beginning and ending disk addresses.

2. Information from the Permanent File Directories concerning each file in the area; file name, format, beginning file address, ending file address, file size, record size, granule size, and resident foreground program indicator.

The forms of the command are

1.

```
:MAP aa,aa,aa,...
```

2.

```
:MAP ALL
```

Examples:

1. Map specified permanent disk areas:

```
:MAP BP,D4
```

This example outputs a map of the permanent areas BP and D4 of the LO device.

2. Map all permanent disk areas:

```
:MAP ALL
```

This example outputs a map of all permanent areas to the LO device.

An example of a MAP SP output is illustrated in Figure 26.

:SMAP The :SMAP command produces a special, short map of the files in the specified areas. The command has

```
AREA   DEVICE=   #WORDS/   SECTORS/   BEGIN     END       WRITE
       ADDRESS   SECTOR    TRACK      SECTOR    SECTOR    PROTECT
 SP    0CCF0       256        11          1      2815        S


                                FLGS  AREA RELATIVE    GRANULE   RECORD   FILE     APPROX   EXTEND
FILENAME,ACCOUNT  ATMT                BEGIN  END        SIZE      SIZE     SIZE     RECORDS  SIZE
                          ORG         SECTOR SECTOR     (BYTES)   (BYTES)  (RECS)   RFMAT    (SECTR)
CPWFILE                   0             1    256       1024      1024      256       0
TFI                       0           259    260       1024      1024        2       0
LOAD                      0           261    272       1024      1024       12       0
JCP                       0           273    288       1024      1024       16       0
CPRMAP                    0           289    318       1024        80      550      10
RADBOOT                   0           319    319       1024      1024        1       0
CPR                       0     S     320    444       1024        80     1242     256
OLOAD                     0     S     445    504       1024      1024       60       0
EDIT                      0     S     505    550       1024      1024       26       0
ANALYZE                   0     S     551    606       1024      1024       76       0
AP                        0     S     607    675       1024      1024       69       0
SYSTODEF                  0     S     676    685       1024      1024       10       0
FREFILE             0     0 F   S     686    690       1024        64       15      85       2
CRASH               0     0 F         691    700       1024      1024        0      10      512
RAFEDIT                   0     S     701    756       1024      1024       48       2

NUMBER OF FILES:          15

REMAINING SECTORS:      2060
SECTORS UNCOVERABLE:       0
```

Figure 26. MAP SP Output Example

the same format as the :MAP command. The output on the LO device (using the M:LO DCB) for each area specified consists of

1. The name of the area.
2. The name of each file, its account, and number of records; or, if there are no files, the message "AREA CONTAINS NO FILES".

:LMAP The :LMAP command maps the library files of the specified permanent directory to the LO device using the M:LO DCB. For each area (SP or FP), all library files in the area are mapped. The library map includes information about the object modules in the library files that consists of the name of each module, its relocatable length, and definition and references in the module.

The form of the command is

```
:LMAP aa[,aa)]
```

where aa specifies the SP or FP area.

Example:

```
:LMAP SP
```

This example specifies the files in the SP area and its associated library are mapped.

:CATALOG The :CATALOG command is used to get file size and organization information either about a particular file or about a group of files in a particular account and/or area. The two forms of the command determine the type of processing.

Form 1: a particular file.

```
:CATALOG (fid)[,(fid),...]
```

where fid is the standard file identifier.

This form will display the file information for the named files.

Form 2: for a group of files.

```
:CATALOG (fid')[,([f][,t])]
```

where

fid' is a standard fid with a null file name (e.g., .SP or .ACCOUNT or .).

f is a starting sort key. Its format is the same as the 'name' field of a fid. It need not be an actual file name.

t is an ending sort key in the same format as f above.

This form will display all files in the area(s) and account specified. If an area is given explicitly, only that area will be displayed. If an account is given explicitly, only files in that account will be displayed. The account is defaulted to the user's account if it is not specified.

The information displayed consists of the filename, the account name, the area the file is in, its organization, and the number of records in the file. The output is sorted alphabetically by file name and account name.

The "f" and "t" parameters allow the list of files to be limited to a particular range. Neither "f" nor "t" must be actual file names. If "f" ("t") is given, files with names alphabetically before (after) it will not be displayed. If "f" ("t") is not given, the first (last) possible name is used as the default.

:DUMP The :DUMP command dumps, in hexadecimal, the designated random or sequential access file onto the LO device (using the M:LO DCB). All permanent areas plus the IOEX Access area (XA), Background Temp area (BT), and Checkpoint area (CK) can be dumped. The RADEDIT will sequentially access the designated file or area to be dumped. Files are dumped by records, areas by sectors. The EBCDIC is not given.

The forms of the command are

1.

```
:DUMP (FILE,fid)[,(SREC,value)][,(EREC,value)]
```

where

fid Is any allotted file.

SREC,value specifies the starting record (in decimal) to begin the dump.

EREC,value specifies the last record to be dumped.

2.

```
:DUMP zz[,(SREC,value)] [,(EREC,value)]
```

where

zz is any disk area.

SREC,value specifies the starting sector (in decimal) to begin the dump.

EREC,value specifies the last sector to be dumped.

Examples:

1. Dump specified file:

```
:DUMP (FILE,BP,TEST)
```

This example specifies that the TEST file in the BP area is to be dumped onto the LO device.

2. Dump specified records:

```
:DUMP (FILE,BP,TEST),(SREC,10),(EREC,20)
```

This example specifies that records 10 through 20 of the TEST file in the BP area are to be dumped onto the LO device.

3. Dump specified sectors:

```
:DUMP BP,(SREC,6),(EREC,9)
```

This example specifies that sectors 6 through 9 of the BP area are to be dumped onto the LO device.

4. Dump all of specified disk area:

```
:DUMP BP
```

This example specifies that all of the BP area is to be dumped onto the LO device.

:XDMP The :XDMP command dumps the specified input onto the LO device (using the M:LO DCB) in hexadecimal and EBCDIC. The input is read and edited according to the device specified: cards for a card reader; sectors for a disk; blocks for a tape.

The form of the command is

:XDMP(yyndd | .zz | fid | op) [, (FILE, value)][, (FROM, value)]

[, (TO, value)]

where

yyndd is any valid input device.

.zz is any area. The period is necessary to distinguish it as an area, not a file.

fid is any allotted file.

op is any oplabel which points to any of the above.

FILE, value applies only to tape input devices and specifies the file in which the edit will start. :XDMP always assumes it is positioned at the beginning of file 1 and will skip value-1 tape marks to position to the correct file. The default is 1.

FROM, value specifies the number of the first record (card, sector or tape block within the specified FILE) to be :XDMPed. The default is 0 for disk devices, 1 for all other cases.

TO, value specifies the number of the last record within the specified FILE to be :XDMPed. The default depends on what is being :XDMPed: for a file it is EOT; for an area, EOA; for a disk device, the last sector; for other devices, the 524,287th record.

The processing of the TO parameter, whether specified or defaulted, depends on the FILE parameter. If a FILE number is specified, TO refers to records within that file only, and :XDMP will not process records beyond the tape mark (EOF) for that file (that is, the maximum value of TO is forced to the number of records in the specified file).

When FILE is not specified, TO specifies the last record to process, without regard to single tapemarks (EOFs). If double tapemarks (EOFs) are not found, TO-FROM+1 records will be :XDMPed. Processing will never go beyond double tapemarks.

Examples:

1.
```
:XDMP 9TA80
```

The tape on 9TA80 will be :XDMPed from its current position to the double EOFs.

2.
```
:XDMP TO TO,5
```

The device or file addressed by oplabel TO will have the next 5 records :XDMPed.

3.
```
:XDMP TESTFILE.D3 TO,10 FROM,2
```

File TESTFILE in area D3 will be :XDMPed from sector 2 through 10.

4.
```
:XDMP DPAF0 TO,4
```

Disk device DPAF0 will have its first five sectors :XDMPed.

:SAVE The :SAVE command saves the specified disk area(s) on the BO device (using the M:BO DCB) for subsequent restoration. The BO device must be a magnetic-tape device. The image of the designated area(s) and the CP-R bootstrap are written on BO in self-reloadable format. The BO output contains a bootstrap loader, followed by the disk image of the CP-R bootstrap, and the designated area(s). Sectors containing all zeros are suppressed. Executing the bootstrap loader causes the disk image to be read into memory and restored to the disk(s) without CP-R control. The bootstrap also types on TYA01 the date that the save was made, and any identifying text string included in the :SAVE command. The BO output can also be used to restore the disk via the :RESTORE command. The BO device is rewound and the data saved is verified. The message

'SAVE TAPE OK'

is output and the tape rewound off-line (unloaded) if the verification pass completes successfully.

The form of the command is

```
:SAVE {[ALL,]zz,zz,...} [,'tape id']
      {ALL             }
```

where

ALL specifies all allocated areas except BT, CK, IS and OS.

zz can be any disk area.

'tape id' is a message of up to 40 characters to identify the save tape. When the tape bootstrap is executed this message will be output to the operators console (TYA01).

Examples:

1. Save specified areas on secondary storage:

```
:SAVE SP,BP,D2,'SAVE BACKGROUND PROGS-
  -AND DATA'
```

This example specifies that the SP, BP, and D2 areas, with a preceding bootstrap and an identifying text string, are to be saved on the BO device for subsequent reloading.

2. Save all areas on secondary storage:

```
:SAVE ALL
```

This example specifies that all disk areas except BT, CK, IS or OS, with a preceding bootstrap, are to be saved on the BO device for subsequent reloading.

:RESTORE The :RESTORE command restores the specified permanent disk areas that were saved by the :SAVE command.† Input is read from the BI device (using the M:BI DCB), and the bootstrap is ignored. A CHKWRT is employed to verify the data restored.

The form of the command is

```
:RESTORE zz,zz,...
```

Example:

```
:RESTORE SP,BP,D2
```

This example specifies that the areas SP, BP, and D2 (previously saved with a :SAVE directive) are to be restored.

:BDSECTOR The :BDSECTOR command specifies a group of sectors on a disk that are not to be used by RADEDIT. Sector 0 of an area cannot be specified as it contains the first directory sector for that area.

The form of the command is

```
:BDSECTOR yyndd,{number-number}[number-number]...
               {number       }[number       ]
```

Example:

```
:BDSECTOR DPAF0,300-311,401
```

This example specifies that RADEDIT use of sectors 300 through 311 and sector 401 of the disk on DPAF0 is to be inhibited.

†RESTORE must be performed on the same device type from which SAVE was performed, i.e., the device must have the same sector size and same number of sectors per track.

:GDSECTOR The :GDSECTOR command specifies that sectors previously inhibited for use by a :BDSECTOR command are to be made available for subsequent use. The disk and sectors must be the same as in a previous :BDSECTOR.

The form of the command is

```
:GDSECTOR yyndd,{number-number}[number-number]...
               {number       }[number       ]
```

Example:

```
:GDSECTOR DPAF0,401,306-311
```

This example specifies that the previously inhibited sectors 306 through 311 and sector 401 are to be made available.

:PFIL, :PREC, :REWIND, :SFIL, :UNLOAD, are identical to the corresponding JCP control commands (see Chapter 2), except for the leading colon. They permit tape positioning to be done without leaving RADEDIT.

:END The :END command may be used to terminate the RADEDIT processor. The form of the command is

```
:END
```

ERROR MESSAGES

The error messages output by the RADEDIT and their meanings are given in Table 20.

RADEDIT outputs error messages on the OC and LL devices. If OC and LL are assigned to the same device, duplication of messages on LL is suppressed. If an operator response is required, RADEDIT will call the "WAIT" routine. If the background is operating in the "attend" mode, the operator will be allowed to initiate a console interrupt and key in one of the following three commands:

C continue and read next record from the C device.

X abort RADEDIT and return control to the system.

COC continue and read a record from the OC device (used only in conjunction with the error message "ERROR ITEM xx").

Otherwise, RADEDIT will take the action indicated in Table 20.

If RADEDIT aborts because of an irrecoverable I/O error, the physical device name is included in the abort message.

DISK RESTORATION MESSAGES

The messages itemized in Table 21 are written on the keyboard/printer during disk restoration via the bootstrap loader produced by SAVE. Unless otherwise specified, the computer will go into a WAIT after writing a message.

Table 20. RADEDIT Error Messages

Message	Meaning	Action Taken
AREA xx CANNOT CONTAIN A RESIDENT FOREGROUND PROGRAM	Illegal area specified. Only the FP area can contain a resident foreground program.	Operation is aborted.
AREA xx CKSM ERROR	A checksum error exists on the SAVE tape in the specified area.	Operation is aborted.
AREA xx CONTAINS NO FILES	Specified area contains no files.	RADEDIT continues.
AREA xx INCOMPATIBILITY	Attempting to restore specified area onto a different type of disk from which it was saved, or the area to be restored is too large for the same area using the current Master Directory.	Operation is aborted.

Table 20. RADEDIT Error Messages (cont.)

Message	Meaning	Action Taken
AREA xx IS NOT ALLOCATED	Specified area was not allocated at SYSGEN.	Operation is aborted.
AREA xx DOES NOT CONTAIN A LIBRARY	An area other than SP or FP was specified that does not contain a library.	Operation is aborted.
AREA xx IS NOT MAINTAINED BY RADEDIT	An attempt has been made to use area CK, XA, or BT which is not maintained by RADEDIT.	Operation is aborted.
AREA xx TRUNCATED	Specified area being restored is larger than the same area using the current Master Directory, but the data that was lost contained all zeros.	Operation continues.
BUFFER SMALLER THAN DATA READ	Data read exceeds the amount of available buffer space.	Operation is aborted.
CKSM ERR ON SAVE TAPE	A checksum error has been encountered while verifying the disk SAVE tape.	Operation is aborted.
CKSM ERROR	Last record in the object module being read has a checksum error.	If the operator response is C, RADEDIT reads the next record from the specified device.
DUPLICATE DEF xxxxxxxx	Relocatable Object Module being copied to the library contains duplicate definitions.	RADEDIT skips to the end of the module. A key-in of C causes RADEDIT to read the next record from the specified device.
DUPLICATE FILE	An attempt has been made to allocate a file using a name which already exists.	Operation is aborted.
EOT on {yyndd / area, name / OP, = {yyndd / area, name}}	Unexpected end-of-tape was encountered on the specified device or file.	Operation is aborted.
ERROR ITEM xx	Item number xx on the command is in error.	If the operator response is C, RADEDIT reads the next record from the C device. If the operator response is COC, the next record is read from the OC device. This will enable operator to rectify a directive error.
ILLEGAL BINARY RECORD	An illegal binary record (first byte not X'1C', X'3C') has been read with an object module.	If the operator response is C, the RADEDIT reads the next record from the specified device.
ILLEGAL FILE NAME	An attempt has been made to allocate a file using GO, OV, or X1-X9 as a file name.	Operation is aborted.
ILLEGAL LOAD ITEM xx	Relocatable Object Module to the library contains an illegal load item.	RADEDIT skips to the end of the module. A key-in of C causes RADEDIT to read the next record from the specified device.

Table 20. RADEDIT Error Messages (cont.)

Message	Meaning	Action Taken
ILLEGAL OPTION xxx	Option specified is not permitted on a :COPY command.	Operation is aborted.
ILLEGAL USE OF :COPY	The specified combination of input and output devices on the :COPY command is prohibited.	Operation is aborted.
INVALID RSIZE. UNBLOCKED ORGANIZATION GIVEN	Maximum record size for a blocked file has been exceeded. Unblocked organization given.	RADEDIT continues.
NOT ENUF BCKG SPACE	Insufficient background space to perform the requested operation.	Operation is aborted.
DISK OVERFLOW	Allocating the amount of disk storage indicated by the "FSIZ" parameter on the :ALLOT command would exceed the space available in the area specified.	Operation is aborted.
RADEDIT I/O ERROR xx AT LOC xxxx	An I/O error has occurred at the indicated location.	Operation is aborted.
WARNING: RECORD SIZES DIFFER ON INPUT AND OUTPUT FILES	Record sizes differ on copying from disk file to disk file.	Operation is continued.
REFERENCES TO F4:COM NOT ALLOWED	An external definition or reference F4:COM encountered in a Relocatable Object Module being copied to the library.	RADEDIT skips to the end of the module. A key-in of C causes RADEDIT to read the next record from the specified device.
ROM DOES NOT CONTAIN A DEF	Relocatable Object Module being copied does not contain an external definition.	A key-in of C causes RADEDIT to read the next record from the specified device.
SAVE TAPE OK	Disk SAVE tape has been verified correctly.	No action.
SEQ ERROR	Last record in the object module being read has a sequence error.	If the operator response is C, RADEDIT reads the next record from the specified device.
FILE xxxxxxxx DOES NOT EXIST	File does not exist in the specified area or account.	RADEDIT continues.
SPECIFIED ROM DOES NOT EXIST	Relocatable Object Module does not exist within the specified library.	Operation is aborted.
SREC VALUE GREATER THAN EREC VALUE	Parameter error on the :DUMP command. The last record to be dumped precedes the initial record to be dumped.	Operation is aborted.
AREA xx NOT FOUND ON SAVE TAPE	Specified area cannot be found on the disk SAVE tape during a :RESTORE operation.	Operation is continued.
ABOVE AREAS NOT RESTORED	The areas specified by the preceding messages were not on the SAVE tape.	RADEDIT aborts.

Table 20. RADEDIT Error Messages (cont.)

Message	Meaning	Action Taken
DATA IN SECTOR xxxxx [TO SECTOR xxxxx] MAY BE LOST IN AREA zz	The indicated sector(s) reported errors when being read from the disk. The data may not be correct on the SAVE tape. When the area is RESTORED, these sectors should be inspected for the repeated string 'LOSTDATA' which was used to replace the data not read.	Operation continues.
WARNING: ERRORS WRITING SAVETAPE. CHECK LISTING.	Self-explanatory.	Operation continues.
FILE xxxxxxxx DELETED	The beginning of the named file is included in the bad sector(s) and was deleted from the directory.	Operation continues.
FILE xxxxxxxx TRUNCATED	The end of the named file is included in the bad sector(s) range, and was truncated at the last good sector.	Operation continues.
yyndd WRT RESTRICTED	Specified disk is software write-protected.	Operator should take appropriate action: interrupt and key in "SYC". Or, if the job is not allowed to write on protected areas of the disk, interrupt and key in "X" to abort.

Table 21. Disk Restoration Messages

Message	Meaning	Resulting Action
CKSM ERROR	A checksum error has occurred in reading the SAVE tape.	If the WAIT condition is cleared, the bootstrap loader continues and accepts the bad record.
DISK RESTORED OK	The disk restoration has been successfully completed.	Control is transferred from the disk bootstrap.
TRK = xxxx DATA = ALL ZEROS	Specifies the contents of the disk controller address register in hexadecimal at the time of a check write error.	If the data being written contains all zeros, this information is output. If the WAIT condition is cleared, the bootstrap loader continues.
yyndd ERROR, SB = xxxx	A parity or transmission error has occurred on device yyndd. Both the device status byte and operational status byte are displayed following "SB=".	There is no recovery.

Table 21. Disk Restoration Messages (cont.)

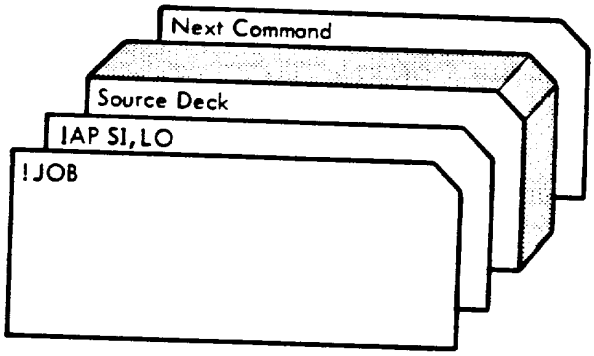
Message	Meaning	Resulting Action
yyndd UNRECOG., SB = xxxx	An unrecognized status has been returned from the indicated device. Both the device status byte and operational status byte are displayed following "SB=".	Upon clearing the WAIT condition, the operation is retried.
yyndd UNUS. END, TDV = xxxx	An unusual end status has been returned from the specified device. Both the TDV status byte and operational status byte are displayed following "SB=".	There is no recovery on a read operation. On a write operation, the write is tried again after the WAIT is cleared.
yyndd WRT PROT	The disk is write-protected.	Program will attempt the disk write after an SY key-in.

12. PREPARING THE PROGRAM DECK

The following examples show some of the ways program decks may be prepared for CP-R operation. Unless stated otherwise, standard default cases for device assignments are assumed.

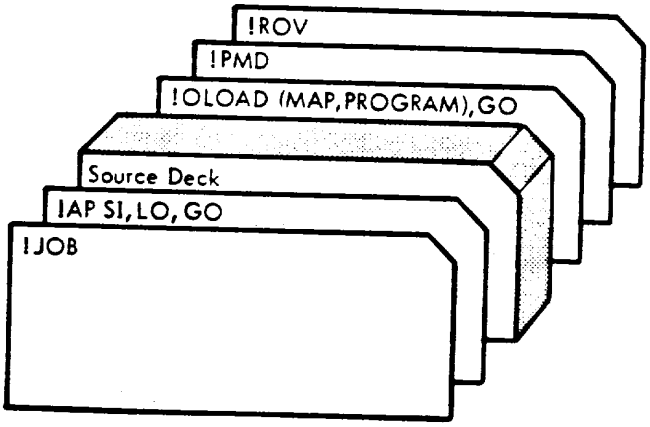
AP EXAMPLES

ASSEMBLE SOURCE PROGRAM, LISTING OUTPUT



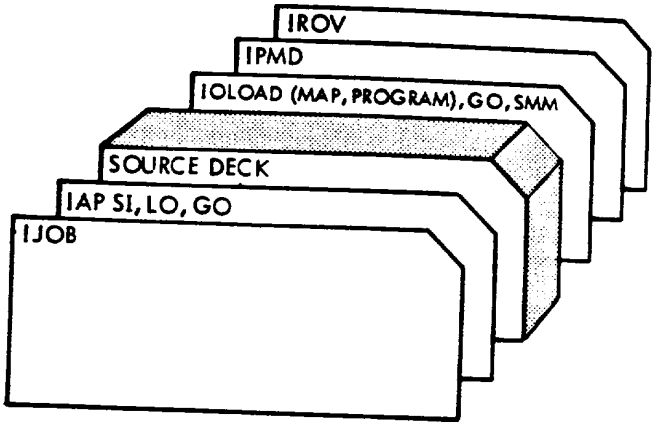
In this example, the symbolic input is received from the SI device and the listing output is produced on the LO device.

ASSEMBLE SOURCE PROGRAM, LISTING OUTPUT, LOAD AND GO OPERATIONS



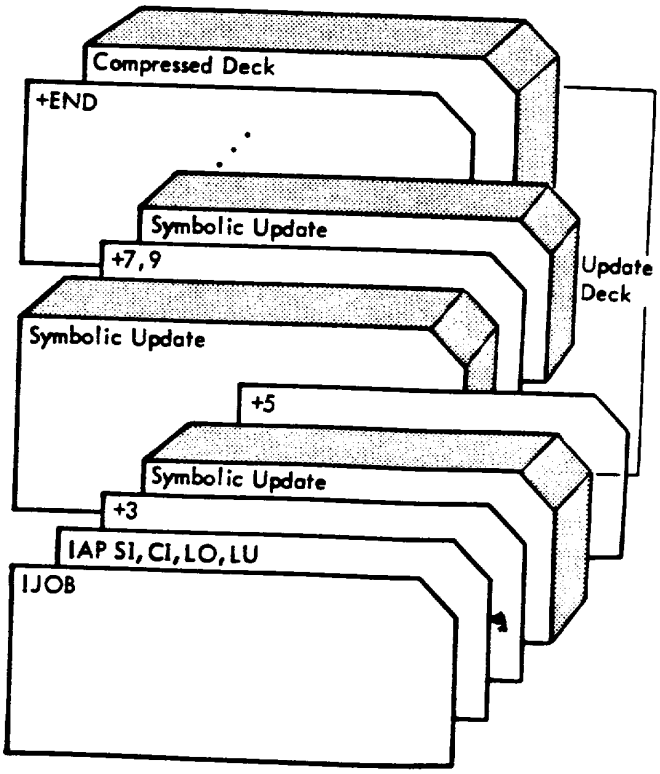
In this example, the binary object program produced from the assembly is placed in a temporary (GO) file from which it is later loaded and executed. The resultant file is always temporary and cannot be retained from one job to another. The Overlay Loader loads the program root into the OV file for execution. A postmortem dump is specified.

ASSEMBLE AND LOAD A PROGRAM WRITTEN FOR UNMAPPED (RBM) BACKGROUND EXECUTION



This example is like the previous one, except that the program either uses SEGLOAD calls, or uses memory outside its segments. The SMM (Simplified Memory Management) option on the !OLOAD command causes the program to be loaded in a manner that allows these actions, but can result in much less efficient real memory utilization.

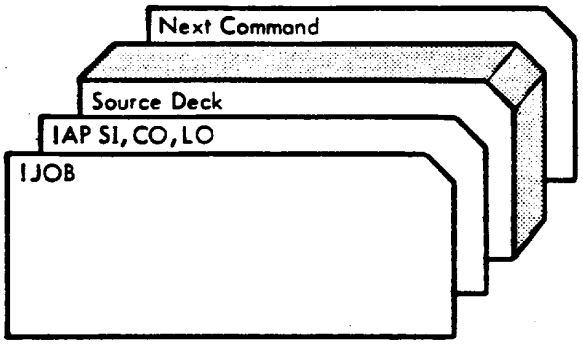
ASSEMBLE FROM COMPRESSED DECK WITH SOURCE AND UPDATES, LISTING OUTPUT



In this example, the compressed input (deck) is received from the CI device, listing output is produced on the LO device,

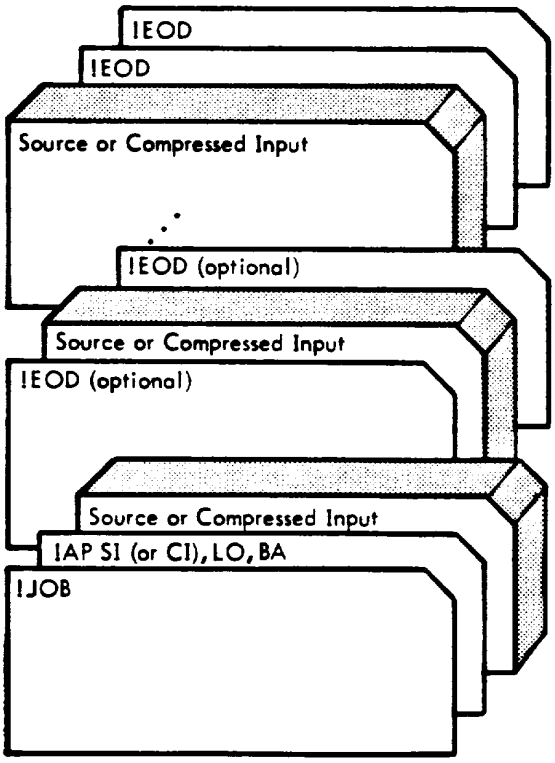
and listing of the update deck is also produced on the LO device. The update deck is enclosed in the bracket.

ASSEMBLE SOURCE PROGRAM, COMPRESSED OUTPUT ON CARDS, LISTING OUTPUT



In this example, the compressed card output is produced on the CO device.

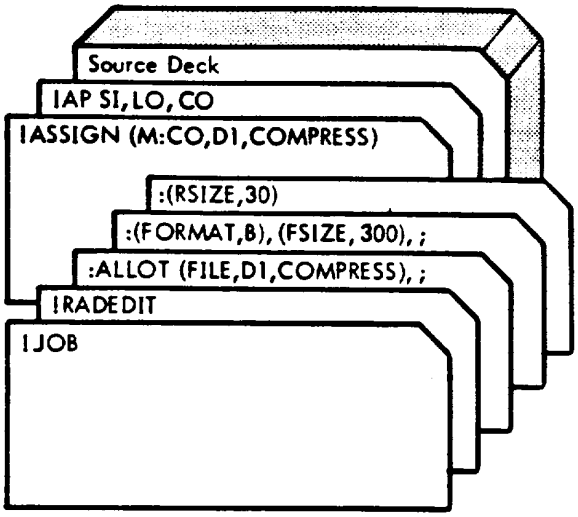
ASSEMBLE SOURCE OR COMPRESSED PROGRAM IN BATCH MODE, LISTING OUTPUT



In this example, successive assemblies are performed with a single AP command until a double EOD is read. The device assignments and options on the AP command apply to all assemblies within the batch. A program is considered terminated when an END directive is processed.

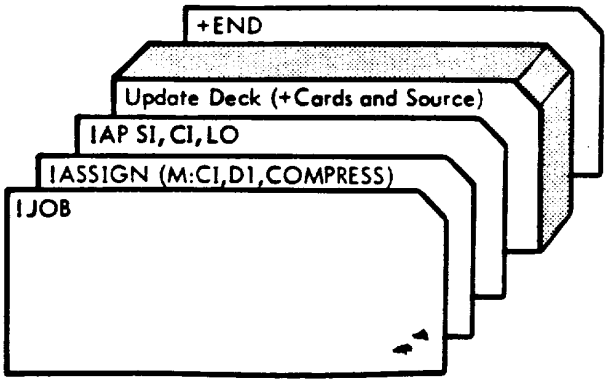
When batch assemblies consist of successive updates from card input to compressed programs from disk or tape, the updates are terminated by a +END card and should not be separated by !EOD cards. There must be a one-to-one correspondence of update packets to compressed programs. End-of-job is signaled by end-of-file conventions applied to the CI device.

ASSEMBLE SOURCE PROGRAM, COMPRESSED OUTPUT ON DISK FILE, LISTING OUTPUT



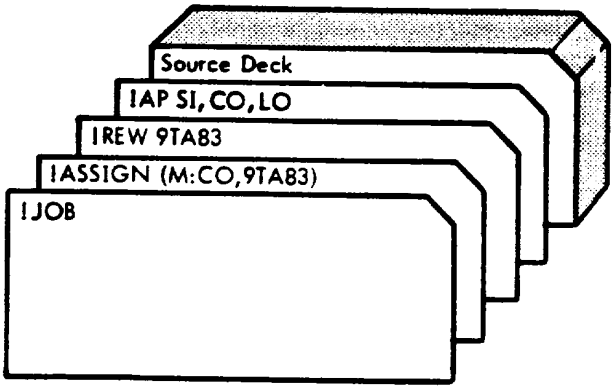
In this example, the CO device is assigned to a disk file called COMPRESS in a background data area of the disk. The compressed output is written on the COMPRESS file.

ASSEMBLE COMPRESSED DECK FROM DISK FILE, SOURCE UPDATES FROM CARDS, LISTING OUTPUT



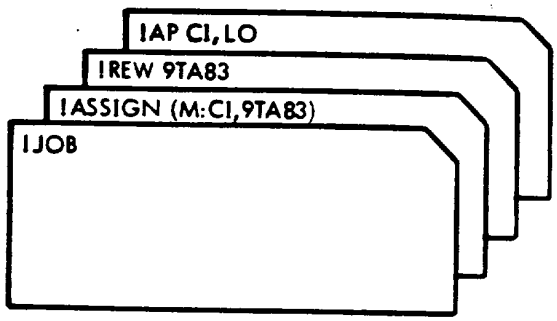
In this example, the CI (compressed input) device is assigned to the COMPRESS file in a background data area of the disk. The source update deck will be read from the SI device. In effect, this will update the assembly given in the previous example.

ASSEMBLE SOURCE PROGRAM, WRITE COMPRESSED OUTPUT ON 9-TRACK TAPE, LISTING OUTPUT



In this example, the CO device is assigned to the designated 9-track magnetic tape unit to receive the compressed output.

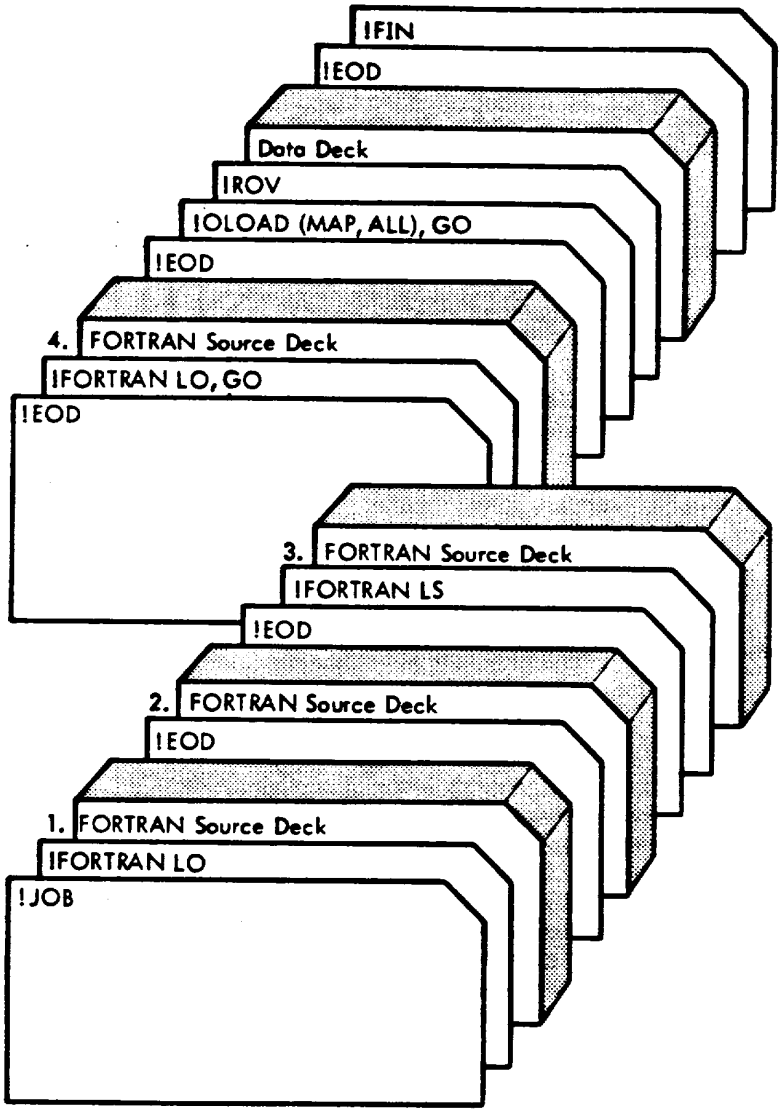
ASSEMBLE COMPRESSED PROGRAM FROM 9-TRACK TAPE, LISTING OUTPUT



In this example, the CI device is assigned to the designated magnetic tape to read the compressed input to be assembled. This is the next logical job step to follow the previous example.

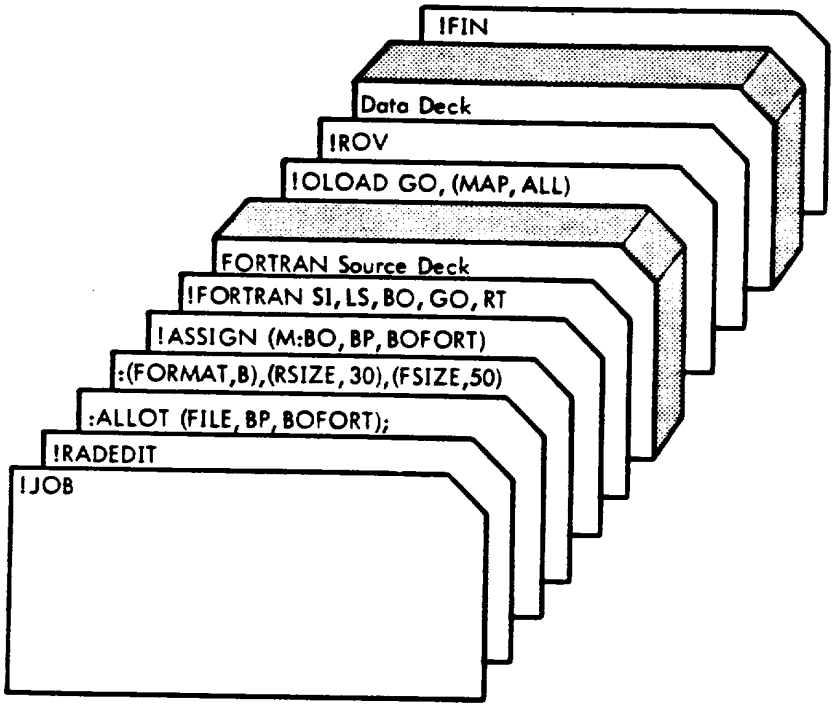
FORTRAN JOB EXAMPLES

COMBINED FORTRAN COMPILATIONS, PLUS FORTRAN COMPILE AND EXECUTE



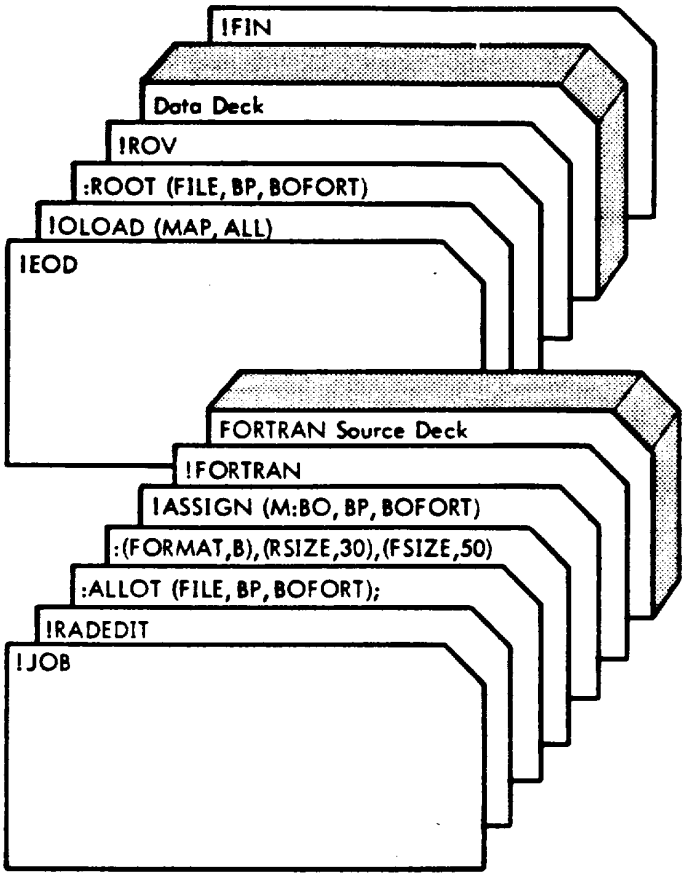
In this example, the first two source decks are compiled with mixed (source and object language) listed output. The next source program (3) is compiled with a source listing (only) being output. The final source deck (4) will compile with the object module being output on both the BO and GO files. The Loader inputs the object module from the GO file to form the Root and outputs the executable program to the OV file. The program (called OV) is executed via the !ROV command.

COMPILE AND EXECUTE FORTRAN SOURCE PROGRAM WITH REAL-TIME LINKAGES



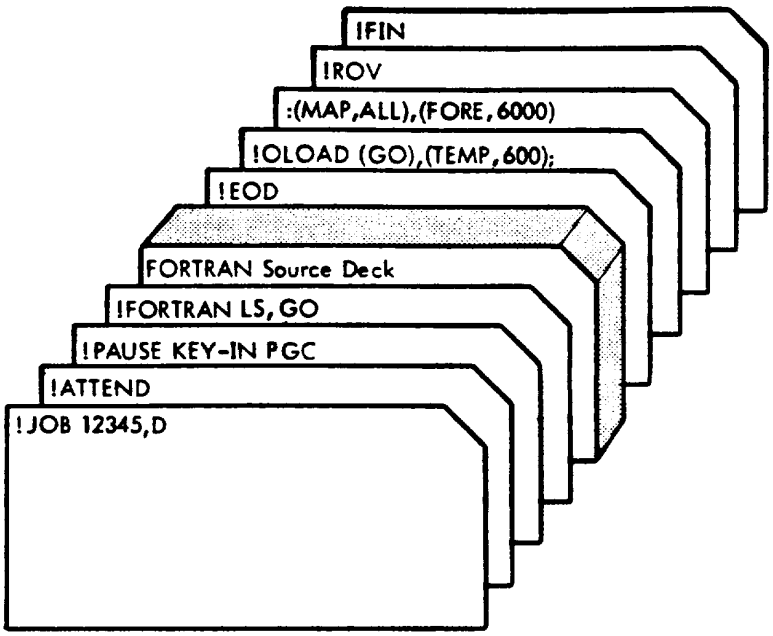
In this example, RADEDIT allots a file called BOFORT in binary format to the Background Program area of the disk. The specified record size is 30 words; the file size is 50 records. The IASSIGN command assigns the M:BO DCB (binary output) to file BOFORT. The IFORTRAN command specifies that symbolic input is to be read from the SI device, the source input is to be listed (LS would be redundant if LO was specified in addition to LS), and the binary output is to be written out on both the BO and GO files. The relocatable binary output on the file "BOFORT" may be used as input to the Overlay Loader at a later date. Since the real-time option (RT) is specified, the linkage will be set up for real-time subroutines. The Overlay Loader (!OLOAD) reads input from the GO file, outputs an ALL map, searches the System Library (by default), and writes the executable program into the OV file. The program is run via the IROV (Run OV) command. Note that a program with real-time linkages can also run in the background in nonreal-time mode.

COMPILE AND EXECUTE PROGRAM USING LS, BO DEFAULT OPTIONS



In this example, the LS (list source) and BO (binary output) are assumed by default on the !FORTRAN command. The System Library is searched via the default option on the !OLOAD command.

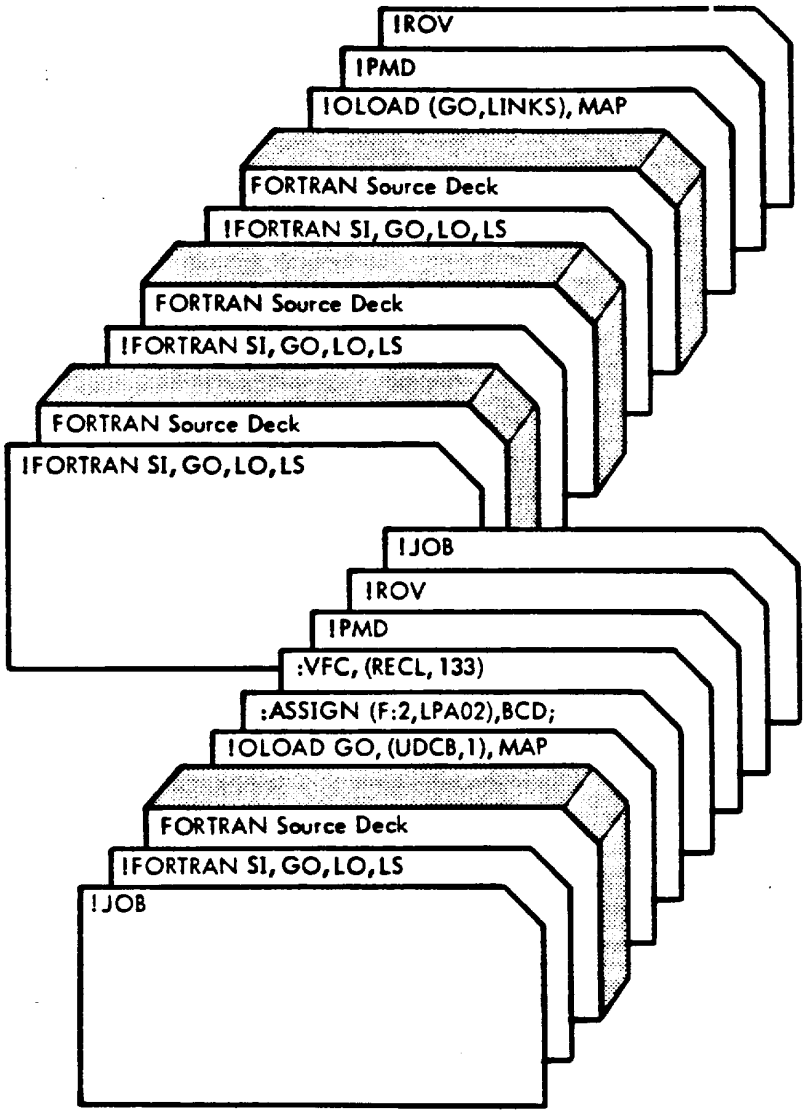
COMPILE A FORTRAN PROGRAM AND SETUP FOR EXECUTION IN FOREGROUND AREA



In this example, the !ATTEND command will inhibit the system ABORT routine so that corrective action can be taken at the console in case of error. The FORE option on the !OLOAD command gives the FWA of the program in the Foreground area of memory.

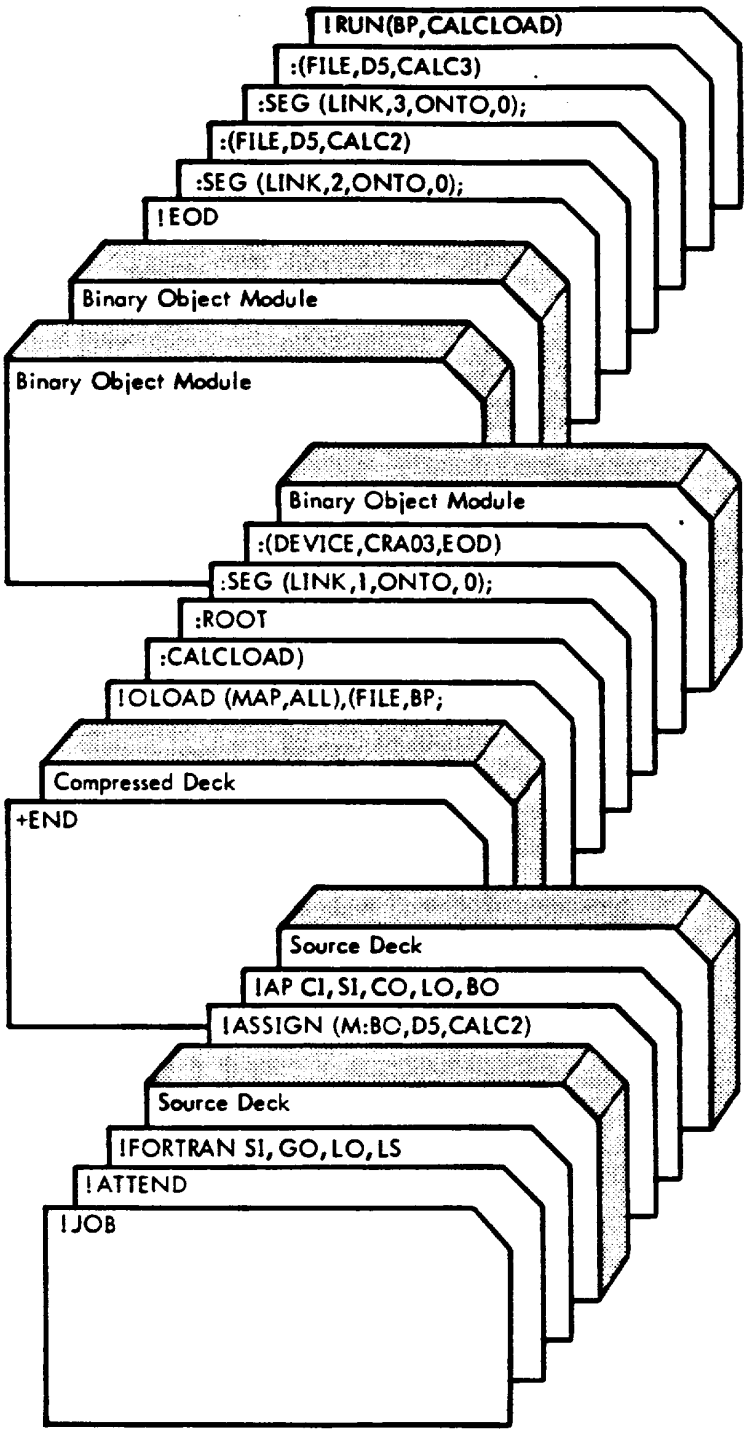
OVERLAY LOADER EXAMPLES

BATCH, USING GO LINKS



In this example, the GO file is rewound by the initial !JOB command for the first FORTRAN compilation. The Overlay Loader loads from the GO file to form a root and outputs on the OV file for execution. A SHORT map will be output. A postmortem dump is requested if the background aborts. The next !JOB command rewinds the GO file and three FORTRAN jobs are compiled, with the binary object modules output on GO to form ROM1,ROM2,ROM3. The Overlay Loader loads the first ROM for the root, the second ROM for segment 1, and the third ROM for segment 2. Note that :SEG cards are not required. The programs are executed from the OV file. A SHORT map is output. A postmortem dump is specified in case an abort occurs.

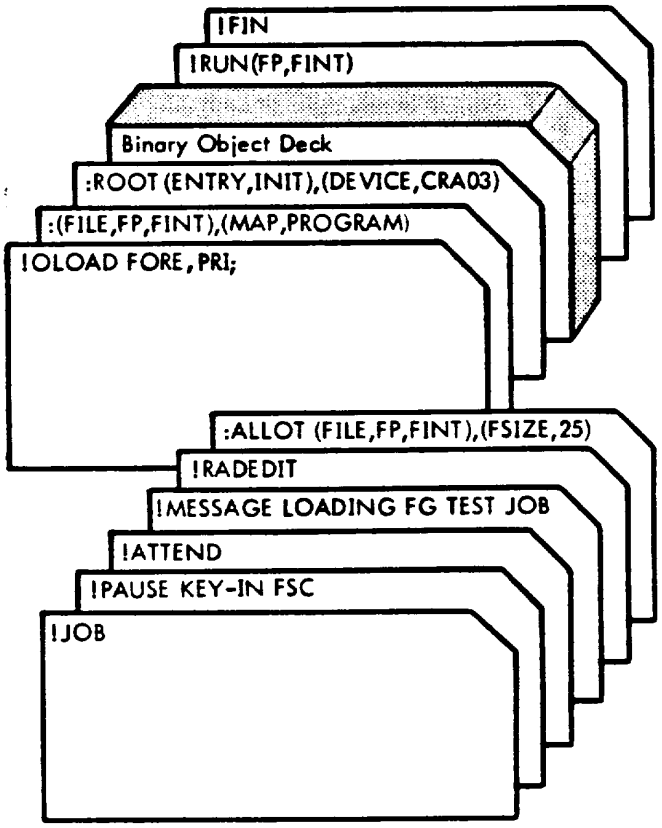
SEGMENTED BACKGROUND JOB



In this example, the JOB card rewinds the GO file, the FORTRAN source deck is compiled, and the binary object module is output on GO. The compressed source deck is updated and the binary object module is output to file CALC2 in the D5 area (previously allocated by RADEDIT). The ROMs designated on the :ROOT and :SEG commands are loaded, and the loaded program is output to CALCLOAD in the BP area. The :ROOT command causes the ROM created by FORTRAN to be loaded from the GO file and creates the Root. The ROMs following the first :SEG command are loaded until !EOD is encountered and segment 1 is then created. The next :SEG command loads the ROM assembled by Macro-Symbol on the CALC2 file in the D5 area and creates segment 2. The last :SEG command loads one ROM from the CALC3 file in the D5 area (ROM previously created by an assembly or compilation). The !RUN command executes the loaded segmented program.

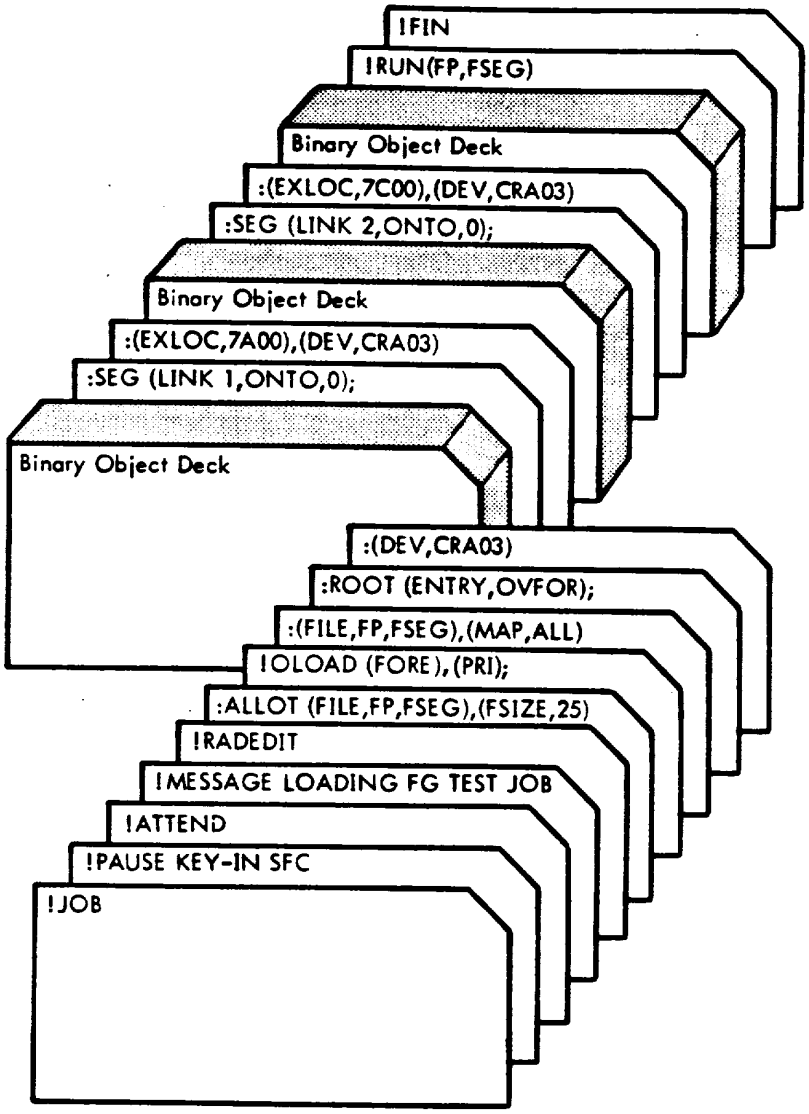
FOREGROUND JOB EXAMPLES

LOAD AND EXECUTE FOREGROUND PROGRAM



In this example, RADEDIT allots a file, (FINT) in the Foreground Programs (FP) area of the disk. The Overlay Loader loads the binary object deck in the file FINT in core image format. The !RUN control command causes execution of the foreground program as a primary task (PRI on OLOAD command) in the CPR job. A PROGRAM map is specified.

LOAD AND EXECUTE SEGMENTED FOREGROUND PROGRAM



In this example, RADEDIT allots space for a file called FSEG in the Foreground Program (FP) area of the disk. The Overlay Loader loads a root and two segments into FSEG in core image format. The overlaid program is executed (as a primary task) via the !RUN control command. An ALL map is requested.

LOAD TWO FOREGROUND SECONDARY PROGRAMS SHARING A READ-AND-EXECUTE PROCEDURE

```
:(SHA,JOB),(ACC,RX),(EXLOC,A000)
:SEG (LIN,1),(FIL,D1,CURVEFIT):
:ROOT (FIL,D1,FP2BI)
!OLOAD MAP,FORE,SEC,(FILE,FP,FP2)
:(SHARE,JOB),(ACCESS,RX),(EXLOC,A000)
:SEG (LINK,1),(FIL,D1,CURVEFIT);
:ROOT (FIL,D1,FP1BI)
!OLOAD MAP,FORE,SEC,(FILE,FP,FP1)
!JOB
```

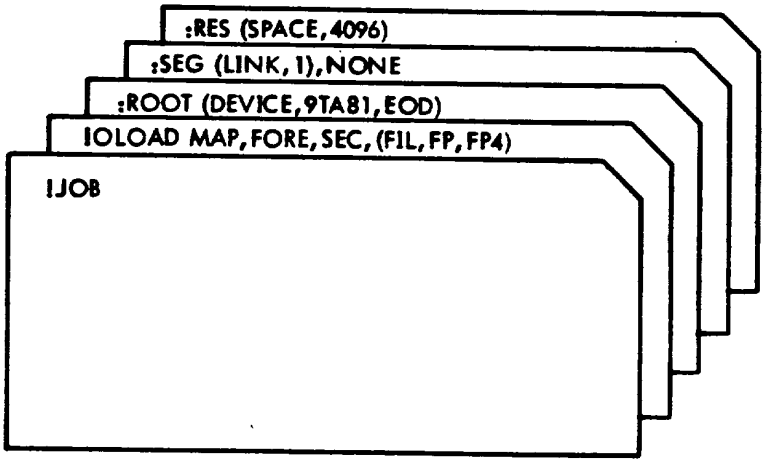
This example causes two foreground secondary programs to be loaded. All object code comes from files in the D1 area, presumed to be generated by past assemblies/compilations. Both programs load the same object code as segment number 1 (FIL,D1,CURVEFIT). This segment is restricted to read-and-execute access (ACCESS,RX). If the programs both run in the same job, and concurrently activate the segment, the same copy of the segment (i.e., the same memory map for the virtual memory concerned) will be made available to both programs (SHARE,JOB).

LOAD FOREGROUND SECONDARY PROGRAM WITH A SEGMENT FIXED IN REAL MEMORY

```
:(EXLOC, 10000),FIX
:SEG (LINK,1),(FIL,D1,FP3SEGBI):
:ROOT (FIL,D1,FP3BI)
!OLOAD MAP,FORE,SEC,(FIL,FP,FP3)
!JOB
```

Segment 1 is defined with the FIX option. This forces it to be allocated real pages so that the real address of any location in the segment is the same as its virtual address. Note that the real pages must lie in a foreground preferred memory partition, and must not be assigned for other use.

LOAD PROGRAM WITH DYNAMICALLY-ALLOCATED SEGMENT



In this example, a foreground secondary program is loaded into file FP4 in the FP area. The object code for the root is obtained from tape drive 9TA81, up to the first tape mark. Segment 1 is composed of no object code; but a DSECT of size 4096 words is reserved, and is identified by the symbol 'SPACE'. Since the size of the segment is entirely due to the :RES command, its load module image has length zero. This means that the segment may be the subject of GETPAGE and RELPAGE CALs, which dynamically alter the part of the segment for which real memory is allocated.

13. LINE EDITOR (EDIT)

INTRODUCTION

CP-R Edit is a line-at-a-time editor for on-line or batch creation, modification, and manipulation of files of EBCDIC text. Edit functions are controlled by a stream of single-line commands read through the M:SI DCB. The command language provides for the following types of operation:

1. Creating a sequenced EBCDIC file.

3. Selective printing of lines or partial lines of a file.

3. Inserting, reordering, and deleting of lines or groups of lines in a file.

4. Editing the content of lines or groups of lines, including character string insertion, replacement, shifting, and deletion.

5. Resequencing or reordering by sequence number of whole files.

6. Merging parts of several files, when assisted by certain RADEDIT functions.

Use of Edit involves two files. The subject file is the source of the data to be edited, but is not directly affected by editing. The scratch file is built from the subject file when editing is begun. It is an indexed direct-access disk file containing all of the information from the subject file, indexed by sequence number. All editing is done on the scratch file. The subject file may be rebuilt from the scratch file at certain well defined times in the editing session, or the session may be terminated without altering the subject file.

CALLING EDIT

Edit is called by being named in either the batch control stream or a TEL control stream

```
EDIT
```

SUBJECT FILE FORMAT

An Edit subject file is a file of EBCDIC data in fixed or variable length records of not more than 140 characters. It may be compressed, blocked, or unblocked. Each record is treated as one line of text.

A line sequence number may be specified in the last eight bytes of a record. If these sequence numbers are to be used by Edit, they must appear in the last eight bytes of every record. The records need not be in sequence number order and more than one record may have the same sequence number. Only the last record with a given sequence number will be retained by Edit, if it uses sequence numbers from the subject file.

SEQUENCE NUMBERS

A sequence number is a positive decimal number which is less than 10000 and a multiple of .001 (i.e., no more than three decimal places). When specified in a subject file record, a sequence number occupies the last eight bytes of the record. It may include a decimal point, decimal digits, and leading and trailing blanks, but must include at least one digit. Special processing of sequence numbers is executed if the save file sequence mode is set: When the Edit scratch file is built, if a record has a valid sequence number in its last eight bytes, it is removed. When the scratch file is saved, the last eight bytes of each saved record will be its sequence number. For fixed-length records, the last eight bytes will be overwritten by the sequence number. For variable length records up to 132 bytes, the record will be extended by the eight bytes. If the record is over 132 bytes already, bytes 133-140 will be the sequence number.

FILE IDENTIFIERS

In the commands involving a file identifier (EDIT and SAVE) the standard CP-R file identifier is used, with the restriction that it may not be of the 'area,name' format.

INPUT/OUTPUT CONVENTIONS

The input/output conventions used in Edit are independent of whether execution is batch or on-line. Commands are read through M:SI and logged through M:LL if it is assigned to a different device than M:SI. Error messages and command prompting will be written through M:LL always. Output other than error messages, command prompting, and command logging will go through M:LO. The results are:

1. The command source need not be a user's console.

2. The medium assigned to M:LL will contain a complete, non-redundant log of control activity.

3. If M:LO is assigned differently from M:LL, it will receive text output free of any control messages.

MULTILINE RECORDS

On a terminal unit having an inherent line-width limit of less than 140 (e.g., Teletype models 33, 35, and 37), a single, multiline record may be entered into a file (using the IN command, for example) in either of two ways:

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

2. Using the simulated local carriage return sequence ESC NET for the same purpose.

Either method permits entering a record of up to 139 characters plus NET on virtually any terminal unit.

An example of a multiline record is presented in Figure 27.

BREAK FUNCTION

The BREAK key always causes an immediate interruption in Edit activity, with any partially completed input being discarded and any waiting output being delivered to the terminal. Edit stops any command in progress and reverts to accepting command input from the user.

If the command in progress when an interrupt occurs is a display command (for example, TY), the display will stop within the next several lines after the interrupt is given.

For commands that produce no display while operating on a range of records, the point of interrupt is reported by a message which denotes the sequence number(s) of the record(s) being processed at the time of the interrupt. Edit then types this message

--X TO ABORT

and prompts for a single character input. If the user enters an X, the operation aborts; if he enters any other character, the operation continues.

If a command is being executed and the Break key causes an interrupt during an I/O operation (e.g., READ, WRITE, OPEN, DELETE record), the I/O operation is completed. After the I/O is completed, the user may continue execution of the command to normal conclusion or may immediately abort the command. With record or intrarecord commands (see Command Structure), the current Edit file remains open. If a file command is aborted, all files in use are closed.

ERROR RESPONSE

I/O errors other than EOF encounter cause Edit to abort. EOF on the M:SI DCB cause Edit to execute an END command. Other errors generally result in a message. In batch execution, a WAIT will then be executed (which results in an abort if unattended batch mode is in effect), followed by reading another command. In on-line execution, the WAIT is omitted.

EDIT COMMANDS

COMMAND STRUCTURE

Edit commands fall into the following three categories:

1. File commands: Commands that apply to an entire file. These commands may be given at any time.

2. Record commands: Commands that act upon one record or a group of records within a file. These commands may be given only after a file has been selected for editing.

3. Intrarecord commands: Commands that make changes within an individual record. These commands generally manipulate character strings and may be given only after a specific set of records has been selected by a command of type 2, above (either the SE, SS, or ST command).

Line number 4.000 is input as a multiline record in the following manner:

```
4.000 THIS IS AN EXAMPLE OF A MULTILINE ESC NET
RECORD. A RECORD CAN CONTAIN UP TO 140 ESC NET
CHARACTERS INCLUDING THE CARRIAGE RETURN. NET
```

If this record were displayed by Edit, it would appear as

```
4.000 THIS IS AN EXAMPLE OF A MULTILINERECORD.  A RECORD CAN CONTAIN U
P TO 140 CHARACTERS INCLUDING THE CARRIAGE RETURN.
```

Note that the user did not type a space after the word 'multiline' and that Edit did not assume a space. Also, the system "folds" the record indiscriminately when the physical line width limit is reached.

Figure 27. A Multiline Record

FILE COMMANDS

The file commands will be discussed in the following order:

EDIT	Select file for editing.
SAVE	Save the edited data.
END	Exit to executive.
SEQ	Set save file sequencing mode.
BP	Set blank preservation mode.

EDIT Select a File for Editing

The EDIT command initiates editing on a new file. Its format is:

EDIT fid1 OVER fid2 [,start[,step]]

or

EDIT fid2

where

fid1	is the file identifier for the subject file, and
fid2	is the file identifier for the scratch file.

On receiving this command, Edit will first determine if there is an edit operation already in progress. If there is, the old subject file will be rebuilt from the old scratch file and both files will be closed. Then, the new scratch file will be built from the new subject file. The sequence numbers for the lines of the file are determined by the "start" and "step" parameters. The default value for each is 1. If either is specified, the sequence number will start with "start" and be incremented by "step" for each record. If neither is specified, Edit examines the last eight bytes of the first record of the file. If this is a sequence number, the sequence number for each record will be determined by its last eight bytes, and an error will be reported if any record does not have a valid sequence number there. If the first record does not end with a sequence number, "start" and "step" of 1 will be used. If the second form is used, "fid2" must identify a previously-built scratch file or an empty file. The subject file is not defined.

If line numbers are taken from the subject file, and are not in order, they will still be indexed correctly. If more than one line has the same number, only the last one is used. When the temporary file is finally saved, records are written in line number order. However the sequence numbering is determined, if the save file sequencing mode is on, any record containing a valid sequence number in its last eight bytes will have these bytes truncated.

SAVE Save the Scratch File

The SAVE command allows the user to explicitly request that the data in the scratch file be saved. Its format is:

SAVE [ON / OVER fid]

If no options appear, the subject file (if it is defined) is rebuilt. If the "fid" (file identifier) parameter appears, the identified file is built. If "ON" appears, the file "fid" must not exist, and will be allocated. If "OVER" appears, "fid" will be allocated if necessary. If "fid" must be allocated, it will have the same structure as the subject file, with adequate size. If the file on which the save is made is estimated to be too small, two behaviors are possible: If "fid" was not specified, an error results, but the subject file is unchanged. If "fid" is specified, the save operation is attempted. In the the case of a compressed save file, the estimate is intentionally high so the save operation may succeed. If the save does not fit, an error results.

END Terminate Edit Control

The END command causes Edit to close its files and release control. By default, Edit will rebuild the subject file from the scratch file at this time, if the subject file is defined. A keyword on the END command overrides this. The command format is:

END[NS]

If "NS" appears, the scratch file will not be saved over the subject file. If "NS" does not appear, but the subject file is estimated to be too small for a complete save, an error results and the subject file is unchanged.

SEQ Set Save File Sequencing Mode

The SEQ command sets the save file sequencing mode on or off. The SEQ command has the following format:

*SEQ {ON / OFF}

When "ON", a save operation (resulting from an EDIT, SAVE, or END command) will result in the sequence number of each record being written in its last eight bytes. An EDIT command will result in truncation of the last eight bytes of any record which contains a valid sequence number there. When "OFF", all records are acquired and saved without alteration. The initial mode is "ON".

BP Set Blank Preservation Mode

BP sets the blank preservation mode ON or OFF. The BP command has the format shown below.

*BP [{ON / OFF}]

When "ON", all strings of blanks are preserved during intra-record operations. When "OFF", blank strings are compressed to a single blank or expanded as required to retain column alignment of nonblank fields. The default mode is "OFF".

When a string is inserted or replaced in a manner that changes the number of characters in a record, the record format is adjusted as follows.

When "ON", all strings of blanks are preserved during intrarecord operations. When "OFF", blank strings are compressed to a single blank or expanded as required to retain column alignment of nonblank fields. The default mode is "OFF".

When a string is inserted or replaced in a manner that changes the number of characters in a record, the record format is adjusted as follows.

When the blank preservation mode is off, the blanks between two successive strings are not preserved. When a string operation causes the first of two strings to be expanded or contracted, the number of blanks between the two strings are decreased or increased so that the second string stays in the same columnar position. (If the first string expands, the number of blanks between the two strings decreases; if the first string contracts, the number of blanks increases.) At least one blank must be left between strings.

When the blank preservation mode is on, the blanks between the two strings are preserved. That is, when the first string expands or contracts, the second string is moved to the left or right so that the same number of blanks remains between the two strings.

For example, the following string substitution command

```
*/8/S/LINK/(RET)
```

substitutes the string "LINK" for the string "8" in the instruction

```
$10  BAL,8  SUB
```

adjusting blanks as indicated below:

old	$10 BAL,8 SUB
new (BP-OFF)	$10 BAL,LINK SUB
new (BP-ON)	$10 BAL,LINK SUB

Although the BP command is discussed with the file commands, it may also be used when Edit is in the record mode.

COMMAND STREAM CONTROL

GO and RET Change the Source of Commands

These commands allow the execution of a program of Edit commands from the file being edited. Additionally, RET allows conditional premature termination of a line of Edit commands being applied to a range of lines. The formats are

```
GO   s
RET
```

where

"s" is a line sequence number.

GO causes Edit commands to be obtained by reading the edited file sequentially, beginning with the first line whose sequence number is greater than or equal to "s".

RET causes immediate return of control to M:SI, so may be used to break out of either GO mode, or of executing intrarecord commands on a range of lines.

These commands may not be used in step mode, but may be used in either record or intrarecord mode. If an intrarecord range is in effect, these commands do not change it. If they are conditionally specified in a line of commands acting on a range of records, the execution of the line of commands will be terminated whenever GO or RET is executed, even if there are more records in the range.

Execution of commands from the file being edited terminates on

- execution of RET; or
- executing to the end of the edited file; or
- an error in syntax or execution; or
- BREAK signal, followed by an "X" keyin.

Control then returns to M:SI.

Example of executing from the edited file:

```
*TY 1-9999 (RET)                    list the file
  1.000  A
  2.000  BX
  3.000  CXX
  4.000  D
9000.000  SE 9100-9200; 1P/DE/
9100.000  2
9101.000  4
9500.000  RET
*GO 9000 (RET)                      execute commands from
*                                   it
SE 9100-9200; 1P/DE/                edit commands to exe-
  2 STRINGS CHANGED                 cute next
*
DE 2                                execute commands just
*                                   built
DE 4
*
```

```
RET                          return control to M:SI
*TY 1-10 ↵                   list results
    1.000 A
    3.000 CXX
*
```

Example of premature termination of intrarecord processing on a range of records:

```
*SE 1-5 ↵
*TY ↵
    1.000 A
    2.000 BX
    3.000 CXX
    4.000 D
*  1P/Z/; 2/X/Y; RET ↵
    3 STRINGS CHANGED
*TY ↵
    1.000 ZA
    2.000 ZBX
    3.000 ZCXX
    4.000 D
*
```

RECORD EDITING COMMANDS

The record editing commands may only be given after a file has been selected for editing via the EDIT command. If the user does not select a file for editing before giving a record editing command, Edit prints the message:

```
-NO FILE NAMED
```

The record editing commands will be discussed in the following order:

IN, IS	Insert records.
DE	Delete records.
TY, TC, TS	Type individual records.
MD, MK	Reorder records within a file.
FD	Delete records containing a specified character string.
FT	List sequence numbers and contents of records containing a specified character string.
FS	List sequence numbers of records containing a specified character string.
RN	Renumber record.
CM	Insert commentary.
SE	Select a group of records for character operations.
SS, ST	Select records for step mode operation.

IN Insert New Records

IN causes Edit to insert new records into a file. The IN command has the format shown below.

```
*IN[n[,i]]
```

New records are inserted starting at the record with sequence number n, with each successive record being sequenced from n with increment i. If i is omitted, the increment size specified in the most recent insertion command is used. If no such commands have been given, the value 1 is assumed by default. If n is omitted also the insertion is made at the most recent insertion line plus the increment value. If there were no prior insertions, the assumed value for n is 0. Commands which can set the default increment or the default next insertion line are MK, MD, IN, IS, and C. If a record with sequence number n exists in the file, it is replaced by the newly inserted record n.

Edit prompts the user with the first sequence to be inserted, and repeats the prompt for each subsequent insertion, increasing the sequence number by the increment i.

The insertion can be terminated in one of three ways. If a null record (↵ only) is supplied, the insertion terminates. An equivalent action takes place if an incremented sequence equals or exceeds a sequence existing in the file. In the latter case, the console bell is rung. The insertion is also terminated (and an error message is printed) if an attempt is made to insert a record having a record number greater than 9999.999.

Example:

```
*EDIT SOURCEFILE ↵
*IN 100, .1 ↵
100.000    10 A=2.5 ↵      Replaces the existing record.
100.100        B=0. ↵
*                          Record insertion terminates
                           because sequence number
                           100.200 existed previously;
                           the console bell is rung.
```

IS Insert New Records

The IS command is identical to the IN command in function and format except that Edit does not prompt with sequence numbers. The format of the IS command is:

```
*IS[n[,i]]
```

Example:

```
*EDIT SOMEFILE ↵
*IS  100, .1 ↵
10  A = 2.5 ↵
    B = 0 ↵
↵

*
```

DE Delete Records

DE causes Edit to delete all records whose sequence numbers lie in a specified range. The DE command has the form shown below.

```
*DE n[-m]
```

where

- n specifies the number of the first record to be deleted.
- m specifies the number of the last record to be deleted. If m is omitted, only record n is deleted.

Example:

```
*DE 50 ↵          Deletes record 50.000 only.
*DE 50-60.5 ↵     Deletes all records in the range
                  50.000 through 60.500, inclusive.
```

TY Type Records, Sequence Numbers Included

TY causes Edit to write to the M:LO DCB the sequence numbers and the contents of specified columns of one or more records. In addition, it causes Edit to enter the intrarecord mode as though an SE command had been given. The TY command has the format shown below.

```
*TY n[-m][,c[,d]]
```

Edit types records in the range n to m, and types only the portions between columns c and d. If m is omitted, only record n is typed. If the values for c and d are not given, c has a value of 1 and d has a vaule of 140 by default.

Example:

```
*EDIT SOURCEFILE ↵
*TY 1-2,4,8 ↵
1.000   EQU
1.200   SYST
1.400   REF
1.600   DEF
1.800   PAGE
2.000   ITIAL
*
```

TC Type Compressed

TC causes Edit to write to the M:LO DCB the sequence numbers and the contents of specified columns of one or more records. Any nonblank strings within the columns are shifted to the left to compress each blank string to a single blank. This compression affects only the output; the records themselves are not affected. TC is the same as TY with all blank strings compressed to a length of one. Like TY, TC causes Edit to enter the intrarecord mode as though an SE command had been given. The TC command has the format:

```
*TC n[-m][,c[,d]]
```

Edit types records in the range n to m, and types only the portions between columns c and d. If m is omitted, only record n is typed. If the values for c and d are not given, c has a value of 1 and d has a value of 140 by default.

Example:

```
*EDIT SOURCEFILE ↵
*TC 1-2,1,7 ↵
1.000 A EQU
1.200  SYS
1.400 B REF
1.600 C DEF
1.800 PAGE
2.000 *INITIA
*
```

TS Type Records, Sequence Numbers Not Included

TS causes Edit to write to the M:LO DCB the contents of specified columns of one or more records, without accompanying sequence numbers. In addition, it causes Edit to enter the intrarecord mode as though an SE command had been given. The TS command has the format shown below:

*TS n[-m][,c[,d]]

Edit types records in the range n to m, and types only the portions between columns c and d. If m is omitted, only record n is typed. If the values for c and d are not given, c has a value of 1 and d has a value of 140 by default.

Example:

```
*EDIT SOURCEFILE
*TS 1-2,1,8
A    EQU
     SYS
B    REF
C    DEF
     PAG
*INITIA
*
```

MD Move and Delete Records

MD causes Edit to move records from one specified range to another. The original records are deleted as they are moved. Records in the destination range are also deleted. The MD command has the form shown below.

*MD n[-m],k[-p][,i]

where

- n specifies the sequence number of the first record that is to be moved and deleted.
- m specifies the sequence number of the last record that is to be moved. If omitted, only n is moved and deleted.
- k specifies the lower limit (i.e., sequence number) of the range of destination records that will be deleted.
- p specifies the upper limit of the range of records to be deleted. If omitted, only k is deleted. However, records from the range n-m are still moved to record k and following until a record is encountered, that originally followed record k in the file. (When such a record is encountered, no more records are moved.)
- i specifies the increment value to be used for renumbering records. If omitted, the most recent increment value specified in a record edit command is used. If no such commands have been given, the default value is 1.

The first record (n) is renumbered as k. Successive records from the range n-m are renumbered consecutively higher, incremented by i.

It is important to note that the ranges n-m and k-p may not overlap.

As each record from the range n-m is moved, it is deleted from the original range (n-m). At the end of this operation, a message is printed specifying the new sequence number of the last record moved from the range n-m.

Example:

```
*EDIT BETA
*MD 5-21, 100-101, .02
--DONE AT 100.32
```

If the increment is too large to permit all records in the range n-m to be moved into the space between k and the next record after p, a message is printed specifying the sequence numbers, from both ranges, of the last record moved.

In this case the original contents of range k-p will be lost, but only those records in the range n-m that have actually been moved will have been deleted. Thus, the user can perform another move (with a smaller increment) to move the remaining records in the range n-m.

Example:

```
*EDIT BETA
*MD 10-30, 100-110, 1
--CUTOFF AT 110. (20.)
```

20 is the number of the last record that was moved.

MK Move and Keep Records

MK is identical to MD except that the records in the range n-m are not deleted as they are moved; thus a copy of records in the range n-m is made. The MK command has the form shown below.

*MK n[-m],k[-p][,i]

FD **Find and Delete Records**

FD causes Edit to search for a specified string between specified columns. If the string is found, the record containing it is deleted from the file. The FD command has the form shown below.

```
*FD n[-m],/string/[,c[,d]]
```

where

n specifies the sequence number of the first record to be searched.

m specifies the sequence number of the last record to be searched. If omitted, only record n is searched.

/string/ specifies the character string identifying the record to be deleted.

c specifies the lower limit (i.e., column number) of the field to be searched. The default value is 1.

d specifies the upper limit of the field to be searched. The default value is 140.

The specified string must be entirely contained within columns c through d to cause deletion. At the end of this operation, a message is printed telling how many records were deleted.

```
*EDIT FILEA ↵
*FD 5-20.4,/DATA/,10,18 ↵
--006 RECS DLTED
```

If there are no records in the specified range containing the indicated string, Edit prints the following message:

```
--NONE
```

FT Find and Type Record and Sequence Number

FT causes Edit to search for a specified string between specified columns. If the string is found, Edit writes the sequence number and the contents of the record to the M:LO DCB. (The string must be entirely contained within the specifiec columns.) The FT command has the format:

```
*FT n[-m],/string/[,c[,d]]
```

The parameter specifications are the same as those for the FD command.

Example:

```
*EDIT SOMEFILE ↵
*FT 1-100,/LW/,10 ↵
5.000          LW,3          DATA
9.000          LW,2          TABLE,7
21.480         LW,10         LOC+5,8
73.000         LW,9          FLAG
*
```

If there are no records in the specified range containing the indicated string, Edit prints the message

```
--NONE
```

FS Find and Type Sequence Number

FS causes Edit to search a given range of records for a specified character string between designated columns. Edit will write to the M:LO DCB the sequence number of each record satisfying the search criteria. The FS command has the format:

```
*FS n[-m],/string/[,c[,d]]
```

The parameter specifications are the same as those for the FD command.

Example:

```
*EDIT SOMEFILE ↵
*FS 10-20,/BE/,10,11 ↵
15.000
18.000
*
```

If there are no records in the specified range containing the indicated string, Edit prints the following message:

```
--NONE
```

RN **Renumber Record**

RN causes Edit to renumber a specified record. The RN command has the form shown below.

```
*RN n,k
```

This has the same effect as deleting record n and then entering a new record with sequence number k with the same contents as n. Sequence number k must not already exist.

CM Insert Commentary

CM causes Edit to insert commentary into specified columns of each successive record beginning at a specified sequence number. The CM command has the format shown below.

```
*CM n,c
```

where

- n is the record number.
- c is the column number.

The sequence number of each record is typed and then the user types in the data he wants inserted starting at column c. The data he types in is blank filled to the right through column 140, as required. A null record terminates the command. It is not necessary to delimit commentary with slashes.

Example:

```
*EDIT SOURCEFILE ↵
*CM 37.6, 40 ↵
37.600 * COMMENT 1 ↵
37.800 * COMMENT 2 ↵
40.500 * ↵
*
```

SE Set Intrarecord Mode

SE causes Edit to accept successive lines of intrarecord commands. The SE command has the format shown below.

```
*SE n[-m][,c[,d]]
```

Each input line of intrarecord commands is applied, in order, to columns c through d of every record in the range n through m. If m is missing, only record n is processed. The default values for c and d are 1 and 140, respectively.

If several commands are entered on one line, all commands on the line are executed on one record before the next record is processed. The first occurrence of a file or record-editing command terminates the effect of the SE command. All string-matching operations in the intrarecord mode apply only to the strings lying entirely within columns c through d.

SE may be used on the same input line with other intrarecord commands, but when so used, it must be the first command on the line.

SS Set and Step

SS causes Edit to start at a specified record and proceed to each record in succession, accepting one line of intrarecord commands to update the current record. The SS command has the format shown below.

```
*SS n[,c[,d]]
```

The first record to be updated has the sequence number n. Intrarecord string-matching operations will only be effective on strings that lie wholly within columns c through d. The default values for c and d are 1 and 140, respectively.

Edit prompts for commands for each successive record with the sequence number, followed by a double asterisk. The SS command is terminated by typing a null record in place of an intrarecord command.

ST Set, Step, and Type Record

This command is similar to SS except that the content of each record is written to the M:LO DCB along with its sequence number, prior to accepting a command. The ST command has the format shown below.

```
*ST n[,c[,d]]
```

The parameters of the command and the error messages which Edit types are the same as those for the SS command.

INTRARECORD EDITING COMMANDS

The intrarecord commands make changes within an individual record. They generally manipulate character strings. These commands may only be given after the user selects an intrarecord mode with the SE, SS, or ST commands.

The intrarecord commands will be discussed in the following order:

Y or N	Conditional command continuation.
CL	Change column limits for string matching.
S	Substitute string.
D	Delete string.
P	Insert string preceding.
F	Insert string following.
O	Overwrite string.
E	Overwrite string; blank fill.
R or L	Shift string.
A	Align columns by shifting.
C	Copy with another line number.

DE	Delete individual line.
TS TY	Type individual records.
JU	Jump to new sequence.
NO	No change.
RF	Reverse blank preservation flag.

Commands in the intrarecord group may be linked together through use of the semicolon (;). The following command sequence would select a line, type the original, edit, and type the new version:

```
*SE 100; TY; /TEMP/S/B/;/JK/F/+BETA/;TY ↵
```

After any semicolon, the command may be continued on the same or the following line, with identical results.

The following conventions are used with intrarecord commands:

1. j/string/x

   means that command x is to operate on the jth occurrence of the indicated string found between columns c through d as specified by an SE, SS, ST, or CL command. If j=0, this means that the command is to operate on all occurrences of the string between columns c and d. If j is missing, the default is 1. A single / may be included in the string by typing two slashes in succession.

2. k x

   means that command x is to operate on the character contained at column k, where k need not lie between columns c and d of the SE, SS, ST, or CL command. If k is '999', the column following the last non-blank is indicated.

Whenever an S, D, P, F, O, E, R, L, or A command is given, Edit responds in one of the following ways:

1. A prompt alone indicates that either no change or one change occurred as a result of the command.

2. The message

   x STRINGS CHANGED

   followed by a carriage return and a prompt indicates that x changes occurred as a result of the command.

The following general error is possible:

-MISSING SE No SE command was given. Either an SE, SS, or ST command must be given in response to this message.

Before reading the intrarecord command descriptions, it is important to note the following information:

Note: In any intrarecord command that seeks a matching string in the image, only those strings that lie totally within the specified column bounds will be found. Partial matches to a column boundary will be ignored. In subsequent examples, references to columns c and d pertain to the column boundaries given in the SE, SS, ST, or CL command.

Y and N Continue if Yes and Continue if No

These intrarecord commands provide for conditional execution of other intrarecord or branch commands. The format is

```
[. . .;]n/string/Y[;. . .] or
[. . .;]n/string/N[;. . .]
```

where "n" is an optional nonzero occurrence count.

Both commands affect the execution of any other commands which follow them on the same command line or its continuations. The commands which follow Y are executed only if the string defined in the Y command is matched. The commands which follow N are executed only if the string is not matched. In the case where a range of lines is being edited, the decision is made independently for each line.

Example:

```
*TY 1-1000 ↵
    1.000  A
    2.000  BX
    3.000  CXX
    4.000  D
*/X/Y; TY ↵
    2.000  BX
    3.000  CXX
*2/X/Y; TY ↵
    3.000  CXX
*/X/N; TY ↵
    1.000  A
    4.000  D
*2/X/N; TY ↵
    1.000  A
    2.000  BX
    4.000  D
*
```

CL Change String-Search Column Limits

This intrarecord command allows modification of the column limits for string matching operations. The format is

[...;]CL[c[,d]][;...]

Where 'c' is the new first column, and 'd' is the new last column for string-matching operations. If 'd' is omitted, it defaults to 140. If both 'c' and 'd' are omitted, the defaults are 1 and 140, respectively. The new column limits remain in effect until the next SE, SS, ST, or CL command.

Example:

```
*TS 1
NO CHANGE
*CL1,4;/A/Y;/NO/S/ /;TS
NO CHANGE
*CL5,8;/A/Y;/NO/S/ /;TS
NO CHANGE
*CL5,8;/A/Y;CL1,4;/NO/S/ /;TS
   CHANGE
```

S Substitute String

S causes Edit to locate a specified string ($string_1$) between columns specified by an SE, SS, or ST command and replaces it with another string ($string_2$). The S command has the format shown below.

*[j]/$string_1$/S/$string_2$/

The image to the right of $string_1$ is adjusted right or left as required, if the lengths of $string_1$ and $string_2$ differ. $String_2$ may extend past column d if d < 140.

If j = 0, all occurrences of $string_1$ between columns c and d are replaced by $string_2$. Otherwise, only the jth occurrence is replaced. If j is missing, the default value is 1.

Example:

Command	Effect		
*/LW/S/CW/		LW,R5 ALPHA+2 CW,R5 ALPHA+2	old new
*/10/S/5/		LW,R10 B LW,R5 B	old new
*/$10/S/ENTRY/	$10 ENTRY	LW,R5 ALPHA LW,R5 ALHPA	old new
*/ALPHA/S/B/		LW,R5 ALPHA+2,R6 LW,R5 B+2,R6	old new
*2/5/S/55/	15 15	C=DISQRT(TEMP+2.5 *BASE) C=DSQRT(TEMP+2.55 *BASE)	old new

D Delete String

D causes Edit to locate a given occurrence of an indicated string, between columns specified by an SE, SS, or ST command, and delete it. The D command has the format shown below.

*[j]/string/D

If j = 0, all occurrences of the string between c and d are deleted. Otherwise, only the jth occurrence is deleted. If j is omitted, the default value is 1.

Example:

```
*EDIT SOMEFILE ↵
*TY 7 ↵
    7.000    STW,4    ALPHA    ANSWER
*SE 7 ↵
*/ANSWER/D ↵
*TY 7 ↵
    7.000    STW,4    ALPHA
```

P Precede String

P causes Edit to start before the first character of a given occurrence of a specified string ($string_1$) or column k and insert another string ($string_2$), pushing characters of the first string to the right as required to make room. The P command has the format shown below.

* j /$string_1$/P/$string_2$/

or

*kP/$string_2$/

$String_2$ may legally extend beyond column d if d < 140. The first character of $string_2$ will occupy the column vacated by the first character of $string_1$, etc.

If j = 0, Edit will insert $string_2$ before all occurrences of $string_1$ between columns c and d. However, after $string_1$ has been found once and $string_2$ inserted before it, scanning for the next occurrence resumes at the next character after $string_1$, as adjusted by the insertion. If j is not equal to zero, the command will only affect the jth occurrence of $string_1$. If j is omitted, the default value is 1.

Example:

```
*SE 17.69 ↵          (set intrarecord mode)
*TS;0/AA/P/./;TS ↵   (type; edit; type)
AAAAAAA              (original record)
.AA.AA.AAA           (edited record)
```

F Follow String

F causes Edit to start after the last character of a given occurrence of a specified string ($string_1$) or column k and insert another string ($string_2$), pushing everything from this column right as required to make room. The F command has the format shown below.

$$\underline{\ }*[j]/string_1/F/string_2/$$

or

$$\underline{\ }*kF/string_2/$$

The j specifies that the jth occurrence of $string_1$ between columns c and d (specified by an SE, SS, or ST command) is to be followed by $string_2$. If j is omitted, the default value is 1. In the case where j = 0, Edit inserts $string_2$ at all occurrences of $string_1$ between columns c and d. Scanning for the next occurrence of $string_1$ resumes following the last character of $string_2$. If a given occurrence of $string_1$ is shifted beyond column d due to previous insertions, it will not be scanned.

$String_2$ may legally extend past column d if d < 140.

Example:

Command	Effect		
*/AB/F/+2/	LW, R6 LW, R6	AB, R2 AB+2, R2	old new

O Overwrite

O causes Edit to start at the column occupied by the first character of a given occurrence of a specified string ($string_1$) or column k and overwrite with another string ($string_2$). No blank preservation or other adjustment is done and all columns not overwritten remain unchanged. The O command has the form shown below.

$$\underline{\ }*[j]/string_1/O/string_2/$$

or

$$\underline{\ }*kO/string_2/$$

$String_2$ may overwrite beyond column d if d < 140. The j specifies that the jth occurrence of $string_1$ between affected columns is to be overwritten by $string_2$. If j is omitted, only the first occurrence is overwritten. If j = 0, all occurrences are overwritten. In the case where j = 0, $string_2$ is not scanned by Edit after $string_1$ is overwritten. Edit begins scanning with the column following $string_2$.

E Overwrite and Extend Blanks

E causes Edit to start at the column occupied by the first character of a given occurrence of a specified string ($string_1$) or column k and overwrite with another string ($string_2$). The E command has the format shown below.

$$\underline{\ }*j/string_1/E/string_2/$$

or

$$\underline{\ }*kE/string_2/$$

Blanks are extended from the end of $string_2$ through column d (where d is the upper limit of the column range selected by an SE, SS, or ST command). $String_2$ may overwrite beyond column d if d < 140, but blank extension only occurs through column d.

The j specifies that the jth occurrence of $string_1$ between affected columns is to be overwritten by $string_2$. If j is omitted, only the first occurrence is overwritten. The specification j = 0 may not be specified, since blank extension precludes multiple substitutions within the same record.

R or L Shift Record Image

R or L commands cause portions of the record image to be shifted right (R) or left (L). The R and L commands have the form shown below.

$$\underline{\ }*[j]/string/\begin{Bmatrix}R\\L\end{Bmatrix}s$$

or

$$\underline{\ }*k\begin{Bmatrix}R\\L\end{Bmatrix}s$$

The string must lie wholly within columns c and d specified by the current SE, SS, or ST command. The specified substring may contain embedded blanks, but the string to be shifted terminates with the first blank following the specified substring.

The j specifies that the jth occurrence of the specified substring between affected columns is to be shifted, together with all subsequent contiguous nonblank characters. If j is omitted, only the first such occurrence is shifted. Note that j = 0 may not be specified for this command.

L Shift Left

The jth field that begins with the indicated string (or column k) is shifted left s positions. If blank preservation (see the BP command) is ON, all of the fields to the right of the string are shifted left, intact, and the fields to the left of the string are overwritten (i.e., destroyed). If blank preservation is OFF, blanks are inserted to the right of the jth field, and the fields to the left of the string are overwritten. The shift may legally overwrite below column c.

R Shift Right

The jth field that begins with the indicated string (or column k) is shifted right s positions. If blank preservation is ON, blanks are inserted to the left of the string and all of the fields to the right of the string are shifted right, intact. If blank preservation is OFF, blanks are inserted to the left of the string and are removed to the right. With blank preservation OFF, the image area to the right of the string may be compressed, but at least one blank will be left between each field; that is, overwriting does not occur in a shift right. The shift may legally push characters beyond column d, if d is less than 140.

In the following examples, blank preservation is OFF.

Command	Effect
*/L/R1	$10 LW,R6 B old $10 LW,R6 B new
*/L/R9	$10 LW,R6 B old LW,R6 B new
*/L/L1	$10 LW,R6 B old LW,R6 B new

A Align Specified Columns

The A command causes either a right or left shift of a part of the record being edited to align one specified column with another. The details of the shift operation are as described for the R and L commands. The form of the command is

[. . .;] c A d[;. . .]

where "c" specifies the column to be aligned and "d" specifies the column to which to align. The "c" and "d" items can be any of

k

an integer column number,

/string/

specifying the first column of the first occurrence of the string, or

n/string/

specifying the first column of the "n"th occurrence of the string. ("n" can not be zero for this command.)

Examples:

```
*SE 1-3; TS
   -ONE
   XXXXX      -TWO, TWO
-THREE  THREE  THREE
*/-/ A 5; TS
   -ONE
   -TWO                TWO
   -THREE THREE THREE
```

Note the distinction between left-shift (TWO) and right-shift (THREE) behavior.

```
*SE 20; TS
*ONE   *TWO   *THREE
*3/*/A 2/*/; TS
*ONE   *THREE
```

C Copy Current Line

This intraline command causes the line being edited to be copied with a specified or provided sequence number. The original line is unaffected by this command. The format is

[. . .;] C[s][;. . .]

where "s" is a sequence number at which to copy the line. If "s" is not specified, the current default increment is added to the sequence number to which the last C applied, to provide the new sequence number. Commands which can set the default increment or the default next insertion line number are MK, MD, IN, IS, and C.

Example:

```
*TY 1-1000
    1.000  A
    2.000  BX
    3.000  CXX
    4.000  D
*IS 101
.
*SE 1-99; /X/Y;C
*TY 1-1000
    1.000  A
    2.000  BX
    3.000  CXX
    4.000  D
  101.000  BX
  102.000  CXX
```

DE Delete Current Record

If used without a specified sequence number, DE becomes an intrarecord command, which specifies that the record currently open for editing be deleted. The form is

[. . .;] DE

Note that it must be the last command in a command line, since once it is executed, there is not anything left of the currently open line.

Example:

```
*TY  1-1000 ↵
 1.000    A
 2.000    BX
 3.000    CXX
 4.000    D
*/X/Y; DE
*TY ↵
 1.000    A
 4.000    D
```

TS Type Record, Sequence Number Not Included

TS causes Edit to write to the M:LO DCB the contents of the record currently open for editing under control of an SE, SS, or ST command. (Unlike the record-editing version, the intrarecord version of TS does not allow column specification.)

The TS command has the format shown below.

```
*[. . .;] TS [; . . .]
```

The three dots indicate that intrarecord commands may precede or follow the TS command.

Example:

```
*SE 5; TS ↵
L1 LW,5K
*19O/KLB/; TS; 37O/GET KLB/; TS ↵
        (overwrite, type, overwrite, type)
L1 LW,5 KLB
L1 LW,5 KLB    GET KLB
```

Because all commands on a single input line are executed for the first record before the second record is processed, etc., TS will type each line in turn after all editing up to the TS command has been done.

Example:

```
*SE  10- 10.2 ↵
*2/A/F/,4/;TS ↵
DATA,4  X'FF'      (10.0)
DATA,4  0.5        (10.1)
DATA,4  GQX,X'0B'  (10.2)
```

TY Type Record, Sequence Number Included

TY is the same as TS, except that each line is written with its sequence number. (Unlike the record-editing version, the intrarecord version of TY does not allow column specification.)

The TY command has the format shown below.

```
*[. . .;] TY [; . . .]
```

JU Jump

JU causes the SS or ST command to jump to a specified record and then continue stepping from that point. JU may only be used while in the "step" mode (i.e., while under the control of an SS or ST command). The JU command has the form shown below.

```
*[. . .;] JU n
```

Record n may be forward or backward from the current sequence number at the time JU is given. The dots indicate that JU may be used on compound lines (i.e., a line with more than one command on it), but in such a case JU must be the last command on the line.

NO Make No Change

NO may be used only while in the "step" mode and specifies that no editing is desired on the current active line under the set. The NO command has the format shown below.

```
*NO
```

Example:

```
*ST 27.5 ↵
27.500          LW,6   BLK
*NO ↵
30.000          STW,6  ALT
*/ALT/F/+19 ; TY ; JU 34 ↵
30.000          STW,6  ALT + 19
34.000          A1,F   X'91'
*
```

RF **Reverse Blank Preservation Mode**

RF causes the current setting of the blank preservation mode (see the BP command) to be reversed temporarily. The RF command has the form shown below.

*[. . . ;] RF ; . . .

or

*. . . ; RF[; . . .]

The mode is reversed only for the duration of the input line in which RF appears and only for those commands which follow the RF command, and blank preservation is restored to its initial setting when a new input line is entered (i.e., at the time a new prompt character is given). Thus, to have any effect, RF must always be used as part of a compound input line and must be followed by other commands.

Example:

*SE 10; TY ↵

10.000 L5 LW,4 X GET CURRENMT ADDR

*RF;/NM/S/N/;TY ↵

10.000 L5 LW,4 X GET CURRENT ADDR

Without using RF in this case (assuming that BP OFF is the initial setting), one would get two blanks after CURRENT. In all cases, the BP mode is restored to the value it had before any RF commands were given.

MESSAGES

During the course of executing any command, Edit may communicate with the user through a variety of messages. These messages are directed to the LL oplabel.

Possible messages are summarized in Table 22. The following conventions are used in regard to message formats:

1. A message preceded by two periods is a comment on some system-oriented operation. For example,

   ..COPY DONE

2. A message preceded by two minus signs indicates the occurrence of some event (during the execution of a command) of which the user should be aware; the command is not aborted. For example,

   --EOF HIT AFTER xxxx.xxx

3. A message preceded by a single minus sign is an error message describing a condition that aborts the current command and causes any others on the same line to be skipped. For example,

   -P1:NO SUCH REC

Such a message is particularized as to cause by the following prefixes:

Prefix	Cause of Error
-Ck:	The kth command of the previous line caused the error.
-Pk:	The kth parameter of the first command on the previous line caused the error.
-CkPj:	The jth parameter of the kth command of the previous line caused the error.

EDIT COMMAND SUMMARY

Table 23 is a summary of Edit commands. The left-hand column gives the command formats. The right-hand column gives the command functions and options.

Table 22. Edit Messages

Message	Meaning
-BAD COL. NO. PAIR	The columns specified are not in the range 1 through 140, or c > d.
--Cn: 'ALL' IGNORED	The value 0 was specified for j. Since this value is not meaningful for the command, the value 1 has been assumed.
-Cn: CMND ILGL HERE	The nth command of the previous line is invalid and the intrarecord mode has been terminated.
-Cn: COL > LIMIT	The value specified for k is greater than d for the nth command.
-Cn: COL < LIMIT	The value specified for k is less than c for the nth command.
-Cn: ILGL SYNTAX	Invalid command syntax has been used.

Table 22. Edit Messages (cont.)

Message	Meaning
-Cn: NO SUCH REC	The record specified in a JU command (the nth command) does not exist.
--Cn: OVERFLOW	The nth command of the previous line has caused characters to be shifted past column 140. Processing continues.
--Cn: UNDERFLOW	Characters were lost to the left of the record.
-Cn: UNKN CMND	The nth command specified is not one recognized by Edit.
--CUTOFF AT x(y)	A specified operation could not be completed because of a conflict between an existing sequence number and a new one. The value x is the current sequence number of the last record affected (formerly record y).
--DONE AT x	A specified operation has been completed. The value x is the current sequence number of the last record affected.
--EOF HIT AFTER xxxx.xxx	One or both sequence numbers specified are higher than the highest one in the file. xxxx.xxx is the last record in the file.
-ILLEGAL LINE NR IN SUBJECT FILE	Sequence numbers were being obtained from the subject file records and a record was found without a correct sequence number in its last eight bytes.
--LINES REORDERED	Sequence numbers were being obtained from the subject file records and they were not in numeric order. They will be reordered numerically.
-MAX. SEQ. NO. EXCEEDED	An attempt was made to insert a record with a record number greater than 9999.999.
-MISSING SE	No SE, SS, or ST command is currently in effect. The specified intrarecord task has been aborted.
-NO FILE NAMED	The command requires a file identification and none was given.
-NO SUCH FILE	A specified file does not exist.
--NONE	There are no records in the specified range containing the indicated string.
-NOT ON/OFF	A parameter other than ON or OFF has been specified in a BP or CR command.
--NOTHING TO MOVE	No records (to be moved) were found in the specified range.
--OVERFLOW	More than 140 characters have been typed on a line or characters have been shifted past column 1 or 140. Excess characters are lost.
-P1: NO SUCH FILE	A COPY command has specified that a nonexistent file is to be copied.
-P1: NO SUCH REC	A specified record does not exist. The command has been aborted.
-P2: COL ERROR	Column c is greater than 140.
-P1: FILE EXISTS	A SAVE ON command specified the name of an existing file.
-P2: REC EXISTS	A specified record already exists. The command has been aborted.
-P3: NOT INCR	An invalid increment has been used in a BUILD command.
-Pn: BAD FID	An invalid fid identification has been used.

Table 22. Edit Messages (cont.)

Message	Meaning
-Pn: ILGL SEQ#	A required sequence number is missing or contains more than three fractional digits.
-Pn: ILGL STRG	The specified string is too long.
-Pn: ILGL SYNTAX	An invalid syntax has been specified for a parameter.
-Pn: NOT CNT	An invalid occurrence count (j) has been specified in an intrarecord command.
-Pn: NOT COL#	An invalid column number has been specified.
-Pn: NOT SEQ#	A required sequence number is invalid.
-Pn: NOT STRG	An invalid character string has been specified.
-Pn: NULL STRG	A null character string has been specified.
-Pn: PARAM MISSING	A required parameter has not been specified.
-Pn: SEQ2<SEQ1	The second sequence number is less than the first in a range specification.
xxxxxxx RECORDS DELETED	xxxxxxx specifies the number of records deleted as the result of either a DE or MD command.
--REPEATED LINE NR...DELETED	Sequence numbers were being obtained from the subject file records and a duplication was found. Only the last line is retained.
--RNG OVERLAP	Specified ranges of sequence numbers overlap. The command has been ignored.
--SAVE FILE TOO SMALL: FSI = yyyy	The estimated size needed to save is "yyyy" records. If the command was SAVE without parameters, END, or EDIT, the save file is unchanged and the command had no effect. Otherwise, the save was partially completed.
--SCRATCH FILE OVERFLOW	The scratch file is too small for the most recent command to be completed. Multiline commands may have been executed on some lines. No lines have been partly completed.
x STRINGS CHANGED	As the result of an intrarecord command, x strings have been changed.

Table 23. Edit Command Summary

Command	Description
[...;] cAd [;...] where c and d are k or [j]/string/	Aligns the first specified column with the second. Columns are specified by either the column number k, or as the first column of the specified string. Option: j specifies the jth occurrence of the specified string. Default is the first occurrence.

Table 23. Edit Command Summary (cont.)

Command	Description
BP [{ON / OFF}]	Sets the blank preservation mode. When ON, all strings of blanks are preserved during intrarecord operations. When OFF, blank strings are compressed to a single blank or expanded as required to retain column alignment of nonblank fields. The default mode is "off".
[. . .;] C [s][;. . .]	Copies the current edit line with another sequence number. Option: s specifies the sequence number for the copy. Default is the sequence number used by the prior insertion command plus the increment most recently specified in any prior command. If there was no prior insertion command, s is zero plus the increment. If there is no prior increment, 1 is used.
CL [c [,d]]	Changes the string-search column limits for intrarecord commands. Options: c is the new first column number. If it is not specified, it defaults to 1. d is the new last column number. If it is not specified, it defaults to 140.
CM n,c	Causes Edit to insert commentary (given by the user) into specified columns (starting at column number c) of each successive record beginning at the specified sequence number n.
[j]/string/D	Locates a given occurrence of the indicated string. Between columns specified by an SE, SS, or ST command, and deletes it. Option: j specifies that only a particular occurrence (the jth occurrence) of the string in the specified columns is to be deleted. If j equals zero, all occurrences of the string in the specified columns are to be deleted. If j is omitted, the default value is 1.
[. . .;] DE	Deletes the current edit line.
DE n[-m]	Deletes all records whose sequence numbers lie in a specified range beginning at n. Option: m indicates the number of the last record to be deleted. If m is omitted, only record n will be deleted.
[j]/$string_1$/E/$string_2$/ or k E/$string_2$/	Starts at a column occupied by the first character of a given occurrence of a specified string ($string_1$) or column k and overwrites with another string ($string_2$). Blanks are extended from the end of $string_2$ through column d (which is specified in an SE, SS, or ST command). Option: j specifies that the jth occurrence of $string_1$ between affected columns is to be overwritten by $string_2$. If j is omitted, only the first occurrence is overwritten; j may not be zero.
EDIT fid1 OVER fid2 [,start[, step]] or EDIT fid2	Prepare for editing the data in file "fid1" (if specified) using file "fid2" as the scratch file. Options: start and step specify the sequence numbering.

Table 23. Edit Command Summary (cont.)

Command	Description
END [NS]	Closes all active files and returns control to the terminal executive language (TEL). Option: No save of the edited data on the subject file.
[j]/string$_1$/F/string$_2$/ or kF/string$_2$/	Starts after the last character of a given occurrence of a specified string (string$_1$) or column k and inserts another string (string$_2$), pushing everything from this column right as required to make room. Option: j specifies that the jth occurrence of string$_1$ between columns c and d (specified by an SE, SS, or ST command) is to be followed by string$_2$. If j is omitted, the default value is 1. If j equals zero, string$_2$ is inserted at all occurrences of string$_1$ between columns c and d.
FD n[-m],/string/[,c[,d]]	Searches for the specified string between specified columns in a specified range of records beginning at record n. If the string is found, the record containing it is deleted from the file. Options: m specifies the sequence number of the last record to be searched. If omitted, only record n is searched. c specifies the lowest column number of the field to be searched. The default value is 1. d specifies the highest column number of the field to be searched. The default value is 140.
FS n[-m],/string/[,c[,d]]	Searches for the specified string between specified columns in a specified range of records beginning at record n. Each time the string is found, the sequence number of the record is printed. Options: Same as for FD.
FT n[-m],/string/[,c[,d]]	Searches for the specified string between specified columns in a specified range of records beginning at record n. Each time the string is found, the sequence number and the contents of the record are printed. Options: Same as for FD.
[...;]GO s	Begins using the Edit file as a command source by reading sequentially starting at sequence number s. If executed in intrarecord mode, causes the specified command to be accessed, whether the end of the edit range was reached or not, but does not change the edit range.

Table 23. Edit Command Summary (cont.)

Command	Description
IN[n[,i]]	Inserts new records into a file starting at record n. Edit prompts the user with the sequence number of each record to be inserted. Options: n specifies the next record to insert. If omitted, the last insertion record plus the increment is used. i specifies an increment amount for successive record numbers. If i is omitted, the increment size specified in the most recent record editing command is used. If no such command has been given, the default value is 1.
IS[n[,i]]	Inserts new records into a file starting at record n. Edit does not prompt with sequence numbers of the records to be inserted. Options: n specifies the next record to insert. If omitted, the last insertion record plus the increment is used. i specifies an increment amount for successive record numbers. If i is omitted, the increment size specified in the most recent record editing command is used. If no such commands have been given, the default value is 1.
[. . . ;] JU n	Causes the SS or ST command to jump to the specified record n and then continues stepping from that point. Option: The dots indicate that JU may be used on a line with more than one command on it, but in such a case JU must be the last command on the line.
MD[n -m],k[-p][,i]	Moves records within a file from a range beginning at n to a range beginning at k. The original records are deleted. Options: m specifies the sequence number of the last record that is to be moved. If omitted, only record n is moved. p specifies the upper limit of the range of records to be deleted. If omitted, only record k is deleted. However, records from the range n-m are still moved to record k and following. i specifies the increment value to be used for renumbering records. If omitted, the most recent value specified in a record edit command is used. If no such commands have been given, the default value is 1.
MK n[-m],k[-p][,i]	MK is identical to MD except that the records in the range n-m are not deleted as they are moved. Options: Same as for MD.

Table 23. Edit Command Summary (cont.)

Command	Description
[. . .;][j]/string/N[;. . .]	Causes the remainder of the commands on the same command line (and any continuations) to be executed only if the specified string is not found in the edit record. (Independently determined for each record in the edit range.) Option: j specifies the jth occurrence of the string. Otherwise, the first occurrence is sought.
NO	Specifies that no editing is to be performed on the current active line.
[j]/$string_1$/O/$string_2$/ or kO/$string_2$/	Starts at the column occupied by the first character of a given occurrence of a specified string ($string_1$) or column (k) and overwrites with another string ($string_2$). Option: j specifies that only a particular occurrence (the jth occurrence) of the string is to be overwritten. If j equals zero, all occurrences of the string are to be overwritten. If j is omitted, the default value is 1.
[j]/$string_1$/P/$string_2$/ or kP/$string_2$/	Starts before the first character of a given occurrence of a specified string ($string_1$) or column k and inserts another string, pushing characters of the first string to the right as required to make room. Option: Same as for the O command.
[j]/string/$\begin{Bmatrix}R\\L\end{Bmatrix}$s or k$\begin{Bmatrix}R\\L\end{Bmatrix}$s	Shifts portions of the record right (R) or left (L) the number of positions indicated by s. The field to be shifted begins with the indicated string or column k. Option: j specifies that the jth occurrence of the specified substring between affected columns is to be shifted, together with all subsequent contiguous nonblank characters. If j is omitted, only the first such occurrence is shifted. Note that j = 0 may not be specified for this command.
[. . .;]RET	Returns immediately to M:SI for next command. If executed while in GO mode (that is, while obtaining commands from the edited file), GO mode is terminated. If executed as an intrarecord command, the current command line will be discarded in favor of the next command from M:SI even if the whole edit range has not been processed, but the edit range will not be changed.
[. . .;] RF ; . . . or . . . ; RF[; . . .]	Causes the current setting of the blank preservation mode ("ON" or "OFF") to be reversed temporarily (for the current line only). Option: The dots indicate that other commands are present on the line.
RN n, k	Renumbers a specified record from number n to number k.

Table 23. Edit Command Summary (cont.)

Command	Description
[j]/string$_1$/S/string$_2$/	Locates a specified string (string$_1$) between columns specified by an SE, SS, or ST command and replaces it with another string (string$_2$). Option: j specifies that only a particular occurrence (the jth occurrence) of string$_1$ is to be replaced. If j equals zero, all occurrences of string$_1$ are to be replaced. If j is omitted, the default value is 1.
SAVE [ON / OVER fid1]	Forces saving of the scratch file. Option: fid1 specifies the file on which to save. Default is the subject file.
SE n[-m][,c[,d]]	Causes Edit to accept successive lines of intrarecord commands to be applied to records beginning at record n. Options: m specifies the number of the last record to which the intrarecord commands are to be applied. If omitted, the intrarecord commands are only applied to record n. c specifies the smallest column number of the range of columns to which the intrarecord commands are to be applied. The default value is 1. d specifies the largest column number of the range of columns to which the intrarecord commands are to be applied. The default value is 140.
SEQ {ON / OFF} SS n[,c[,d]]	Sets the save file sequencing mode. Causes Edit to start at a specified record (record n) and proceed to each record in succession, accepting one line of intrarecord commands to update the current record. Options: c specifies the smallest column number of the range of columns to which the intrarecord commands are to be applied. The default value is 1. d specifies the largest column number of the range of columns to which the intrarecord commands are to be applied. The default value is 140.
ST n[,c[,d]]	Causes Edit to start at a specified record (record n) and proceed to each record in succession, accepting one line of intrarecord commands to update the current record. The sequence number and contents of each record are typed prior to accepting a command. Options: Same as for the SS command.

Table 23. Edit Command Summary (cont.)

Command	Description
TC n[-m][,c[,d]]	Types the sequence numbers and the contents of specified columns of one or more records beginning at record n. Any nonblank strings within the columns typed are shifted to the left to compress each blank string to a single blank. Options: m specifies the number of the last record to be typed. If omitted, only record n is typed. c specifies the smallest column number of the range of columns to be typed. The default value is 1. d specifies the largest column number of the range of columns to be typed. The default value is 140.
[. . . ;] TS[; . . .]	Types the contents of the record currently open for editing under control of an SE, SS, or ST command. Option: The dots indicate that other commands may be present on the line.
TS n[-m][,c[,d]]	Types the contents of specified columns of one or more records beginning at record n. Options: Same as for the TC command.
[. . . ;] TY[; . . .]	Types the sequence number and contents of the record currently open for editing under control of an SE, SS, or ST command. Option: The dots indicate that other commands may be present on the line.
TY n[-m][,c[,d]]	Types the sequence numbers and the contents of specified columns of one or more records beginning at record n. Options: Same as for the TC command.
[. . .][j]/string/Y[; . .]	Causes the remainder of the commands on the same line (and any continuations) to be executed only if the specified string is found in the edit record. (Independently determined for each record in the edit range. Option: j specifies the jth occurrence of the string. Otherwise, the first occurrence is sought.

14. TERMINAL JOB ENTRY

Terminal Job Entry (TJE) provides real-time-shared computing service that allows multiple on-line terminals to be connected to the central computer. There are three general categories of time-sharing service provided to on-line users. They are on-line file management, on-line multi-task execution and debugging, and on-line entry of jobs into the batch stream. Some of these services are provided by processors that are listed in Table 24 and are discussed in more details in the following paragraphs.

Table 24. On-Line User Processors

Processor	Function
TEL	Executive language control of all terminal activities.
EDIT	Composition and modification of programs and other bodies of text.
MUST	General file and peripheral utility.

TERMINAL EXECUTIVE LANGUAGE

The Terminal Executive Language (TEL) is the principal terminal language for CP-R. Most activities associated with FORTRAN and assembly language programming can be carried out directly in TEL through requests that take the form of single-line commands and declarations. These activities include such major operations as composing programs and other bodies of text, batching the programs, initiating execution of one or more concurrent tasks, and debugging these tasks. They also include such minor operations as determining batch status and setting simulated tab stops.

EDIT

The Edit processor is a line-at-a-time context editor designed for on-line creation, modification, and handling of programs and other bodies of information. All Edit data is stored on disk storage in a structure of sequence-numbered variable length records.

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.

MUST

MUST is a multi-use service translator which is a subset of RADEDIT and allows for file and peripheral manipulation. It may be used to allot, delete, truncate, dump, or map files and areas. It may also serve to copy files or devices to one another.

THE TERMINAL

Under CP-R terminals are treated as devices in every sense. They may be assigned to operational labels, to DCBs, and by background control commands. Any job may assign to any terminal in any other job and communicate with it.

TERMINAL OPERATIONS

The types of on-line terminals used with TJE may be:

Xerox Model 7015 Keyboard/Printer.

Teletype Models 33, 35, and 37.

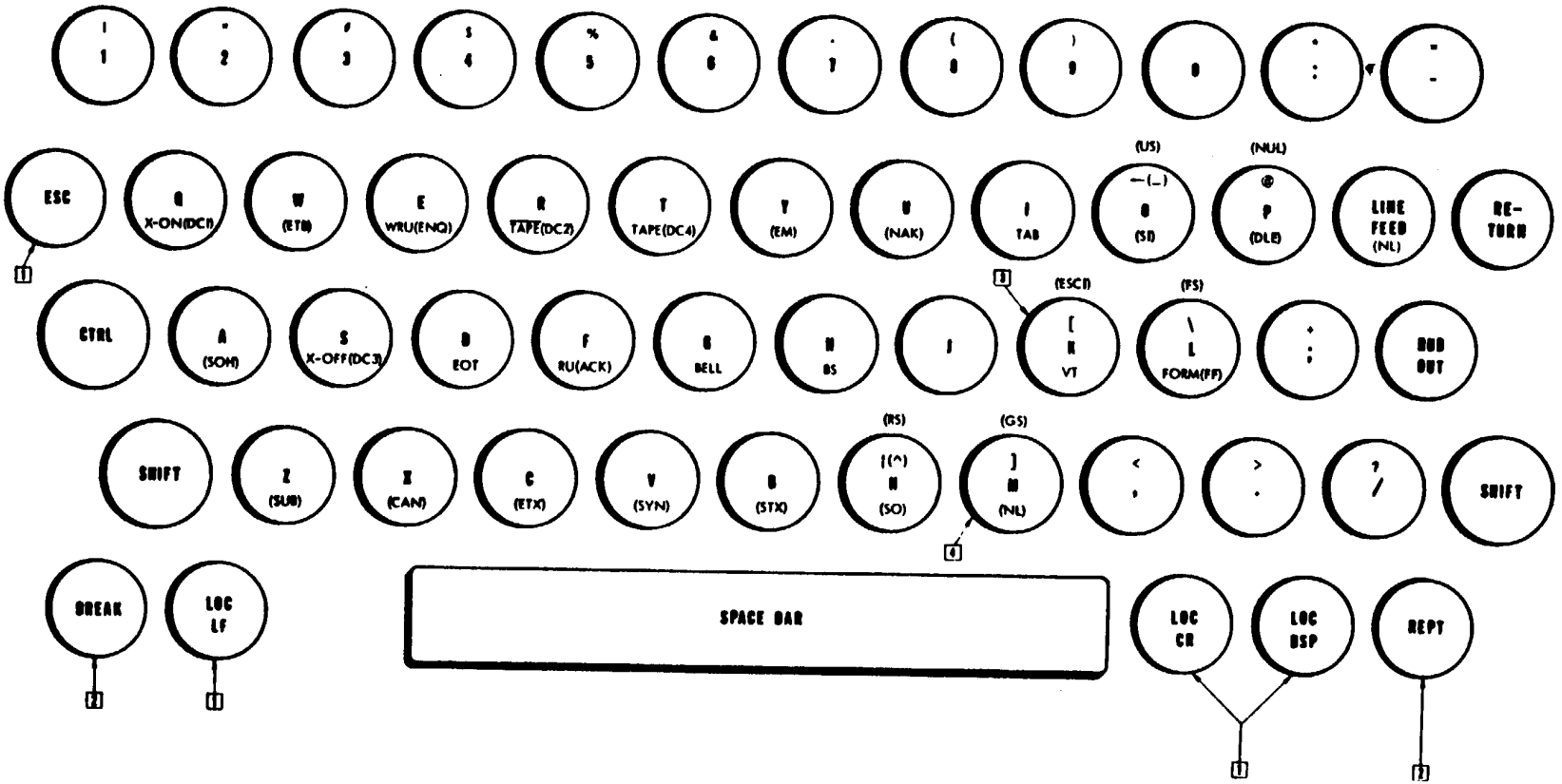
Any terminal compatible with any of the above.

In addition to these remote terminals supported through use of Xerox communication equipment, local teletypes (e.g., TYndd) may be used with TJE. Since local terminals are not processed by the communications handler only these functions avaiable by the local teletype handler are available to the local teletype user. (See LOGON, CONTROL and BREAK key-ins.)

The following sections describes the totality of remote terminal support (see Figure 28) without delineation of those functions not available to local terminals.

Six facets of terminal operations are described in this section. They are

1. Initiating and ending on-line sessions.
2. Typing lines.
3. Typing commands.
4. Detecting and reporting errors.
5. Interrupting CP-R.
6. Paper tape unit.



[1] These keys are missing on some models.

[2] This key is positioned elsewhere on some models.

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

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

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

Figure 28. Model 33 Teletype Terminal Keyboard

INITIATING AND ENDING ON-LINE SESSIONS

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

```
XEROX CP-R AT YOUR SERVICE
ON AT (time and date)
LOGON PLEASE:
```

The system then waits for user account and name to be entered in the following format:

```
account,name
```

Account must be from one to eight characters and the name must be from one to twelve characters in length. Any of the following characters may be used in user account and name:

A-Z a-z 0-9 _ $ * % : # @ - backspace

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

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

If the identification is valid and consistent with CP-R records, the user gives the job identification, TEL types its prompt character (!) at the left margin, and then awaits the first command. The system sends an error message to the terminal and repeats the log-on request if the identification is garbled or otherwise invalid. The error messages are

```
? under the field in question

account,name??
```

It may not always be possible to log on. If an error prevents the log-on, the message LNxxx yy or TYndd yy (e.g., LN007 66) will be typed, where "yy" (i.e., 66) represents a type completion. Depending on the error, the user may be logged off immediately or allowed to try again. The system tries five time to log each user on before dismissing him. Note that an error message of this type may appear anytime during a user's session due to various reasons (e.g., lack of space) causing an immediate log-off.

Following a successful log-on, the user will receive the following message:

!!TERMINAL JOB {LNxxx / TYndd} account,name ON

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

!!TERMINAL JOB {LNxxx / TYndd} account,name OFF

Note that "LNxxx or TYndd" not only identifies the user's terminal, but also the user's job.

TYPING LINES

The rules governing the typing of lines are concerned with operations such as erasing characters or lines, terminating lines, pagination, tabbing, and so on. These rules are common to TEL, all subsystems, and programs that carry on a line-by-line dialogue with the user.

PROMPT CHARACTERS†

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

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

TEL	!
EDIT	*
MUST	:

ECHOING CHARACTERS – ESC E

For terminals operated in echoplex (full-duplex) mode, character echoing – display on the terminal's output device of characters typed in – by the system is normally 'on', but can be turned off, and on again, etc., at the user's discretion (e.g., to suppress printing of passwords or other security-related information). Successive uses of the ESC E key sequence toggles the echoplexing on/off state. For terminal units operated in local-printing (half-duplex) mode, characters typed at the terminal are automatically printed by the terminal. When operating in local-printing mode, the user will need to turn echoplexing off to avoid redundant echoing by the system. (In half-duplex, a direct electrical connection exists between the keyboard and printer, via the modem unit.)

---

†See M:PC Service Call, Chapter 5.

ERASING CHARACTERS – RUBOUT OR ESC RUBOUT

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

ERASING THE CURRENT INPUT LINE – ESC X

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

CANCELLING ALL INPUT AND OUTPUT – CONTROL X

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

RETYPING THE CURRENT LINE – ESC R

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

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

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

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

2. Using the simulated local carriage return sequence ESC RET or ESC LINE FEED for the same purpose.

TERMINATING LINES

When TEL or a processor that carries on a line-by-line dialogue is in use, an "end-of-message" is signaled by depressing either the RETURN or LINE FEED key. ESC and then F signals "end-of-file". Each read operation at the terminal specifies a maximum number of characters to be read (never more than 140). If this number is reached, "end-of-message" is signaled. TEL read operations are restricted to a maximum of 80 characters.

TYPING AHEAD

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

SIMULATING TAB STOPS – ESC T

The user can enter tabulation characters into his terminal input, either with the CONTROL and I key combination or ESC I sequence on teletypewriter units, or the TAB key on terminal units that have it. The system simulates tabbing by typing (echoing) successive blank characters. Tab-stop values for this simulation can be set or changed by the TABS command. This tab simulation is under the user's control: it is 'on' at the beginning of a terminal session and set 10, 19, 37, but the user can turn it off, and back on again, with the ESC T key sequence (i.e., successive use of the ESC T sequence has a toggle-switch effect on tab simulation, each use reversing the previous on or off state). With tab simulation on, any tab characters either sent from the terminal or received for transmission to the terminal are replaced at the terminal by an appropriate string of blanks (in lieu of mechanical tabulation). If no tab-stop values are set, each tab character is replaced by a single blank. The state of tab simulation does not determine whether or not blanks are substituted for a tab character in the input stream received by the processor or program requesting the input.

SIMULATING TAB CHARACTERS–ESC I AND CONTROL I

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

SETTING THE TAB RELATIVE MODE – ESC C

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

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

RESTRICTING INPUT TO UPPER CASE – ESC U

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

INTERPRETING UPPER CASE AS LOWER CASE – ESC)

Receipt of the ESC) sequence causes

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

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

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

EXITING THE LOWER CASE INTERPRET MODE – ESC (

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

TYPING COMMANDS

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

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

DETECTING AND REPORTING ERRORS

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

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

The error messages and actions initiated by the errors are contained at the end of each chapter for the processor to which they apply.

INTERRUPTING CP-R

CP-R can be interrupted whenever one of its processors or a user task is in control. Subsequent control depends on which interrupt keys are used and which processor or user program is in control.

CONTROL Y, ESC Y, OR ESC ESC

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

BREAK†

If a processor or a task explicitly requests break control, the task can be interrupted by depressing the BREAK key. This action gives control to the task that has explicitly asked for it (see M:INT).

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

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

ESC Q

Teletype users may request acknowledgment from the system at any time by use of the ESC Q sequence. The system will respond by sending two exclamation points (!!) to the terminal. No other action is taken by the monitor.

†See M:INT Service Call, Chapter 5.

PAPER TAPE INPUT

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

1. Insert the paper tape in the paper tape reader.
2. Depress the X-ON (Q^C) key. This will start the reading of the tape by the paper tape reader under control of the computer.
3. Depress the X-OFF (S^C) key to turn off the paper tape mode (read operation).

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

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

Restrictions:

1. Line feed (LF) characters received after any other activation condition is reached are ignored.
2. The full duplex paper tape facility requires the X-ON, X-OFF option on the Teletype.

HALF DUPLEX PAPER TAPE READING MODE

A special mode is available for half duplex terminals that are reading paper tape. While in this mode, no attempt is made by the monitor to turn the tape reader off or on. Input is accepted until available buffer space is exhausted. No program output, prompt characters, or echoes are sent to the terminal because the mode renders the terminal incapable of accepting output.

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

SUMMARY OF TELETYPE FUNCTIONS

Table 25 summarizes the functions of Teletype terminals.

Table 25. Summary of Teletype Services

Function	Key-in
Get log-on message	BREAK
Erase line	ESC X
Tab relative	ESC C
Suppress lowercase	ESC U
Uppercase shift	ESC (
Lowercase shift	ESC)
Erase last character	RUBOUT
Tab	ESC I, CONTROL I
End of input	NL, RETURN
Line continuation	ESC CR, ESC LF, LOC CR
Retype	ESC R
Toggle tab simulation mode	ESC T
End of file	ESC F
Monitor escape (to TEL)	ESC ESC, CONTROL Y, ESC Y, or 4 BREAKs
Break	BREAK
Half duplex paper tape	ESC P
Toggle ECHO mode	ESC E
Acknowledge	ESC Q
Erase all input and output	CONTROL X

TERMINAL EXECUTIVE LANGUAGE

Once account verification and log-on have been accomplished, the user is in control of his own foreground job which contains one task, TEL. The name of the job and the name of the terminal which has logged are identical. TEL initializes the tab settings to 10, 19, and 37 and assigns all appropriate operational labels for the job to the terminal. Once prompted by TEL the user may begin his activities. These activities include:

Major operations

- Composing program and data files.
- Loading tasks and initiating execution.
- Initiating debugging operations.
- Creating multi-task environment.
- Submitting batch jobs.
- Calling previously submitted batch jobs.
- Calling processors.
- Interrupting, continuing, and terminating execution.

Minor operations

Logging off.

Assigning operational labels.

Setting simulated tab stops.

Stopping and starting tasks.

Terminating tasks.

MAJOR OPERATIONS

An AP or FORTRAN program may be composed on-line by the Edit processor, which is called by the EDIT command. The resultant file may be submitted for assembly and loaded to the batch system.

After successful batching the resultant load module may be executed by use of RUN or INIT. Both primary and secondary may be created at any level, with or without time sharing. Thus a user may create and control any multi-task structure desired.

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

An executing program becomes a static module whenever it is interpreted or whenever an error occurs. This static core module can then be restarted with the CONTINUE or GO command.

MINOR OPERATIONS

The user may assign operational labels, stop and start tasks, previously created with INIT, or terminate these tasks. He may also set tab settings or terminate the entire job by logging off.

ERROR HANDLING

Whenever a syntactical error occurs, TEL outputs a "?" under the offending field. If the syntax is correct but an error occurs subsequently, a "xx?" is output; where, "xx" represents the type completion.

TEL COMMANDS

TABS This command permits up to 12 simulated tabs, in ascending sequence to be set. Whenever a tab character (Control I) is received from a terminal, spaces are sent to the terminal to position the carrier to the next stop. These spaces are also given to the reading program.

This command has the form

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

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

OFF This command indicates that the user is done. All tasks associated with this terminal job will be aborted.

MESSAGE This command causes a message to be sent to the machine operator.

This command has the form

MESSAGE text

where text may be from 1 to 50 characters.

STDLB The assignment thus invoked is permanent for this terminal job.

This command has the form

STDLB label, {device / fid / oplabel}

where

label specifies one of the standard operational labels as defined at SYSGEN. C and OC may not be reassigned.

device specifies a device name to which the operational label is to be assigned (e.g., 9TA81).

fid is a CP-R file identifier for a file to be assigned to the label.

MEDIA This command will cause the file to be listed on the list device presently assigned.

This command has the form

MEDIA fid

where fid is a CP-R file identifier.

BATCH This command causes the file to be entered into the symbiont input queue of batch jobs to be run.

This command has the form

BATCH fid

where fid is a CP-R file identifier.

A batch file may contain only one background job.

JOB This command causes the status of the batch job to be output.

This command has the form

JOB identification

where identification is the hexadecimal value returned by BATCH.

CANCEL This command causes the batch job to be cancelled.

This command has the form

CANCEL identification

where identification is the hexadecimal value returned by BATCH.

SETNAME This command establishes or cancels a relationship between a task name and a file name for the terminal job which uses it. Whenever an INIT service is requested for the SETNAMEd task, the related file name is used as the load module name for the task. The number of relationships which may be established is limited by the JPT parameter of the SYSGEN :RESERVE command.

This command has the form

SETNAME taskname[, load module]

where

taskname is a 1 to 8 character name.

load module is a 1 to 8 character name. If omitted, the currently established relationship for 'taskname' is cancelled.

[RUN] This command causes the load module to be loaded in core and if requested, gives control to DEBUG, and if not, gives control to the program. Note that the command RUN need not be included and is thus implied.

This command has the form

[RUN] taskid[UNDER DEBUG]

where

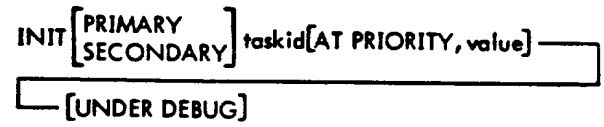
taskid is a CP-R file identifier for the file which contains the desired load module. It has the non-standard default of system processor (or system processor alternate) area and system account, if neither area nor account are specified. Other defaults are standard.

DEBUG identifies the assembly language debugging system.

Note that the RUN command always produces a synchronous stream of tasks. That is, all tasks run in the same task in which TEL runs. See INIT for asynchronous tasking.

INIT This command causes the load module indicated to be loaded in core and either it or DEBUG will be given control. The task thus initialized runs asynchronous to the TEL task (i.e., it is independent of the TEL task).

This command has the form



where

PRIMARY or SECONDARY defines the task being initialized as either a hardware dispatched task or a software dispatched task, respectively. The default is SECONDARY.

taskid is a CP-R file identifier for the file which contains the desired load module. If no equivalence has been established via SETNAME, taskname is the load module name.

value is a 2 or 4 character hexadecimal field for SECONDARY task initialization only. The first 2 characters are the interrupt level minus X'4F' of the dispatcher level to be used. The last two characters represent the software level within the dispatcher level. Software priority values X'F0' – X'FF' are reserved. If the PRIORITY value is not included, the task being initialized will run at the same level as that executing TEL. All tasks will be time sliced as are the TEL tasks.

DEBUG identifies the assembly language debugging system.

DEBUG This command causes the named task to be placed under DEBUG control. Only one task in the terminal job may be associated with DEBUG.

This command has the form

DEBUG [taskname]

where taskname is a 1 to 8 character name associated with the task via INIT. DEBUG may be used without the optional parameter to activate debug in a synchronous task (stated with RUN).

EXIT This command causes the TEL task to exit. The TEL task will not be activated again until a TEL activation sequence is executed from the terminal. This allows the other tasks in the job to use the terminal input facilities in a non-ambiguous manner. An implicit exit is done for the DEBUG command or for an INIT command with a DEBUG option.

STOP Prevents the secondary task from further execution.

START Allows a previously STOP'ed task to continue execution.

The form of the STOP and START command is

{STOP / START} taskname

where taskname is 1 to 8 characters name associated with the secondary task via INIT.

EXTM This command causes the named task to be aborted.

This command has the form

EXTM taskname

where taskname is a 1 to 8 character name associated with the task via RUN or INIT.

QUIT This command may be executed when TEL control is received from a task initiated via RUN (i.e., synchronous). It causes the program to be aborted.

CONTINUE or GO These commands may be executed when TEL control is received from a program initiated via RUN (i.e., synchronous). They cause the program to continue execution.

INTERRUPTING, RESUMING, AND TERMINATING EXECUTIONS

Any processor or user program running synchronously with the TEL task may be interrupted by use of Yc. Upon interruption, all TEL commands are legal except RUN. Before another run may be entered, QUIT must be used.

MUST OPERATIONS

MUST is a proper subset of RADEDIT. Only those commands inappropriate for on-line use are unavailable. The following commands are available:

:ALLOT
:DELETE
:TRUNCATE
:CLEAR
:DUMP
:XDMP
:MAP
:LMAP
:SMAP
:CATALOG
:COPY
:DPCOPY
:END

The command formats may be found in the RADEDIT chapter.

To ease on-line usage, the user will have an optional syntactical variation available. All occurrences of '(', ')', (', or ')' may be replaced by one or more spaces. For example,

:ALLOT (FILE,D3,TEST),(FSIZE,50)

or

:ALLOT FILE,D3,TEST FSIZE,50

are both acceptable.

MUST is called by the RUN command by:

!MUST

TJE ACCOUNT MAINTENANCE

The system manager is responsible for maintaining the AI file in the SP area which contains all legal log-on accounts.

The file must be blocked, sequential, and contains 80-byte records.

The file must be maintained with Edit and contain line numbers in columns 73-80.

The file must be arranged alphabetically with one record per account and multiple subaccount records per account record.

An account record contains the account number beginning in column 1 (e.g., K173021).

A subaccount record, which must immediately follow the account record, contains the account and subaccount separated by a comma (e.g., K173021,AB32).

For example:

```
Column
1 2 3 4 5 6 7 8 9 10 11 12 13                    73 74 75 76 77 78 79 80
A 9 5 2 . . . . . . . . . . . . . . . . . . . . 1 . 0 0 0
A 9 5 2 , C 3 2 1 0 5 . . . . . . . . . . . . . 2 . 0 0 0
A 9 5 2 , N 5 6 . . . . . . . . . . . . . . . . 3 . 0 0 0
K 1 7 3 0 2 1 . . . . . . . . . . . . . . . . . 4 . 0 0 0
K 1 7 3 0 2 1 , A B 3 2 . . . . . . . . . . . . 4 . 1 0 0
K 1 7 3 0 2 1 , X 9 . . . . . . . . . . . . . . 5 . 0 0 0
K 1 7 3 0 2 1 , Z Z Z . . . . . . . . . . . . . 1 0 . 0 0 0
```

STANDARD SYMBOLS, CODES AND CORRESPONDENCES

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) Table 26, CP-R 8-bit Computer Code, i.e., the Xerox Extended Binary-Coded-Interchange Code (EBCDIC); and (2) Table 27, CP-R 7-bit Communication Code, i.e., American National Standard Code for Information Interchange (ANSCII).

STANDARD CHARACTER SETS

1. EBCDIC (Table 26).

   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 (Table 27).

   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 Table 28, CP-R Symbol-Code Correspondences and Table 29, ANSCII Control-Character Translation Table.

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 26. CP-R 8-Bit Computer Codes (EBCDIC)

Least Significant Digits: Hexadecimal	Binary	Most Significant Digits: 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	NUL	DLE	LF only	ESC F	SP	&	-						SP		-	0
1	0001	SOH	X-ON	FS	CAN			/		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					d	m	u	[[1]	D	M	U	4
5	0101	HT	LF NL	EM	ESC (					e	n	v	] [1]	E	N	V	5
6	0110	ACK	SYN	/	ESC)					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					h	q	y		H	Q	Y	8
9	1001	ENQ	EM	CR only	ESC E					i	r	z		I	R	Z	9
A	1010	NAK	SUB	EOT	ESC C	¢ [2]	!	¬ [1]									
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				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.

Table 27. CP-R 7-Bit Communication Codes (ANSCII)

Least Significant Digits: Decimal (rows)	Binary	Most Significant Digits: 0 (x000)	1 (x001)	2 (x010)	3 (x011)	4 (x100)	5 (x101)	6 (x110)	7 (x111)
0	0000	NUL	DLE	SP	0	@	P	`	p
1	0001	SOH	DC1	! [5]	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
10	1010	LF NL	SUB	*	:	J	Z	j	z
11	1011	VT	ESC	+	;	K	[[4]	k	{
12	1100	FF	FS	,	<	L	\	l	¦
13	1101	CR	GS	-	=	M	] [4]	m	} [4]
14	1101	SO	RS	.	>	N	^ [4]	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 ⌊⌉.)

Table 28. CP-R 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	
02	2	STX	12-9-2	0-2	start of text	
03	3	ETX	12-9-3	0-3	end of text	
04	4	EOT	12-9-4	0-4	end of transmission	
05	5	HT	12-9-5	0-9	horizontal tab	
06	6	ACK	12-9-6	0-6	acknowledge (positive)	
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 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	
13	19	DC3	11-9-3	1-3	device control 3	DC3 is RS on 2741s, 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	LF outputs CR and LF.
16	22	SYN	11-9-6	1-6	sync	
17	23	ETB	11-9-7	1-7	end of transmission block	
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	
1E	30	RS	11-9-8-6	1-14	record separator	
1F	31	US	11-9-8-7	1-15	unit separator	
20	32	LF only	11-0-9-8-1	1-5	line feed only	20 through 2F are used by COC for output only. These codes are duplicates of the label entries that caused activation. The 20-2F entries output a single code only and are not affected by any special COC functional processing.
21	33	FS	0-9-1	1-12		
22	34	GS	0-9-2	1-13		
23	35	RS	0-9-3	1-14		
24	36	US	0-9-4	1-15		
25	37	EM	0-9-5	1-9		
26	38	/	0-9-6	2-15		
27	39	'	0-9-7	5-14		
28	40	=	0-9-8	3-13		
29	41	CR only	0-9-8-1	0-13	carriage return only	
2A	42	EOT	0-9-8-2	0-4		
2B	43	BS	0-9-8-3	0-8		
2C	44	)	0-9-8-4	2-9		
2D	45	HT	0-9-8-5	0-9	tab code only	
2E	46	LF only	0-9-8-6	1-5	line feed only	
2F	47	SUB	0-9-8-7	1-10		
30	48	ESC F	12-11-0-9-8-1		end of file	30 through 3F cause COC to perform special functions.
31	49	CANCEL	9-1		delete all input and output	
32	50	ESC X	9-2		delete input line	
33	51	ESC P	9-3		toggle half-duplex paper tape mode	
34	52	ESC U	9-4		toggle restrict upper case	
35	53	ESC (	9-5		upper case shift	
36	54	ESC)	9-6		lower case shift	
37	55	ESC T	9-7		toggle tab simulation mode	
38	56	ESC S	9-8		toggle space insertion mode	
39	57	ESC E	9-8-1		toggle echo mode	
3A	58	ESC C	9-8-2		toggle tab relative mode	
3B	59	ESC LF	9-8-3		line continuation	
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 28. CP-R Symbol-Code Correspondences (cont.)

EBCDIC† Hex.	EBCDIC† Dec.	Symbol	Card Code	ANSCII††	Meaning	Remarks
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			
43	67		12-0-9-3			
44	68		12-0-9-4			
45	69		12-0-9-5			
46	70		12-0-9-6			
47	71		12-0-9-7			
48	72		12-0-9-8			
49	73		12-8-1			
4A	74	¢ or `	12-8-2	6-0	cent or accent grave	
4B	75	.	12-8-3	2-14	period	
4C	76	<	12-8-4	3-12	less than	
4D	77	(	12-8-5	2-8	left parenthesis	
4E	78	+	12-8-6	2-11	plus	
4F	79	| or ¦	12-8-7	7-12	vertical bar or broken bar	
50	80	&	12	2-6	ampersand	
51	81		12-11-9-1			
52	82		12-11-9-2			
53	83		12-11-9-3			
54	84		12-11-9-4			
55	85		12-11-9-5			
56	86		12-11-9-6			
57	87		12-11-9-7			
58	88		12-11-9-8			
59	89		11-8-1			
5A	90	!	11-8-2	2-1	exclamation point	
5B	91	$	11-8-3	2-4	dollars	
5C	92	*	11-8-4	2-10	asterisk	
5D	93	)	11-8-5	2-9	right parenthesis	
5E	94	;	11-8-6	3-11	semicolon	
5F	95	~ or ¬	11-8-7	7-14	tilde or logical not	
60	96	-	11	2-13	minus, dash, hyphen	
61	97	/	0-1	2-15	slash	
62	98		11-0-9-2			
63	99		11-0-9-3			
64	100		11-0-9-4			
65	101		11-0-9-5			
66	102		11-0-9-6			63, 65, 68, and 69 are unassigned.
67	103		11-0-9-7			
68	104		11-0-9-8			
69	105		0-8-1			
6A	106	^	12-11	5-14	circumflex	On Model 7015 ^ is ^ (caret).
6B	107	,	0-8-3	2-12	comma	
6C	108	%	0-8-4	2-5	percent	
6D	109	_	0-8-5	5-15	underline	Underline is sometimes called "break character"; may be printed along bottom of character line.
6E	110	>	0-8-6	3-14	greater than	
6F	111	?	0-8-7	3-15	question mark	
70	112		12-11-0			
71	113		12-11-0-9-1			
72	114		12-11-0-9-2			
73	115		12-11-0-9-3			
74	116		12-11-0-9-4			73, 75, 77, and 78 are unassigned.
75	117		12-11-0-9-5			
76	118		12-11-0-9-6			
77	119		12-11-0-9-7			
78	120		12-11-0-9-8			
79	121		8-1			
7A	122	:	8-2	3-10	colon	
7B	123	#	8-3	2-3	number	
7C	124	@	8-4	4-0	at	
7D	125	'	8-5	2-7	apostrophe (right single quote)	
7E	126	=	8-6	3-13	equals	
7F	127	"	8-7	2-2	quotation mark	

†Hexadecimal and decimal notation.

††Decimal notation (column-row).

Table 28. CP-R Symbol-Code Correspondences (cont.)

EBCDIC† Hex.	EBCDIC† Dec.	Symbol	Card Code	ANSCII††	Meaning	Remarks
80	128		12-0-8-1			80 is unassigned.
81	129	a	12-0-1	6-1		81-89, 91-99, A2-A9 comprise the lowercase alphabet. Available only in 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			
8B	139		12-0-8-3			8A through 90 are unassigned.
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			
9B	155		12-11-8-3			9A through A1 are unassigned.
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			
AB	171		11-0-8-3			AA through AE are unassigned.
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	
B3	179	}	12-11-0-3	7-13	right brace	
B4	180	[	12-11-0-4	5-11	left bracket	
B5	181	]	12-11-0-5	5-13	right bracket	On Model 7015, [is ↓. On Model 7015,] is ¬.
B6	182		12-11-0-6			
B7	183		12-11-0-7			
B8	184		12-11-0-8			B0 and B6 through B8 are unassigned.
B9	185		12-11-0-9			
BA	186		12-11-0-8-2			
BB	187		12-11-0-8-3			
BC	188	[	12-11-0-8-4		left bracket	BC, BD, and BF are used by COC for output of ANSCII 5-11, 5-13, and 7-14, respectively.
BD	189	]	12-11-0-8-5		right bracket	
BE	190	last data	12-11-0-8-5		last data	
BF	191	¬	12-11-0-8-7		logical not	

†Hexadecimal and decimal notation.

††Decimal notation (column-row).

Table 28. CP-R Symbol-Code Correspondences (cont.)

EBCDIC† Hex.	EBCDIC† Dec.	Symbol	Card Code	ANSCII††	Meaning	Remarks
C0	192	SP	12-0	2-0	blank	Output only.
C1	193	A	12-1	4-1		C1-C9, D1-D9, E2-E9 comprise the
C2	194	B	12-2	4-2		uppercase alphabet.
C3	195	C	12-3	4-3		
C4	196	D	12-4	4-4		
C5	197	E	12-5	4-5		
C6	198	F	12-6	4-6		
C7	199	G	12-7	4-7		
C8	200	H	12-8	4-8		
C9	201	I	12-9	4-9		
CA	202		12-0-9-8-2			CA through CF are unassigned.
CB	203		12-0-9-8-3			
CC	204		12-0-9-8-4			
CD	205		12-0-9-8-5			
CE	206		12-0-9-8-6			
CF	207		12-0-9-8-7			
D0	208		11-0			D0 is unassigned.
D1	209	J	11-1	4-10		
D2	210	K	11-2	4-11		
D3	211	L	11-3	4-12		
D4	212	M	11-4	4-13		
D5	213	N	11-5	4-14		
D6	214	O	11-6	4-15		
D7	215	P	11-7	5-0		
D8	216	Q	11-8	5-1		
D9	217	R	11-9	5-2		
DA	218		12-11-9-8-2			DA through DF are unassigned.
DB	219		12-11-9-8-3			
DC	220		12-11-9-8-4			
DD	221		12-11-9-8-5			
DE	222		12-11-9-8-6			
DF	223		12-11-9-8-7			
E0	224	–	0-8-2	2-13	minus	Output only. E1 is unassigned.
E1	225		11-0-9-1			
E2	226	S	0-2	5-3		
E3	227	T	0-3	5-4		
E4	228	U	0-4	5-5		
E5	229	V	0-5	5-6		
E6	230	W	0-6	5-7		
E7	231	X	0-7	5-8		
E8	232	Y	0-8	5-9		
E9	233	Z	0-9	5-10		
EA	234		11-0-9-8-2			EA through EF are unassigned.
EB	235		11-0-9-8-3			
EC	236		11-0-9-8-4			
ED	237		11-0-9-8-5			
EE	238		11-0-9-8-6			
EF	239		11-0-9-8-7			
F0	240	0	0	3-0		
F1	241	1	1	3-1		
F2	242	2	2	3-2		
F3	243	3	3	3-3		
F4	244	4	4	3-4		
F5	245	5	5	3-5		
F6	246	6	6	3-6		
F7	247	7	7	3-7		
F8	248	8	8	3-8		
F9	249	9	9	3-9		
FA	250					
FB	251					
FC	252					
FD	253					
FE	254		12-11-0-9-8-6			FE is not assigned.
FF	255	DEL	12-11-0-9-8-7		delete	Special – neither graphic nor control symbol.

†Hexadecimal and decimal notation.

††Decimal notation (column-row).

Table 29. 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	None	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	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	DC1	None	DC1 (11)	DC1
DC2 (12)	R^{c}	DC2	DC2	None	DC2 (12)	DC2
DC3 (13)	S^{c}	DC3	DC3	None	DC3 (13)	DC3
DC4 (14)†	T^{c}	DC4	DC4	None	DC4 (14)	DC4
NAK (15)†	U^{c}	NAK	NAK (15)	None	LF/NL (15)	CR and LF (0A)
SYN (16)†	V^{c}	SYN	SYN	None	SYN (16)	SYN
ETB (17)†	W^{c}	ETB	ETB	None	ETB (17)	ETB

†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 29. ANSCII Control-Character Translation Table (cont.)

Input					Output	
ANSCII	TTY Key	Echoed	Prog. Receives (EBCDIC)	Process	EBCDIC	Transmitted (ANSCII)
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	None	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)	} (7E)
~(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 26 and 27. All special graphics map as shown, allowing for Table 26, 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 26 and 27; for Models 33 and 35, and 7015 terminals, however, lower case alphabetics are automatically translated to upper case.	

15. SYSTEM GENERATION

System generation adapts the CP-R system to the specific requirements of a user environment and configuration. The system generation process consists of four phases: BOOT26, SYSGENLOAD, SYSGEN, and SYSLOAD.

BOOT26

Is a simple ROM bootstrap which only loads the SYSGENLOAD ROM from the same device as the original bootstrap.

SYSGENLOAD

Is a more complex ROM loader which loads the SYSGEN/SYSLOAD ROM. Setting SSW3 allows for the selection of a ROM source other than the bootstrap device.

SYSGEN/SYSLOAD

Is a single ROM, but runs as two separate processes.

SYSGEN

Reads control cards and constructs all system tables. It may either go directly into the SYSLOAD process, or it may output a rebootable memory image which contains the SYSLOAD process and the constructed tables. Setting SSW3 allows the operator to select the command input and listing devices, otherwise default device addresses and types will be assumed, (see :SYSGEN description).

SYSLOAD

Reads the system ROMs and forms the system on the system disk. Optionally, a map of both memory and disk allocation may be printed, (see :SYSLD description).

A sample map output by SYSGEN is illustrated in Figure 29.

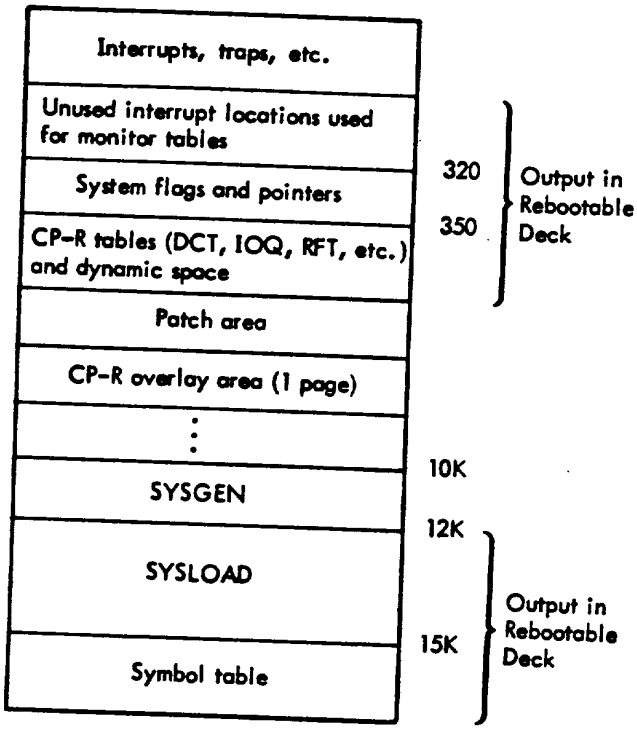
Control is transferred to SYSLOAD either following the completion of the SYSGEN operation or loading of the rebootable SYSLOAD deck.

Sense switch settings affect the way SYSGEN operates. These effects will be described under "Input Parameters".

MEMORY ALLOCATION

MEMORY LAYOUT AFTER SYSGEN

After SYSGEN has executed and before control is transferred to SYSLOAD, memory has the layout displayed as follows:



CP-R STRUCTURE

The control program, or monitor, is assembled in several different modules, the largest of which consists of the following nonoptional resident routines:

1. CP-R Control Task.
2. Control Panel Task.
3. Tasks to process the various traps.
4. The following monitor functions: Dynamic space management, overlay manager, entry and exit logic, and general CAL exit/entry logic and I/O control.

The other CP-R modules consist of the optional resident routines, the CP-R overlays, and the Job Control Processor (JCP). All CP-R is assembled as relocatable object modules and loaded by SYSLOAD.

The optional resident routines are the floating-point simulation routines and decimal simulation routines; they are input during SYSLOAD as required.

```
$SITE 550 PROTOTYPE 1 CPR D00
$PROC     (MIOP,22),(MIOP,23)
$PROC     (MI,26),(PI,27)
$PROC       (MIOP,0),(MI,6),(PI,7),(CPU,5)
$MONITOR  (CORE,64)$
$         (SPAI,D1)$
$         (ACCOUNT,F8)$                    FORE/BACKGROUND ACCOUNTING
$         (ERROR)$                         ERROR LOGGING
$         (BOUND)$                         BOUND RESIDENT OVERLAYS
$         (OLAYEP,1000)$
$         (CRASH)$                         CRASH SAVING
$         (DEBUG)$                         DEBUG CAPABILITIES
$         (REBOOT)$
$         (SSOLAY,LOAD)$
$         (ROLAY,TRAPS)                    TRAP MODULE RESIDENT
$RESERVE  (FFPOOL,2)$                      FOREGROUND BLOCKING POOLS
$         (TSPACE,2000)$
$         (FRAD,10)$
$         (BRAD,10)$
$         (RSDF,12,5800)$                  USE FOR PRIMARIES
$         (FIOQ,10)$                       FOREGROUND IOQ ENTRIES
$         (BIOQ,20)$                       BACKGROUND IOQ ENTRIES
$         (FMBOX,20)$                      FOREGROUND MAIL BOXES
$         (BT,6)$                          BACKGROUND TEMP FILES
$         (MPATCH,256)$
$         (TASKS,15)$                      MAXIMUM # OF TASKS
$         (JOBS,10)$                       MAXIMUM # OF JOBS
$         (JPT,25)$
$         (JENQ,3)$
$         (TENQ,2)$
$         (TASKS,24)$
$         (LDMD,20)
$COC (DEVICE,A05),(LINES,7),(HARDWIRE,1,0)
$COMMENT RING BUFFER DEFAULTS TO 2** OF LINES
$COMMENT COCBUFFERS DEFAULT TO NUMBER OF LINES
$COMMENT INTERRUPTS DEFAULT TO 60 AND 61 AS IN AND OUT
$CHAN
$DEVICE   TYA01,4592
$CHAN
$DEVICE   LPA02,3465,(DEDICATE,S)
$CHAN
$DEVICE   CRA03,7120,(DEDICATE,S)
$CHAN
$DEVICE   TYA0B,4591
$CHAN
$DEVICE   9TA80,3335
$DEVICE   9TA81,3345
$DEVICE   9TA82,3347
$DEVICE   9TA83,3345
$CHAN
$DEVICE   DPAE0,7270$
$          (SP,2100)$
$          (CK,420)$
$         (D1,1400,B)$
$         (BT,2400)$
$         (IS,20,50)$
$         (OS,50,250)$
$         (FP,1800)$
$         (BP,1800)$
$         (DS,ALL,B)
$DEVICE   DPAE1,7270
$CHAN
$DEVICE   DPVC0,3243
$DEVICE   DPVC1,3243
$DEVICE   DPVC2,3243
```

Figure 29. SYSGEN Map Example

```
$DEVICE  DPVC3,3243
$DEVICE  DCVC6,3214
$CHAN
$DEVICE  DPWF0,3283
$CHAN
$DEVICE  DPWF1,3277
$CHAN
$DEVICE  DPWF2,3283
$STDLN   (DC,TYA01);
$        (LP,LPA02);
$        (CR,CRA03);
$        (T0,9TAP0);
$        (T1,9TAP1);
$        (T2,9TAP2);
$        (T3,9TAP3);
$        (C,CR);
$        (LO,LP);
$        (LL,LO);
$        (DO,LP);
$        (BO,T1);
$        (SI,CP);
$        (CI,CP);
$        (BI,T2);
$        (BI,T3);
$        (MO,LP);
$        (DI,DC);
$        (DL,LP);
$        (SE,T1);
$        (ER,0);
$        (P1,LP);
$        (P2,LP);
$        (DP,CR)
$CTINT   (CT,64),(MI,69)
$INTLN   (I0,62);
$        (I1,63);
$        (I2,64);
$        (I3,65);
$        (I4,66)
$ALLOBT  (GO,10);
$        (OV,20)
$SYSLD   (IN,9TAP3,3345);
$        (MAP,LPA02,3465);
$        (V,D00);
$        (UPD)

 PATCH FWA=01C00
 FMBOX FWA=00014
 FGD,  FWA=05AC0
 BCKG, FWA=06000


**** CPR TABLE ALLOCATION ****
DCT1     =006A   DCT1P    =006A   DCT1A    =006A   DCT2     =0074   DCT3     =007B
DCT4     =0082   DCTDISCI=0089   DCT5     =008F   DCT6     =0095   DCT7     =009C
DCT8     =00A5   DCT9     =00B7   DCT10    =00CA   DCT11    =00D3   DCT12    =00E5
DCT13    =00F6   DCT14    =011C   DCT15    =0123   DCT16    =021A   DCT17    =0128
DCT18    =0132   DCT19    =0139   DCT20    =024E   DCT20A   =0253   DCT21    =0258
DCTDEBUG=0262    DCTRBM   =0267   DCTDCB   =026C   DCTJID    =0273   DCTMOD   =0279
DCTMODX  =0293   DCT#IO   =029C   DCT#ERR  =02AF   DCTSDBUF=02C0   DCTTJE   =02D3
DCTSYM1  =02DA   DCTSYM2  =02DE   DCTSYM3  =02F1   DCTCD    =02F6   DISCNSPT=02FB
DISCNWPS=02FD    DISCMAXS=02FF   DISCMINS=0304   DISCSSFT=030A   DISCTSFT=030C
DISCCSFT=030E    DISCNTPC=0310   DISCNCYL=0313   LOGFLAG  =013E   SITE     =0316
CNFGADDR=0324    CNFGTYPE=0327   CNFGSTAT=0329   MDDCTI   =0332   MDDISCI  =0335
MDBOA    =0338   MDEOA    =0342   MDFLAG   =034C   MDNAME   =034F   S:SPAI   =0006
S:ISFSIZ=0032    S:OSFSIZ=00FA   S:JENQ   =0003   S:TENQ   =0002   S:DEBUG  =0140
CIT1     =04DC   CIT2     =04DF   CIT3     =04E2   CIT5     =04E5   CIT6     =0000
```

Figure 29. SYSGEN Map Example (cont.)

```
IOQ1     =04E8   IOQ2     =04F0   IOQ3     =04F8   IOQ4     =0500   IOQ5     =0508
IOQ7     =0510   IOQ8     =0517   IOQ9     =0536   IOQ10    =0546   IOQ11    =054E
IOQ12    =0555   IOQ13    =0572   IOQ14    =0580   IOQECB   =0587   IOQFRHDR=05D5
RFT1     =05F2   RFT2     =0625   RFT3     =0647   RFT4     =0664   RFT5     =0673
RFT6     =0681   RFT7     =069E   RFT8     =06A6   RFT9     =06AE   RFT10    =06B6
RFT11    =06C4   RFT12    =06E0   RFT13    =06FD   RFT14    =0705   RFT15    =070D
RFT16    =0715   RFT17    =071C   RFTESZ   =073A   RFTE#    =0755   RFTACNT  =0762
LMI1     =079A   LMINAME  =079A   LMI2     =07C7   LMIPCB   =07C7   LMIFWA   =07C7
LMI3     =07DC   LMIJID   =07DD   LMILWA   =07DD   LMI4     =07F3   LMIPL    =07F3
LMICTXT  =07F3   LMI5     =080A   LMISTAT  =080A   LMI6     =0815   LMISDT   =0815
LMIRTS   =082A   LMI8     =0858   LMIMAXS  =0858   LMI9     =085E   LMIMAXR  =085E
LMIUSERS=085F   LMI#     =085F   LMISECB  =0863   LMIRECB  =0879   LMIAET   =088F
LMIRFT   =08A5   STISPCE  =08B8   STIXRTS  =08D8   STIRTSB  =08F4   STIJID   =0930
STILMID  =0938   STIPRIO  =093F   STITCB   =095C   STIOVID  =097A   STICOUNT=0989
STITIME  =0990   STIDNXT  =09AE   STIRNXT  =09B6   STISTAT  =09BE   STITICK  =09C6
STIQMIN  =09D5   STIQMAX  =09DD   RDLIPRIO=09E5   RDLISTI  =09E6   RDLITCB  =09E6
RDLIADD  =09E8   RDLILVL1=09E9   RDLIGRP1=09EA   RDLILVL2=09EB   RDLIGRP2=09EC
SJI1     =09EC   SJI2     =09F8   SJI3     =0A12   OPLB1    =0A16   OPLB2    =0A23
OPLB3    =0A2A   S:SD     =0A31   S:EDT    =0A32   ALARMREC=0A33   INTLB1   =0A4E
INTLB2   =0A51   OVLOAD1  =0A54   OVLOAD2  =0A88   OVISK    =0AF0   OVLOAD3  =0B59
OVIMA    =0B74   OVIECB   =0BA8   OVICT    =0C11   FPIEP    =0C2B   EPIB     =0FEC
S:TRACE  =0FED   S:TRACEN=0FEE   S:TRACES=0FEF   S:TRACET=0FF0   S:TRACEP=0FF1
RMPT     =0FF0   PPT      =0FF7   WLOCK    =0FFE   S:UTIME  =01F4   S:JPT    =0019
IOLOCK   =100F   MMROLTAB=36A3   S:XBB    =0000   COM:DN   =1037   CO:AIIL  =1038
CO:ADIL  =1039   CO:IIL   =103A   CO:DIL   =103B   CO:STAT  =103C   CO:OUTRS=103D
CO:RCVDN=103E   CO:XDATA=103F   CO:TRNDO=1040   CO:RCVDO=1041   CO:XSTOP=1042
CO:LST   =1043   CO:RINGE=1044   COM:RBS  =1045   CO:OUT   =1046   COM:II   =1047
COM:IO   =1048   COCMPB   =1049   COCOC    =104A   LB:UN    =104C   ARSZ     =104E
BUFCNT   =1050   MODE     =1052   MODE2    =1054   MODE3    =1056   MODE4    =1058
MODECPR  =105A   COCTERM  =105C   RSZ      =105E   CPI      =1060   CPOS     =1062
COCII    =1064   COCIR    =1068   COCOI    =106C   COCOR    =1070   TL       =1074
EOMTIME  =1078   COD:LPC  =107C   COD:HWL  =107E   CO:INO   =1080   CO:INN   =0000
CO:OUTO  =1084   CO:CMND  =108A   COCBUF   =108E   MRBA     =0050   LCOC     =0000
LNOL     =0007   COA:IIG  =0002   COA:OIG  =0002   S:TEMPN  =0910   S:TEMP   =10F0
OLAYFWA  =1A00
6KLIMIT  =5030   LDPRE    =50D0   LDPOST   =512F   COCOP    =3EBB   COCIP    =3C4C
COCINIT  =4151   COCNTT   =0003   TJE:NUM  =3A51   TJE:NOW  =3A52   TEXWORKS=3A50
TEXSTCB  =3A54   TEXSTACK=3A85   MEDRWB   =3949   MEDRWA   =3942   MEDRVFCB=3971
MEDRVFCA=3968   MEDRTASK=3914   MEDRSTPS=3977   MEDRSTCB=397C   MEDRSTAK=39AD
MEDRRB   =393A   MEDRRA   =3932   MEDROTMP=3917   MEDRODCB=3927   MEDROC   =3958
MEDRMASN=3910   MEDRJOB   =3912   MEDRITMP=3916   MEDRIMAG=3A38   MEDRIDCB=391C
MEDRFPTX=395C   MEDRERPS=391A   MEDRCTRL=3910   MEDRCHKW=3954   MEDRCHKR=3950
MEDRBB2  =3919   MEDRBB1  =3918   MEDIARES=3910   ROLL99   =36F6   MMTSLEV  =36FB
MMTSK99  =36F7   MMTPREQ  =36A1   MMTPAGES=36A0   MMTBKGP  =36A2   MMSTCB   =36BC
MMRTS    =36FC   MMRSTIME=36F8   MMRSRGSV=36ED   MMRPRIO   =36FA   MMROLMAP=36AB
MMRMAPLG=36A6   MMRLOOP   =36A7   MMRLEVEL=36F9   MMRINDX   =36A4   MMRGF:RE=36BB
MMRENDX  =36A5   MMAST    =36EC   XJOB     =35D9   XFIN     =35DA   XEXU     =35D6
MCBKG    =35DC   MCSTART  =367D   MCSIG    =3683   MCJOBN   =364D   MCJOB    =3649
MCPPT    =3657   MCDCB    =3650   MCBUF    =3658   TESTACAL=35C3   SYMCRSH  =3613
SYMCOUNT=35E8   SYMBSTCB=3490   SYMBSTAK=34C1   SYMB     =35E7   STOPME   =3557
SMSG7    =360E   SMSG6    =3607   SMSG5    =3603   SMSG4    =35FE   SMSG3    =35FA
SMSG2    =35F6   SMSG1    =35F0   SMENDAC  =3693   SMCUP3   =3698   SM:C     =35E9
SIZEOS   =0028   SIZEIS   =0018   SIZEFPTJ=0006   SEX1     =3691   SEXE1    =35D1
RCTIMER  =3645   RCPOST   =3642   RCPOLL   =363F   RCPBITS  =3633   RCOPEN   =3634
RCLOSE   =3638   RCGIOBLK=362D   RCFPT    =3619   RCDLTE   =363C   RCDCB    =361D
RCASSIGN=3628   RCAREA   =362E   RCACNT   =3631   FPTWEOF  =35AE   FPTSTMED=35A6
FPTSTIM  =35AD   FPTSIG   =35B3   FPTPSTBK=35B8   FPTPRECB=35CC   FPTPREC  =35C7
FPTPOLBK=35B8   FPTPFILB=35C0   FPTOPEN  =3598   FPTLP    =3558   FPTJOBDL=35AB
FPTJOB   =35A0   FPTDELF  =35B1   FPTCR    =3580   FPTCLOS  =359C   FMALLOT  =368D
DFLTP    =35D2   CKSIG    =3689   RFJJJ:00=35D7   BATCHMSG=35DE   BATCHF   =35EA
ADDRRW   =35EC   ADDRMSC  =35EF   SCALL8D  =2882   494A4B  =1DBA   4CHAR   =2E34
0001FE00=1D6F   ZEROS    =1DC0   Y82      =1DA4   Y8       =1D00   Y7F      =1D89
Y69      =1DA2   Y68      =1DA1   Y67      =1DA0   Y60      =1D9F   Y6F      =1DA3
Y6       =1D98   Y48      =1D9A   Y4       =1D01   Y31      =1D99   Y3030    =1D80
Y3       =1D98   Y22      =1D9E   Y2       =1D02   Y155A5A  =1D7D   Y15      =1D9C
Y1C      =1D9D   Y1       =1D03   Y083     =1D8F   Y08      =1D04   Y07      =1D94
```

Figure 29. SYSGEN Map Example (cont.)

```
Y05       =1D93   Y04       =1D05   Y03       =1D92   Y02       =1D06   Y01       =1D07
Y008      =1D08   Y007F      =1D83   Y007       =1D8D   Y006      =1D8E   Y004      =1D09
Y003      =1D8C   Y002      =1D0A   Y0011     =1D88   Y001      =1D0B   Y0008     =1D0C
Y0006     =1D8A   Y0002     =1D0E   Y000A      =1D81   Y00FF     =1D7F   Y00FE     =1D82
Y00E      =1D91   Y00C      =1D90   Y0F        =1D97   Y0C       =1D96   Y0A       =1D95
YFF8      =1D86   YFF7E     =1D85   YFF4       =1D87   YFFFFFF   =1D7C   YFFFF     =1D7E
YFFFE     =1D84   YFF       =1D88   YF         =1DAA   YE        =1DA9   YD        =1DA8
YC        =1DA7   YBE       =1DA5   YA         =1DA6   X9        =1D35   X8000     =1D10
X80       =1D18   X8        =1D34   X70        =1D3E   X7FFFF    =1D2E   X7FFF     =1D28
X7F       =1D27   X7D       =1D3F   X7         =1D23   X6000     =1D50   X60       =1D3D
X6        =1D38   X5        =1D22   X404040    =1D58   X4000     =1D11   X40       =1D19
X4        =1D1D   X380      =1D4C   X3000      =1D4F   X30       =1D3C   X3F       =1D26
X3        =1D20   X20200    =1D57   X20000     =1D0E   X2000     =1D12   X200      =1D16
X20       =1D1A   X2        =1D1E   X1800      =1D4E   X1700     =1D4D   X1000001 =1D5C
X10000    =1D0F   X1000     =1D13   X10        =1D18   X1FFFF    =1D2D   X1FF      =1D29
X1        =1D1F   XPSDCMMD  =1DAD   XF7        =1D47   XF0       =1D46   XFF8F     =1D53
XFF7F     =1D52   XFFF7     =1D55   XFFF0      =1D54   XFFFF00   =1D5A   XFFFFFF   =1D2F
XFFFF     =1D2C   XFFFE     =1D56   XFF        =1D28   XFE7D     =1D51   XFE       =1D48
XFD       =1D4A   XFC       =1D49   XF8        =1D48   XF        =1D24   XE7       =1D44
XE0       =1D43   XEF       =1D45   XE         =1D3B   XD60      =1D39   XD3600    =1D3A
XDF       =1D42   XCF       =1D41   XBF        =1D40   XB        =1D36   XA        =1D37
UTIMES    =1E39   UTIME     =1E38   USRCOM     =2E83   TTICKS    =21CF   TTFINAL   =321F
TSISTICK  =21CE   TSISEC    =21CD   TSTICK     =21CC   TRTN10    =312C   TRAP90    =311E
TRAP50    =30CE   TRAP46    =3074   TRAP45     =306A   TRAP44    =3060   TRAP43    =3056
TRAP42    =304C   TRAP41    =3042   TRAP40     =3030   TRAP4D    =30B6   TRAP4C    =309C
TRAPX     =2FC7   TRAP      =302E   TMX5       =23CB   TMX4      =23D1   TMX2      =23CF
TMX1      =23CE   TMVALF3   =2454   TMTRIG     =2410   TMTPRIO   =23F7   TMSETERR  =241C
TMREECB   =2491   TMRDLTRG  =2417   TMENQGO    =247F   TMDQR     =21EA   TMDQA     =21FE
TMDQ      =21D4   TEMP      =1DF2   TDTRIG     =1E38   TDSTLIDL  =217B   TDSTL     =2130
TDRDLVL   =1E36   TDRDLRET  =20E9   TDRDLGP     =1F37   TDRDL     =20E2   TDDISPX   =21C6
TASK99    =1E11   TSLDEV    =1F3A   SYSACNT     =1F50   SYMREENT  =1E16   SPRPSD    =1DEC
SIGPOST   =23D6   SERDEV    =28A0   SENSESW     =2955   SEEKADDR  =2954   SECTPERM  =2786
SECTPERB  =2787   SDPSTACK  =2DCA   SDPOSTER    =2DBA   SCEND     =2288   SC:YEAR   =1E1A
SC:SEC    =1E1C   SC:LIST   =1E1D   SC:INIT     =1E20   SC:DCB     =1E1F   SC:DAY    =1E1B
SC:BBCH   =1E1E   RUN99     =1E10   ROLL98     =1E12   RIPTOFF   =2988   RIPOFF    =297E
RESCHED   =28CB   REQCOM    =2B8F   REPCOM     =2E41   RENT:SN   =1E14   RENT:D    =1E15
RELTEMPI  =328E   RELTEMP   =32C0   RE:ENT     =2F2F   RCS0      =2C05   RC28      =2C1F
RBMSD     =1FB8   RBMSAVE   =1FF5   RBMEXITD   =200B   RBMEXITA  =2007   RBMEXIT   =2003
QUEUE     =2818   QUANTUM   =21D0   QSWAP      =21D2   QREG      =295A   QMIN      =21D1
QMAX      =21D3   PUSHLOG   =2D2E   POST10     =27FA   POST07    =27F8   POST06    =27F6
POST05    =27F4   POST03    =27F2   POST01     =27EF   POSTEOD   =2596   POST      =27FC
PONPSD    =1DE4   POLLPOST  =23E1   POFFPSD    =1DE8   PLSEG#    =1E13   PFITRIP   =2FAE
PFIPSD    =2F20   OUTMSG    =2DD4   OPTIONS    =1E35   OMANTPSD  =3428   OMANPAGE  =3436
OMANFWA   =3437   OMANFPP   =346D   OMANFP     =344F   OMAN#PGS  =3438   OMAN      =330C
OCDCT     =2956   NEXTQUE   =28F5   NEWIOCK    =2E8E   M9        =1D29   M8        =1D28
M7        =1D27   M6        =1D26   M5         =1D25   M4        =1D24   M32       =1D32
M31       =1D31   M3        =1D23   M25        =1D30   M24       =1D2F   M2        =1D20
M19       =1D2E   M17       =1D2D   M16        =1D2C   M15       =1D2B   M12       =1D2A
M1        =1D1F   MTWCMMD   =1DAC   MSG9       =2E2C   MSG6      =2E28   MSG5      =2E23
MSG4B     =2E17   MSG4A     =2E11   MSG2       =2E07   MSG1      =2E05   MSGOUT    =2DD9
MOVEBYTS  =23C5   MFITRIP   =2FAF   MFIPSD     =2F14   L1015     =1DB6   LOSTLOGS  =2D61
LOGSTACK  =2D4E   LASTPRTY  =2F0A   KEYBUF     =32FC   KI:UTIME   =1E3A   KI:DPIDLE  =2E28
JOB99     =1E18   JOBPRI    =1E09   JOB#       =1E08   JCBRBM     =1F3E   JCABKG    =1F70
IOSCU     =2B37   IOPSD     =1DD4   IOLOG      =2C59   IOINC     =2910   IOERROR   =2CA9
IOCRASH   =25A4   IOCLOCK   =1E0C   IOALT      =2ABC   IMAGE     =1E32   GUTEMP    =32BD
GTBL      =32AB   GOODLOGS  =2D60   GINC       =32A3   GETWD     =237A   GETTEMPI  =326A
GETTEMP   =326C   GETPSII   =27B9   GETPSI     =27AF   GETPS     =278D   GETPII    =27B3
GETPI     =27A7   GETP      =2794   GETFPTNS   =278D   GETFPTN   =2794   GETEFADR  =27BD
F7FFFFFF  =1D78   FMRELRAD  =2755   FMQUEUE    =261D   FMGETRAD  =265B   FMDELECB  =2650
FMABORT   =2767   FIGPANEL  =297A   FGLTRIG    =2462   FGLPUBL   =1E34   FGLCBP    =1E32
FGLBBP    =1E33   FGDPBD    =1DD0   FFF7FFFF   =1D6A   FFF1FFFF  =1D6C   FFFF7FFF  =1D66
FFFFF800  =1D7B   FFFFFF00  =1D5A   FFFFFFE1   =1D70   FFFFFFEF  =1D71   FFFFFFBF  =1D75
FFFFFE00  =1D59   FFFFFEFF  =1D60   FFFDFBFF   =1D61   FFFF8FFF  =1D67   FFFEFFFF  =1D65
FFFDFFFF  =1D64   FFFDFDFF  =1D63   FFFDB5FF   =1D62   FFFDFEFF  =1D6B   FEFFFFFF  =1D77
FDFFFFFF  =1D76   FCW7TDEC  =000E   FCW7TBIN   =000B   FCWTMMT   =0003   FCWMT     =0001
FCR7TDEC  =000D   FCR7TBIN  =000A   FCROFFMT   =0009   FCRMT     =0000   FCREWMT   =0008
FCRB9T    =0002   FCFSRMT   =0005   FCFSFMTS   =0014   FCFSFMT   =0007   FCBSRMT   =0004
```

Figure 29. SYSGEN Map Example (cont.)

```
FCBSFMTS=0015  FCBSFMT =0006  FBFFFFFF=1D5D  FBACCNT2=2081  FBACCNT1=207F
EXITABRT=239E  EXIT    =239C  ERROR60 =27ED  ERROR59 =27E9  ERROR58 =27E7
ERROR54 =27E3  ERROR48 =27DF  ERROR44 =27DD  ERROR40 =27D3  ERROR4A =27E1
ERROR2E =27D9  ERROR03 =27D5  ERROR01 =27D3  ERRORCA =27D7  EODREC  =1DA9
ENQPOST =23EA  ENDAC   =2C28  EMRSECB =24E5  EMRRECB =24C7  EMPOSTYC=2526
EMPOST  =2506  EMDELECB=24AD  FFFFFFFF=1D5F  EDT99   =1E0F  DIRECODE=1DC2
DIOPSD  =1DD8  DFFFFFFF=1D7A  DELCOM  =2E81  DEFFFFFF=1D79  DCBERR  =2813
DBGBREAK=316E  DATA1C1D=1DC4  C001FFFF=1D6D  C0000030=1D6E  COCOCOCO=1D72
CVTFILE =25A9  CVTDISC =25CA  CVTAREA =25B6  CUP3POST=1DF0  CUPUCB  =2DAE
CUPCORE =2DA6  CUPCOD4 =1D05  CUPCOD3 =1D92  CUPCOD2 =1D06  CUPCOD1 =1D07
CT1     =224F  CTSTCB  =1E3C  CTRTS   =1E5C  CTRIG   =246F  CTLSUB  =2FF0
CTIOSTK =1E26  CTALTPSD=3020  CRSHXPSD=2328  CRASDLAY=2988  CRASHRES=2950
CRASHREG=2828  CRASHPSD=2E8E  CRASHMSG=2818  CRASHMAP=2838  CRASHLOC=2958
CRASHI3 =2948  CRASHI2 =2940  CRASHI1 =2938  CPPSD   =1DDC  COUNT6S =1E0B
COUNT5S =1E0A  COUNTER4=1E0D  COUNTID =1E17  CONVADDR=27D1  COMLIST =2E3B
CLOCKPSD=1DE0  CLK3SAVE=1FF3  CLK2SAVE=1FF0  CLK1SAVE=1FED  CKXABT  =1DB4
CFLAG   =1FEC  CCRUF   =1FCE  CAL234  =1DB8  CALIPSD =1DCC  CALSTOX =2323
CALREG  =1DF2  CALLSD10=289C  CALLSD  =2868  CALLOP  =25F7  CALLQ   =25F4
CALFLAG =1DAF  CALEXIT =2331  CALERR  =2325  CALENTRY=31E8  BRKSUB  =2FD1
BLANKS  =1DBE  BKGSTOP =227C  BKGSTART=2280  BKALTPSD=300E  BITABLE =1D00
BINITFPT=1E19  BFFFFFFF=1D74  BCRASH0 =2F0A  BALRBMSV=1DB2  BADCAL  =2FC7


**** CPR PROGRAM ALLOCATION ****
JCP     =0EB8  LOAD    =0AB9  TEL     =00D7  FGL1    =00CA  FGL2    =0119
FGL3    =0135  SCHED   =0147  SCNEXT  =012A  ABEX    =01A6  KEYSCN  =00ED
KEY1    =01EF  KEY2    =01C6  KEY3    =01ED  KEY4    =0187  KEY5    =01F9
KEY6    =01F8  KEY7    =00F8  TEX1    =016A  TEX2    =01F7  TEL1    =0135
TEL2    =0149  TEL3    =0133  DUMP    =01E7  BKL1    =0103  ARM     =01FE
ENQ     =01FF  DELETE  =0156  RWBFIL  =017C  FINDBB  =01D8  DEVI    =0129
ASSIGN  =0132  IOEX    =011D  OPENX   =01DA  CLOSEX  =011A  REWIND  =01F4
REWDEV  =0086  IPLHM   =01D2  IPLSYM  =011A  MMD1    =01E8  MMD2    =01FA
MMD3    =01E2  MMD4    =01F8  MMD5    =01F8  MMD6    =014E  PL01    =016B
SIGNAL  =01B5  SJOB    =01ED  SNAM    =0101  RUN     =0171  PINIT   =01F1
STDLB   =01C5  EXTM    =0152  WAIT    =01C9  TIO1    =013A  TIO2    =016A
TT      =015D  TTJOB   =00F4  JOB1    =01EF  JOB2    =00C6  SYM2    =01CB
SYM3    =01F5  SYM5    =00AA  TRAPS   =48A0  TMGETP  =0195  ALLOT   =00FA
ESU     =01BF  CRB     =01B7  CRD     =01D5  CKD     =01DA  CKD2    =017E
LOG     =01FF  CRASH   =00AD  MED1    =01EB  MED2    =01FE  CRS2    =01FC

DBDW    =01C8  DBC1    =01E6  DBC2    =01DD  DBC3    =01F3  DBS1    =01F0
DBS2    =01EF  DBS3    =01F3  PRINT   =01A1  TMTYC   =4D50  CHECK   =4F20
READWR  =4720  SDBUF   =45F0  SYM1    =4960  SYM4    =4AB0  RWFILE  =4510
RWDEV   =43E0  GETNRT  =4820  DISC    =5140  TAPE    =5680  CARD    =5530
TTY     =5030  MEDIA   =3910  LP      =5410  TEX     =3A50  COCIO   =3C30
SEX     =3090  TERM    =4310  TIO3    =3B20  MMROOT  =36A0  ABORT   =43C6
ACTV    =1A2E  ANALYSE =1A88  BKGSEQ  =1A86  BKLASSN =1A39  BREAK   =1A5A
CFUPDIR =1A87  CHECKA  =4F65  CHKBAL  =4F35  CHKBALA =4F33  CKENACT  =4EB6
CKENACTS=4EB5  CKENACT1=4EBA  CKENACT2=4EBA  CKINTADR=4EEF  CKINTLAB=4EE6
CLOSE   =47ED  CLOSEDCB=47F1  CLOSRFIL=1A40  COCRIP  =427D  COCSRDV  =42A1
COCTIME =4107  COOP    =4912  CORRES  =1AE8  CPRSYMBL=1A07  CRFIL   =1A6C
CSEARCH =1A43  DBKG    =1AB3  DCBBUSY =4756  DEACTV  =1A2C  DEBUG   =1B94
DELFPT  =4F25  DEQ     =1A24  DEVN    =1A00  DFGD    =1BDC  DFGDBAL =1BD3
DFM     =1ABE  DISARM  =1A00  DRC     =1B12  DVF     =1B08  EMARECB  =1B1F
EMARECBX=1B1F  EMBLDECB=1A7A  EMDATAI =1AC3  EMDATAO =1AEA  EMGETECB=4DCF
EMGETEM =4DCB  FMGETFPT=4DF8  EMSETR3 =5018  EMSETR3A=5019  EMWAIT  =4E0B
ENQARNM =1AE4  ENQCHK  =1A8E  ERRSEND =1B81  FGLBADLM=1A3C  FGLMEMCK=1A3E
FGLMSG  =1A74  FGLOKLM =1A58  FINDDIR =1A99  FINDDIRX=1A9A  FMBLDECB=48CF
FMCHECK =4FA6  FMCKWP  =45C8  FMCK1   =4FC4  FMCK2   =4FC9  FMCK3   =4FCF
FMDELETE=1A3F  FMJCL   =1A80  FMMASTX =45D8  FMOPL2AD=4843  FMTCL   =1AF8
FPTBSY  =4781  GENCHARS=1A57  GETANAME=1B86  GETDCBAD=4820  GETDCTX=482A
GETIOID =1A09  GETTIME =1BAC  HOURLOG =1B39  IBBPARAM=1B6E  INITLOG =1AC6
INSDBUF =4657  JMTENQ  =1AE3  JMTERM  =1A22  JOBDLTE =1A06  JOBDLTEA=1A04
JOBMSG  =1A9F  JORSCAN =1B5E  JSCAN   =1B74  JTRAP   =4D05  KEY1A04 =1A39
KJOB    =1A76  LOADACI =384F  LOADMAP =3848  LOGPURGE=1B6F  MEDIACAL=1A67
MEDIATSK=1A00  MEDRLOOP=3962  MEDRLOPA=3970  MEDR900 =3961  MEDOEXIT=1B24
```

Figure 29. SYSGEN Map Example (cont.)

```
MED090  =1B30   MED094  =1B4B   MED099  =1B53   MED6N0  =1A00   MED800  =1A55
MED810  =1A5N   MED820  =1ADC   MED830  =1B60   MED840  =1B6C   MED860  =1BAA
MMABNM  =1AB5   MMACT   =1AB4   MMCAL   =1BB0   MMCHECK =1A1F   MMDEACT =1A4A
MMERASE =1A72   MMEXEC  =1A2A   MMFETCH =38BB   MMFMP   =37E4   MMFOV   =1A9C
MMGETP  =1ACC   MMGJRP  =1AE5   MMGP    =1A3B   MMGPPS  =1A00   MMGSTM  =3760
MMGTRP  =1AE2   MMICHK  =1A18   MMLOCK  =1A05   MMMOVE  =38CC   MMOMFPP =38DC
MMPOST  =1AE2   MMRDS   =1B85   MMRECB  =1BB7   MMRELP  =1B0B   MMRELS  =1BEC
MMRELBD =1BC4   MMPFILE =1AD4   MMRILW  =37ED   MMRISEG =1A00   MMRJRP  =1B40
MMROLL  =1A05   MMROLLIN=1A23   MMROUT  =1A00   MMRP    =1A00   MMRPPB  =1A73
MMRPPSI =1A70   MMRPREF =1A33   MMRREAD =1B53   MMRRFILE=1B26   MMRSTM  =37A6
MMRTRP  =1B3A   MMRWRITE=1B87   MMSAC   =1A00   MMSDS   =1B64   MMSEGCK =1B43
MMSETVPN=1BA8   MMSTART =1B2F   MMSTOP  =1A72   MMSTORE =38C4   MMSWAP  =38E8
MMSWLK  =1B8A   MMTJOB  =1BD1   MMTPRIM =1BB2   MMTSEC  =1B48   MMUNLOCK=1A00
MMVVPN  =1B1D   MODIFY  =1A85   OFFVERBG=1B3B   ONOFFMSG=1AF0   OPEN    =47D5
OPENDCB =47F6   OSEARCH =1A18   OUTSDBUF=45F1   PFIL    =1AEB   PFILDEV =1A5C
PINTABNM=1B38   PMD     =1A90   POLL    =1A7B   POLLABNM=1B0C   POLLCHK =1ABF
PPOST   =1B0F   PRECDEV =1A48   PRECORD =1A23   PREFMODE=1AC0   PROMPT  =1B1B
PUBLIB  =1A00   RBLOCK  =1B50   RCCUPF  =1A94   RCCUPJ  =1A88   RCEXU   =1A29
RCFEOD  =1A87   RCGETF  =1A07   RCJOB   =1A4C   RCOOP   =4AB4   READDIR =1BB5
RECALARM=1B9D   RELADBUF=1A5C   RETELING=1A0A   RLS     =1ACE   RWRFILE =0000
RWUFILE =0000   SIOLC   =0000   SCAN    =1A8A   SCEMPTY =1A02   SCFIND  =1A06
SCHEDC  =0000   SCMSG   =1AD2   SCUPDATE=1AA9   SEARCHAI=1AB6   SEGLOAD =1AEF
SETNAME =1A00   SETOVR  =4923   SETPRI  =1A7A   SETUP   =1BC8   SIGABNM =1A51
SIGCHK  =1A30   SIGNAL1 =1A17   SIMIKEY =1AF0   SMBACKUP=1A04   SMBCDMEX=4B76
SMBSYF  =1B00   SMCLOSF =1AE7   SMDEVERR=1A39   SMDFIS  =1A56   SMDFOS  =1A5A
SMFEOD  =1A43   SMFINDF =1B86   SMGETF  =1B2F   SMGETOF =1A04   SMGETQF =1BA4
SMHEXBCD=4B6A   SMINIT  =1B32   SMJOBFIN=1B92   SMSTART =4B81   SMSTOP  =4B84
SMWMSG  =1B1D   SMXKEY  =1A1D   SMXTND  =1A4A   SNAP    =1A00   START   =1B3E
STATUS  =1A85   STIMABNM=1BA3   STIMER  =1B8C   STLBCHK =1B2D   STOP    =1B4E
STPIO1  =1A31   STPIO2  =1A33   STRTIO1 =1A36   STRTIO2 =1A38   SYMCUP  =1AF7
TDLOAD  =385E   TELCNTRL=1A03   TELERROR=1B20   TELEXEC =1A02   TELREAD =1A1A
TEST    =1A89   TESTBUF =4848   TESTLOOP=1A04   TESTWTG =48C0   TEXBUFFR=1A5C
TEXEXEC =1A00   TEXIT   =4CF7   TI      =1A02   TICRASH =1AF5   TIME    =1B5F
TIO2ABEN=1AE8   TIO3DBUG=3C1F   TIPFT   =3C08   TISAST  =1B39   TISCHN  =3BF9
TISD    =383E   TISDECHN=3BE4   TISEARCH=3B20   TISREAD =1B00   TIS1S   =1A00
TI821   =1A37   TMABORT =4310   TMABRTT =4316   TMCKADP =4DAE   TMCKADR =4D76
TMDCBERR=1A4B   TMDELAET=1B67   TMDEQ   =1A6E   TMENQ   =1A34   TMFINDJ =1A17
TMFINDT =1A4C   TMGETID8=1A10   TMGETJID=1A1E   TMGETTID=1A50   TMGRA   =4D93
TMLM    =4368   TMSETE  =1A08   TMSETPSD=4F81   TMSETREG=4F78   TMSTOP  =1B71
TMSTOPLN=1B64   TMTERM  =4313   TMTRMJ  =4330   TMTRMT  =4318   TMTYCB  =4D54
TMTYC8  =4D55   TMTYC15 =4D5C   TMTYC158=4D59   TMVADR  =4D72   TMWALL  =1A0A
TRAPCRSH=4C85   TRAP5   =4BD3   TRAP70  =4D0F   TRTN    =4CFE   TRTY    =4CF5
TRUNCATE=1A7B   TTDEBUG =1BD4   TTLN    =1A4C   TTPRIM  =1A57   TYPE    =1A50
USEARCH =1A00   VERACCNT=1B1A   WAITALL =1A05   WAITANY =1A89   WBLOCK  =1B4A
WCGETJOB=1AB9   WCOOP   =4B22   WCSTBYM =1ACF   WEOF    =1A11   WEOFDEV =1A2C
WLBLOCK =1A09   WRITDIR =1BB9   $#ALTENT=1B2F   $#ASSGN =1AA9   $#BRNCH =1BB2
$#CALENT=1B39   $#CONSG =1A05   $#DMPRET=1A05   $#DODMP =1A2A   $#DOINIT=1B8A
$#DORET =1B66   $#DOSNAP=1B79   $#DOTRAP=1B90   $#DOVAL =1B52   $#DUMP  =1A07
$#EXCT  =1B85   $#INSRT =1A1F   $#LOOK  =1B4F   $#MODFY =1A05   $#NAME  =1A5F
$#PATCH =1B38   $#QUIT  =1B97   $#REMOV =1AE6   $#RERCHK=1B48   $#SCAN  =1A04
$#SCNDR =1AB4   $#SCNLCX=1A04   $#SCNLWX=1A0C   $#SCNWDX=1A08   $#SNAP  =1A9D
$#SNAPIT=1AFF   $#TINIT =1A20   $#TRAPEX=1BA1   $#TRAPIN=1A58   $#WKSQZ =1B97

FGL2B04 =0000   FGL2B30 =0000   FGL3B81 =0000   MTTYPE  =0000   OTEMP   =0000
TEMP1   =0000   CONAISCH=0000   CONVSECT=0000   CONV1   =0000   CONV3   =0000
CPLCCU  =565C   CPLCOUT =565A   CRDIN   =5530   CRDINCU =5532   CRDOCU  =55C2
CRDOUT  =55C0   CRFIL4  =0000   DUMMYMD =0000   ENDAISCH=0000   FCRCRAUT=0000
FCRCRBIN=0000   FCRKPWE =0000   FCRKPWNL=0000   FCRKPWOE=0000   FCWCPBCD=0000
FCWCPBIN=0000   FCWKP   =0000   FCWKPWNL=0000   FCWNL   =0000   KBTCU   =5042
KBTID   =5030   MDL7261 =0000   MDL7266 =0000   MDL7275 =0000   POSITAI =0000
RWDEVF  =0000   TELBUFFR=0000   RFTADD  =0000   RFTBADD =0000   RFTFADD =0000
SCT4A   =0000   GIOEX   =1A89   SCANEX  =1AB8   EXITCRS =1BB1   EXITESU =1BB5
EXITK2  =1B9C   EXITK3  =1B9F   EXITK4  =1BA2   EXITK5  =1BA5   EXITK6  =1BA8
EXITK7  =1BAB   KEY1EXB =1BED   KEY2A04 =1A04   KEY2EXB =1BC4   CRSHEX  =1B97
```

Figure 29. SYSGEN Map Example (cont.)

```
KEY3A04 =1A1A   KEY3EXB =1B9F   KEY4A04 =1A1A   KEY4EXB =1A2C   KEY5A04 =1A19
KEY5EXB =1A25   KEY6A04 =1A1B   KEY6EXB =1A2D   KEY7A04 =1A19   KEY7EXB =1A28
FGL1EXIT=1A17   FGL2EX  =1A3A   FGLMCX  =1A63   FGL3X   =1A3D   SCMEDX  =1A18
SCMSGX  =1AE7   SCSUBSX =1AC3   RKLASSNX=1A90   RKLEXIT =1A15   ABEXEXIT=1A34
ABEXRAX =1ABF   DEXIT   =1B8A   DFGDX   =1BDB   DISCCU  =5145   DISCIO  =5140
DPAK    =51E7   DPAKCU  =5239   LPPOST  =545D   LPPRE   =5410   MTAP    =56B0
MTAPCU  =56B8   PRTCU   =5431   PRTLCCU =5431   PRTLCOUT=5412   PRTOUT  =5410
TAPEPOST=5680   TAPEPRE =5680   346XPOST=547A   346XPRE =5414   7TAP    =56E3
7TAPCU  =56B8   CARDPOST=5567   CARDPRE =5565   CONNEXIT=1B48   DISCTDFT=0000
ENQX1   =1A00   ENQX2   =1A01   ENQX4   =1A02   ENQX8   =1A05   DELEXIT =1A31
DELTRAPX=1A32   FMDELX1 =1A6D   RWBFX2  =1A01   RWBFX10 =1A08   RWBFX11 =1AAA
RWBFX2  =1A01   RWBFX3  =1A02   RWBFX4  =1A03   RWBFX5  =1A04   RWBFX6  =1A05
RWBFX7  =1A06   RWBFX8  =1A07   FBBX3   =1A93   FBBX1   =1A00   FBBX2   =1A01
OPEN1A  =1A90   DEVERR  =1B25   OPEN1B  =1A91   DEVEX2  =1B27   ASSIGNX =1AA9
CLOSE1A =1A33   CLOSE1B =1A3A   CLOSE1C =1A3C   OPEN2B  =1A95   CFUPDX  =1ABF
REWEX01 =1AD0   REWX1   =1A06   REWX2   =1A14   REWERR  =1A07   REWEXIT =1A04
SETUPX1 =1BE2   WEOFX10 =1A18   REWDX1  =1A00   REWX3   =1A26   REWX4   =1AEE
REWDX4  =1A28   REWDX5  =1A20   GCEXIT  =1AAC   REWDX2  =1A28   REWDX3  =1A26
IOXEXIT =1B1B   IOXTRAPX=1B1A   POLLABX =1B0E   TYPREXIT=1A4F   TYPREX1 =1A4E
SIGX2   =1A02   SIGX4   =1A03   SIGX5   =1A04   SIGX   =1A00   SIGX1   =1A01
JCALERR =1BC7   JCALEXIT=1BC6   MEDCEXIT=1A66   SIGX8  =1A07   SIGX9   =1A08
SNAMEX02=1A3C   SCALERR =1A32   SCALEXIT=1A31   SNAMEXIT=1A59   SNAMEX01=1A3B
RUNX    =1A96   SCERRX  =1AC9   SCNORMX =1AC6   SCALTRPX=1A33   STLBCHKX=1B41
PINTXABN=1A04   PINTXR8 =1A05   PINTXSIG=1A03   SCTRAPX =1AC8   PINTX   =1A00
SMGETFX =1B9D   SYM2X1  =1A01   SYM2X8  =1A03   PINTX1  =1A01   PINTX5  =1A02
SYM3X5  =1A04   SYM3X6  =1A06   SYM5X1  =1A01   SYM3X3  =1A01   SYM3X4  =1A03
TRMX    =1A07   TRMXR8  =1A00   TRMXTERM=1A01   SYM5X2  =1A03   SEGEXIT =1B43
TMXABT  =1A01   TMXBAD  =1A00   TMXCALX =1A02   TRMXTRMJ=1A02   TMXAB   =1A03
TTX     =1A05   FMJCEXIT=1AD4   JMTENQZ =1AF2   WAITXR8 =1A04   TTLNX   =1A55
TTJX    =1A05   ZETMX8  =1A01   ALLEXIT =1ABD   JMTRMX  =1A79   TTJOBX  =1A21
JOBEX   =1B73   JOBNX   =1B70   JOBSCX  =1BA4   ALLTRAPX=1AD0   JOBERR3 =1B48
JORDX   =1A50   JOBMSGX =1AC4   JOB2X1  =1A01   JOBSCXE =1BA5   JTRAPX  =1B45
ESUMXIT =1A6E   JMEXIT2 =1B89   SAIEXIT =1B59   JOB2X2  =1A03   SETPX   =1A90
REX2    =1BB3   CRB2XIT =1A06   CRS2XIT2=1A21   CRSXIT  =1B0B   REX1    =1BB1
CRDXIT  =1AF2   TIO1EX01=1B22   TISARENX=1B38   CKDXIT  =1A51   CKD2XIT =1AB9
TIO2EX03=1AD1   TIO2EX04=1AD3   TISEXIT =1AF4   TIO2EXIT=1B36   TIO2EX02=1AC8
LOGXIT  =1A68   MMCALX  =1BC2   MMO1EXIT=1BE6   PLO1FXIT=1A0A   ERRXIT  =1BDC
MMO4EXIT=1BF9   MMO5EXIT=1BF6   MMO6EXIT=1B4C   MMO2EXIT=1BF8   MMO3EXIT=1BE0
DBUGCALX=1BBC   DBUGINTX=1B92   DBERR1  =1A01   IPLMMEX =1B19   ISYMEXIT=1AB4
SNAPEXIT=1B26   USRIN1  =1A03   DBERR2  =1A01   DOVALEXT=1A98   GODUMPIT=1A06
DBERR3  =1A01   DUMPEXIT=1A9F   SCNDROUT=1B26   TRAPRTN =1AA1   USRIN2  =1A03
SCANOUT =1AC2   TTDBUGX =1BED   USRIN4  =1A03   USRIN3  =1A03   DBERR4  =1A01
SNAPOUT2=1B78   TRAPINEX=1BA0   TRAPOUT =1BA6   SEARCHEX=1A4F   SNAPOUT1=1B76
SCNEXIT =1A14   USRIN6  =1A03   MED1EXIT=1A0D   BRAEXIT =1BD8   DBERR6  =1A01
MED2XITA=1A54   MED2XITB=1A5C   ANLY8EXT=1ADB   MED1XITA=1B85   MED2EXIT=1A53
VERACCEX=1B39   ANLERREX=1BED   OVERBGEX=1B65   OFFMSGEX=1B55   TEXBBRTN=1A82
CNTLEXIT=1A14   TEL2EXIT=1A00   TEL3RD  =1B20   CNTRLEXT=1A2D   TEL1EXIT=1A00
RBM     =1D00   RBMEND  =58DF   FP8IM   =0000   DEC8IM  =0000   BYT8IM  =0000
CV8IM   =0000   DELTA   =0000

**** DISC ALLOCATION ***
AREA # DEVICE  MODEL   FIRST    LAST     SIZE    HP    NWPS
SP   0 DPAE0   7270        1    2100     2100     8     256
FP   1 DPAE0   7270     6391    8190     1800     8     256
BP   2 DPAE0   7270     8191    9990     1800     8     256
BT   3 DPAE0   7270     3921    6320     2400     8     256
CK   5 DPAE0   7270     2101    2520      420     8     256
D1   6 DPAE0   7270     2521    3920     1400     8     256
I8   7 DPAE0   7270     6321    6340       20     8     256
O8   8 DPAE0   7270     6341    6390       50     8     256
D5   9 DPAE0   7270     9991   47999    38009     8     256
**** END MAP ****
```

Figure 29. SYSGEN Map Example (cont.)

The CP-R overlays (which may optionally be made resident) consist of the subtasks of the CP-R Control Task and the various user services, which include:

Key-in Processor

Background Abort/Exit Routine

Postmortem Dump

Foreground Root Loader

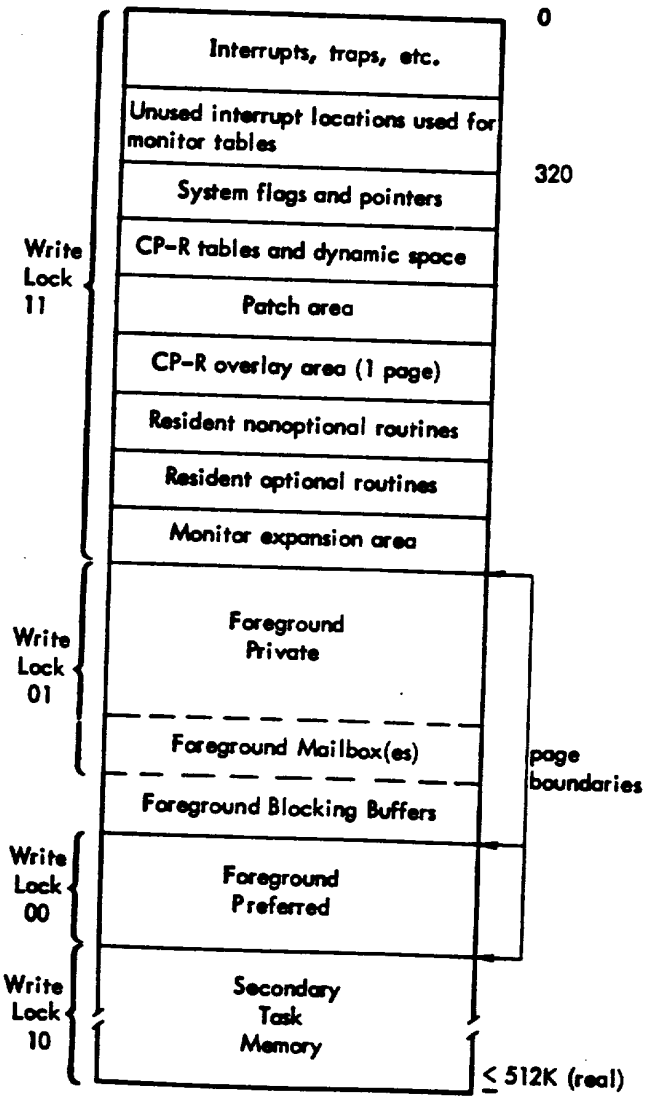
Background Root Loader

Foreground user services

Background user services

MEMORY LAYOUT AFTER SYSLOAD

After SYSGEN and SYSLOAD have executed, memory would have the following typical layout:



DISK AREAS

During SYSGEN, the permanently mounted disk space is divided into areas. There is a maximum of 23 standard areas. The names of these areas, their common usage, and other information is given below. Additional user defined areas are identified by any unique two-character name and are assigned space at the same time as the standard areas. The only restriction placed upon names of user defined areas is that they may not match operational label names. The size and location of the areas are fixed by SYSGEN and can be changed only by another SYSGEN.

With the exception of those files that contain the generated system, the definition of files on disk storage is not a function of SYSGEN.

The standard SYSGEN areas, their names and their normal order of allocation is as follows:

Area Name	Area Mnemonic	Write Access
System Program	SP	SYSTEM
Foreground Program	FP	SYSTEM
Background Program	BP	SYSTEM
Foreground and Background Data	D1 through DF	Foreground or Background as specified
Input Symbiont	IS	Background
Output Symbiont	OS	Background
IOEX Access	XA	IOEX
Checkpoint	CK	SYSTEM
Background Temp	BT	Background
User Defined	xx	As specified

The user can specify a given area to physically reside on any disk in the system if the system contains more than one disk; however, each area must be wholly contained on one disk. The user must also specify a disk to be the system disk, which will contain the SP area and receive the CP-R disk bootstrap.

The System Program area of the disk contains the monitor, service processors (Overlay Loader, RADEDIT), system processors (AP, FORTRAN, etc.), and the System Library.

The Foreground Program area of the disk should contain the user's foreground programs, the Public Library, and the User Library, if they exist.

The Background Program area should contain any permanent user background programs.

The IOEX Access area can be written only by IOEX and should normally be the only area of the disk that IOEX is allowed to access.

The foreground and background Data areas can be used to store the appropriate type of user data. Up to 15 Data areas (D1-DF) are allowed, to accommodate a user with multiple disks. If the Accounting Log file, AL is used, it must be allocated on the D1 area with a background-write protect code.

The Input Symbiont area contains background jobs waiting to run (or running) while the Output Symbiont area contains output from those jobs.

The Checkpoint area is used to save the contents of rolled-out segments. The Background Temp area can be allocated to a maximum of nine scratch files (X1-X9) plus the GO and OV files.

If a user does not choose to specify the sizes for the different disk areas, SYSGEN will attempt to allocate, on the system disk, the default sizes given in Table 30. If there is not enough system disk space to allocate default sizes, a disk OVERFLOW message will be output.

Table 30. Disk Area Default Sizes

Area	Default Size	Comments
System Program	1500 sectors	Large enough to contain all system processors, one per file, in load-module form; the system library format; and the monitor in absolute format.
Foreground Program	0	User is required to specify number of tracks for all areas not used by system programs.
Background Program	0	
Foreground/ Background Data	0	
Input Symbiont	0	A nominal area size is approximately 250 sectors (256 words per sector).
Output Symbiont	0	A nominal area size is approximately 1000 sectors (256 words per sector).
IOEX Access	0	
Checkpoint	n sectors	Where n = the size of real memory in sectors.
Background Temp	m sectors	Where m = remainder of disk. Disk size is determined from the ENTRACK parameter on the :DEVICE command.

The areas will be physically located on disk in the same order as they were input during SYSGEN. The System Program area will be the first area on the system disk unless the user inputs the SP area in a different order. All disk areas required by the user must be specified except those areas that SYSGEN automatically allocates by default.

Beginning at the starting sector input on the :DEVICE command, SYSGEN will allocate the number of sectors for each area without leaving empty spaces between areas. A bad area on a user's disk can be skipped via an input to the RADEDIT at the time the user's files are defined. The system disk will include the CP-R Bootstrap and therefore the actual space available will be one sector less.

BACKGROUND TEMP AREA

The scratch files (X1 through X9) of the Background Temp area of the disk will be automatically allocated and defined by the Job Control Processor prior to execution of a background program, unless the user wishes to override these defaults via an !ALLOBT control command. During SYSGEN, the user will not specify any standard sizes for the scratch files, X1-X9. The X1-X9 files are reallocated before execution of each background job step.

The GO and OV files are also in the BT area of the disk. These files are more permanent than the X1-X9 files and are maintained throughout an entire job. The user has the option to override the default permanent size of GO and OV at SYSGEN time via the :ALLOBT command. The permanent size, determined at SYSGEN, of either file, can be altered through the background job stack via an !ALLOBT control command.

An example of allocation of a system disk is given in Figure 30.

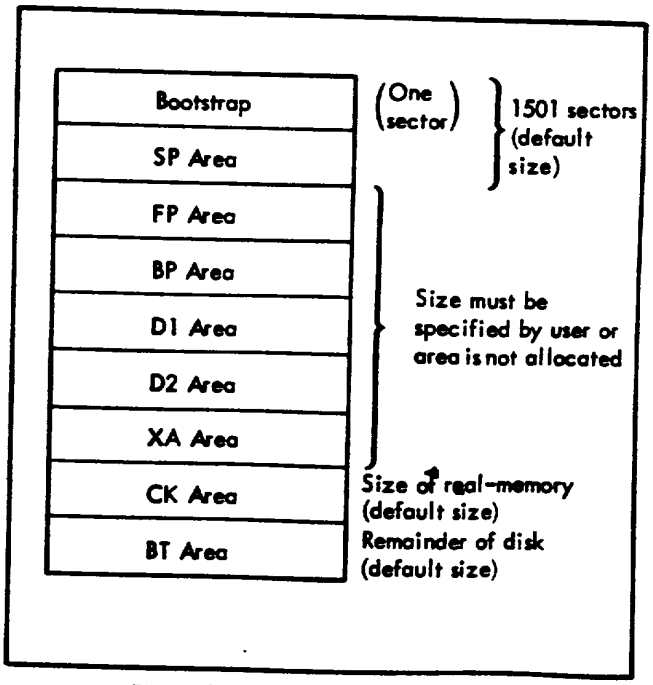


Figure 30. Disk Allocation Example

Table 31 gives the default sizes and types for GO, OV, and X1-X9, and the order in which the files are allocated. Note that X1-X9 are at the front of the BT area, and GO and OV are at the opposite end.

TABLES ALLOCATED AND SET BY SYSGEN

DEVICE CONTROL TABLE (DCT)

The DCT table is allocated by SYSGEN and several of the entires in the table are set by SYSGEN (i.e., device type, device number, dedicated to foreground bit, etc.). The DCT contains one entry for each device input by the user on the :DEVICE command, and the order of the entries is the same as the order of the :DEVICE commands. Note that there will be only one entry in the DCT for each disk.

RAD FILE TABLE (RFT)

The RAD file table is allocated by SYSGEN from the FRAD and BRAD entries on the :RESERVE command, and should contain sufficient entries to reflect the maximum number of open disk files that can exist simultaneously. The user will input the number of RFT entries to be reserved for foreground programs and the number to be reserved for background programs. The background is not allowed to use more than the number of RFT entries allocated for the background. However, the foreground can use all RFT entries if they are needed. The rationale for having foreground/background disk files as opposed to a single pool of files is that a background program could erroneously use all the file entries, thus preventing the operation of a foreground program.

MASTER DIRECTORY

The Master Directory is entirely set up by SYSGEN in the resident monitor portion of memory and contains the following information about each area on disk: the sector address of each area, the disk to which the area is assigned, the sector size and number of sectors per track; a bit that states if an area has been allocated; and the write protection code for the area.

OVERLAY INVENTORY (OVI)

The Overlay Inventory table is entirely set up by SYSLOAD and contains the information the monitor needs to load a CP-R overlay. This information consists of an overlay identifier, the seek address of the overlay, the real memory address, the use count, and the number of bytes in the overlay. The overlays are put on a page boundary during the generation of the system.

I/O QUEUE TABLE (IOQ)

The IOQ table is allocated by SYSGEN from the FIOQ and BIOQ entries on the :RESERVE command. The user inputs the maximum number of I/O operations that can be queued at one time for the foreground and background. The restrictions on the use of the foreground IOQ table are the same as for the RAD File Table.

LOAD MODULE INVENTORY (LMI)

The Load Module Inventory contains an active entry for each foreground program loaded into memory. Requests to load a foreground program can be made from either another foreground program or by the operator. Space for this table is allocated by SYSGEN from the FRGD entry on the :RESERVE command.

Table 31. GO, OV, X1-X9 Default Sizes

File Name	File Type	Default Size	Comments
X1 X2 X3 X4 X5 X6 X7 X8 X9	Unblocked	Determined by Job Control Processor at execution time.	File type and record sizes can be changed through a Device Mode function call or through an !ALLOBT command.
OV	Unblocked	128 sectors	Default output for Overlay Loader. Used mainly to test a program that has no permanent file defined, or to test a new version of a program without destroying the current version.
GO	Blocked (120 bytes/ logical record)	128 sectors	Used by FORTRAN and AP for "assemble and go" type operations.

SYSTEM JOB INVENTORY (SJI)

The System Job Inventory contains an active entry for each job in the system. Requests to start a new job or to stop an existing one can be made either by a foreground program or by the operator. Space for this table is allocated by SYSGEN from the JOBS entry on the :RESERVE command.

SYSTEM TASK INVENTORY (STI)

The System Task Inventory contains an active entry for each task in the system. Requests to connect a new task or to stop an existing one can be made either by a foreground program or by the operator. Space for this table is allocated by SYSGEN from the TASKS entry on the :RESERVE command.

OPERATIONAL LABEL TABLE (OPLBS)

The OPLBS table is built by SYSGEN from the information input on the :STDLB command. The table has a minimum of eleven entries that contain the standard system operational labels. Since operational labels are referenced via an index value in the DCB, each of the eleven standard operational labels have a fixed index value. If the user adds his own operational labels to the table, the user operational labels are assigned an index value, starting with 12, in the order in which they are input on the :STDLB command. The standard operational labels and index values are

Op Label	Index Value	
C	1	Standard operational labels
OC	2	
LO	3	
LL	4	
DO	5	
CO	6	
BO	7	
CI	8	
SI	9	
BI	10	
SO	11	
xx	12	User-defined operational labels; (index value dependent upon order on :STDLB command).
yy	13	
:	:	
SE	User + 1	Generated after user operational labels if ALARM and ERRORLOG are specified.
ER	User + 2	
DI	User + 3	Generated if DEBUG is specified.
DL	User + 4	
DP	User + 5	
P1	User + 6	
P2	User + 7	

INTERRUPT LABEL TABLE (INTLB)

The INTLB table is set up by SYSGEN from information contained on the :INTLB command. The table contains the name of each interrupt and the location to which the interrupt is assigned.

INPUT PARAMETERS

After SYSGEN and SYSLOAD are loaded by SYSGENLOAD, Control is transferred to SYSGEN. SYSGEN types out the message

```
CP-R SYSGEN
```

If Sense Switch 3 is set, SYSGEN will type the message

```
IN, OUT DEVICES
```

and request input from TYA01. The operator must then type in a valid :SYSGEN command.

```
:SYSGEN (IN,yyndd,mmmm)[,(OUT,yyndd,mmmm)]
```

where

IN specifies the device in the format yyndd from which the remainder of the SYSGEN control commands will be input.

- yy is a device type code and must be either CR, or TY (see below for a description of the codes).
- n is the IOP; legal values are A-H corresponding to IOP's 0-7.
- dd is the hardware device number of the device.
- mmmm is the hardware model number.

OUT specifies an optional output device on which the input commands are to be logged or the map, if requested, is to be output. The device type code must be either the TY or LP.

If Sense Switch 3 is reset, default IN, OUT devices will be used. The default command is

```
:SYSGEN (IN,CRA03,7120),(OUT,LPA02,7440)
```

Following input of the :SYSGEN command, the SYSGEN control commands are input through the specified device.

The following device types are standard under CP-R and should be input in the yy portion of a yyndd parameter in all SYSGEN control commands:

Device Type Code	Device
NO	Not a standard device. A special purpose device for use with IOEX.
TY	Typewriter
LP	Line printer
CR	Card reader
CP	Card punch
9T	9-track magnetic tape
7T	7-track magnetic tape
DC	RAD
DP	Disk pack
PL	Plotter
LD	Logical Device

SYSGEN CONTROL COMMANDS

The SYSGEN control commands are given below. SYSGEN control cards follow the same syntax rules that apply to other control commands. The general control card syntax is described at the beginning of Chapter 2. The :MONITOR and :RESERVE commands (in that order) must be input prior to the :DEVICE command. A :DEVICE command must precede a :STDLB command that references that device.

:PROC This command defines the processors in a Xerox 550 system. It is not used on Sigma systems. The information from this command is used to place a record of the system configuration in the Error log file. It also drives the processor polling activity subsequent to certain fault situations.

The :PROC command has the form:

```
:PROCESSOR (option), (option) . . . , (option)
```

The options are:

MIOP, value	PI, value	SU, value
HSR, value	MI, value	CPU, value

The values are the hexadecimal processor addresses. The keyword is the processor type.

Example:

```
:PROCESSOR (HSR, B)
```

defines cluster 1 unit 3 as a high speed RAD IOP.

Note: The MIOP containing the console teletype must be unit 0 in cluster 0.

:MONITOR The :MONITOR command specifies monitor and CPU options. The :MONITOR command must precede the :RESERVE command and must precede the :DEVICE command for the system disk.

The command has the form

```
:MONITOR (option)[,(option)...,(option)]
```

where the options are

CORE, size specifies the real memory size, in decimal units of K (where 1K = 1024 words), of the target computer (computer for which the SYSGEN is being run). The default value for CORE is 32K words.

FPSIM† specifies that the floating-point simulation package is to be loaded by SYSLOAD. If this parameter is absent, either the floating-point hardware exists or floating-point is not needed for the target computer.

DECSIM† specifies that the decimal instruction simulation package is to be loaded by SYSLOAD. The absence of this parameter indicates that either the decimal instruction hardware exists or the decimal package is not needed for the target computer.

BYTSIM† specifies the byte string instruction simulation package is to be loaded by SYSLOAD.

CVSIM† specifies the convert instruction simulation package is to be loaded by SYSLOAD.

ALLSIM† specifies that all software instruction simulation packages are to be loaded by SYSLOAD.

REBOOT specifies that the default action taken following a system alarm will be a Re-Boot and Initialization.

ALARM specifies that all code associated with handling system alarms will be included in the generated system.

†Standard CP-R allows the use of instruction simulators when the hardware will not handle a given instruction. If an instruction simulator is not loaded during SYSGEN and that type of instruction is executed, CP-R will simulate a TRAP to location 40.

ERRORLOG specifies that all code associated with handling error logging will be included in the generated system.

DEBUG specifies that all code associated with DEBUG will be included.

LPP,value is number of lines per printer page. The default is 37. This value is used by processors that perform their own vertical format control of the printer.

ACCT specifies that the monitor is to perform job accounting. Absence of the ACCT parameter indicates that no job accounting is to be kept.

OLAYEP,value is the number of entry point and exit point locations in the system. The value defaults to an assembly parameter (see PROGRAM ALLOCATION section of map for a list of entry and exit points).

ROLAY,name specifies that the named overlay module† is to be included and be made resident.

NROLAY,name specifies that the named overlay module† is to be included (as nonresident).

BOUND specifies that resident overlays are to start on even X'10' locations.

SPAI,xx specifies that disk area "xx" is the system processor alternate area, which acts as an extension to area SP for the location of programs called by the JCP processor name commands or TEL RUN commands.

:RESERVE The :RESERVE command allocates areas of memory and the various variable-length monitor tables. The :RESERVE command must precede the :DEVICE command for the system disk.

The :RESERVE command has the form

```
:RESERVE (option)[,(option)...,(option)]
```

where the options are

TSPACE,size specifies the number of words to be allocated for the CP-R dynamic-space pool. The default number is formulated as follows:

each IOQ entry	8 words
each TASK entry	192 words
each JOB entry	32 words
each JPT entry	4 words/job

The SYSLOAD map value S:TEMPN may be examined to determine the resultant value.

†The named module and its size (nonresident) or base address (resident) may be found in the SYSGEN map under "CP-R PROGRAM ALLOCATION" as the first entries. The user may choose size or performance trade-offs by use of these options.

RSDF,size[,start] specifies the size and starting address of a foreground-private memory partition, to be reserved for the exclusive use of primary tasks. (Several such partitions may be declared by multiple uses of this option.) "Size" is the decimal number of contiguous real pages to be reserved; "start" is the hexadecimal starting address of the partition, which must be on a page boundary in the first 128K of memory and may not fall within any other defined partition. The default for "start" is the first page boundary above the resident monitor. If the option does not appear, no foreground-private partition is allocated.

This space is available for all primary foreground programs on a first-come, first-served basis. A program is given its predetermined memory space (determined when it is linked by the Overlay Loader) when loaded for execution. No other program can use this space until the program is unloaded.

MPATCH,size specifies the decimal number of word locations to be reserved for modification to and expansion of the monitor. The default size is zero.

FFPOOL,size[,start] specifies the number of foreground blocking buffers to be reserved, for use by primary tasks only, either as a separate foreground-private partition or as an area within the first partition defined via RSDF. "Size" is the decimal number of 256-word buffers to be allocated; "start" is the hexadecimal starting address of the blocking-buffer partition, which must be on a page boundary in the first 128K of memory and may not fall within any other defined partition. If "start" is omitted, the blocking buffers will form an area at the end of the first-defined foreground-private partition such that the starting address will default to the upper bound of $RSDF_0$ less the size (in words) of the allocated buffer pool. If the FFPOOL option does not appear, no blocking buffers are allocated; one buffer, however, is required to initiate a foreground primary load module.

LDMD,value†† specifies the maximum number of foreground load modules that can reside in memory at any one time. This parameter will be used to allocate space for the load module inventory that is used to manage the foreground private area. The default size is zero. Maximum allowable value is 250.

TASKS,value†† specifies the number of entries to reserve in the System Task Inventory. This number must be adequate to meet the peak requirements of all foreground tasks. The default size is zero. The maximum allowable value is 250.

††SYSGEN will add entries for the CP-R job and the background job.

JOBS,value[†] specifies the number of entries to be reserved in the System Job Inventory. One entry exists for each active job. The default size is zero. The maximum allowable value is 30.

JPT,value specifies the number of entries to be reserved in Job Program Table for each job (see the SETNAME service call). The default size is zero. The maximum allowable value is 63.

FRAD,value specifies the number of entries to reserve in the RAD File Table for foreground disk files. This number should reflect the maximum number of foreground disk files that could be open simultaneously. Note that the background disk pool is also available to the foreground. The default value is zero.

BRAD,value specifies the number of entries to reserve in the RAD File Table for background disk files. This number should reflect the maximum number of background disk files that can be opened simultaneously. The default value is 5, which is sufficient to accommodate the requirements of the system processors. The value indicated should not include the files in the BT area of the disk.

FIOQ,value specifies the maximum number of foreground I/O operations that can be queued at any one time. This parameter determines the space allocated for foreground entries in the I/O queue table. Note that the background queue table is also available to the foreground. The default value is zero. Note that this must include queue entries for COC input and output requests.

BIOQ,value specifies the maximum number of background I/O operations that can be queued at any one time. This parameter determines the space allocated for background entries in the I/O queue table. The default value allows three entries to be placed in the queue table.

Note that the sum of FIOQ and BIOQ must be less than 256, or an error indication will be given.

FMBOX,size[,start] specifies a foreground mailbox area to be reserved, for use by primary tasks only, either as a separate foreground-private partition or as an area within the first partition defined via RSDF. "Size" is the decimal number of words to be allocated for the mailbox; "start" is the hexadecimal starting address of the mailbox partition, subject to the same rules as given for RSDF and FFPOOL. If "start" is omitted, the mailbox will form an area at the end of the first-defined foreground-private partition but preceding the default blocking-buffer area, if any, such that the starting address will default to the upper bound of $RSDF_0$ less the size of the buffer-pool area, if any, and the size of the mailbox area. If the FMBOX option does not appear, no foreground mailbox is reserved.

JENQ,value when added to the number of job exclusive devices, specifies the maximum number of ENQs allowed within a job for a system level controlled item. This parameter determines the size of the AET which is linked to a job.

TENQ,value specifies the maximum number of ENQs allowed within a load module for a job-level-controlled item. This parameter determines the size of the AET which is linked to a load module.

BT,value specifies the maximum number of Background Temp files (X1-X9) that will ever be used. The default value is 6, that is, files X1, X2, X3, X4, X5, X6. Six files are sufficient for the System Processors. The files defined are X1-Xn, where n is the input value. The value must be less than 10.

PMEM,size,start specifies the size and starting address of a foreground-preferred memory partition, to be reserved for the use of primary or secondary tasks. (Several such partitions may be reserved by multiple uses of this option.) "Size" is the decimal number of contiguous real pages to be reserved; "start" is the hexadecimal starting address of the partition which must be on a page boundary in the first 128K of memory and may not fall within any other defined partition. If no PMEM options appear, no foreground-preferred partitions are allocated.

BMEM,size specifies the maximum number of pages of secondary-task memory that may be allocated to the background. This value defaults to 32 if BMEM does not appear. It may be altered by the BMEM operator key-in.

:CHAN This command groups peripheral devices (see :DEVICE below) according to channel controller. All :DEVICE commands following a given :CHAN command are assumed to be part of that channel. At least one :CHAN command must be used, and each :CHAN command must precede the :DEVICE command (or commands) to which it applies.

The :CHAN command has the form

```
:CHAN [(DUAL,n1d10,n2d20)]
```

where DUAL, n_1d_10, n_2d_20 specifies that this channel has dual-access capabilities, and that any following device command that has n_3d_3 (as in yyn3d3d), which matches n_1d_1 or n_2d_2, will be considered a dual-access device. (Note that since all dual-access devices are also multiunit controller devices, n_3 and d_3 are, in effect, IOP and channel respectively.) This dual feature allows the device to be referenced as either yyn_1d_1d or yyn_2d_2d, thus improving throughput and reliability.

[†]SYSGEN will add entries for the CP-R job and the background job.

The :CHAN command has the form

```
:CHAN [(DUAL,n1d10,n2d20)]
```

where DUAL, n_1d_10, n_2d_20 specifies that this channel has dual-access capabilities, and that any following device command that has n_3d_3 (as in yyn3d3d), which matches n_1d_1 or n_2d_2, will be considered a dual-access device. (Note that since all dual-access devices are also multiunit controller devices, n3 and d3 are, in effect, IOP and channel respectively.) This dual feature allows the device to be referenced as either yyn_1d_1d or yyn_2d_2d, thus improving throughput and reliability.

:DEVICE The :DEVICE command introduces peripheral units into the system. One :DEVICE command is required for each peripheral unit to be used. The order of the :DEVICE commands determines the Device Control Table index value that the device will receive. If an error is made in any field of the command, the entire command must be input again.

The :DEVICE command has the form

```
:DEVICE yyndd,mmmm[,S][,(option)][,(option)]...
```

where

yyndd specifies the device name (see the :SYSGEN command for a description of yyndd). The device name must be the first field after :DEVICE.

mmmm is the model number. Table 32 shows the model numbers for all supported devices. Table 33 shows the model numbers and structural parameters for all disk devices. Model numbers of NO or NONE are allowed for special devices which are going to be used with IOEX access only.

S specifies (for a disk device only) this as the system disk. The system disk receives the CP-R bootstrap, SP area, and any default allocations. If a SP area is defined, then that disk will become the system disk and this option is not required.

The options are:

DEDICATE,value specifies various forms of restricted use declared for the device.

A value of "S" specifies the device is a symbiont device. The device must be a card reader or a line printer. Note that symbiont devices may be used like nonshareable devices (see "J" value) when not in use by symbiont.

A value of "F" specifies that the device may only be used by foreground programs.

A value of "J" specifies that the device used may not be shared by more than one job.

A value of "X" specifies that the device may only be used by IOEX, without request queuing.

DEBUG,xxx specifies the hexadecimal address of an external interrupt level to be associated with the device when it is used as a DEBUG control console; meaningful only if device is TYndd.

Note: The following options are applicable only for a disk device.

SSEC,value specifies the decimal absolute sector number of the beginning of the disk space used by CP-R. Sectors are numbered consecutively starting with 0. The SSEC value must be less than the ENSEC value. The default SSEC value is 0.

ENSEC,value specifies the decimal absolute sector number of the last sector number of the space used by CP-R. The ENSEC value must be greater than the SSEC value and less than or equal to the default value. The default value is shown in Table 32 and depends on the disk model number.

NSPT,value specifies the decimal number of sectors per track.

NWPS,value specifies the decimal number of words per sector.

NTPC,value specifies the decimal number of tracks per cylinder. (In the case of fixed-head disks, it is the total number of tracks.)

NCYL,value specifies the decimal number of cylinders available for use. (In the case of fixed-head disks, there is just one cylinder.)

SSFT,value specifies the decimal value for a left shift which positions the sector number (right justified in a word) to the correct position for a seek address.

TSFT,value specifies the decimal value for a left shift which positions the track number (right justified in a word) to the correct position for a seek address.

CSFT,value specifies the decimal value for a left shift which position the cylinder number (right justified in a word) to the correct position for a seek address.

area,value specifies the decimal number of sectors to be allocated to the designated area.

If the remainder of the disk is to be allocated to an area, "ALL" can be input instead of a decimal value.

The various forms in which this option can be written are:

System areas

SP,value	BP,value	XA,value
FP,value	CK,value	BT,value

Table 32. System Device Model Numbers and Parameters

Model Number	Mnemonic	Type Number	CLIST Size	Device Type
7202	DC	7	8	90 word RAD – 128 tracks
7203	DC	7	8	90 word RAD – 256 tracks
7204	DC	7	8	90 word RAD – 512 tracks
7212	DC	7	8	High speed RAD
7232	DC	7	8	256 word RAD – 512 tracks
3214	DC	7	8	NS RAD
7242	DP	12	14	Double spindle disk pack
7246	DP	12	14	Single spindle disk pack
7270	DP	12	14	406 CYL 7242
7275	DP	14	14	Disk 33
7261	DP	14	14	45M byte disk pack
7266	DP	14	14	90M byte disk pack
7251	DC	7	8	DIABLO removable disk
7252	DC	7	8	DIABLO fixed disk
3242	DP	14	14	Cartridge disk
3243	DP	14	14	Cartridge disk
3277	DP	14	14	3333 disk
3283	DP	14	14	6660 disk
7362	7T	9	6	7T tape (37 IPS)
7372	7T	9	6	7T tape (75 IPS)
7322	9T	8	8	9T tape (75 IPS)
7323	9T	8	8	9T tape (150 IPS)
7332	9T	8	8	9T Potter
7333	9T	8	8	9T Potter
3325	9T	16	8	9T 550
3335	9T	16	8	9T 550
3344	9T	16	8	9T 550
3345	9T	16	8	9T 550
3346	9T	16	8	9T 550
3347	9T	16	8	9T 550
7440	LP	6	4	LP
7441	LP	6	4	LP
7445	LP	6	4	LP
7446	LP	6	4	LP (BDP)
7461	LP	6	4	LP
7462	LP	6	4	LP
7463	LP	6	4	LP
7464	LP	6	4	LP
7465	LP	6	4	LP
7466	LP	6	4	LP
3461	LP	6	4	NS LP
3462	LP	6	4	NS LP
3463	LP	6	4	NS LP
3464	LP	6	4	NS LP
3465	LP	6	4	NS LP
3466	LP	6	4	NS LP
7450	LP	6	4	LP (Low Cost)
7020	TY	1	10	TTY
7012	TY	1	10	TTY
4591	TY	1	10	KSR 35
4592	TY	1	10	DIABLO HYTYPE
7120	CR	4	2	CR
7121	CR	4	2	CR
7122	CR	4	2	CR
7140	CR	4	2	CR
3132	CR	4	8	NS 300 CPM CR
3134	CR	4	8	NS 600 CPM CR

Table 32. System Device Model Numbers and Parameters (cont.)

Model Number	Mnemonic	Type Number	CLIST Size	Device Type
3130	CR	4	8	NS 1000 CPM CR
7160	CP	5	39	CP
7165	CP	5	2	CP (Low Cost)
7062	PR	2	8	PR
7020	PR	2	8	PR
7060	PR	2	8	PR
7063	PP	3	8	PP
7020	PP	3	8	PP
7060	PP	3	8	PP
7530	PL	13	2	PL
7531	PL	13	2	PL
NO	NO	0	0	No device (IOEX)
NONE	NO	0	0	No device (IOEX)
LDEV	LD	19	0	Logical device

Table 33. Disk Device Model Numbers and Parameters

Model Number	NSPT	NTPC	NCYL	NWPS	Description
7202	16	128	1	90	90 word RAD - 128 tracks
7203	16	256	1	90	90 word RAD - 256 tracks
7204	16	512	1	90	90 word RAD - 512 tracks
7212	82	64	1	256	High Speed RAD
7232	12	512	1	256	256 word RAD - 512 tracks
3214	11	256	1	256	NS RAD
7242	6	20	200	256	Double spindle disk pack
7246	6	20	200	256	Single spindle disk pack
7270	6	20	400	256	406 CYL 7242
7275	11	19	404	256	Disk 33
7261	11	20	200	256	45M byte disk pack
7266	11	20	404	256	90M byte disk pack
7251	16	407	1	90	DIABLO removable disk
7252	16	407	1	90	DIABLO fixed disk
3242	7	2	400	256	Cartridge disk
3243	7	2	400	256	Cartridge disk
3277	12	19	404	256	3333 disk
3283	12	19	808	256	6660 disk

Symbiont areas

IS, value [, fsize]

OS, value [, fsize]

where "fsize" defines the size of symbiont area files in records. The default "fsize" is 25 records for the input area and 200 records for the output area. Note that the maximum number of files allowed for a job in either the IS or OS area is 256, therefore, fsize should be selected so that this maximum will never be exceeded for a single job.

User areas

uu, value [, {P / B / F / S}] where uu is a two-character name of the user defined area to be allocated.

Note that for the data areas uu, a type may be given to specify usage and write protection. This may be one of

- P public
- B background (this is the default).
- F foreground
- S system

Public areas have no software write protection; any task may write to them. They provide a pool of space for file operations which omit the specification of a disk area name.

The remaining three areas are protected from accidental access by requiring that the area name be given explicitly.

Background areas have no other protection; any task may write to them.

Foreground areas are protected from the background unless the "SY" key-in is in effect.

System areas are protected from both background and foreground unless the "SY" key-in is in effect.

The SP, CK, and BT areas will be allocated by default if not specified. The CK and BT areas may be removed by specifying a size of zero sectors.

Inclusion of either a [,S] option or a [SP,value] defines the device as a system disk and any of the above default allocations will be made on this disk.

The default size allocated for the SP area is 1500 sectors. The default size allocated for the BT area is 'ALL'. The default size allocated for the CK area is large enough to hold all of memory.

Examples:

```
:DEVICE CRA03,7120
```

Higher performance card reader, device number 3, on IOP number 0, undedicated.

```
:DEVICE LPA02,7450,(DEDICATE,F)
```

Lower performance line printer, device number 2, on IOP number 0, dedicated to the foreground.

```
:DEVICE NOC09,NO
```

Nonstandard device, device number 9, on IOP number 2, dedicated to IOEX.

```
:DEVICE DCB90,7202,S,(FP,15),(D1,10,F),(D2,10,B)
```

A 7202 RAD, device number 90, on IOP number 1, to be used as the system disk starting on sector zero, with default sizes for the SP, CK, and BT areas, and the input sizes for the FP, D1, and D2 areas.

:ALLOBT The :ALLOBT command establishes the permanent sizes of the GO and OV files contained in the background temp area, i.e., the system default sizes of these files at execution time.

The :ALLOBT command has the form

```
:ALLOBT (file name,size)[,(file name,size)]
```

where

file name specifies the name of the file, which must be either GO or OV.

size specifies the decimal number of sectors to be allocated for the specified file. The size input becomes the permanent size for the specified file and overrides the SYSGEN default sizes given in the "Background Temp Area" subsection.

:MDEF The :MDEF command defines new models of peripheral devices. This allows SYSGEN to accept :DEVICE commands for types of devices not included in Tables 32 and 33.

The :MDEF command has the form

```
:MDEF   mmmm [,(option)][,(option)]...
```

where

mmmm specifies the model number. Redefinition of model numbers is also allowed.

The options are:

MNEMONIC,yy specifies the two-character mnemonic used in the yyndd format.

SIMILAR,mmmm specifies a similar model number from which all characteristics are copied. Additional options may then be used to modify selected characteristics.

TYPE,n specifies the decimal type number placed in DCT4. (See CP-R Technical Manual 90 30 88.)

SDBUF,n specifies the decimal number of words allocated as a buffer. This is only valid when the system is assembled with #SIDEBUF set.

DCT3,x specifies the hex value to be placed in DCT3. (See CP-R Technical Manual 90 30 88.)

CLIST,n specifies the decimal number of words allocated as a command list buffer.

PREHAND,eeeeeeee specifies the EBCDIC name of the I/O pre-handler to be used for this device.

POSTHAND,eeeeeeee specifies the EBCDIC name of the I/O post-handler to be used for this device.

IOTX,n specifies the decimal value of the internal SYSGEN I/O table index. (See CP-R Technical Manual 90 30 88.)

The following options are valid only for disk devices. (Mnemonic = 'DC' or 'DP'.) They establish default disk characteristics, and are identical to the disk options on the :DEVICE command.

NSPT,n
NTPC,n
NCYL,n
NWPS,n
SSFT,n
TSFT,n
CSFT,n
} see :DEVICE command

:MOD The :MOD commands allow patches to be made to SYSGEN itself.

The :MOD command has the form

```
:MOD  (SYSGEN,addr)[,xxxxxxxx][,xxxxxxxx]
```

where

addr is the absolute hex address of the start of the patch.

xxxxxxxx is the absolute hex value to be stored starting at addr. Successive values are stored in successive addresses.

:COC The :COC command specifies the characteristics associated with the COC device. The :COC command has the form

```
:COC (option),(option),...
```

The options are:

DEVICE,ndd specifies in hexadecimal, the address of the COC device.

OUT,value specifies the hexadecimal address of the COC output interrupt. The default is 61. Output must have a lower priority than input.

IN,value specifies the hexadecimal address of the COC input interrupt. The default is 60. Input must have higher priority than output.

LINES,value specifies the decimal number of lines. The default is 8.

BUFFERS,value specifies the decimal number of 4-word buffers necessary for COC I/O usage. The default value is the number of lines times 3. The recommended value is the number of users times 3.

RING,value specifies the decimal number of words for the COC input buffer. The default is 2 bytes per line for the first 30 lines plus 1 byte per line above 30.

33,value,..... specifies the decimal line numbers that are attached to Model 33 Teletypes.

35,value,..... specifies the decimal line numbers that are attached to Model 35 Teletypes.

37,value,..... specifies the decimal line numbers that are attached to Model 37 Teletypes.

7015,value,... specifies the decimal line numbers that are attached to Xerox Model 7015 keyboard printers.

Model 33 is the default terminal.

HARDWIRE,value,.. specifies the decimal line numbers that are hard wired to terminals. Lines which are "hardwired" will not be timed out nor will they be logged on automatically.

RATE,rate,value,. specifies the "rate" in decimal characters per second and value specifies the line numbers in decimal associated with the "rate". The rate must be in the range 1 to 255. One RATE option is used for each terminal speed on the COC. The default "rate" is 10.

NONTJE,value,... specifies the decimal line numbers that are not to be logged on as TJE lines. This allows the user to declare lines whose only purpose will be to be read or written.

The following restrictions apply to multiple :COC commands. All COC input interrupts must be higher than all COC output interrupts. All COC input interrupts must fall within one interrupt group and all COC output interrupts must fall within one group. The :COC commands must be ordered and sequential with respect to the DIO addressing logic and must begin with zero. COC0 input interrupt must be the highest of all COC interrupts.

:STDLB The :STDLB command defines all standard system operational label assignments for the generated system and all standard user operational labels and their assignments. Note that operational labels cannot be assigned to disk files during SYSGEN. The STDLB command must be input following the :DEVICE commands.

The :STDLB command has the form

```
:STDLB (label,name)[,(label,name)...]
```

where

label specifies a system operational label or a user operational label. All user operational labels must consist of two alphanumeric characters. Any system operational labels not specified on a :STDLB command will receive by default a permanent assignment of zero. The order of the user's labels determines a label's position in the operational label table, and therefore determines the OPLB value that might be present in a user DCB (see the table in the example below). No label will be allowed that is the same as a Background Temp file name (GO, OV, X1-X9) or the same as a disk area.

name specifies a physical device name to which the operational label is permanently assigned, a numeric zero, or a previously assigned operational label. In the latter case, the operational label will be assigned to the same device as the label to which it is assigned. If "0" is specified, the permanent assignment is zero.

The :STDLB command example

```
:STDLB (C,TYA01),(OC,C),(LO,LPA02),(LL,LO),
  (BI,CRA05),(SI,CRA05),(SO,CRA06),(XX,CRA05),
  (ZZ,LPA02)
```

would set up the operational label table given below.

	Label	OPLB Index$_{(10)}$	Permanent Assignment
Standard System Oplabels	C	1	TYA01
	OC	2	TYA01
	LO	3	LPA02
	LL	4	LPA02
	DO	5	0
	CO	6	0
	BO	7	0
	CI	8	0
	SI	9	CRA05
	BI	10	CRA05
	SO	11	
User Oplabels	XX	12	CRA05
	ZZ	13	LPA02

:CTINT The :CTINT command specifies the interrupts to which the CP-R Dispatchers and Control Task is to be connected; the highest address used for interrupts; and the interrupt that will be used for deferred I/O interrupt processing.

The :CTINT command has the form

```
:CTINT[(CT,address),...,(HI,address)][,(IO,address)]
```

where

CT,address specifies the hexadecimal address of the higher-priority one of a pair of adjacent interrupts to be used by CP-R for dispatching secondary tasks. As many of these pairs as desired may be specified (within the constraints of the existing interrupt levels). The lowest-level dispatcher (for the CP-R Control Task) will use the lowest-level interrupt specified and a 'null' level below it, i.e., the lowest-level "pair" defines only one actual interrupt level. This level will run the Control Task and the background task. If "address" has the value zero, no interrupt is available for the Control Task. In this case, SYSGEN will specify the Control Panel interrupt and its use will be shared by the monitor. The default value for the Control Task Interrupt is location X'61'. Note that if multiple CT parameters are input, then they must input in priority order (i.e., highest dispatcher first).

HI,address specifies the highest address, in hexadecimal, needed for an interrupt. SYSGEN will assume that all memory locations greater than HI are unused as interrupts and will attempt to allocate monitor tables in this area. The default value is X'13F'. Normally, CT and HI would have the same value.

IO,address specifies the absolute hexadecimal interrupt location to which the bookkeeping associated with an I/O interrupt will be deferred (cleaning up the interrupting operation and getting something else going). If "address" is 0, the highest CT interrupt level will be used. The value of address in all other cases must be equal to or less than the CT address. If the IO parameter is not present, the aforementioned bookkeeping will take place at the I/O interrupt level.

By using this parameter, a user may defer I/O processing to an external level below that of his task(s) which require this high response.

:INTLB The :INTLB command provides the capability of associating a label with an interrupt location. The label may then be used in system service calls in lieu of an actual interrupt address.

The :INTLB command has the form

```
:INTLB (label,loc)[,(label,loc),...,(label,loc)]
```

where

label specifies a two-character alphanumeric label.

loc specifies the absolute hexadecimal interrupt location to be associated with the label.

Duplicate labels are diagnosed as errors. The key-in INTLB may be used to change the assignment of the label from one interrupt location to another.

:PUNCH The :PUNCH command specifies that a rebootable version of SYSLOAD is to be punched after SYSGEN has read its last control command.

The :PUNCH command has the form

```
:PUNCH yyndd,mmmm
```

where yyndd,mmmm specifies the device (e.g., CPA04) on which the rebootable copy of SYSLOAD is to be punched.

:FIN The :FIN command signals the end of the control commands for the SYSGEN phase. Upon reading the :FIN command, SYSGEN will punch a rebootable version of SYSLOAD, output the SYSGEN-phase map if requested, and continue to SYSLOAD.

The :FIN command has the form

```
:FIN[MAP]
```

where

MAP specifies that a MAP is to be output on the same device being used to log the SYSGEN control commands. If no output device was specified on the :SYSGEN command, the MAP is output on TYA01.

:SITE The :SITE command, if present, is used in job header identification and for error logging purposes.

The :SITE command has the form

```
:SITE  text
```

where

SITE .... .

text is up to 56 bytes of text which is used to uniquely identify the site for which the SYSGEN was made.

:COMMENT The :COMMENT command allows insertion of comment lines into a SYSGEN control deck on command lines which have no other effect.

The :COMMENT command has the form

```
:COMMENT any test string up to column 80
```

:SYSLD The :SYSLD command also signals the end of the control commands to SYSGEN. The :SYSLD command causes SYSGEN to output the rebootable deck of CP-R, if requested, and then exit to a SYSLOAD entry where no further control command input is required.

The :SYSLD command has the form

```
:SYSLD[(IN,yyndd,mmmm)][,(OUT,yyndd,mmmm),
    [,{ALL|FAST|UPD|OVR}][,(V,xxx)][,(MAP[,yyndd,mmmm])]
```

IN specifies the device to be used for loading the control-program modules. The device must be either CR, 9T, or 7T. See the SYSGEN command for the yyndd and mmmm definitions. The default is the bootstrap device.

OUT specifies the device to receive a hard copy of the CP-R disk bootstrap. If the system disk allocation starts on sector zero, this field is optional; otherwise, an output device must be specified. The output device must be either CP, 9T, or 7T.

{ALL / FAST / UPD / OVR} specifies the SYSLOAD mode of operation. The "ALL" and "FAST" parameters indicate that all defined areas of the disk are to be initialized to zero. The "UPD" parameter indicates that existing data on the disk must be saved and only the new version of CP-R should be output to the disk. The "OVR" parameter is identical to "ALL" and "FAST", except that no clearing or referencing of any area will take place. See "SYSLOAD", below, for a further description of these options. The default value for this parameter is "UPD".

V specifies the version number of the system being loaded. Up to four alphanumeric characters can be input for the version. The version will be logged on LL at the start of each job and logged with each postmortem dump.

MAP specified that a MAP is to be output at the completion of SYSLOAD on the yyndd device and on the file CPRMAP which will be created for it in the SP area. The device type specified must be either LP or TY. The default is the 'OUT' device defined on the :SYSGEN command.

See the SYSGEN control command for a detailed description of yyndd,mmmm.

SYSLOAD

When the SYSGEN phase has been completed, or when the rebootable SYSLOAD deck punched by SYSGEN has been loaded, control is transferred to SYSLOAD. SYSLOAD loads the required CP-R modules as determined by the SYSGEN input parameters and outputs these to the disk. It then outputs the disk bootstrap and the System Program area directory to the disk. When SYSLOAD terminates it enters an idle state. If necessary, the user can now load the system and user programs on the disk by following the sequence outlined later in this chapter. If a :SYSLD command was not input to SYSGEN or input after rebooting the SYSLOAD deck, SYSLOAD will initially output the following messages on the TYA01 device:

```
CP-R SYSLOAD INPUT OPTIONS
```

The options input on the TYA01 device must be made via the :SYSLD command.

All writes made on the disk during the SYSLOAD phase will be checked to ensure that the data was correctly recorded.

MAP OPTION

The MAP option allows the listing of a map of both memory and disk allocation. The memory allocation is displayed as two portions. The first is titled 'CP-R TABLE ALLOCATION'. It is again divided into two parts. The first is a listing of tables allocated during the SYSGEN process itself. The second is a listing of the symbols defined in the permanently resident modules of the system. The second major part is titled 'CP-R PROGRAM ALLOCATION' and is a listing of the overlay entry and exit points.

The disk allocation lists the area name, number, starting sector number, ending sector number, size and protection for all areas included in the SYSGEN.

SYSLOAD loads three groups of CP-R modules in the following sequence: optional resident routines (FPSIM, DECSIM, CVSIM, BYTSIM); mandatory resident modules; the overlays and JCP.

The three groups of modules must be loaded in the stated order, but (for example) individual CP-R overlays need not be in any special order. All these routines can be input as one package and SYSLOAD will select and load only the routines that were requested during SYSGEN, making it unnecessary to rearrange the decks of modules if requirements change.

Each monitor module, in absolute object form, is identified to SYSLOAD via a DEF item, and any module not required is passed over. EODs are allowed between modules, and the final module in a group must be followed by an EOD. If all the required modules are not present in a group, SYSLOAD outputs the following alarm on TYA01:

```
MISSING ID name1,name2,...
```

where $name_n$ is the name of the missing routine, the name being indicated by the single DEF item in the module. SYSLOAD assumes the missing routines are not required and continues.

SYSLOAD writes the required overlays on the disk as they are loaded and sets up the information needed to enter the overlays in the OVI table. Modules that are not overlays will be loaded directly into the resident-monitor memory image and later written out with the resident portion.

When the SP-area directory is written on the disk by SYSLOAD, it will contain entries for four files named CPRFILE, JCP, CPRMAP, and RADBOOT. RADBOOT is the file that contains a copy of the disk bootstrap, the operational version of which being the only program on the disk not contained within a disk area. Therefore, it cannot be accessed if a disk dump or save is required, and for this reason a copy of the bootstrap is kept in an SP-area file.

The CPRFILE, JCP, CPRMAP, and RADBOOT files will be the first four files in the SP area. The CPRFILE will include any patch or expansion area that the user has requested.

After CP-R and the SP directory are output to the disk, SYSLOAD sets up the appropriate command list in the bootstrap to enable the bootstrap to reload CP-R from the disk. The bootstrap is then written both into the RADBOOT file and onto the starting sector of the system disk (STTRACK option on :DEVICE command).

If the system disk allocation starts at other than sector zero, a copy of the bootstrap is written on the device specified by the OUT keyword on the :SYSLD command. The user can then boot in CP-R by loading the "hard" copy of the bootstrap. This permits having more than one monitor system on disk and still being able to boot in CP-R by reading in a hard copy of the bootstrap.

ALL OPTION

The ALL option specifies that a complete system load is to occur and that all disk areas should be initialized. The ALL option is necessary for the initial SYSLOAD or if the disk allocation has changed so drastically that all areas on the disk have moved. SYSLOAD will zero out all defined areas of the disk.

FAST OPTION

The FAST option is identical to ALL except that only the first sector of each area is cleared to zeros. For disk packs, this will provide a considerable saving in the time required for SYSLOAD. The RADEDIT may be used subsequently to clear any areas that the user specifically requires to be cleared to zeros.

OVERRIDE OPTION (OVR)

The override option is identical to the ALL and FAST options except no disk area referencing or clearing takes place. Only area SP will be written to. This permits the user to add new disk areas while leaving other areas intact, and also permits one CP-R installation to generate a system for another CP-R installation with different disk areas and/or different secondary storage devices (only the SP areas must match).

UPDATE OPTION

The UPD option can be used whenever there is an existing system on the disk, and the user wishes to load a new version of the monitor or change some of the SYSGEN parameters. It is not necessary to go through a SYSGEN to load a new version of the monitor. It is only necessary to load the rebootable SYSLOAD deck and go through a normal SYSLOAD, specifying the "UPD" option on the :SYSLD command.

To change any SYSGEN-defined parameters, it is necessary to input the complete set of SYSGEN control commands. That is, there is no attempt to merge the new version of the monitor with the existing version on the disk.

If the user does not want to disturb any of the disk areas, the areas must be input with the same size and in the same order as the initial SYSGEN. If the size of a disk area has to be changed or a new disk area has to be added, all disk areas (except CK or BT) must be reloaded from the first changed area to the end of the disk. Therefore, the areas most subject to change in size should be allocated to the end of the disk so that the minimum number of areas are affected by a change. An area that must be moved can be saved and restored intact by using the RADEDIT Save and Restore functions. It is normally to the user's advantage to take the default size and allocation for the CK and BT areas since these are automatically allocated at the end of the disk and may be changed without affecting any other area.

To inform the user as to which areas on the disk have moved, SYSLOAD reads in the disk bootstrap from the existing version, determines where the monitor is located on the disk, and then inputs the Master Directory from the existing version. If the disk address is changed for any of the SP, FP, BP, XA, or Dn areas, SYSLOAD outputs an appropriate alarm, requests permission to continue, and then zeros out the first sector in each area that has moved, thus effectively erasing all data in the area.

The old system is examined to determine which disk areas have changed their allocation. If the size, location, or device have changed, an alarm is output and the operator is requested to indicate that the process should continue or not.

The size of the old CP-R file is examined to determine if it has grown larger than the previous system. If it has, then the SP area must be completely reloaded in the same manner as a FAST or ALL option. This is because the CP-R file is the first file in the area.

If the first word address of background is different in a new version from that of the existing version, the alarm

```
RELOAD, BGKG, PROGRAMS
```

is output. All programs that execute in the background, both system processors and user background programs, would then have to be reloaded and absolutized for their new core execution location.

If SYSLOAD determines that the new version is completely compatible with the existing version, the message

```
RELOAD, NOTHING
```

is output.

After typing the necessary RELOAD alarms, SYSLOAD loads the optional resident routines, the mandatory resident modules, and the overlays as described under the ALL option.

ALLOCATION OF SP AREA

When SYSLOAD has executed, the Systems Program area of the disk will have the following layout:

SP Directory: Entries for CPRFILE, JCP, CPRMAP, RADBOOT	Relative to sector 0
CPRFILE (Resident monitor and CP-R Overlays)	Sectors 1 to m
TEL File (if TJE is included)	Sectors m+1 to n
JCP File	
CPRMAP File	Sectors n+1 to p
RADBOOT File	Sector p+1
Unused SP AREA	Sectors (p+2) to q

SYSGEN AND SYSLOAD ALARMS

All alarm messages that can be output during SYSGEN and SYSLOAD are defined in Table 34.

LOADING SYSTEM PROCESSORS AND USER PROGRAMS

After SYSLOAD completes its operation, it will read CP-R into memory from secondary storage and transfer control to the CP-R initialization routine.

CP-R will type the message

XEROX CP-R VERSION xxxx

and execute a WAIT instruction. The operator should then place his job stack in the C device to load the appropriate programs, perform an interrupt, and key in a "C". Control will be transferred to the Job Control Processor to read the first control command.

If one of the ALL, FAST, or OVR options is specified to SYSLOAD, the system processors must be rebuilt. Additionally, if the UPD option is specified, but a message indicates the need to reload the SP area or system processor alternate area, or relink background programs, part or all of the system processors must be rebuilt.

The CP-R initialization routine provides the files in which the OLOAD and RADEDIT processors will reside, in the SP area (or system processor alternate area, if one is defined).

First, the OLOAD processor is linked by the JCP Loader, in a single-level overlay structure. OLOAD is then used to link the more complicated overlay structure of RADEDIT.

These two processors are then used to allot and link other processors and utility files.

Figure 31 shows an abbreviated control deck for linking the system processors from ROMs on oplabel TO.

Table 34. SYSGEN and SYSLOAD Alarm Messages

Alarm	Meaning	Recovery Action
	Non-I/O Alarms	
BI CKSM ERR	A checksum error has occurred in the object module being input.	The message 'CONTINUE?' will be output. The operator should type 'YES' or 'NO' to indicate what to do. If he types 'YES', the problem will be ignored and the process will continue. If he types 'NO' the process will be aborted.
BI SEQ ERR	A sequence error has occurred in the object module being input.	The message 'CONTINUE?' will be output. The operator should type 'YES' or 'NO' to indicate what to do. If he types 'YES', the problem will be ignored and the process will continue. If he types 'NO' the process will be aborted.
BT AREA TOO SMALL	The space allocated to the BT area is insufficient to hold the default sizes of the GO and OV files.	There is no recovery from this condition except to rerun the SYSGEN to either allocate more disk space for the BT area, or reduce the default size of the GO and/or OV file.

Table 34. SYSGEN and SYSLOAD Alarm Messages (cont.)

Alarm	Meaning	Recovery Action
	Non-I/O Alarms (cont.)	
CK AREA TOO SMALL	The amount of disk space allocated for the CK area is not sufficient to hold the initial size of background.	Either the disk areas must be reallocated (requires a rerun of SYSGEN) or a checkpoint cannot be done with the initial size of background.
DUP. DEF, xxxxxxxx	The same DEF has been encountered in two object modules, probably indicating that two copies exist of the same object module.	The message 'CONTINUE?' will be output. The operator should type 'YES' or 'NO' to indicate what to do. If he types 'YES', the problem will be ignored and the process will continue. If he types 'NO' the process will be aborted.
EOF BEFORE END ITEM	During the loading of an object module, SYSLOAD has encountered a misplaced EOD or EOF.	The message 'CONTINUE?' will be output. The operator should type 'YES' or 'NO' to indicate what to do. If he types 'YES', the problem will be ignored and the process will continue. If he types 'NO' the process will be aborted.
ERR,CONTROL BYTE = xx	The xx control byte in the object module being loaded cannot be processed by SYSLOAD.	The message 'CONTINUE?' will be output. The operator should type 'YES' or 'NO' to indicate what to do. If he types 'YES', the problem will be ignored and the process will continue. If he types 'NO' the process will be aborted.
ERROR ITEM xx	An error has occurred in item xx of the last control command input. Every item (except the :), followed by a blank or a comma, is counted in determining the one in error. If xx is one greater than the last item input, a nonoptional item was not input.	Control will be transferred to TYA01 to allow the user to correct the error. Unless stated otherwise (where the individual commands are described) all items preceding the incorrect one have been processed, and only items starting with and following the incorrect one need by input. If the user desires to input nothing from TYA01 and to transfer control back to the original input device, a single colon (:) should be input on TYA01. If an error occurs on a continuation card, a card containing a control command must follow.
ILL. DEF, xxxxxxxx ILL. REF, xxxxxxxx	The specified DEF or REF is not recognized by SYSLOAD during the loading of an object module.	The message 'CONTINUE?' will be output. The operator should type 'YES' or 'NO' to indicate what to do. If he types 'YES', the problem will be ignored and the process will continue. If he types 'NO' the process will be aborted.
INPUT ORDER ERROR	The :MONITOR, :RESERVE, or :DEVICE command for the system disk has been input in the wrong order.	Catastrophic error. Rerun SYSGEN from the start.
MISSING ID	See ALL option.	See ALL option.
NO SYSTEM DISK	No disk has been designated as the system disk.	Catastrophic error. SYSGEN must be rerun from the start.
OBJ. MOD. NOT RECOG.	The current object module being loaded by SYSLOAD is not recognized by SYSLOAD.	The message 'CONTINUE?' will be output. The operator should type 'YES' or 'NO' to indicate what to do. If he types 'YES', the problem will be ignored and the process will continue. If he types 'NO' the process will be aborted.
OC LABEL NOT ASSIGNED	The OC operational label has not been assigned to a typewriter device.	There is no recovery from this error. The OC label must be assigned in order for the system to function.

Table 34. SYSGEN and SYSLOAD Alarm Messages (cont.)

Alarm	Meaning	Recovery Action
	Non-I/O Alarms (cont.)	
DISK OVERFLOW	The total number of tracks input on the :DEVICE command have exceeded the total available size.	The :DEVICE command must be completely re-input, with the sizes of the areas appropriately changed.
RELOAD XX AREA RELINK BACKGROUND PROGRAMS RELOAD NOTHING	These messages are output when an update option was used on the :SYSLD command. They indicate various incompatibilities between the new and the old systems.	The message 'CONTINUE?' will be output. The operator should type 'YES' or 'NO' to indicate what to do. If he types 'YES', the problem will be ignored and the process will continue. If he types 'NO' the process will be aborted.
UNABLE TO FIND OLD CP-R	During an update run, SYSLOAD was unable to locate the old version of CP-R on the disk.	The message 'CONTINUE?' will be output. The operator should type 'YES' or 'NO' to indicate what to do. If he types 'YES', the problem will be ignored and the process will continue. If he types 'NO' the process will be aborted.
	I/O Alarms	
yyndd BUSY IOP n BUSY	The indicated device or IOP has returned a busy status.	SYSGEN will keep attempting the I/O operation. Probably indicates a hardware problem.
yyndd ERROR,SB=xxxx yyndd PARITY,TRK=xxxx	A transmission error has occurred with the indicated device. SB=xxxx indicates the contents of the TIO status bytes in hexadecimal. If a parity occurs while clearing the disk, the bad track, as returned in the sense order, is also logged in hexadecimal.	SYSLOAD continues attempting the I/O operation, unless a parity has occurred while clearing the disk. In this case, this alarm and the parity alarm will be logged and the disk clearing will continue.
yyndd FAULT, TDV=xxxx	A hardware fault has occurred on the indicated device. The TDV status byte is also output in hexadecimal.	SYSGEN continues attempting the I/O operation. Repair and ready the indicated device.
yyndd MANUAL	The indicated device is in manual mode.	Ready the device.
yyndd UNRECOG	The device indicated by yyndd is unrecognized by the system.	SYSGEN will enter a "WAIT" state. Probably an invalid device number was input, and the SYSGEN will have to be rerun from the start. If the "WAIT" state is cleared, SYSGEN will retry the I/O operation.
yyndd UNUS. END, TDV=xxxx	An unusual end status has been returned from the indicated device. The TDV status byte is also logged in hexadecimal.	SYSGEN continues attempting the I/O operation.
yyndd WRT PROT	The indicated magnetic tape or disk is hardware write-protected.	For a magnetic tape, insert a write ring and ready the tape. For a disk, reset the hardware Write Protect switch and then clear the "WAIT" state so SYSLOAD can retry the I/O operation.

```
!JOB BLD,PROCESSORS.
!ATTEND
!STDLB  (BI,TO)
!PAUSE  KEYIN SYC
!ALLOBT (FILE,GO),(FSI,0),SAVE
!ALLOBT (FILE,OV),(FSI,0),SAVE
!ALLOBT (FILE,X1),(FSI,800),(RSI,30),(FOR,B),SAVE
!LOAD (OUT,SP,OLOAD),(SEGS,6),MAP
!OLOAD (FILE,SP,RADEDIT),(TEMP,50),(MAP,PROG),LIB,FORE
:ROOT (OPLB,BI,EOD)
:SEG (LINK,999),(EXLOC,10000),NONE
:RES (XBPEND,20000)
:SEG (LINK,1000,ONTO,999),(EXLOC,8000),(OPLE,BI,EOD);
:    (SHARE,SYS),(ACCESS,RX),ILOAD
:SEG (LINK,1001,ONTO,1000),(OPLB,BI,EOD),(SHARE,SYS),(ACCESS,RX)
:SEG (LINK,1002,ONTO,8000),(OPLB,BI,EOD),(SHARE,SYS),(ACCESS,RX)
:SEG (LINK,1003,ONTO,1000),(OPLE,BI,EOD),(SHARE,SYS),(ACCESS,RX)
:SEG (LINK,1004,ONTO,1000),(OPLB,BI,EOD),(SHARE,SYS),(ACCESS,RX)
!RADEDIT
:ALLOT (FILE,SP,CRASH),(RSIZE,256),(FSIZE,514)
:ALLOT (FILE,SP,ERRFILE),(RSIZE,16),(FSIZE,100),(FORMAT,B)
:ALLOT (FILE,SP,AI),(RSI,20),(FOR,B),(FSI,100)
:ALLOT (FILE,D1,AL),(RSI,8),(FOR,B),(FSI,500)
:ALLOT (FILE,SP,EDIT),(FSI,100),(RSI,256)
!OLOAD (FILE,SP,EDIT),(TEMP,25),(LIB),(MAP,PRO),FOR
:ROOT (OPL,BI,EOD)
:SEG (LIN,1066),ILOAD,(OPL,BI,EOD),(SHARE,SYS),(ACC,RX)
!RADEDIT
:TRUNCATE (FILE,SP,EDIT)
:SQUEEZE SP
```

Figure 31. Linking the CP-R System Processors (abbreviated)

16. HARDWARE CONFIGURATION GUIDELINES

INTRODUCTION

This chapter is intended as an aid to the system manager in selecting the proper hardware for his CP-R system. A reasonable selection of hardware can be made only on the basis of a thorough understanding of the particular application's requirements. Requirements that must be evaluated include the number of discrete interrupt levels required, the amount of secondary storage required for programs and data, memory requirements to satisfy resident foreground needs and concurrent batch (if desired), and peripheral equipment requirements for the data media desired. Figure 32 illustrates the lowest-cost minimum CP-R configuration. Figure 33 illustrates a more typical CP-R configuration.

HARDWARE INTERRUPT REQUIREMENTS

The maximum number of primary foreground tasks that are expected to operate concurrently determines the number of hardware-interrupt levels required, in addition to those needed by CP-R itself. The association of an interrupt level with a task establishes the priority of the task. The task's worst-case response time (to an external stimulus) is dependent upon the maximum CP-R inhibit time and the possible activity of higher priority tasks. Tasks that are prioritized above the I/O interrupt task must contend only with the CP-R typical inhibit of 100 μsec and possible interference among themselves. (With error logging, the worst-case inhibit time is 400 μsec.)

Tasks associated with an interrupt priority higher than I/O cannot utilize any CP-R services that involve I/O; thus these higher interrupt levels are typically associated with high priority tasks that utilize the Direct I/O interface with SIUs and defer CP-R services to related, lower-priority tasks.

Tasks that operate at a priority lower than the I/O interrupt level can utilize all CP-R services including I/O. Any number of these tasks may operate concurrently up to the

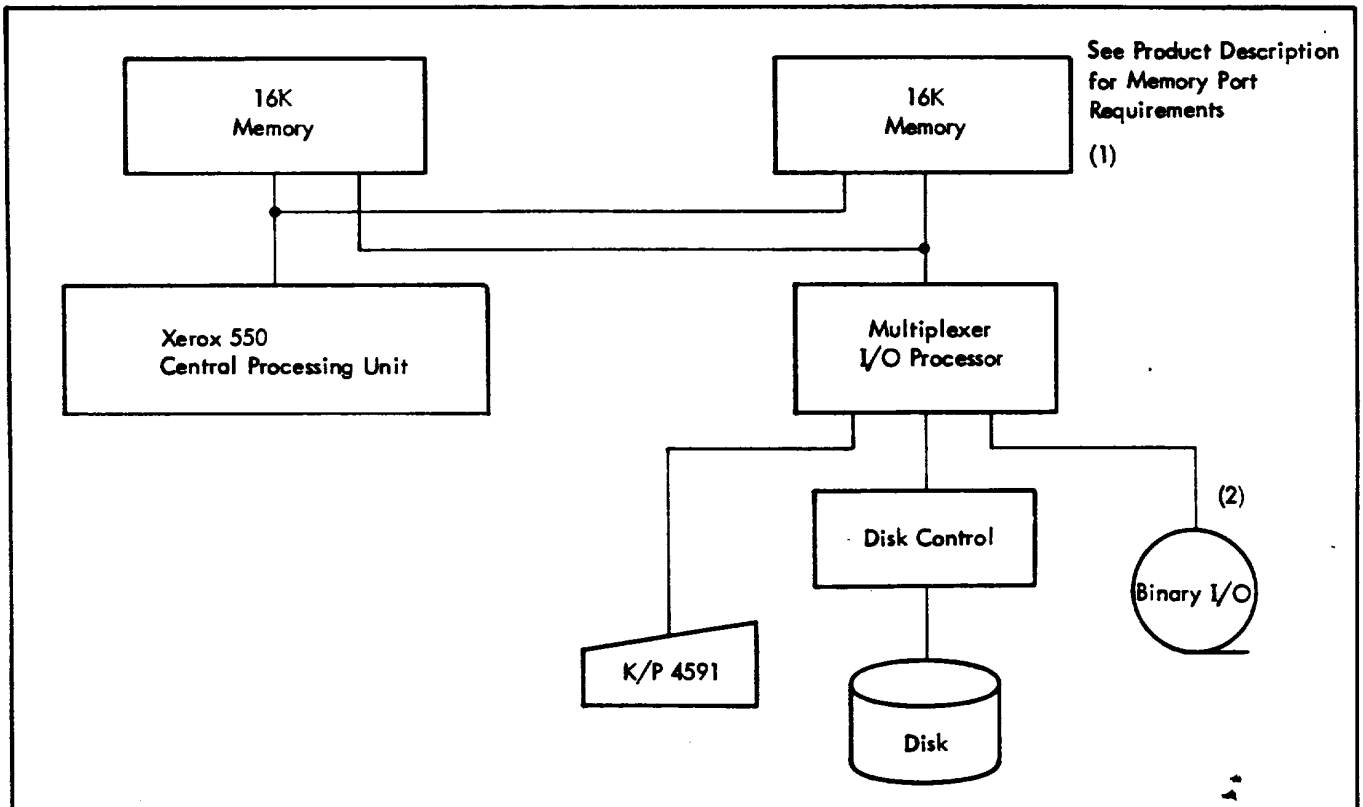


Figure 32. CP-R Lowest-Cost, Minimum Configuration

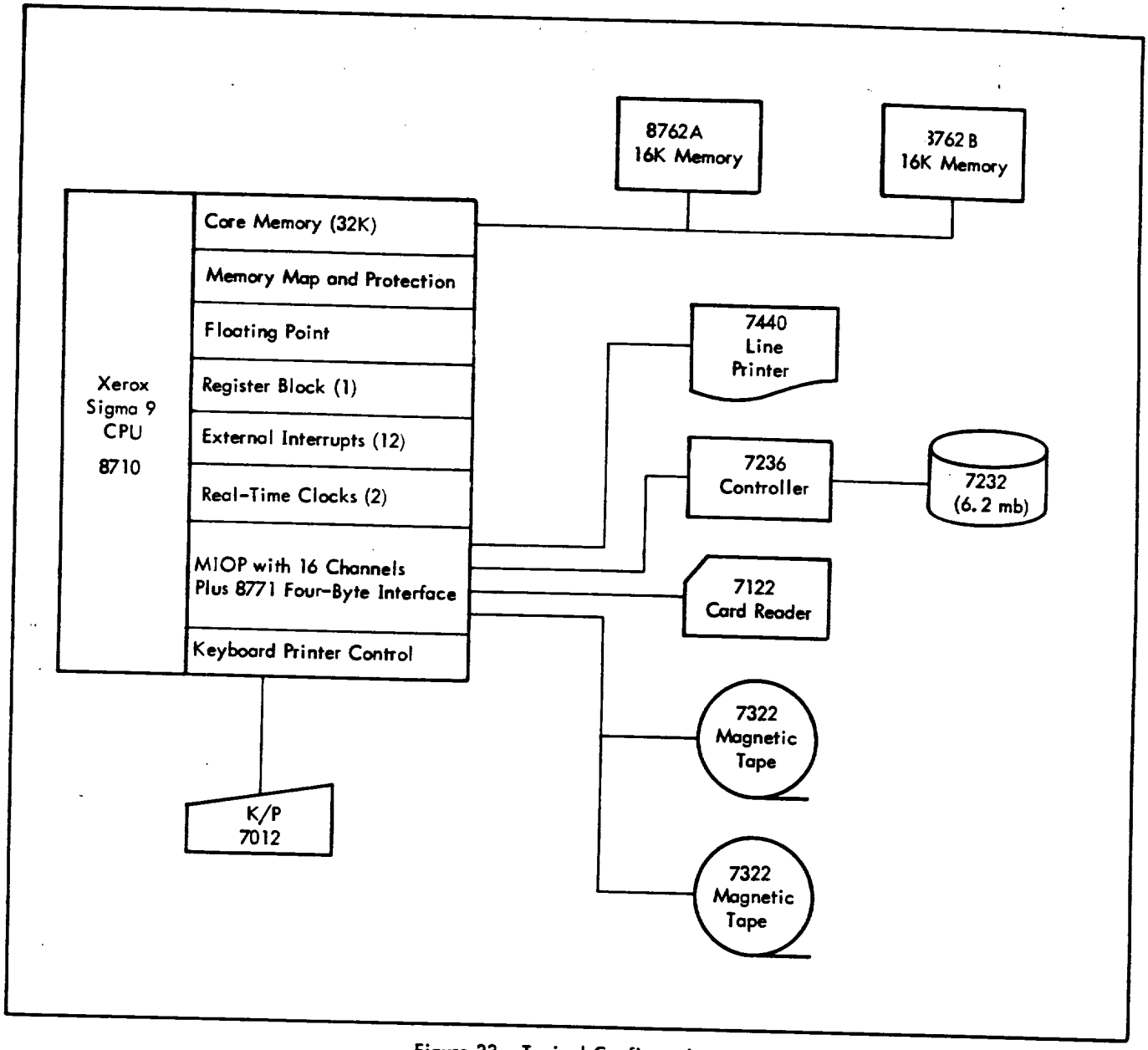


Figure 33. Typical Configuration

number of available lower-priority hardware interrupts minus two for each CP-R dispatcher level. (Each CP-R dispatcher requires the two hardware interrupt levels.) A dispatcher may be connected to the lowest interrupt level, in which case, the other interrupt level is null. For tasks below the I/O level, the I/O interrupt can cause an interference of up to approximately 500 microseconds.

There is an option in CP-R to allow I/O end-action and I/O cleanup to be postponed to a level lower than the I/O level. The user specifies this hardware level, if desired. Primary tasks between this level and the I/O level have less interference (due to I/O interrupt activity).

If 7611 (COC) equipment is available for the terminal job entry system, two external interrupts are required for each 7611.

Although external hardware-interrupt groups on a Sigma system can be arranged in any priority order the user desires, CP-R requires of a Sigma configuration (and assumes) that group 2 is higher than group 3, group 3 higher than group 4, and so forth. Therefore, if any one group is to be above the I/O level in priority, it must be group 2.

The same hardware interrupt can be used "serially" by different tasks, but only one can be connected to a given interrupt at any one time. Table 35 gives a summary of the Sigma 9 and Xerox 550 interrupt structures.

Table 35. Interrupt-Structure Summary

Maximum Number of External Interrupts	Clocks†	Priority Levels > I/O	Priority Levels < I/O††
Sigma 9: 224 standard and optional combined.	2 standard 2 optional	208 optional	224 optional
Xerox 550: 48 standard and optional combined.	4 standard	12 optional	36 optional

†Clock 4, set at a frequency of 500 Hz, is dedicated to provide CP-R services relating to I/O timeouts, job accounting, and background-execution-time limit.

††CP-R requires that two hardware interrupt levels be dedicated to each CP-R dispatcher.

MAIN STORAGE (MEMORY) REQUIREMENTS

MEMORY SPACE REQUIREMENTS FOR CP-R

The minimum CP-R configuration which would include keyboard/printer, magnetic tape, and disk I/O routines only, and provide a minimum number of disk device-files and operational labels, requires about $12,000_{10}$ cells for the monitor and its tables. This minimum memory space requirement would increase as handlers are added for additional peripherals, as optional software routines are chosen during SYSGEN, and as additional files, operational labels, or user tasks are allocated during SYSGEN. The residence-space requirements for CP-R will vary from 12,000 to 22,000 cells, depending upon the user's configuration. If background processing is desired, the user must allow an additional minimum of 9,000 cells for background space in order to accommodate the CP-R Job Control Processor, the standard CP-R subsystems, and the Assembly Program (AP). See the "System Generation" chapter for information on the space requirements of the various monitor options.

MEMORY SPACE REQUIREMENTS FOR CP-R PROCESSORS

The memory space required for concurrent background processing is a function of the longest overlay path of the set of processors in question, plus the maximum work-space and blocking-buffer requirement of that set. The standard CP-R language processors (FORTRAN IV and AP) require resident work space for symbol-description tables during compilation or assembly. Thus, the number of source statements that can be processed in a single compilation or assembly is directly proportional to the amount of background memory available over and above the longest-overlay-path requirement.

All of the CP-R processors are designed to utilize memory efficiently and will take advantage of any excess space that may be available.

Table 36 shows the various combinations of processors that are operable in the context of several sample CP-R/memory configurations.

Table 36. Processor Availability in Sample Configuration

	Total Memory in System		
Configuration	32K	48K	64K
12K CP-R plus 8K resident (primary) foreground, no secondary foreground, no PUBLIB.	9K BKGND All CP-R Processors except FORTRAN IV and SL-1	17K BKGND All Processors (AP assembly of about 10,000 lines)	25K BKGND All Processors
12K CP-R, 1K resident (primary) foreground, 16K secondary foreground, no primary PUBLIB.	16K BKGND† All Processors†	24K BKGND† or 8K BKGND†† All Processors†	32K BKGND† or 16K BKGND†† All Processors

†Only when secondary foreground is inactive and rolled out.

††Concurrent with secondary foreground (real-time).

MEMORY SPACE REQUIREMENTS FOR USER-FOREGROUND PROGRAMS

The amount of memory required for user's primary foreground programs is the total size requirements for all concurrently resident primary programs and their blocking buffers, plus the size of the nonresident foreground area if any.

The overhead for centrally-connected primary tasks is 26 words per interruptible task, for the task control block and the task's PSD, plus temp space as follows:

Monitor Services Requirements	Amount of System Temp Stack Required
None	0
All services	150

More detailed information is given under "Overlay Loader" and "System Generation".

The user temp stack size is dependent upon the routines used. See the FORTRAN IV reference manual for library sizes.

Blocking buffers are required for each operational label (or FORTRAN device number) that is associated with a blocked disk file. The blocking buffer size is always 256 words.

SECONDARY STORAGE REQUIREMENTS

The optimal type of secondary storage for a CP-R system is dependent upon the requirements of the installation. The storage capacity required may dictate the use of disk packs, or perhaps where access time is very important and the majority of transfers are small, a fixed-head disk would be appropriate. (Note that for small average-transfer sizes on the order of 500-1000 bytes, access time is more critical than transfer rate.)

CP-R requires one magnetic tape unit, which also provides an optimum backup capability for CP-R and disk packs. Table 37 gives a comparison of the access time, capacity, and other pertinent characteristics of the various secondary storage devices supported by CP-R on Sigma 9 hardware.

Table 37. Comparison of Secondary Storage Devices

Device or Device/Controller Combination	Capacity[1] (in megabytes)	Average Access Time (in milliseconds)	Maximum Transfer Rate (in kilobytes)	MIOP Bandwidth[2] Requirements	
				Sigma 9 MIOP	Add if thru a 7720
7202	.75 mb	16.9 ms	187.5 kb	37%	7%
7203	1.5 mb	16.9 ms	187.5 kb	37%	7%
7204	3.0 mb	16.9 ms	187.5 kb	37%	7%
7212	5.3 mb	16.9 ms	3000.0 kb	NA[11]	NA
7232/7231	6.3 mb	16.9 ms	384 kb	77%	16%
7232/7236[3]	6.3 mb	16.9 ms	384 kb	34%	5%
7246[4]	24.5 mb	87.5 ms	312 kb	50% - 29%	13% - 3%
7251[5]	2.3 mb	50.5 ms	312 kb	50% - 29%	13% - 3%
7260	45 mb	42.5 ms	512 kb	51% - 39%	4% - 5%
7270	49 mb	47.5 ms	312.5 kb	50% - 29%	13% - 3%
7275	86 mb	42.5 ms	806 kb	77% - 65%[6]	7%
7315/16	23 mb @ 800 bpi[7]	5 ms	60 kb	14%	3%
7332	46 mb @ 1600 bpi[7]	5 ms	120 kb	22% - 13%	6% - 1%
7333[8]	46 mb @ 1600 bpi[7]	3.25 ms	240 kb	25%	2%
7362	16 mb @ 556 bpi[7]	10 ms	20 kb	9%	1%
7372	23 mb @ 800 bpi[7,9]	5 ms	60 kb[10]	13%	3%

Notes:
1. CP-R requires approximately .5 megabytes for system storage, including standard processors.
2. CP-R does not manage bandwidth control, thus the installation must be properly configured to avoid data overruns. When two figures are shown, the second figure indicates bandwidth requirements when the four-byte interface is included; this figure is not shown if not applicable. The user can combine

Table 37. Comparison of Secondary Storage Devices (cont.)

Notes: (cont.)

several devices on a logical channel (using the CHAN command at SYSGEN) even though they are on separate physical channels.

3. CP-R requires the 7236 controller if the extended-width interface is desired. The 7231/7235 combination may not be used.

4. CP-R treats each 7242 as two 7246 disk packs. All disk packs except the system-residence pack may be declared to be removable.

5. CP-R treats each 7251 as a 720x type device. Each 7252 is treated as two 7251 devices. The removability feature of the 7251/52 is not managed by CP-R. A restricted form of removability for 725x cartridges is available (see "Removable-Disk Devices"). There is no support for flawed tracks under CP-R for this device, unlike for other disk devices.

6. The 7275 uses the burst-mode feature of the Sigma 9 MIOP, which is included with all systems configured for Model 7275. The second figure reflects the bandwidth requirement when using burst mode.

7. Assuming 2400-foot reels and no interrecord gaps. Reduce this figure by the percentage of the tape required for interrecord gaps. This value is a function of record size since gap is fixed (0.75 inches for 7362/72; 0.5 to 0.75 inches (average 0.6 inches) for 7315/16).

8. Model 7333: only with extended-width interface (1039).

9. Model 7372 transfers six bits of information per byte.

10. Model 7372: reduce transfer rate by 25 percent for packed binary mode.

11. Requires Sigma Selector IOP (SIOP) with Model 7211 controller.

The amount of secondary storage required for a CP-R system is a function of the secondary storage requirements of CP-R itself, plus user requirements for program and data storage, plus background temporary file space.

BT AREA STORAGE REQUIREMENTS

If a user wants to allocate enough Background Temp area to assemble an average 5000-line source program, approximately 200 kilobytes of disk storage would be needed. (The smallest Xerox disk unit, Model 7202, has a .75 megabyte storage capacity.

USER SECONDARY STORAGE REQUIREMENTS

Figure 34 illustrates the hierarchy of CP-R file management. It is important to note that since a file may not span physical device boundaries, the capacity of secondary storage devices at an installation may limit the maximum file size. Furthermore, the maximum size of any file or area is 32 megabytes, independent of device capacity.

REMOVABLE DISK PACKS AND CARTRIDGES

DISK AREA DEFINITION AND PACK INITIALIZATION

Each disk drive provides from 24 to 86 megabytes of storage, which can be divided into from 1 to 21 areas. (The total number of areas in the system may not exceed 21.) An area can be up to 32 mb in size.

Also, part or all of a pack can be used as a direct (random) file with no area limitations, by means of a device-access type of assignment.

7244 PACK COPY

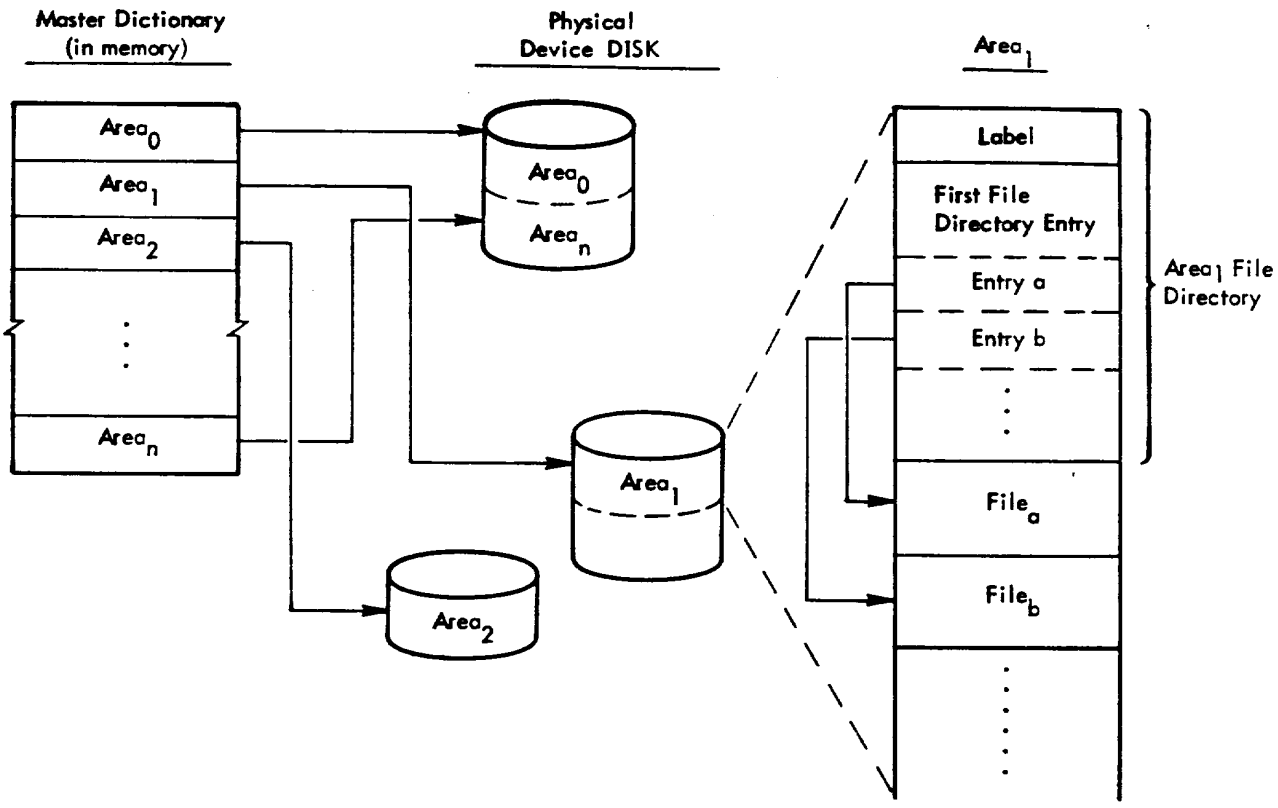
The RADEDIT command DPCOPY may be used to create a backup copy of a pack on another pack. The user must first assign a DCB to each device, with a device-type assignment, and then execute RADEDIT.

7251/52 DISK AREA DEFINITION

All areas for the removable 7254 disk cartridges (for Model 7251 or 7252 Cartridge Disk Systems) must be entirely defined at system generation time (SYSGEN). Any cartridge mounted on a given spindle will automatically assume the attributes of that spindle as defined at SYSGEN. If areas must be redefined, a complete 'ALL' SYSGEN must be performed.

7254 CARTRIDGE INITIALIZATION

SYSGEN will automatically initialize the first and second sector of each area defined on any cartridges mounted at SYSGEN. When interchangeable cartridges are to be



The following rules apply:

- A physical device (e.g., DISK) may contain several areas. An area is totally contained on one physical device.
- An area may contain one or more files. A file is totally contained within one area.

Figure 34. CP-R Disk File Management

initialized subsequent to SYSGEN, they must be mounted on the appropriate spindle and a RADEDIT CLEAR command given for each area associated with that spindle. This must be done prior to using a RADEDIT MAP or ADD command. Files subsequently defined will be available whenever that cartridge is mounted.

7254 CARTRIDGE MOUNTING

Any cartridge mounted on a spindle automatically assumes the area definitions defined at SYSGEN for that spindle; therefore, interchangeable cartridges may only be mounted on a spindle compatible with that cartridge. The cartridges are compatible only when they have been initialized with the exact area allocations as were defined at SYSGEN for that spindle. Cartridges must be initialized, as described previously, prior to use.

7254 CARTRIDGE REMOVAL

Cartridges may not be removed until all files contained on that cartridge are closed and the device-file numbers are released. Cartridges used exclusively by background should only be removed or replaced between sequences of jobs, i.e., following a FIN command and preceding a JOB command. Cartridges used by foreground tasks can be safely removed only on the basis of a thorough understanding of the foreground application. It is recommended that the foreground tasks communicate to the operator when all file activity has ceased (files closed and device-file numbers released).

Special precaution must be taken for 7252 Cartridge Disk units. Since the fixed disk as well as the removable disk is stopped when replacing the cartridge, all file activity must be quiesced for both the fixed and removable portions of the device (which are independently addressable). If the

SP area resides on the fixed disk, the computer must be placed in a completely idle state (no active background or foreground, no key-ins pending). Failure to do so will result in an ALARM when an attempt is made by CP-R to access an overlay. Foreground tasks that access files on the fixed portion of a 7252 drive may also require the computer to be in the idle state to avoid the 'non-operational' condition. Obviously a critical real-time environment cannot tolerate the computer being in 'idle', therefore, proper allocation of areas and spindles is essential (e.g., two 7251 devices may be required rather than a single 7252).

7254 CARTRIDGE COPIES

In order for 7254 back-up cartridges to be created, two 'removable' spindles should be available with identical area-size definitions but with different area-name assignments. As indicated previously, the cartridge to be copied to must have been previously initialized. RADEDIT may then be used to provide an area-to-area copy.

If an installation has only two spindles with one being the system-residence device (i.e., 'nonremovable') and desires to create backup cartridges, the cartridges to be copied must themselves contain a copy of CP-R. The cost of this is approximately 10 percent of the cartridge capacity. Although it is possible to perform a cartridge copy on a single 7252 unit (i.e., a copy of the fixed-disk portion to the cartridge), the procedure would be cumbersome and error prone. It is therefore recommended that two 7251s be utilized instead of a single 7252. (Multiple 7252s or a combination of 7252 and 7251 are equivalent to multiple 7251s in this respect, of course, ignoring the cost factor.)

PERIPHERAL EQUIPMENT REQUIREMENTS AND OPTIONS

KEYBOARD/PRINTER – REQUIRED

CP-R requires that the minimum configuration include a keyboard/printer for operator control. This must always be located on MIOP 0, device address 01. (All other device addresses are determined at SYSGEN time.)

CHARACTER-ORIENTED COMMUNICATIONS (COC) – OPTIONAL

- One buffered input/output channel dedicated to a 7630 Controller, with eight lines.
- External DIO interface feature.
- Two external interrupts dedicated to the COC controller.

The COC software support requires approximately 2K words of resident memory space, including some line-buffer space.

ADDITIONAL PERIPHERAL OPTIONS

Consult the "System Generation" chapter for information about other optional peripheral devices supported by CP-R.

MISCELLANEOUS HARDWARE OPTIONS

REGISTER BLOCKS

Additional register blocks can be used by primary real-time tasks to reduce context-switching time on task entry and exit. If terminal job entry is selected, register block one is required.

POWER ON/OFF

Although this feature is standard in Sigma 9 hardware, CP-R software does not provide a generalized support for it, since recovery sequences are so application-dependent in a real-time system. Users interested in a power fail-safe capability can handle these interrupts by means of a primary task that provides for the installation's specific requirements in this regard.

APPENDIX A. SYSTEM CP-R

System CP-R is written for use under the AP (level A01 or greater) assembler.

SYSTEM CP-R PROCEDURE REFERENCES

For any CAL that requires a Function Parameter Table (FPT), System CP-R provides a procedure that can generate four different CAL and FPT combinations. The combination is chosen by a keyword in the procedure CF(2) field.

DEFAULT FORM

The default format is used when the CF(2) field of the procedure is absent. It has the form

```
LABEL1, LABEL2      M:CAL      parameters
```

In this form, the CAL instruction is generated at the position of the procedure reference. The FPT is generated in the next available space of a separate CSECT, which is allocated by the first use of a procedure of this form. The parameters define the size and content of the FPT. LABEL1, if it exists, is defined as the address of the CAL. LABEL2, if it exists, is defined as the address of the first word of the FPT. Further labels are ignored.

IN-LINE FPT FORM

Another form is provided to avoid generating a separate CSECT. It is similar to the default form, but generates the FPT preceded by a branch instruction following the CAL. The branch address is the first word after the FPT. It has the form

```
LABEL1, LABEL2      M:CAL, INLN      parameters
```

The keyword "INLN" in the CF(2) field specifies the in-line FPT form of the procedure. The labels and parameters are used in the same way as for the default form.

FPT-ONLY FORM

This form generates only the FPT, at the position of the procedure reference (with no branch instruction preceding the FPT). It has the form

```
LABEL1, ..., LABELn      M:CAL, FPT      parameters
```

The keyword "FPT" specifies this form of the procedure. The parameters are treated as in the default form, but all labels are defined as the first word address of the FPT.

CAL-ONLY FORM

The last form generates only the CAL, at the position of the procedure reference. It has the form

```
LABEL1, ...LABELn      M:CAL, CAL      address
```

The keyword "CAL" specifies this form of the procedure. The labels are all defined as the address of the CAL. The "address" field is the word address of an FPT, optionally with indexing or indirection.

PARAMETER FIELD

In the first three procedure forms above, the argument field is a list of parameters defining an FPT. With a few exceptions, each parameter in the list is a parenthesized group consisting of a keyword, possibly followed by a value. Certain keywords ("flags") indicate only a single bit of information by their presence or absence. If a value follows one of these flags, it is considered an extra value. Other keywords normally are followed by a value. If the value is omitted, zero is assumed.

If a parameter group specifies a value for an FPT field, a label may be defined as the address of the word containing the field. This is done by including the label as the third item in the parameter group. For example,

```
M:INIT      (TASK, 'ABCD', HERE)
```

Defines the label 'HERE' as the address of the task name field in the INIT FPT. Flag parameters which signify the inclusion of a full word (i.e., 'TYC' or 'TYPE2') may label that word by including the label as the second item in the parameter group.

Two prominent exceptions to the "(keyword, value)" format for parameters are the DCB address for the I/O CALs and the file identification for several other CALs. The former is a single unkeyed value, which must be first in the parameter list (this provides compatibility with System BPM). The latter is the keyword "FILE" followed by either one or two text values. If two values follow, they are assumed to be the area name and the file name, in that order. If only one value follows, it is assumed to be the file name. In either case, a label may follow the last parameter. It will take the address of the file name field as its value.

In any case where a parameter is a single item rather than a group of items, the parentheses may be omitted.

If an FPT word is optional, it is generated only if a parameter is referenced which affects that word. The parameter presence bits are set accordingly.

SYSTEM CP-R ERRORS

Generally, the procedure system generates as much as it can determine from the procedure reference, notifying the user with error messages that certain information was unusable.

Extra items in a parameter group are unused. If a value item is in error, a zero value is used. If a keyword is in error, the parameter group which it heads is ignored (which may cause an FPT optional word to be omitted).

CAL-PARAMETER GROUPS

The system CP-R CAL procedures each translate a set of parameters for a group of CALs with similar FPTs. Within such a group, any parameter will be accepted for any CAL, even though the parameter might be meaningless or illegal for that CAL. It is the user's responsibility to know which parameters are necessary for the CALs used.

SPECIFIC CAL-PARAMETER GROUPS

Listed below in Figure A-1 are the CALs in the groups for each procedure in system CP-R, followed by the parameters processed by that procedure.

INDEX BY SERVICE NAME

SERVICE	CODE	GROUP	SERVICE	CODE	GROUP	SERVICE	CODE	GROUP
ABORT	X'03'	2	ACT	X'52'	11	ALARM	X'6A'	11
ALLOT	X'5A'	11	ARM	X'04'	12	ASSIGN	X'08'	17
BIOSTP	X'0E'	20	BIOSTRT	X'0F'	20	CALRTN	X'60'	4
CHECK	X'29'	1	CLOSE	X'15'	1	CON	X'04'	12
CORRES	X'2B'	7	DCB		22	DEACT	X'53'	11
DEBUG	X'65'	11	DELETE	X'5B'	11	DELFPT	X'47'	1
DEQ	X'4D'	11	DFMODE	X'22'	19	DISABLE	X'01'	10
DISARM	X'03'	12	DISCON	X'03'	12	DRC	X'0B'	19
ENABLE	X'02'	10	ENQ	X'4C'	11	ERASE	X'54'	11
ERSEND	X'66'	1	EXDA	X'0A'	2	EXIT	X'01'	2
EXTM	X'49'	11	GETASN	X'09'	17	GETPAGE	X'57'	11
GETTIME	X'61'	5	HIO	X'15'	21	INIT	X'48'	11
INT	X'0E'	3	IOACT	X'17'	20	IODEACT	X'16'	20
JOB	X'67'	15	JTRAP	X'5D'	9	KJOB	X'64'	11
LOCK	X'55'	11	MASTER	X'08'	4	MEDIA	X'59'	17
MODIFY	X'4F'	11	OPEN	X'14'	1	PC	X'2C'	3
PFIL	X'1C'	1	POLL	X'45'	11	POST	X'46'	11
PREC	X'1D'	1	PRECORD	X'1D'	1	PREFMOD	X'50'	16
PRINT	X'01'	6	READ	X'10'	1	RECALM	X'69'	11
RELPAGE	X'58'	11	REW	X'01'	1	RLS	X'0B'	8
RUN	X'0C'	8	SEGLOAD	X'01'	13	SETNAME	X'51'	11
SIGNAL	X'43'	11	SIO	X'12'	21	SIOSTP	X'10'	20
SIOSTRT	X'11'	20	SJOB	X'63'	11	SLAVE	X'07'	4
START	X'4A'	11	STATUS	X'4E'	11	STDLB	X'62'	17
STIMER	X'44'	11	STOP	X'4B'	11	TDV	X'14'	21
TERM	X'08'	2	TEST	X'42'	11	TEXIT	X'5F'	11
TIME	X'10'	3	TIO	X'13'	21	TRAP	X'14'	9
TRIGGER	X'00'	10	TRTN	X'05'	2	TRTY	X'5E'	11
TRUNC	X'5C'	11	TYPE	X'02'	6	UNLD	X'03'	1
UNLOCK	X'56'	11	VFC	X'05'	19	WAIT	X'09'	2
WAITALL	X'40'	11	WAITANY	X'41'	11	WEOF	X'02'	1
WRITE	X'11'	1	XXX	X'03'	2			

Figure A-1. Specific CAL-Parameter Groups

```
* * * * * * * * * * * * * * * * * GROUP 1 * * * * * * * * * * * * * * * * *

                        NAME          CAL       FPT CODE

                        M:CHECK       CAL1,1    X'20'
                        M:CLOSE       CAL1,1    X'15'
                        M:DELFPT      CAL1,7    X'47'
                        M:FRSEND      CAL1,7    X'66'
                        M:OPEN        CAL1,1    X'14'
                        M:PFIL        CAL1,1    X'1C'
                        M:PREC        CAL1,1    X'1D'
                        M:PRECORD     CAL1,1    X'1D'
                        M:READ        CAL1,1    X'10'
                        M:REW         CAL1,1    X'01'
                        M:UNLD        CAL1,1    X'03'
                        M:WFOF        CAL1,1    X'02'
                        M:WRITE       CAL1,1    X'11'


KEYWORD                 MEANING AND SAMPLE
(WITH ALTERNATES)

(NO KEYWORD)            DCB ADDRESS (REQUIRED TO BE FIRST)
                        M:READ  DCB
ABN                     ABNORMAL RETURN ADDRESS (MAY BE INDIRECT)
                        M:READ  DCB,(ABN,XXXX)  OR  (ABN,*XXXX)
BBA                     BLOCKING BUFFER ADDRESS FOR OPENING FILE
                        (MAY BE INDIRECT)
                        M:READ  DCB,(BBA,XXXX)   OR   (BBA,*XXXX)
BLOCK                   GRANULE NR FOR DIRECT READ, OR ADDRESS OF SAME
                        M:READ  DCB,(BLOCK,N) OR (BLOCK,*XXXX)
BTD                     BYTE IN WORD TO START, OR ADDRESS OF SAME
                        M:READ  DCB,(BTD,N) OR (BTD,*XXXX)
BUF,BUFFER              BUFFER ADDRESS (MAY BE INDIRECT)
                        M:READ  DCB,(BUF,XXXX) OR (BUF,*XXXX)
BUSY                    BUSY RETURN ADDRESS
                        M:READ  DCB,(BUSY,XXXX)
CKWRT                   WRITE WITH CHECKREAD TO DISK
                        M:WRITE DCB,(CKWRT)
DCB                     INDICATES DCB BEING CHECKED
                        M:CHECK DCB,(DCB)
DIAG                    OPERATION WILL BE VALID ONLY IF DEVICE
                        IS IN DIAGNOSTIC MODE
                        M:READ DCB,DIAG
EAADDR                  END ACTION ADDRESS
                        M:READ  DCB,(EAADDR,XXXX)
EACSA                   END ACTION COMPLETION STATUS ADDRESS
                        M:READ  DCB,(EACSA,XXXX)
EAINTLB                 END ACTION INTERRUPT LABEL
                        M:READ  DCB,(EAINTLB,'I3')
EAINTNR                 END ACTION INTERRUPT NUMBER
                        M:READ  DCB,(EAINTNR,X'65')
ERR,ERROR               ERROR RETURN ADDRESS (MAY BE INDIRECT)
                        M:READ  DCB,(ERR,XXXX)  OR  (ERR,*XXXX)
FPT                     INDICATES FPT BEING CHECKED
                        M:CHECK DCB,(FPT)
FUNC                    FUNCTION CODE FOR SPECIAL DEVICE HANDLER,
                        OR ADDRESS OF SAME
                        M:READ  DCB,(FUNC,X'12')  OR  (FUNC,*XXXX)
FWD                     READ OR SPACE FORWARD
                        M:READ  DCB,(FWD)
IGNERR                  IGNORE ERRORS (ABORT OVERRIDE)
                        M:READ  DCB,(IGNERR)
IOQ12                   DATA TO BE PASSED TO SPECIAL DEVICE HANDLER
                        IN IOQ12 TABLE, OR ADDRESS OF SAME
                        M:READ  DCB,(IOQ12,X'1234')  OR  (IOQ12,*XXXX)
LWAIT                   LONG WAIT INDICATOR (PRIME ROLLOUT CANDIDATE)
                        M:READ  DCB,(LWAIT)
```

Figure A-1. Specific CAL-Parameter Groups (cont.)

```
GROUP 1:  (CONTINUED)

N               NUMBER OF UNITS TO SKIP OR ADDRESS CONTAINING SAME
                M:PREC  DCB,(N,25) OR (N,*XXXX)
NOCK            NO CHECK ON OPERATION (DELETE ON POST)
                M:READ  DCB,(NOCK)
PROMPT          PROMPT CHARACTER TO BE WRITTEN BEFORE A READ FROM
                A KEYBOARD-PRINTER DEVICE, OR ADDRESS OF SAME
                M:READ DCB,(PROMPT,'-')  OR  (PROMPT,*ADDR)
PTRI            PAPER TAPE READ IMMEDIATE
                M:READ  DCB,(PTRI)
PTRD            PAPER TAPE READ DIRECT
                M:READ  DCB,(PTRD)
REV,BACK        READ OR SPACE REVERSE
                M:READ  DCB,(REV)
SIZE,RECL       BYTE LENGTH OF TRANSFER, OR ADDRESS OF SAME
                M:READ  DCB,(RECL,N) OR (RECL,*XXXX)
SKIP            SPACE PAST EOF ON SKIP OPERATION
                M:PREC  DCB,(SKIP)
TIME            TIMEOUT COUNTER
                M:READ  DCB,(TIME,10)
TYPE1           INDICATES TYPE I I/O
                M:READ  DCB,(TYPE1)
WAIT            WAIT FOR COMPLETION OF OPERATION BEFORE RETURN
                M:READ  DCB,(WAIT)
WFPT,TYPE2      INDICATES TYPE II I/O, FORCES PRESENCE OF POST WORD
                M:READ  DCB,(TYPE2)

* * * * * * * * * * * * * * * * * GROUP 2 * * * * * * * * * * * * * * * * *

                NAME      CAL

                M:ABORT   CAL1,9  X'03'
                M:EXDA    CAL1,9  X'0A'
                M:EXIT    CAL1,9  X'01'
                M:TERM    CAL1,9  X'08'
                M:TRTN    CAL1,9  X'05'
                M:WAIT    CAL1,9  X'09'
                M:XXX     CAL1,9  X'03'

                THE ABOVE CALS HAVE NO PARAMETERS

* * * * * * * * * * * * * * * * * GROUP 3 * * * * * * * * * * * * * * * * *

              NAME      CAL       FPT CODE

              M:INT     CAL1,8    X'0E'
              M:PC      CAL1,1    X'2C'
              M:TIME    CAL1,8    X'10'

        THESE CALS EACH HAVE A SINGLE UNKEYED PARAMETER:
        THE TIME RETURN BLOCK ADDRESS FOR M:TIME,
        THE BREAK PROCESSOR ADDRESS FOR M:INT,
        AND THE NEW PROMPT CHARACTER FOR M:PC
                        M:TIME  XXXX
```

Figure A-1. Specific CAL-Parameter Groups (cont.)

```
* * * * * * * * * * * * * * * * * GROUP 4 * * * * * * * * * * * * * * * * *

                    NAME        CAL       FPT CODE

                    M:CALRTN    CAL1,7    X'60'
                    M:MASTER    CAL1,8    X'08'
                    M:SLAVE     CAL1,8    X'07'

                THE ABOVE CALS HAVE NO PARAMETERS

* * * * * * * * * * * * * * * * * GROUP 5 * * * * * * * * * * * * * * * * *

                    NAME        CAL       FPT CODE

                    M:GETTIME   CAL1,7    X'61'

 KEYWORD            MEANING AND SAMPLE

 STIME              RETURN SYSTEM CLOCK VALUE (SECONDS SINCE MIDNIGHT)
                    M:GETTIME  (TIME)
 UTIME              RETURN USER CLOCK VALUE (SECONDS SINCE SYSTEM BOOT)
                    M:GETTIME  (UTIME)

* * * * * * * * * * * * * * * * * GROUP 6 * * * * * * * * * * * * * * * * *

                    NAME        CAL       FPT CODE

                    M:PRINT     CAL1,2    X'01'
                    M:TYPE      CAL1,2    X'02'

 KEYWORD            MEANING AND SAMPLE

 MESS               ADDRESS OF THE MESSAGE  IN TEXTC FORMAT (MAY BE
                    INDIRECT)
                    M:PRINT (MESS,XXXX)   OR   (MESS,*XXXX)
 TIME               TIMEOUT COUNTER
                    M:TYPE (TIME,20)
 WAIT               WAIT FOR COMPLETION BEFORE RETURNING
                    M:PRINT (WAIT)

* * * * * * * * * * * * * * * * * GROUP 7 * * * * * * * * * * * * * * * * *

                    NAME        CAL       FPT CODE

                    M:CORRES    CAL1,1    X'2B'

 KEYWORD            MEANING AND SAMPLE

 DCB1 ]
      |             THE ADDRESSES OF THE DCBS TO COMPARE (MAY BE INDIRECT)
      |             M:CORRES (DCB1,XXXX)     OR (DCB1,*XXXX)
 DCB2 ]
```

Figure A-1. Specific CAL-Parameter Groups (cont.)

```
* * * * * * * * * * * * * * * * * GROUP 8 * * * * * * * * * * * * * * * * *

                    NAME     CAL      FPT CODE

                    M:RLS    CAL1,5   X'0B'
                    M:RUN    CAL1,5   X'0C'

KEYWORD             MEANING AND SAMPLE

FILEN               FILE NAME FROM WHICH TO LOAD.
                    M:RUN  (FILEN,'HORSE')
INTLB               INTERRUPT LABEL TO TRIGGER ON COMPLETION
                    M:RUN  (INTLB,'I3')
INTNR               INTERRUPT NUMBER TO TRIGGER ON COMPLETION
                    M:RUN  (INTNR,X'65')
ISP                 SUPPRESS STORAGE OF COMPLETION CODE
                    M:RUN  (ISP)
PRIO                PRIORITY AT WHICH TO QUEUE THE RUN
                    M:RUN  (PRIO,X'8000')
SIG                 ADDRESS TO STORE COMPLETION CODE
                    M:RUN  (SIG,XXXX)

* * * * * * * * * * * * * * * * * GROUP 9 * * * * * * * * * * * * * * * * *

                    NAME      CAL      FPT CODE

                    M:JTRAP   CAL1,7   X'5D'
                    M:TRAP    CAL1,8   X'14'

KEYWORD             MEANING AND SAMPLE

(NO KEYWORD)        TRAP RECEIVER ADDRESS (IF INCLUDED, MUST BE FIRST. IF
                    OMITTED, FIRST PARAMETER MUST NOT BE A SINGLE ITEM>
                    M:TRAP  XXXX
ABORT               THE TRAPS INDICATED BY THE FOLLOWING CODES
                    WILL CAUSE TASK ABORT
                    M:TRAP  (ABORT,AA,BB,CC,DD)
IGNORE              RESET THE PERMIT BIT IN THE PSD FOR THE
                    TRAPS INDICATED BY THE FOLLOWING CODES
                    M:TRAP  (IGNORE,AA,BB)
PERMIT              SET THE PERMIT BIT IN THE PSD FOR THE
                    TRAPS INDICATED BY THE FOLLOWING CODES
                    M:TRAP  (PERMIT,AA,BB)
TRAP                THE TRAPS INDICATED BY THE FOLLOWING CODES
                    WILL CAUSE TRANSFER TO THE TRAP RECEIVER
                    M:TRAP  XXXX,(TRAP,AA,BB,CC,DD)
  CAL               CODE FOR THE CAL ERROR SIMULATED TRAP
  DEC               CODE FOR THE DECIMAL ARITHMETIC ERROR TRAP
  FP                CODE FOR THE FLOATING ARITHMETIC ERROR TRAP
  FX                CODE FOR THE FIXED ARITHMETIC ERROR TRAP
  NAO               CODE FOR THE NON-ALLOWED OPERATION TRAP
  PS                CODE FOR THE STACK LIMIT TRAP
  UI                CODE FOR THE UNIMPLEMENTED INSTRUCTION TRAP
  WDG               CODE FOR THE WATCHDOG TIMER TRAP
                    M:TRAP  HANDLER,(ABORT,WDG,CAL,PS,FP,FX),:
                            (TRAP,NAO,UI),(PERMIT,FX),(IGNORE,DEC)
```

Figure A-1. Specific CAL-Parameter Groups (cont.)

```
* * * * * * * * * * * * * * * * * GROUP 10 * * * * * * * * * * * * * * * * *

                    NAME          CAL       FPT CODE

                    M:DISABLE     CAL1,5    X'01'
                    M:ENABLE      CAL1,5    X'02'
                    M:TRIGGER     CAL1,5    X'00'

KEYWORD             MEANING AND SAMPLE
(WITH ALTERNATES)

ERR,ERROR           ADDRESS FOR ERROR RETURN
                    M:TRIGGER  (ERR,XXXX)
INTLB               LABEL OF THE INTERRUPT TO AFFECT
                    M:TRIGGER  (INTLB,'I3')
INTNR               ADDRESS OF THE INTERRUPT TO AFFECT
                    M:TRIGGER  (INTNR,X'70')

* * * * * * * * * * * * * * * * * GROUP 11 * * * * * * * * * * * * * * * * *

                    NAME          CAL       FPT CODE

                    M:ACT         CAL1,7    X'52'
                    M:ALARM       CAL1,7    X'6A'
                    M:ALLOT       CAL1,7    X'5A'
                    M:DEACT       CAL1,7    X'53'
                    M:DEBUG       CAL1,7    X'65'
                    M:DELETE      CAL1,7    X'5B'
                    M:DEQ         CAL1,7    X'4D'
                    M:ENQ         CAL1,7    X'4C'
                    M:ERASE       CAL1,7    X'54'
                    M:EXTM        CAL1,7    X'49'
                    M:GETPAGE     CAL1,7    X'57'
                    M:INIT        CAL1,7    X'48'
                    M:KJOB        CAL1,7    X'64'
                    M:LOCK        CAL1,7    X'55'
                    M:MODIFY      CAL1,7    X'4F'
                    M:POLL        CAL1,7    X'45'
                    M:POST        CAL1,7    X'46'
                    M:RECALM      CAL1,7    X'69'
                    M:RELPAGE     CAL1,7    X'58'
                    M:SCHED       CAL1,7    X'68'
                    M:SETNAME     CAL1,7    X'51'
                    M:SIGNAL      CAL1,7    X'43'
                    M:SJOB        CAL1,7    X'63'
                    M:START       CAL1,7    X'4A'
                    M:STATUS      CAL1,7    X'4E'
                    M:STIMER      CAL1,7    X'44'
                    M:STOP        CAL1,7    X'4B'
                    M:TEST        CAL1,7    X'42'
                    M:TEXIT       CAL1,7    X'5F'
                    M:TRTY        CAL1,7    X'5E'
                    M:TRUNC       CAL1,7    X'5C'
                    M:UNLOCK      CAL1,7    X'56'
                    M:WAITALL     CAL1,7    X'40'
                    M:WAITANY     CAL1,7    X'41'
```

Figure A-1. Specific CAL-Parameter Groups (cont.)

```
GROUP 11:  (CONTINUED)

KEYWORD                 MEANING AND SAMPLE
(WITH ALTERNATES)

ABORT                   ABNORMAL EXTM INDICATOR
                        M:EXTM  (ABORT)
ACNTPTR                 ADDRESS OF 2-WORD BLOCK CANTAINING A
                        FILE ACCOUNT NAME.  MAY BE INDIRECT.
                        M:ALLOT  (ACNTPTR,XXXX)  OR  (ACNTPTR,*XXXX)
ALADR                   ALARM RECEIVER ADDRESS (MAY BE INDIRECT)
                        M:ALMREC  (ALADR,DOWNER)  OR  (ALADR,*K:CRSH)
ALMSG                   ADDRESS OF MESSAGE FOR ALARM CAL
                        (MAY BE INDIRECT)
                        M:ALARM  (ALMSG,MESS)  OR  (ALMSG,*ADDR)
AREA                    FILE AREA NAME
                        M:INIT  (AREA,'FP')
CLASS                   ASYNCHRONOUS EVENT CONTROL CLASS MASK
                        M:SIGNAL  (CLASS,X'FFFF')
DEBUG                   INITIATE TASK UNDER CONTROL OF THE CP-R DEBUGGER
                        M:INIT  (DEBUG)
DEL                     DELETE THE TASK FROM THE SCHEDULING LIST
                        M:SCHED (DEL)
EAADDR                  END ACTION TRANSFER ADDRESS
                        M:INIT  (EAADDR,XXXX)
EACSA                   END ACTION COMPLETION STATUS ADDRESS
                        M:INIT  (EACSA,XXXX)
EAINTLB                 END ACTION INTERRUPT LABEL
                        M:INIT  (EAINTLB,'I3')
EAINTNR                 END ACTION INTERRUPT NUMBER
                        M:INIT  (EAINTNR,X'65')
ERR,ERROR               ERROR RETURN ADDRESS (MAY BE INDIRECT)
                        M:INIT  (ERR,XXXX)  OR  (ERR,*XXXX)
EXCL                    ENQUEUE FOR EXCLUSIVE ACQUISITION
                        M:ENQ  (EXCL)
FILE                    FILE NAME, OR FILE AND AREA NAMES
                        M:ALLOT  (FILE,'BP','TEST')  OR
                        M:ALLOT  (FILE,'TEST'),(AREA,'BP')
FSI                     FILE SIZE IN RECORDS (OR ADDRESS OF SAME)
                        M:ALLOT  (FSI,20)  OR  (FSI,*XXX)
GO                      INIT WITH START
                        M:INIT  (GO)
GSI                     GRANULE SIZE IN WORDS (OR ADDRESS OF SAME)
                        M:ALLOT  (GSI,128)  OR  (GSI,*XXX)
IGNERR                  IGNORE ERRORS (ABORT OVERRIDE)
                        M:START  (IGNERR)
IMMED                   ENQUEUE FOR IMMEDIATE ACQUISITION
                        M:ENQ  (IMMED)
INTV                    INTERVAL TIME FOR SCHED SERVICE
                        M:SCHED (INTV,50)
INTVL                   STIMER VALUE IS AN INTERVAL
                        M:STIMER  (INTVL)
JOB                     JOB NAME
                        M:SIGNAL  (JOB,'JOB2')
JOBL                    JOB LEVEL ENQUEUE
                        M:ENQ  (JOBL)
LWAIT                   LONG WAIT INDICATOR
                        M:GETPAGE  (LWAIT)
NOCK                    NO CHECK REQUIRED ON CAL WITHOUT WAIT (DELETE ON POST)
ORG,FOR                 FILE ORGANIZATION (OR ADDRESS OF SAME): U (UNBLOCKED),
                        B (BLOCKED), OR C (COMPRESSED)
                        M:ALLOT  (FOR,B)  OR  (FOR,*XXX)
PRI                     INIT A PRIMARY TASK
                        M:INIT  (PRI)
PRIO                    PRIORITY AT WHICH DISPATCH IS TO BE DONE
                        (OR ADDRESS OF SAME)
                        M:INIT  (PRIO,X'FFFF')  OR  (PRIO,*XXXX)
```

Figure A-1. Specific CAL-Parameter Groups (cont.)

```
GROUP 11:  (CONTINUED)

RDATA           ASYNCHRONOUS EVENT CONTROL RECEIVED DATA SIZE
                AND ADDRESS
                M:POLL  (RDATA,16,XXXX)
REBOOT          REINITIALIZE SYSTEM AFTER ALARM PROCESSING
                M:RECALM  REBOOT
RESFGD          ALLOT A RESIDENT FGD PROGRAM FILE
                M:ALLOT  (RESFGD)
RNAME           RESOURCE NAME
                M:ENQ  (RNAME,'THING')
RSI             RECORD SIZE IN WORDS (OR ADDRESS OF SAME)
                M:ALLOT  (RSI,20)   OR   (RSI,*XXX)
SDATA           ASYNCHRONOUS EVENT CONTROL SIGNAL DATA ADDRESS
                M:SIGNAL  (SDATA,XXXX)
SEC             INIT A SECONDARY TASK
                M:INIT  (SEC)
SEG             SEGMENT NUMBER FOR MEMORY MANAGEMENT CALS
                M:ACT (SEG,1)
SHARE           ENQUEUE FOR SHARED ACQUISITION
                M:ENQ  (SHARE)
STOP            INIT WITHOUT START INDICATOR
                M:INIT  (STOP)
STRTPTR         ADDRESS OF 2-WORD BLOCK CONTAINING A
                SCHED SERVICE START TIME. MAY BE INDIRECT.
                M:SCHED  (STRTPTR,XXXX)  OR  (STRTPTR,*XXXX)
SYSL            SYSTEM LEVEL ENQUEUE
                M:ENQ  (SYSL)
TASK            TASK NAME
                M:INIT (TASK,'FSEG')
THRESH          STIMER VALUE IS A THRESHHOLD TIME
                M:STIMER  (THRESH)
TIME            TIMEOUT COUNTER
                M:POLL  (TIME,50)
TS              INITIATE TASK FOR TIME-SLICED EXECUTION
                M:INIT  (TS)
TYC             INCLUDE A COMPLETION POSTING WORD IN THE FPT
                M:INIT  (TYC)
VPNL            LOW VIRTUAL PAGE NUMBER (MEMORY MGT)
VPNH            HIGH VIRTUAL PAGE NUMBER (MEMORY MGT)
                M:GETPAGE  (VPNL,32),(VPNH,34)
WAIT            WAIT FOR COMPLETION BEFORE RETURN
                M:INIT  (WAIT)

* * * * * * * * * * * * * * * * * GROUP 12 * * * * * * * * * * * * * * * * *

                NAME        CAL       FPT CODE

                M:ARM       CAL1,5    X'04'
                M:CON       CAL1,5    X'04'
                M:DISARM    CAL1,5    X'03'
                M:DISCON    CAL1,5    X'03'

KEYWORD         MEANING AND SAMPLE
(WITH ALTERNATES)

AI              ADDRESS INCREMENTATION OPTION ON A CAL CONNECTION
                M:CON  (AI)
AM              ENTER CONNECTED ROUTINE WITH FIXED ARITHMETIC MASK SET
CI              ENTER CONNECTED ROUTINE WITH CLOCK GROUP INHIBITED
DE              INTERRUPT TO BE DISABLED
DI              DIRECT CONNECTION
DM              ENTER CONNECTED ROUTINE WITH DECIMAL MASK SET
EI              ENTER CONNECTED ROUTINE WITH EXTERNAL GROUP INHIBITED
II              ENTER CONNECTED ROUTINE WITH I/O GROUP INHIBITED
```

Figure A-1. Specific CAL-Parameter Groups (cont.)

```
GROUP 12:  (CONTINUED)

MS              ENTER CONNECTED ROUTINE IN SLAVE MODE
                M:CON  DE,DI,CI,II,EI,MS,DM,AM
ERR,ERROR       ERROR RETURN ADDRESS (MAY BE INDIRECT)
                M:CON  (ERR,XXXX)  OR  (ERR,*XXXX)
IGNERR          IGNORE ERRORS  (ABORT OVERRIDE)
INTLB           INTERRUPT LABEL OF THE INTERRUPT TO AFFECT
                M:CON  (INTLB,'I3')
INTNR           INTERRUPT NUMBER OF THE INTERRUPT TO AFFECT
                M:CON  (INTNR,X'65')
JOB             JOB NAME FOR CONNECTION
                M:CON  (JOB,'THATJOB')
MTW             MTW INSTRUCTION OR CLOCK VALUE
                M:CON  (MTW,X'33F00000'+XXXX)
NR              NUMBER OF REGISTERS TO SAVE
                M:CON  (NR,8)
RBLK            REGISTER BLOCK NUMBER
                M:CON  (RBLK,2)
START           START ADDRESS FOR CENTRAL CONNECTION
                M:CON  (START,XXXX)
TASK            TASK NAME FOR CONNECTION
                M:CON (TASK,'THISTASK')
TCB             TCB ADDRESS
                M:CON  (TCB,XXXX)
TYC             COMPLETION WORD TO BE RESERVED IN FPT
                M:CON  (TYC)
XPSD            XPSD ADDRESS FOR DIRECT CONNECTION
                M:CON  (XPSD,XXXX)

* * * * * * * * * * * * * * * * GROUP 13 * * * * * * * * * * * * * * * *

                    NAME        CAL      FPT CODE

                    M:SEGLOAD   CAL1,8   X'01'

KEYWORD             MEANING AND SAMPLE
(WITH ALTERNATES)

EAADDR          END ACTION TRANSFER ADDRESS
                M:SEGLOAD  (EAADDR,XXXX)
EACSA           END ACTION COMPLETION STATUS ADDRESS
                M:SEGLOAD  (EACSA,XXXX)
EAINTLB         END ACTION INTERRUPT LABEL
                M:SEGLOAD  (EAINTLB,'I3')
EAINTNR         END ACTION INTERRUPT NUMBER
                M:SEGLOAD  (EAINTNR,X'65')
ERR,ERROR       ERROR RETURN ADDRESS
                M:SEGLOAD  (ERR,XXXX)
SEG             SEGMENT NUMBER TO LOAD
                M:SEGLOAD  (SEG,12)
T               TRANSFER TO SEGMENT WHEN IT IS LOADED
                M:SEGLOAD  T
```

Figure A-1. Specific CAL-Parameter Groups (cont.)

```
* * * * * * * * * * * * * * * * GROUP 15 * * * * * * * * * * * * * * * * *

                    NAME     CAL      FPT CODE

                    M:JOB    CAL1,7   X'67'

KEYWORD             MEANING AND SAMPLE
(WITH ALTERNATES)

DEF                 INDICATES A SYMBIONT FILE MUST BE DEFINED
                    FOR THE INDICATED JOB
                    M:JOB  (DEF)
DEL                 INDICATES THAT ALL SYMBIONT FILES ASSOCIATED WITH
                    THE GIVEN JOB ARE TO BE DELETED
                    M:JOB  (DEL)
ERR,ERROR           ERROR RETURN ADDRESS (MAY BE INDIRECT)
                    M:JOB  (ERR,XXX)  OR  (ERR,*XXX)
FILPTR              ADDRESS OF DATA AREA CONTAINING NAME OF FILE TO SUBMIT
                    FOR BACKGROUND EXECUTION (MAY BE INDIRECT)
                    M:JOB  (FILPTR,XXX)  OR  (FILPTR,*XXX)
IDNR                THE JOB IDENTIFICATION NUMBER OF THE SYMBIONT
                    JOB CONCERNED, OR A POINTER TO SAME.
                    M:JOB  (IDNR,30)  OR  (IDNR,*XXX)
IGNERR              INDICATES THAT ERRORS ARE TO BE IGNORED
                    (ABORT OVERRIDE)
                    M:JOB  (IGNERR)
IN                  INDICATES THAT AN INPUT SYMBIONT FILE MUST BE DEFINED
                    M:JOB  (IN)
OUT                 INDICATES THAT AN OUTPUT SYMBIONT FILE MUST BE DEFINED
                    M:JOB  (OUT)
PRIO                PRIORITY OF THE SYMBIONT JOB CONCERNED, OR
                    POINTER TO SAME
                    M:JOB  (PRIO,5)  OR  (PRIO,*XXX)
QNR                 RETURNED NUMBER OF SYMBIONT JOBS QUEUED
                    AHEAD OF THE ONE CONCERNED, OR POINTER TO SAME
                    M:JOB  (QNR,0)  OR  (QNR,*XXX)
SFILPTR             ADDRESS OF DATA AREA IN WHICH A SYMBIONT
                    FILE NAME WILL BE RETURNED (MAY BE INDIRECT)
                    M:JOB (SFILPTR,XXX)  OR  (SFILPTR,*XXX)
STAT                THE RETURNED STATUS OF THE SYMBIONT JOB
                    CONCERNED, OR A POINTER TO SAME.
                    M:JOB  (STAT,0)  OR  (STAT,*XXX)
STATCK              INDICATES THAT THE CAL IS ONLY TO CHECK THE STATUS
                    OF THE SYMBIONT JOB CONCERNED
                    M:JOB  (STATCK)
TYC                 INDICATES THAT A WORD IS TO BE RESERVED IN THE
                    FPT FOR RETURNING THE CAL TYPE COMPLETION CODE
                    M:JOB  (TYC)
```

Figure A-1. Specific CAL-Parameter Groups (cont.)

```
* * * * * * * * * * * * * * * * GROUP 16 * * * * * * * * * * * * * * * *

                    NAME        CAL      FPT CODE

                    M:PREFMOD   CAL1,7   X'50'


KEYWORD             MEANING AND SAMPLE
(WITH ALTERNATES)

ERR,ERROR           ERROR RETURN ADDRESS (MAY BE INDIRECT)
                    M:PREFMOD  (ERR,XXXX)  OR  (ERR,*XXXX)
IGNERR              IGNORE ERRORS (ABORT OVERRIDE)
                    M:PREFMOD  (IGNERR)
PBA                 PARTITION BASE ADDRESS
                    M:PREFMOD  (PBA,XXXX)
REL                 RELEASE PARTITION FROM ITS CURRENT CLASS
                    M:PREFMOD  (REL)
RES                 RESTORE PARTITION TO ITS FORMER CLASS
                    M:PREFMOD  (RES)
TYC                 COMPLETION STATUS WORD TO BE RESERVED IN FPT
                    M:PREFMOD  (TYC)

* * * * * * * * * * * * * * * * GROUP 17 * * * * * * * * * * * * * * * *

                    NAME        CAL      FPT CODE

                    M:ASSIGN    CAL1,1   X'08'
                    M:GETASN    CAL1,1   X'09'
                    M:MEDIA     CAL1,7   X'59'
                    M:STDLB     CAL1,7   X'62'


KEYWORD             MEANING AND SAMPLE
(WITH ALTERNATES)

(NO KEYWORD)        FOR M:STDLB, THE LABEL TO BE AFFECTED.
                    FOR M:ASSIGN OR M:GETASN, THE DCB CONCERNED.
                    FOR M:MEDIA, A DCB FOR THE FILE TO LIST
                    (MUST BE FIRST PARAMETER. REQUIRED EXCEPT FOR MEDIA)
                    M:STDLB  'LO'
                    M:ASSIGN  DCBADDR
ACNTPTR             ADDRESS OF 2-WORD BLOCK CANTAINING A
                    FILE ACCOUNT NAME.  MAY BE INDIRECT.
                    M:ALLOT  (ACNTPTR,XXXX)  OR  (ACNTPTR,*XXXX)
BOT                 FIRST SECTOR OF FILE, OR ADDRESS OF SAME
                    M:GETASN  DCB,(BOT)    OR    (BOT,*XXXX)
DCTX                DCT INDEX OF DEVICE CONCERNED, OR POINTER TO SAME
                    M:GETASN  (DCTX,0)  OR  (DCTX,*XXX)
DEL                 DELETE SOURCE FILE AFTER MEDIA LISTING
                    M:MEDIA  (FILPTR,XXX),DEL
DEVCPTR             ADDRESS OF 3-WORD BLOCK FOR DISK TRACKS PER
                    CYLINDER, SECTORS PER TRACK, AND WORDS PER
                    SECTOR.  MAY BE INDIRECT.
                    M:GETASN  (DEVCPTR,XXXX)   OR   (DEVCPTR,*XXXX)
DEVPTR              ADDRESS OF A 2-WORD BLOCK TO CONTAIN AN EBCDIC
                    DEVICE NAME (MAY BE INDIRECT)
                    M:STDLB  'LO',(DEVPTR,XXXX)    OR    (DEVPTR,*XXXX)
DS                  DOUBLE-SPACED MEDIA LISTING
                    M:MEDIA (FILPTR,XXX),DS
EAADDR              END ACTION TRANSFER ADDRESS
                    M:STDLB  'LO',EAADDR,XXXX)
EACSA               END ACTION COMPLETION STATUS ADDRESS
                    M:STDLB  'LO',(EACSA,XXXX)
EAINTLB             END ACTION INTERRUPT LABEL
                    M:STDLB  'LO',EAINTLB,'I3')
EAINTNR             END ACTION INTERRUPT NUMBER
                    M:STDLB  'LO',EAINTNR,X'65')
```

Figure A-1. Specific CAL-Parameter Groups (cont.)

```
GROUP 17: (CONTINUE)

ENQ             ENQUEUE ON THE REFERENCED DEVICE IN STDLB
                M:STDLB  'LO',(DEVPTR,XXXX),ENC
EOT             LAST SECTOR OF FILE, OR ADDRESS OF SAME
                M:GETASN  DCB,(EOT)    OR    (EOT,*XXXX)
ERR,ERROR       ERROR RETURN ADDRESS (MAY BE INDIRECT)
                M:STDLB  'LO',(ERR,XXXX)  OR  (ERR,*XXXX)
FILPTR          ADDRESS OF A 3-WORD BLOCK TO CONTAIN AN AREA NAME,
                RIGHT-JUSTIFIED, IN THE FIRST WORD, AND A FILE NAME,
                EXTENDED TO 8 BYTES WITH TRAILING BLANKS, IN THE
                SECOND AND THIRD WORDS (MAY BE INDIRECT)
                M:STDLB  'LO',(FILPTR,XXXX)    OR    (FILPTR,*XXXX)
FOR,ORG         FILE ORGANIZATION OF FILE CONCERNED, OR POINTER
                TO SAME
                M:GETASN  (FOR,0)  OR  (FOR,*XXX)
IGNERR          IGNORE ERRORS  (ABORT OVERRIDE)
                M:STDLB  'LO',IGNERR
MODLPTR         ADDRESS OF A WORD TO CONTAIN AN EBCDIC MODEL NUMBER
                (MAY BE INDIRECT)
                M:GETASN DCB,(MODLPTR,XXXX)   OR   (MODLPTR,*XXXX)
NVFC            NO VERTICAL FORMAT CONTROL INTERPRETATION
                FOR MEDIA LISTING
                M:MEDIA DCB,NVFC
OPLB            OPLABEL TO BE REFERENCED, OR ADDRESS OF SAME
                M:STDLB  'LO',(OPLB,'LL')    OR    (OPLB,*XXXX)
PRIO            PRIORITY AT WHICH OPERATION IS TO BE DONE
                (OR ADDRESS OF SAME)
                M:MEDIA (PRIO,X'FFFF')  OR  (PRIO,*XXXX)
REL             RELEASE THE CURRENT ASSIGNMENT OF AN OPLABEL
                M:STDLB  'LO',REL
RSI,GSI         RECORD OR GRANULE SIZE (RESPECTIVELY FOR BLOCKED OR
                UNBLOCKED FILES) OF THE FILE CONCERNED,
                OR POINTER TO SAME
                M:GETASN  (GSI,0)  OR  (GSI,*XXX)
TIME            TIMEOUT INTERVAL
                M:STDLB  'LO',(TIME,50)
TYC             RESERVE A COMPLETION STATUS WORD IN THE FPT
                M:STDLB  'LO',TYC
WAIT            WAIT FOR COMPLETION BEFORE RETURN
                M:STDLB  'LO',WAIT
WP              WRITE RESTRICTION CODE FOR DISK AREA, OR
                ADDRESS OF SAME.
                M:GETASN  (WP,0)   OR   (WP,*XXXX)
ZERO            EXPLICIT ZERO ASSIGNMENT OR STDLB
                M:STDLB  'LO',ZERO


* * * * * * * * * * * * * * * * GROUP 19 * * * * * * * * * * * * * * * *

                    NAME        CAL       FPT CODE

                    M:DFMODE    CAL1,1    X'22'
                    M:DRC       CAL1,1    X'0B'
                    M:VFC       CAL1,1    X'05'


KEYWORD         MEANING AND SAMPLE
(WITH ALTERNATES)

(NO KEYWORD)    DCB CONCERNED (REQUIRED FIRST PARAMETER)
                M:DFMODE  DCBADDR
ASC             TAPE DRIVE PERFORMS ASCII TRANSLATION
                M:DFMODE  DCBADDR,(ASC)
BCD             BCD MODE DATA
                M:DFMODE  DCBADDR,BCD
BIN             BINARY MODE DATA
                M:DFMODE  DCBADDR,BIN
```

Figure A-1. Specific CAL-Parameter Groups (cont.)

```
GROUP 19:  (CONTINUED)

DRC             DIRECT RECORD CONTROL ON KEYBOARD-PRINTER
                M:DRC  DCBADDR,DRC
D800            TAPE DENSITY IS 800 CPI
                M:DFMODE  DCBADDR,(D800)
D1600           TAPE DENSITY IS 1600 CPI
                M:DFMODE  DCBADDR,(D1600)
EBC             TAPE DRIVE PROVIDES EBCDIC TRANSLATION
                M:DFMODE  DCBADDR,(EBC)
GSI             FILE GRANULE SIZE IN BYTES (OR ADDRESS OF SAME)
                M:DFMODE  DCBADDR,(GSI,30)   OR   (GSI,*XXX)
NODRC           NORMAL RECORD CONTROL ON KEYBOARD-PRINTER
                M:DRC  DCBADDR,NODRC
NOVFC           DO NOT INTERPRET VERTICAL FORMAT CONTROL
                M:VFC  DCBADDR,NOVFC
NRT             NUMBER OF RETRIES (OR ADDRESS OF SAME)
                M:DFMODE  DCBADDR,(NRT,5)   OR   (NRT,*XXX)
ORG,FOR         FILE ORGANIZATION (OR ADDRESS OF SAME): U (UNBLOCKED),
                B (BLOCKED), OR C (COMPRESSED)
                M:DFMODE  DCBADDR,(ORG,C)   OR   (ORG,*XXX)
PACK            PACKED DATA FORMAT
                M:DFMODE  DCBADDR,BIN,PACK
RSI             RECORD SIZE IN WORDS (OR ADDRESS OF SAME)
                M:DFMODE  DCBADDR,(RSI,30)   OR   (RSI,*XXX)
UNPK            UNPACKED DATA FORMAT
                M:DFMODE  DCBADDR,BIN,UNPK
VFC             INTERPRET VERTICAL FORMAT CONTROL
                M:VFC  DCBADDR,VFC

* * * * * * * * * * * * * * * * * * GROUP 20 * * * * * * * * * * * * * * * * * *

                   NAME         CAL      FPT CODE

                   M:BIOSTP     CAL1,5   X'0E'
                   M:BIOSTRT    CAL1,5   X'0F'
                   M:IOACT      CAL1,5   X'17'
                   M:IODEACT    CAL1,5   X'16'
                   M:SIOSTP     CAL1,5   X'10'
                   M:SIOSTRT    CAL1,5   X'11'


KEYWORD         MEANING AND SAMPLE

DCB             ADDRESS OF DCB ASSIGNED TO THE DEVICE CONCERNED
                M:SIOSTP  (DCB,XXXX)
DEV             AFFECT ONLY THE DEVICE, NOT ITS CONTROLLER
                M:SIOSTP  DEV
DEVX            DCT INDEX OF DEVICE CONCERNED
                M:SIOSTP  (DEVX,5)
HIO             ISSUE AN HIO TO THE DEVICE CONCERNED
                M:SIOSTP  HIO
IOP             AFFECT THE IOP, NOT JUST THE DEVICE OR CONTROLLER
                M:SIOSTP  IOP
OPLBX           INDEX OF OPLABEL ASSIGNED TO DEVICE CONCERNED
                M:SIOSTP  (OPLBX,4)
PRADDR          I/O PREEMPTION TRANSFER ADDRESS
                M:IODEACT  'LO',(PRADDR,XXXX)
PRCSA           I/O PREEMPTION COMPLETION STATUS ADDRESS
                M:IODEACT  'LO',(PRCSA,XXXX)
PRINTLB         I/O PREEMPTION INTERRUPT LABEL
                M:IODEACT  'LO',(PRINTLB,'I3')
PRINTNR         I/O PREEMPTION INTERRUPT NUMBER
                M:IODEACT  'LO',(PRINTNR,X'65')
```

Figure A-1. Specific CAL-Parameter Groups (cont.)

```
* * * * * * * * * * * * * * * * GROUP 21 * * * * * * * * * * * * * * * *

                        NAME    CAL       FPT CODE

                        M:HIO   CAL1,5    X'15'
                        M:SIO   CAL1,5    X'12'
                        M:TDV   CAL1,5    X'14'
                        M:TIO   CAL1,5    X'13'


KEYWORD             MEANING AND SAMPLE

DCB                 ADDRESS OF DCB ASSIGNED TO THE DEVICE CONCERNED
                    M:SIO  (DCB,XXXX)
DEVX                DCT INDEX OF DEVICE CONCERNED
                    M:SIO  (DEVX,5)
DIAG                OPERATION WILL BE VALID ONLY IF DEVICE
                    IS IN DIAGNOSTIC MODE
                    M:SIO  DIAG
EAADDR              END ACTION TRANSFER ADDRESS
                    M:SIO  (EAADDR,XXXX)
EACSA               END ACTION COMPLETION STATUS ADDRESS
                    M:SIO  (EACSA,XXXX)
EAINTLB             END ACTION INTERRUPT LABEL
                    M:SIO  (EAINTLB,'I3')
EAINTNR             END ACTION INTERRUPT NUMBER
                    M:SIO  (EAINTNR,X'65')
OPLBX               INDEX OF OPLABEL ASSIGNED TO DEVICE CONCERNED
                    M:SIO  (OPLBX,4)
SIO                 IOCD ADDRESS
                    M:SIO  (SIO,XXXX)
TIME                TIMEOUT INTERVAL
                    M:SIO  (TIME,50)


* * * * * * * * * * * * * * * * GROUP 22 * * * * * * * * * * * * * * * *

            M:DCB       GENERATES A DCB AT THE POSITION OF THE
                        PROCEDURE REFERENCE


KEYWORD             MEANING AND SAMPLE
(WITH ALTERNATES)

ABA,ABN             ABNORMAL ADDRESS
                    M:DCB  (ABN,XXXX)
ACNT                FILE ASSIGNMENT; FILE ACCOUNT NAME IS PARAMETER
                    M:DCB  (ACNT,'ACC 123')
AREA                FILE ASSIGNMENT; AREA NAME IS PARAMETER
                    M:DCB  (AREA,'FP')
AREAX               FILE ASSIGNMENT;  AREA INDEX IS PARAMETER
                    M:DCB  (AREAX,L),(FILEN,'NONZERO') FOR FILE;
                    M:DCB  (AREAX,1),(FILEN,0)  FOR WHOLE AREA
ASC                 TAPE DRIVE PERFORMS ASCII TRANSLATION
                    M:DCB  (ASC)
BIN                 BINARY MODE
                    M:DCB  (BIN)
BTD                 BYTE DISPLACEMENT OF FIRST BYTE IN FIRST WORD OF BUF
                    M:DCB  (BTD,3)
BUF                 BUFFER ADDRESS
                    M:DCB  (BUF,XXXX)
DEV                 DEVICE ASSIGNMENT; DEVICE NAME IS PARAMETER
                    M:DCB  (DEV,'LPA02')
DEVX                DEVICE ASSIGNMENT;  DCT INDEX IS PARAMETER
                    M:DCB  (DEVX,5)
DFAREA              FILE ASSIGNMENT WITH DEFAULT AREA INDICATED.
                    HAS NO PARAMETER.
                    M:DCB  DFAREA,(FILE,'FILENAME')
```

Figure A-1. Specific CAL-Parameter Groups (cont.)

```
DRC             KEYBOARD-PRINTER I/O WITH DIRECT RECORD CONTROL
                M:DCB  (DRC)
D800            TAPE DENSITY IS 800 CPI
                M:DCB  (D800)
D1600           TAPE DENSITY IS 1600 CPI
                M:DCB  (D1600)
EBC             TAPE DRIVE PROVIDES EBCDIC TRANSLATION
                M:DCB  (EBC)
ERA,ERR,ERROR   ERROR ADDRESS
                M:DCB  (ERR,XXXX)
FILE            FILE ASSIGNMENT;  FILE AND OPTIONAL AREA
                NAMES ARE PARAMETERS
                M:DCB  (FILE,'FP','FILENAME')
                       OR
                       (FILE,'FILENAME')
FILEN           FILE ASSIGNMENT;  FILE NAME IS PARAMETER
                M:DCB  (FILEN,'GIRAFFE')
LIST<L          LISTING DEVICE
                M:DCB  (L)
NRT,TRIES       NUMBER OF RETRIES
                M:DCB  (NRT,3)
OPLB            OP LABEL ASSIGNMENT; OP LABEL NAME IS PARAMETER
                M:DCB  (OPLB,'LO')
OPLBX           OP LABEL ASSIGNMENT; OP LABEL INDEX IS PARAMETER
                M:DCB  (OPLBX,1)
PACK            7-TRACK BINARY TAPE< PACKED FORMAT
                M:DCB  (PACK)
RSZ,RECL        RECORD LENGTH IN BYTES
                M:DCB  (RECL,80)
TTL             LENGTH OF DCB IN WORDS
                M:DCB  (TTL,7)
VFC             INTERPRET VERTICAL FORMAT CONTROL CHARACTERS
                M:DCB  (VFC)
ZERO            EXPLICIT ZERO ASSIGNMENT
                M:DCB  (ZERO)
```

Figure A-1. Specific CAL-Parameter Groups (cont.)

APPENDIX B. XEROX STANDARD COMPRESSED LANGUAGE

The Xerox Standard Compressed Language is used to represent source EBCDIC information in a highly compressed form.

Several Xerox processors will accept 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. This form of compressed language should not be output to "compressed" files since the I/O compression may cause loss of data.

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 eight 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 the record, including the checking bytes (≤108).

The rest of the record consists of a string of six-bit and eight-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:

Six-Bit Decimal Item	Function
0	Ignore.
1	Not currently assigned.
2	End of line.
3	End of file.
4	Use eight-bit character which follows.
5	Use n + 1 blanks, next six-bit item is n.
6	Use n + 65 blanks, next six-bit item is n.
7	Blank.
8	0
9	1
10	2
11	3
12	4
13	5
14	6
15	7
16	8
17	9
18	A
19	B
20	C
21	D
22	E
23	F
24	G
25	H
26	I
27	J
28	K
29	L
30	M
31	N
32	O
33	P
34	Q
35	R
36	S
37	T
38	U
39	V
40	W
41	X
42	Y
43	Z
44	.
45	<
46	(
47	+
48	|
49	&
50	$
51	*
52	)
53	;
54	¬
55	-
56	/
57	,
58	%
59	_
60	>
61	:
62	'
63	=

APPENDIX C. REAL-TIME PERFORMANCE DATA

The following timing data was determined for Sigma 9 hardware. Xerox 550 hardware is somewhat slower.

RESPONSE TO INTERRUPTS BY CENTRALLY CONNECTED TASKS

The time used by the system to save the interrupted context and establish the interrupting task context is approximately 45 μsec assuming that 16 registers are being saved. This time includes the XPSD in the interrupt location context and represents the total time between the interrupt becoming active and the start of execution of the first instruction in the task (assuming no interruption by a higher priority task).

I/O INTERRUPT

Following successful completion of an I/O device access, the I/O interrupt will remain active for approximately 150 μsec, assuming that no higher priority interrupts have become active during this period and that cleanup is deferred to an interrupt level other than I/O. Upon clearing the I/O interrupt, the system proceeds at the priority level of the interrupted task or at the cleanup level if its priority is higher than that of the interrupted task.

If I/O cleanup deferral was not requested at SYSGEN, the I/O request will be cleaned up at the I/O interrupt level.

INTERRUPT INHIBITS

At times CP-R must inhibit the interrupt system. The inhibit time is typically less than 100 μsec.

SECONDARY TASK DISPATCH TIME

The amount of time required for the system to dispatch a secondary task is approximately given by

$$\text{time (in } \mu\text{sec)} = 15T + 198 + \sum_{n=1}^{k} (44 + .98\ P_n)$$

where

- T is the number of tasks in the dispatcher queue that are higher in priority than the task being dispatched but which are not currently dispatchable.
- k is the number of segments in the task being dispatched.
- P_n is the number of pages in the nth segment.

CONSOLE INTERRUPT

The Console Interrupt remains active for less than 30 μsec. During this time, a flag in the Control Task is set to indicate the occurrence of the interrupt and the Control Task interrupt is triggered.

OVERLAY LOADING

Overlay loading is accomplished with a negligible percentage of total time devoted to non-I/O system activity. For example, on the Model 7202/7204 RAD, a 1400-word overlay requires approximately 50 msec, assuming average latency (17 mils) and I/O transfer time of 34 msec. To this must be added the time waited to gain access to the RAD (time of request, if any in progress, plus the time of any higher priority requests).

APPENDIX D. COOPERATIVES AND SYMBIONTS

In CP-R the routines to perform card reader and/or line printer operations operate concurrently with the jobs being run. The peripheral system is composed of a "cooperative" and a "symbiont" or symbionts. The cooperative is a monitor routine called as a result of a user's I/O request, whereas a symbiont is a monitor routine that is initiated either by the action of the cooperative or by operator command from the system console. The cooperative is used to transfer information between the user's program and secondary (disk) storage, and symbionts are used to transfer information between sesondary storage and peripheral devices (see Figure D-1).

The symbiont-cooperative system provides for complete buffering between I/O devices and the user's program. Therefore, a user's program never has to wait for an I/O device to complete an action. Also, the current job may be running while the output of the previous job and the job file for the following job are being handled by symbiont operation.

COOPERATIVE

A single cooperative is provided for handling both user input and output files. It is reenterable and can handle any number of device-type files (printer, card reader) per job.

SYMBIONTS

Symbionts execute as a mapped task in the CP-R job and control the action of a symbiont dedicated I/O device having a lower transfer rate than secondary storage. Core storage as well as secondary storage will be used by the symbiont to produce a continuous flow of information to or from these devices. Symbionts will transfer information from a peripheral device to the disk and from the disk to a peripheral device.

Unlike the user's program, which is directed primarily by control commands, symbionts – once initiated – receive all their control from the operator's console. An input symbiont device can be initiated only by the console operator, while an output symbiont device can be initiated either by the operator or the cooperative.

For each device, the symbionts perform one operation at a time with no wait. When no more operations can be initiated, the symbiont task goes inactive. When any of the initiated services is completed, the symbiont task TESTS (CHECK) the completed service and resumes execution following the call that was completed.

Since symbionts are reenterable, a single symbiont may drive several types of devices. For example, the same symbiont may be used to drive many printers and card readers. All the peripheral-dependent information is contained in monitor tables.

A single symbiont is provided in the monitor which handles all standard line printers, card readers, and magnetic tapes.

SYMBIONT-COOPERATIVE HOUSEKEEPING

Two monitor subroutines are provided for automatic maintenance of core storage. One is used to release a blocking buffer after use by the symbionts, and the other is used to obtain a blocking buffer. If a blocking buffer is requested by a symbiont and none is available, the symbiont waits for five seconds and requests again. When one becomes available, normal symbiont activities continue. As each buffer is emptied, either by reading from or storing into secondary storage, it is released. This procedure allows for efficient utilization of blocking buffers.

The symbiont routines consist of a permanently resident portion in monitor space and a nonresident portion which executes in the monitor overlay area. After starting an I/O or selected operation on every applicable symbiont device, the symbiont relinquishes control to the monitor.

Two disk areas are used to separate input from output symbiont data; the Input Symbiont (IS) area and the Output Symbiont (OS) area. The sizes are installation dependent and are set up at SYSGEN. Each area contains compressed format files which are fixed in size, the size being a SYSGEN parameter.

When the input symbiont reads a IJOB card, a file is created in the IS area. The file name is based on an internal sequential number, job ID, which is used for the symbiont key-ins. Input from the device to the file continues until a IJOB or IFIN card is read at which time a new file is created or the symbiont is suspended. If the file size is not large enough to contain all the records for a job, a continuation file is allotted which is readily identifiable as a continuation of the previous file.

A file is created in the OS area by the output cooperative when a user writes to a symbiont device. The file name is based on the executing job's number. Output to the file continues until the file is out of space or a new job begins to execute. When a file has run out of space, a continuation file (still based on the executing job's number) is defined. If a new job has begun to execute, a file with a new identifying job number is defined.

A file in the IS area is deleted by the input cooperative when a read results in an end-of-file indication from file management. Files in the OS area are deleted by the output symbiont. The files will normally be deleted all at once at the end of the job. This allows backspacing of printer output. However, if the DO key-in is in effect, the files will be deleted as they are output.

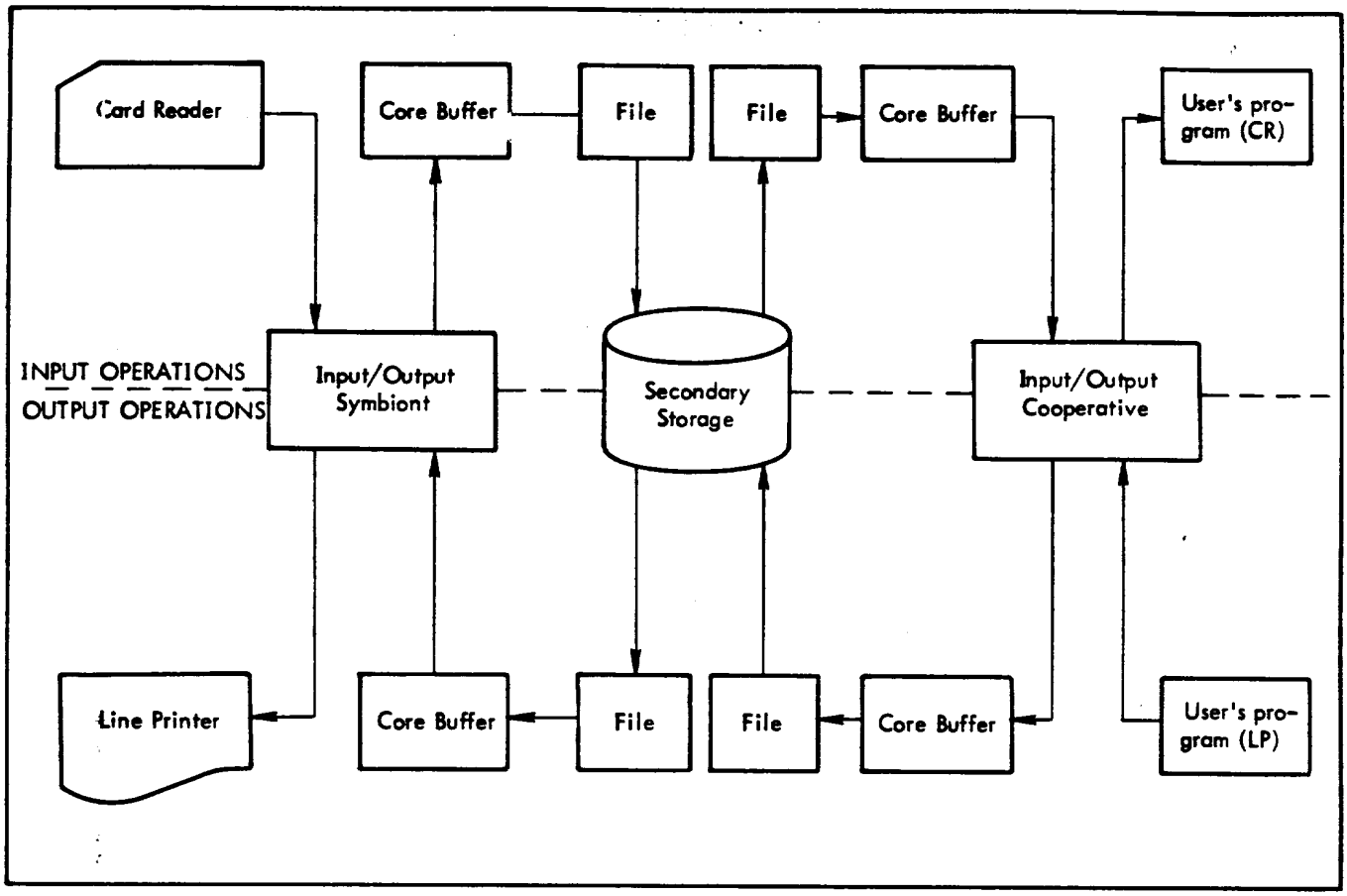


Figure D-1. Information Flow Through Cooperative and Symbionts

APPENDIX E. JCP LOADER

In addition to loading the Overlay Loader and RADEDIT into their respective files in the SP area at SYSGEN time (see ILOAD command in Chapter 2), the JCP Loader can be used to load some user programs. Within the restrictions given below, both nonsegmented and segmented programs can be loaded onto the disk for subsequent execution.

LOADING NONSEGMENTED PROGRAMS

1. The last object module in the program being loaded must be terminated by an IEOD command. Any number of object modules can be loaded.

2. The object module cannot contain any of the following load items: declare dummy section, declare secondary external reference name, forward reference definition, or add or subtract absolute section. The field (in a define field load item) cannot cross a word boundary.

3. FORTRAN compiled programs or programs assembled with basic Symbol, or programs using the LOCAL directive cannot be loaded with this Loader.

4. The JCP Loader creates the OVLOAD and DCB table and the M:SL DCB. The user must code the complete PCB except for the OVLOAD table address, the DCB table address, and the M:SL DCB address. The user must also code all DCBs (except M:SL), the Temp Stack and all other tables referenced in the PCB.

5. The PCB must be the first data loaded in the root.

6. Simplified Memory Management is assumed (see the "Overlay Loader" chapter, in the paragraph concerning the SMM option on the IOLOAD command).

Example:

Load a nonsegmented background program onto its permanent file:

```
:LOAD (IN,9TA82),(OUT,BP,FCOPY),(EXLOC,6100)
```

This example loads a background program from a 9-track tape onto the FCOPY file in the BP area. The program will be loaded to execute at 6100_{16} (EXLOC,6100). The program can consist of any number of object modules, but the last object module must be followed by an EOF. The object modules must obey the restrictions specified for the JCP Loader (see above).

LOADING SEGMENTED PROGRAMS

For loading a program with overlay segments, the above restrictions plus the following restrictions apply:

1. The only overlay structure allowed is a root plus one level of overlay.

2. The root must be the first group of object modules loaded and must be terminated by an IEOD command. Each group of object modules loaded after the root and terminated with an IEOD command will consist of one overlay segment and will be attached to the end of the root. Each overlay segment will be assigned a segment identification of 1 to n, where 1 is assigned to the first segment loaded, 2 assigned to the second segment loaded, etc. The segment identification is used in calling in an overlay segment at execution time.

3. The number of overlay segments must be correctly stated on the ILOAD command.

For multisegment programs, the JCP Loader builds the OVLOAD table at the end of the root and allocates seven words for the M:SL DCB. The entry address for each overlay is taken from the last encountered start item and placed in the OVLOAD table along with the load address, byte count, and segment identification.

The JCP Loader writes the program in core image format onto the appropriate file. The addresses and names of all M: or F: DEFs will be used by the Loader to create the DCB table. The loaded program can be executed via the RUN, ROV, or Name command, depending upon which disk file the program was loaded on. Note that the Loader never loads a program directly into core memory.

APPENDIX F. SYSTEM INITIALIZATION AND PATCHING

Modification of the resident CP-R system (including all system tables), CP-R overlays, and Job Control Processor can be performed at system boot time through patches defined on IMODIFY control commands. Any number of IMODIFY commands can be input, and the stack is terminated by a single !END command. The system is notified that patching is to take place by sense switch settings that are set prior to CP-R initialization.

During CP-R initialization, the Control Panel interrupt is inhibited so that a premature key-in will not prevent the completion of initialization.

At the end of initialization, the message

PLEASE KEYIN DATE AND TIME

will be output on the OC device. The operator must respond with the DT key-in. Any other key-in is considered to be an error.

As each !MODIFY command is processed, the overlay is read into core, modified, and written out to the system disk. Note that unused disk space between the end of an overlay and the end of an overlay's file (that is, unused space on the last sector of an overlay) can be used for a patch area.

INPUT OPTIONS

Sense switch (SSW) settings are used to select initialization options following a system boot. Because SSW's are used in SYSGEN also, care must be taken to allow for the system boot which follows the SYSGEN process.

Foreground programs which are designated resident are normally loaded during the initialization process. The periodic scheduler is also normally reactivated to resume these tasks. Both may be prevented by setting SSW4. No other functions are affected by the setting of SSW4.

Modify control commands and quick patch commands are input depending on the settings of SSW1, SSW2, and SSW3.

Setting SSW1 selects the C device for MODIFY command input. Setting SSW2 selects the OC device for MODIFY command input. Setting SSW1 and SSW2 causes the initialization process to hang following a WAIT instruction before any processing has been done. This allows the sense switch settings to be changed or any critical hand patching to be done before MODIFY or quick patch commands are read.

SSW3 set causes quick patch commands to be read from the same device as the MODIFY commands. If SSW1 and SSW2 are reset, the setting of SSW3 is ignored.

Setting	Meaning
XXX0	Load Resident foreground programs
XXX1	Don't load resident foreground programs
11XX	Hang at the start of initialization
00XX	Read no MODIFY or quick patch commands
100X	Read MODIFY commands from C device
010X	Read MODIFY commands from OC device
101X	Read MODIFY and quick patch from C device
011X	Read MODIFY and quick patch from OC device

QUICK PATCHES

Quick patches may occur at the beginning of the initialization process and use none of the system services. Because of this, they may be used to patch portions of the system used during the reading and processing of MODIFY commands. Also modifications to the early portions of the initialization processor may be made.

SSW3 set selects the reading of quick patches from either the C or OC device. SSW1 and SSW2 are used to indicate which device. Unlike the MODIFY commands, quick patches assume that the C device is a card reader and that the OC is a TTY type device. The reading of quick patches is always followed by the reading of MODIFY commands.

The format for quick patches is:

loc[+] value[+]

where

loc specifies the hex location of the location to be patched. A trailing + indicates that CP-R relocation bias is to be added.

value specifies the value to be stored in that location. A trailing + indicates that CP-R relocation bias is to be added.

Quick patching is terminated by a blank card when reading the C device, and a NL when reading from the OC device.

PATCH COMMAND FORMATS

In the IMODIFY control commands listed below, the brackets only indicate options available in the specification field and are not actually used in the MODIFY statements; the indicated parentheses and commas are required. The general format for IMODIFY commands is

IMODIFY ({module, loc / symbol, loc}),[x]value[,[x]value...]

where

symbol specifies either a table name or a resident overlay name.

module specifies one of the modules described below to be patched.

loc specifies the initial hexadecimal location to be modified and is relative to the beginning of the named module (for ABS, loc is relative location 0).

value specifies the hexadecimal value to be inserted in the location. Successive values go in successive locations. The value field may be repeated and has an alternate form.

(value, symbol)

where symbol is either a table name or an overlay name, whose value will be added to the value field. This alternate form for the value field is also valid for all the examples shown on this page.

x is the relocation flag indicating the address field of the value is to be relocated by adding the beginning address of any of the following:

- I is the Decimal Simulation start address.
- J is the JCP start address.
- L is the Floating-Point Simulation start address.
- M is the Monitor start address.
- O is the CP-R Overlay start address.
- P is the Patch area start address.
- R is the named module start address.
- V is the Convert Simulation start address.
- Y is the Byte-String Simulation start address.
- Z is the module preceding the current module.

PATCH SYSTEM OVERLAY OR JCP

IMODIFY (OLAY,name,loc),[x] value[,[x] value, ... [x] value]

where name is defined in the SYSGEN map under "CP-R Program Allocation" as the first entries. The size (non-resident) or base address (resident) of each overlay accompanies its name.

PATCH SIMULATION ROUTINE

IMODIFY (SROU,name,loc),[x]value[,[x]value,...,[x]value]

where name = FPSIM, DECSIM, BYTSIM, or CVSIM.

PATCH CP-R MONITOR

IMODIFY (CPR,loc),[x]value[,[x]value,...[x]value]

PATCH SYSTEM TABLES

IMODIFY (ABS,loc),value[,value,...value]

MODIFY PATCH AREA

IMODIFY (PATCH,loc),value

TRACE COMMAND FORMATS

In the IMODIFY control commands listed below, the brackets only indicate options available in the specification field and are not actually used in the modify statements; however, the indicated parenthesis and commas are required. The general format for the IMODIFY command is

IMODIFY {(SETRACE,task[,buffer[,events]]) / (TRACE,{ON / OFF})[,entry point,entry point,etc.]}

where

SETRACE changes the three trace parameters that follow. It does not activate or deactivate CP-R tracing.

task may be used to selectively trace a single task. The value is the value in the first byte of word 7 of the user's TCB. The default is zero, which traces all tasks.

buffer is an area to be used by the trace. This area should be large enough for the trace routine, entry point† symbol table, and 24 words for each event traced; that is, $\cong (1792_{10} + (24 * \text{events}))$. The default is the foreground private area.

events if the hexadecimal number of entries to trace before cycling over the oldest entry (this is a circular trace). The default is X'100' events.

TRACE activates (ON) or deactivates (OFF) the trace. If no entry point parameters exist, "ON" traces all entry points and "OFF" turns the trace completely off.†

ON or OFF are described in TRACE, above.

entry point activates (ON) or deactivates (OFF) each entry point listed.†

The user may obtain the circular trace stack by dumping memory beginning at "buffer". The EBCDIC option (if using the DM key-in) should be used to provide readable entry points and overlay names†. The format of memory is as follows:

ENTRY POINT	OVERLAY	PSD	4E & 4F
(2 words)	(2 words)	(2 words)	(2 words)
(16 registers)			

Four words of EBCDIC Xs will follow the last event traced. If the circular stack has been fully used, the Xs will also appear at the end of the stack.

†See "SYSGEN" for a discussion of entry points and overlay names.

CLEAR COMMAND FORMAT

The CLEAR command is provided to initialize file area directories to contain no files. It may be of particular use to clear the symbiont areas when an IS/OS inconsistency is reported at boot time, or to initialize areas SYSGEN'ed but not saved when booting a SAVE tape. It is important to be aware that this command only zeroes the first sector of the area, to give an empty file directory. It does not zero the whole area.

The form of the command is

```
!CLEAR AA[,AA,...]
```

where AA is the name of an area to be given an empty file directory.

!END COMMAND FORMAT

The !MODIFY control command(s) must be followed by a terminating !END command with the form

```
!END [CPR]
```

where CPR indicates that the copy of the CP-R resident on the system disk is to be replaced by the current version of CP-R in core.

SYSTEM PATCHING AND TRACING DIAGNOSTICS

The diagnostic messages in Table F-1 will be output on the OC device for a corresponding error in sense switch settings or !MODIFY commands.

Table F-1. System Patching Diagnostics

Message	Meaning	Action
!!ERROR ITEM XX-CORRECT AND CLEAR B $	If this is OC, only the message portion preceding the hyphen is output. If not OC, the entire message is output.	If this is OC, input next command. If not OC, idle machine, increment address, and RUN.
!!PATCH LOC ERR-CORRECT AND CLEAR B $	If this is OC, only the message portion preceding the hyphen is output. If not OC, the entire message is output.	If this is OC, input next command. If not OC, idle machine, increment address, and RUN.
!!NO PATCH AREA - CLEAR B $ IGNORES MODIFY COMMAND	If this is OC, only the message portion preceding the hyphen is output. If not OC, the entire message is output.	If this is OC, input next command. If not OC, idle machine, increment address, and RUN.

APPENDIX G. CHARACTER-ORIENTED COMMUNICATIONS ROUTINES

The COC Support Package provides on-line terminal communications capability to CP-R users via a single 7611 communications controller. The package is open-ended, and can be tailored to fit a user's particular application or extended to support other terminal types. It is also capable of supporting other character-oriented communications activities, such as store-and-forward message switching.

The set of reentrant subroutines that comprise COC can be included in a Public Library or loaded with user program(s) that call for COC support. In either case, the Overlay Loader is used to link the package with the calling program(s). The package is only available to foreground user programs.

The COC Support Package is capable of handling up to 64 character-oriented terminals. The following devices can be supported:

Device	Xerox Model Number
KSR-35	7015 or 7025
RO-35	7016 or 7026
ASR-35	7017 or 7027
KSR-37	7018

COC handles the following classes of communication mode and terminal device:

- Simplex, capable of one-way communication only.
- Half-duplex, capable of one-way transmission, but reversable.
- Full-duplex, capable of simultaneous transmission in both directions.
- Echoplex, a special case of full-duplex, where a character received from a terminal's keyboard must be sent back (i.e., "echoed") to the terminal to cause printing or display of the character.

The package provides for input/output handling of messages or arbitrary strings of characters. The input routines will accept requests for input on specified lines, accumulate input characters into messages, and deliver the messages to the user program upon request. The output routines deliver a message to a specified device. The package also provides for conversion between the EBCDIC character set used internally, and the external character set generated and/or recognized by the individual terminal device. Special characters are recognized and can cause special actions to be performed, such as message termination.

COC uses a pool of buffers for I/O operations and input messages are accumulated in these buffers as they are received. Output messages are transferred to the buffers during code translation. The buffers are dynamically allocated to the various lines, as needed. The size of the buffer pool is an assembly parameter that must be set large enough to accomodate the number, nature, and speed of the devices supported.

To a great extent, the COC Support Package is parameterized, and is driven by tables that define the characteristics of each device, the optional functions to be performed, and the code translation tables to be used. Special handling is provided for keyboard/displays operating either in character mode or in the optional message mode. Any one of three different control character modes may be selected, enabling the user to exercise greater control over the user-system dialogue.

The COC Support Package is organized into three main groups:

1. I/O command routines that accept user program calls for input, output, and status checking.
2. I/O interrupt routines that service the external interrupt pair connected to the COC, and perform input character translation and buffering, special action checking, output character transmission, and end-of-message action; end-action includes posting of status and, optionally, triggering of a specified external interrupt.
3. COC database that includes COC parameters, line control tables, and code translation and special action tables.

SYSTEM INTERFACE

The package provides the input/output interfacing between the computer and the 7611 COC controller and its associated terminal devices. The COC Support Package requires 1,226 words of core memory, plus nine words per communication line and four words per Buffer Pool block (see "Buffer Pool" later in this Appendix).

The 7611 COC requires an adjacent pair of external interrupts for signalling input and output events, respectively. The input (external) interrupt must have the higher priority of the two. The COC Support Package causes them to be centrally connected by executing a connect CAL for each, during execution of the initialization routine.

Input from the COC equipment is through a single IOP channel and is started by the initialization routine, CSTART. Data chaining is employed to cause the input to flow into a single ring buffer. The I/O interrupt is not used. An external interrupt pulse is generated each time a data character (along with line identification number) is input.

Output to the COC equipment is via the direct I/O system. Each data character is output (along with line identification number) by a Write-Direct instruction. When a transmitter in the COC is ready to send another character on a given line, an external interrupt signal is generated, and it remains set until a Write-Direct either sends another character on the line or turns off the transmitter.

USER PROGRAM INTERFACE

The COC Support Package is first assembled to fit the requirements of the local installation and then linked to user programs. The package may be loaded and stored in executable form in a Public Library (generally the case if it is called by more than one user program) or it may be loaded with the calling program(s) and stored in a User Library.

The interface between COC Support Package and user program is defined in three general areas:

1. Assembly-time parameters supplied by the user to define his particular configuration.

2. Line Control and other tables, available to the user program through DEF directives in the COC source deck. These describe the state of each line, its mode of operation, code conversion to be performed, etc.

3. I/O commands issued in the form of subroutine calls. Since standard FORTRAN IV calling sequences are used, the user programs may be coded in AP or FORTRAN IV.

PREPARATION AND USE OF COC

Since the COC Support Package supports communications with many different devices in several different communication modes, the package is highly parameterized, table driven whenever possible, and assembled specifically for each installation. The steps that must be performed to obtain an operational COC capability are as follows:

1. Assemble the standard COC routines using the AP assembler, specifying the number of lines, types of terminals, size of buffer pool, conversions desired, etc. (see Table G-2). The Relocatable Object Module (ROM) generated may be stored in a System Library but this is not required (see Chapter 6).

2. Create a load module containing the COC Support Package and user program(s) that will interface with it. The !OLOAD command must specify the number of tasks included; two for the COC interrupts, plus any required by the user program(s). The load module must be stored in the foreground program area of the disk file (see Chapter 6).

   An alternative form of step 2 consists of loading the COC Support Package by itself and storing it in a Public Library. Separate calling programs may then be loaded onto RAD (in the FP area), with the COC Support Package identified in a PUBLIB option on each of the !OLOAD commands.

3. Load the program root into core memory and initiate execution of the user program (see Chapter 5).

4. Execute (within the user program) the COC Initialize command CALL CSTART. Upon successful completion of this command, the COC Support Package and the 7611 equipment will be ready to accept input/output commands.

COC COMMANDS AND CALLING SEQUENCES

The I/O calls available to user programs are summarized in Table G-1. They all operate on a "no-wait" basis. That is, a call is accepted, the operation is begun and if the operation cannot be completed immediately, control is returned to the calling program.

Table G-1. COC I/O Commands

Command	Function
CALL CSTART	Initiate COC activity.
CALL CSET	Set line table parameters.
CALL CWRITE	Write output to a terminal.
CALL CREAD	Accept input from a terminal.
CALL CMOVE	Move input to user area.
CALL CHECK	Check I/O line status.
CALL CSTOP	Terminate COC activity.

Table G-1 is intended for ready reference: more detailed descriptions of the commands, their functions, and the calling sequences for both FORTRAN and AP are given below.

CALL CSTART (Initiate COC Activity)

This command initializes the COC Support Package, the 7611 equipment and associated external interrupts, and prepares the associated IOP channel for input. Optionally, CSTART may be called on to initiate activity on a single line, without affecting any of the others. Initialization and startup of the COC routines will be accomplished by the following call:

From FORTRAN programs, the call is

```
CALL CSTART (line, status)
```

From assembly language programs, the call is

LI, 14	2
BAL, 15	CSTART
ARG_1	address of line-number variable (see below).
ARG_2	address of location where status is to be returned.

TURN ON ALL LINES

If line number = 0, CSTART performs the following functions:

1. Initializes the COC input buffer, the COC temporary blocking buffers, and all pointers associated with them.
2. Starts input on the IOP channel to which the COC is connected.
3. Executes CONNECT CALs that centrally connect, arm, and enable the external interrupts to which the COC is attached.
4. Turns on all receivers.
5. Types out error notifications on the operator's console if any receivers have not been successfully turned on.
6. Returns control to the user program with status information stored in the indicated location, and coded as follows:

Code	Status
0	Normal return.
1	One or more receivers not turned on.
2	COC already active: request ignored unless any lines were out of service.
4	Unable to start COC.

If the status code indicates that one or more receivers were not turned on, the user may issue the command CALL CHECK recursively to determine the actual status of the lines of interest.

This form of the call may also be given when some of the receivers have already been turned on. In this case, only steps 4 and 5 are executed, as appropriate.

TURN ON SPECIFIC LINE

If a user program must initiate I/O activity with a specific line, the line number variable is set equal to the number of the line to be started and the above calling sequence is executed.

This form of the calling sequence performs the following functions:

1. Steps 1 through 3 under the "Turn On All Lines" procedure above, are executed only if this is the first call for COC initialization.
2. Only the specified receiver is turned on (the appropriate error notice is output if turn-on is unsuccessful).
3. The status variable is updated as shown above.

CALL CSET (Set Line Table Parameters)

This command is used to alter parameters describing the type and mode of operation of the terminal connected to a specified line. To change the mode and/or terminal type for a specific line, the following call may be executed:

From FORTRAN programs, the call is

CALL CSET (line-number, mode, terminal type)

From assembly language programs, the call is

LI, 14	3
BAL, 15	CSET
ARG_1	address of line number variable.
ARG_2	address of mode variable.
ARG_3	address of terminal-type variable.

The mode variable may have the following decimal values:

Mode Value	Meaning
1	Simplex (input only).
2	Simplex (output only).
3	Full-duplex.
4	Half-duplex (i.e., local-printing, no echo).
16	No-translate mode (neither input nor output characters are to be translated).
132	Echoplex (i.e., Full-duplex, simulating half-duplex).
>255	No change.

Note that the no-translate value must be added to one or the other valid mode values.

The terminal type variable may have the following values:

Type Value	Meaning
0	Model 33 Teletype.
1	Mode 35 Teletype.
2	Model 37 Teletype.
>255	No change.

The above values are identical to those contained in the line control table COCTERMN. New values may be created to handle new terminals, but the Input and Output Translate Pointer Tables (COCIT and COCOT, respectively) must be expanded accordingly.

CALL CWRITE (Write Output to a Terminal)

This command is used to perform the following functions:

1. Convert the indicated character string from the internal EBCDIC representation to the code set recognized by the specified terminal.
2. Store the converted character string in the COC buffer pool.
3. Update the appropriate line control data.
4. Start output to the terminal if it is not already in progress.
5. Return control to the user program.

Output will be started or continued by executing the following call:

From FORTRAN programs, the call is

```
CALL CWRITE (line, data, displacement,byte-count,
└──status[,end]).
```

From assembly language programs, the call is

LI, 14	5 (or 6).
BAL, 15	CWRITE
ARG_1	address of line number variable.
ARG_2	address of data string to be output.
ARG_3	address of byte displacement variable.
ARG_4	address of the byte-count variable.
ARG_5	address of the options/status word.
ARG_6 (optional)	address of the end-action interrupt specification.

where

line is an integer variable that identifies the terminal's line number (1 - Nmax).

data indicates the starting location of the area from which data is to be output (word address resolution).

displacement is an integer variable that gives the location of the first byte to be output, relative to the starting location (i.e., the number of bytes from the start of the data area).

byte-count is an integer variable that specifies the number of bytes to be output.

status is an integer variable that is to contain a code describing the options affecting execution of the Write (set prior to executing the call), and into which is stored a code giving the status of the request when returning to the calling program.

end-action (optional) indicates that an external interrupt is to be triggered at the completion of the output operation. The argument may be a constant or an integer variable having one of two forms:

2Haa, where aa is the two-character label for the interrupt.

3Znnn, where nnn is the hexadecimal address of the interrupt to be triggered.

Each ARG has the standard calling sequence format described in the Extended FORTRAN IV Operations Reference Manual, No. 90 11 43.

Execution of the CWRITE routine causes the following sequence of events:

1. Validity checks are made on the arguments; if any are found to be invalid, control is returned to the calling program with the status returned in the location indicated (i.e., pointed to by the address of argument 5).
2. The specified output string is moved from the user program area into temporary blocking buffers maintained by the COC routines. The characters are then translated into external form as they are moved, and the buffers are chained together as they are allocated, with the first being linked to the output pointer LINKO for

the specified line. If output is currently in progress, the new output string will be chained to the data already queued for output (see the description of Buffer Pool operations later in this Appendix).

3. If output is not already active, the first character is prepared and sent to the COC.

4. Control is returned to the calling program with an indication that the request was accepted and that output is in progress. The calling program may periodically monitor the progress of the output by execution of a CHECK command (i.e., CALL CHECK) for that line.

The options/status variable may have the following legal values upon entry to the Write routine:

Option Code	Option
0	Normal output operation.
1	Abort input if operation is in progress on the specified line, or awaiting completion of a currently active output.
2	Abort output if operation is in progress on the line (a carriage return and line feed will be sent to the terminal).

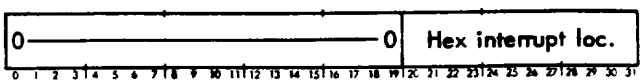
Combinations of the above are indicated by the arithmetic sum of the corresponding codes.

The options/status word will have the following values upon return to the calling program:

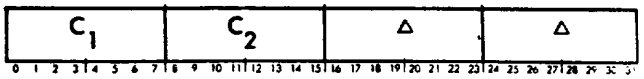
Status Code	Meaning
0	Normal Transfer; request accepted, data moved, and output in progress.
1	Insufficient buffer space; request ignored.
2	Invalid line number; request ignored.
4	Invalid line state; request ignored.
8	Break signal has been received (for information only); request will be processed as specified.

The presence of ARG_6 in the assembly language calling sequence is optional, but the number of arguments contained in register 14 must be adjusted accordingly. The format of the word containing the interrupt specification is as follows:

If interrupt address is given:



If interrupt label is given:



where

C_1C_2 is the Hollerith label assigned to the interrupt at SYSGEN time.

Δ indicates a Hollerith space character.

The priority of the end-action interrupt, and priority of any external interrupts associated with the user program, must be lower than the interrupt pair connected to the COC.

The end-action interrupt must be connected and enabled by the user program (i.e., through Monitor CALs or FORTRAN CONNECT statements).

CALL CREAD (Accept Input from a Terminal)

This command is used to perform the following functions:

1. Mark the input line status as active.

2. Initialize the pointers and counters necessary to begin accepting and saving input from a specified line.

3. Return control to the user.

Input from a terminal may be enabled by execution of the following call:

From FORTRAN programs, the call is

CALL CREAD (line,byte count,status[,end])

From assembly language programs, the call is

LI,14 3 (or 4).

BAL,15 CREAD.

ARG_1 address of line number variable.

ARG_2 address of the byte count variable.

ARG_3 address of status word.

ARG_4 (optional) address of end-action interrupt specification.

where the arguments are all integer variables (or elements of integer arrays), with the following assignments:

line is the communications line number.

byte count is the maximum number of characters to accept with this call.

status is a variable into which will be stored a code giving the status of the request.

end (optional) is the end-action interrupt specification, as described previously.

When CREAD is executed, preparations are made to accept characters received from the specified input line. No temporary buffers are assigned until needed. Control is returned to the calling program (with the status of the request stored in the status variable), and the input operation may then proceed in parallel with program execution.

Input will be terminated upon:

1. Receipt of a character defined as an end of message code (including the Break signal).
2. Receipt of the specified number of bytes.
3. Execution of a Write, or Move Input command with the abort-input option specified.

Note that this call signals only that input is to be accepted, translated to internal form, and echoed if appropriate. It is the responsibility of the user program to effect any preliminary positioning of cursor, carriage, or paper, and clearing of display screen, etc. The Write call is adequate for this purpose. When an input message is complete, the user program must also issue a Move Input call to copy the message from the COC buffers into its own buffer space. The status argument has the following values when control is returned to the caller:

Status Code	Meaning
0	Normal; request has been accepted and input may proceed.
2	Invalid line number; request ignored.
4	Invalid line state; request ignored.
8	Break signal received; request ignored.

CALL CMOVE (Move Input to a User Area)

This command is used to copy the converted input characters that have been received on a specified line into the specified user area. As the data is moved, the allocated buffers are released to the pool. When an input message has been received (whether complete or not), it may be moved from the COC Buffer Pool into the user's program area by execution of the following call:

From FORTRAN programs, the call is

```
CALL  CMOVE (line,buffer,displacement,count,
     status)
```

From assembly language programs, the call is

LI, 14	5
BAL, 15	CMOVE
ARG_1	address of line number variable.
ARG_2	address of user's buffer area.
ARG_3	address of byte displacement variable.
ARG_4	address of byte count variable.
ARG_5	address of options/status word.

When this call is executed, the input received from the specified line is moved into the specified area. Control is returned to the calling program when:

1. An error is dectected in the calling sequence.
2. All received input has been moved.
3. The specified number of bytes have been moved.

When control is returned to the calling program, the following modifications will have been made to the calling sequence:

ARG_3	the byte displacement variable now points to the byte following the last byte moved (relative to the user buffer address).
ARG_4	the byte count variable contains a count of the input that has been moved, or that is yet to be moved (see description of "status", below).
ARG_5	the options have been replaced by the status of the operation.

The following options may be specified in the option/status variable:

Option Code	Meaning
0	Normal operation.
1	Terminate input if not already inactive (end-action interrupt will not be triggered).

If input is still active, or if all data received has not yet been moved, the next Move Input operation will begin where the current operation stops. It is not necessary that input be complete when this call is executed, or that option 1 be specified if input is active (see below).

The status returned in the ARG_5 location is as follows:

Status Code	Meaning
0	Normal operation; all input moved (ARG_4 location contains count of bytes actually moved).

Status Code	Meaning
1	Normal operation; not all input has been moved (ARG_4 location contains a count of the number of bytes remaining to be moved).
2	Invalid line number; request ignored.
4	Input is active (for information only; does not affect this operation).
8	Break signal has been received (for information only; does not affect this operation).

Combinations of the above conditions are indicated by the arithmetic sum of their corresponding codes.

Note that the Move Input routine is responsible only for moving an input message from the linked buffers into a single set of contiguous locations specified in the calling sequence. The user program is responsible for any editing that must be performed on the message.

This call may be executed whether or not input is still active on the line. When executed, the call will move all characters that have been received in the current "message" into the specified area, up to the specified byte count. The temporary buffers are released as data is moved out of them. The byte count variable (ARG_4) is updated accordingly as follows:

1. If all characters so far received have been moved (status code = 0, 4, 8, or 12), the byte count variable will be set equal to the number of characters that were actually moved.

2. If all characters have not been moved; that is, if the specified byte-count was less than the actual number of characters received (indicated by status code = 1, 5, 9, or 13), the byte count variable will be set equal to the number of received characters remaining in the COC buffers at the completion of this call. That is, upon return to the caller, byte count = actual number of characters received less the specified byte count.

The COC Line Tables are updated so that the next Move Input call for the line will begin where the preceding call left off, regardless of whether the call was terminated because the specified byte count was satisfied or because all data was moved that was received up to that point.

The line STATE table is also updated if STATE = 8 (input complete), and all characters have been moved. In this case, the input complete state is cleared, thus permitting subsequent CREAD and CWRITE calls to be accepted for the line. However, if not all data has been moved, the input-complete state will remain set, causing subsequent CREAD and CWRITE calls to be rejected.

CALL CHECK (Check I/O Line Status)

This command is used to determine the mode and status parameters associated with the specified line. It is used principally by programs that must monitor the progress of an I/O operation, rather than waiting to be interrupted upon completion.

When a FORTRAN program must monitor the progress of an I/O operation or check the status of a line, it can issue the following call:

CALL CHECK (line,mode,state,byte count)

where the arguments are all integer variables (or elements of integer arrays), and have the following meanings:

line — contains the line number when the call is executed (i.e., supplied by the user).

mode — will contain the current terminal operating mode upon return to the calling program (see below).

state — will contain the current line state upon return to the calling program (see below).

byte count — will contain a count of the number of characters that have been received from the terminal (if any).

The values returned in the mode and state argument locations are identical to those contained in the corresponding entries of the MODE and STATE line control tables. These tables are described later in this Appendix.

The assembly language calling sequence is not described since assembly language programs have easy, direct access to the Line Control tables.

CALL CSTOP (Terminate COC Activity)

This command is used to terminate COC operation, halt input on the associated IOP channel, and disarm the associated external interrupts. Optionally, CSTOP may be called on to terminate a single line without affecting operation of the others.

TURN OFF ALL LINES

When all operations have been completed with COC equipment, the following call should be executed:

From FORTRAN programs, the call is

```
CALL    CSTOP
```

From assembly language programs, the call is

```
LI, 14      0

BAL, 15     CSTOP
```

This routine closes out all I/O activity in the following manner:

1. Disarms the external interrupt pair assigned to the COC.
2. Halts input on the IOP channel assigned to the COC.
3. Turns off all receivers.

Control is then returned to the calling program, which is responsible for issuing the RLS (release foreground task) CALL.

TURNS OFF SPECIFIC LINE

If a specific line is to be turned off (i.e., the receiver turned off, and the Line STATE marked "out of service"), the following call should be executed:

From FORTRAN programs, the call is

```
CALL    CSTOP (line-number)
```

From assembly language programs, the call is

```
LI,14   1
BAL,15  CSTOP
ARG     address of line number variable
```

This call turns off only the specified receiver. No change is made to the IOP activity or to the external interrupt pair connected to the COC.

I/O INTERRUPT ROUTINES

INPUT INTERRUPT PROCESSING

The external interrupt pulse associated with COC input is generated each time the COC sends a character/line number pair to the IOP. The routine that services this interrupt performs the following functions:

1. Fetches the pair from the IOP ring buffer.
2. Identifies the source.
3. Converts the character via the translation table assigned to the line.
4. Examines the character to see if it requires special handling (e.g., end-of-message).
5. Stores the character in the buffer assigned to the line, allocating an additional buffer when necessary.
6. Marks the input complete if the specified number of characters have been received or if the received character calls for end-of-message activation.
7. Triggers the (optionally) specified external interrupt if end-of-message action is called for.
8. Checks for any additional character pairs (e.g., from other lines) received during the time that the first pair was being processed.
9. Exits.

OUTPUT INTERRUPT PROCESSING

COC generates an interrupt whenever a transmitter indicates it is ready to send another character. The routine that services this interrupt performs the following functions:

1. Determines which line caused the interrupt.
2. Determines whether any data is left to send on that line. If data is left, the next character is sent; if not, the transmitter is turned off, line status is updated, and the (optionally) specified end-action interrupt is triggered.
3. Exits.

END-ACTION PROCESSING

Once started, an input or output operation will proceed in parallel with user program execution (by means of the COC I/O interrupt routines). The user program may determine when an I/O operation is complete by periodically checking the line status. Alternatively, the user program may request that a specific external interrupt be triggered when an operation is complete. The COC CREAD and CWRITE commands permit this interrupt to be specified in the respective calling sequences, and a different interrupt may be specified for each. Completion of output on a line occurs when the specified number of characters have been transmitted or when a "break" signal is received on the line (see below).

Input completion on a line occurs under the following circumstances:

1. When the byte count equals the specified maximum.
2. When a "break" signal is received.
3. When a character is received that is designated as an end-of-message or "activation" character in the translation and/or special action tables.

SPECIAL ACTION PROCESSING

Special actions are performed by the input interrupt routine when certain control characters are received. After

translation, these characters are identified by their code value being in the range

$00_{16} - 3F_{16}$

When a translated character is found to lie in this range, it is used as an index to a branch table. The entries in the table point to the individual subroutines that perform the indicated processing. About half of these special action codes are assigned; the remainder are available to users who wish to tailor the special action processing to suit their own needs.

Special processing is also performed by the output routine, CWRITE. After a character is translated for output, it is tested for a value in the range

$F1_{16} - FE_{16}$

that has odd parity. As with the input interrupt routine, such a character is used as an index to a special action branch table. The special action characters are defined toward the end of this Appendix along with the translation tables.

BREAK SIGNAL HANDLING

The special line signal "break" (also called "long space") is recognized and recorded regardless of line state or condition of transmission. The signal is transmitted from a terminal by depression of the BREAK key on a TTY. When a break signal is received, a bit in the line control table associated with the sending terminal is set. The condition of this bit is reported to the user in the status returned from an I/O command call (i.e., after initiation of a Read, Write, Move, or Status Check command), and the bit is cleared. During input from the terminal, the receipt of a break signal is also reported by placing the character EOT in the input buffer and setting the end-of-message or "activate" condition. If the program using the COC routines wishes to examine the break condition without doing an I/O command, it may examine the bit directly (see "Line Control Tables").

ESCAPE SEQUENCE PROCESSING

The COC Support Package provides the capability to handle two-character escape sequences. That is, the user may cause special processing to occur when an ESC is received, followed by one of a user-designated set of "follower" characters. The user must supply the set of routines to handle the escape sequences when the COC Support Package is assembled or loaded for his installation, (a description of the required tables is given later in this chapter). Note that the routines will be entered with the COC input interrupt "high", hence the processing must be of very short duration.

COC DATABASE

COC PARAMETERS

The parameters itemized in Table G-2 define the configuration of the COC equipment and the COC Support Package as a whole. The parameters must be inserted into COC source deck (or included as updates to a compressed deck) by the user prior to assembly of the package for a given installation.

Table G-2. COC Parameters

Label	Directive	Description
COCNB	EQU	Number of 4-word buffers to be provided (see discussion in subsequent chapter).
COCDN	DATA	Device number of attached COC (IOP assignment).
COCNO	EQU	COC number (DIO address).
COCNL	EQU	Number of lines; 64 lines are the maximum number that may be connected to a 7611 COC controller.†
COCII	EQU	Location of input external interrupt (hexadecimal address).
COCIO	EQU	Output external interrupt.

†The COC input/output commands (i.e., CALL CWRITE, etc.) assume the lines to be numbered from 1 to COCNL. However, when assembly language coders are referring directly to the Line Control Tables, they must decrement the line number before using as an index value.

LINE CONTROL TABLES

The items listed in Table G-3 are maintained for each line supported by the COC. Each item is contained in a separate table of similar items indexed by the line number. All of the items are available to the user program loaded with the COC Support Package (i.e., each item is declared in a DEF directive). The user must declare each item to be accessed with a REF directive in the source code.

In Table G-3 the given label corresponds to the first entry in the table (i.e., to the table entry that applies to line 1). The size given is for each entry in the table. The table contains as many entries as there are lines connected to the COC, as defined by the parameter COCNL.

LINE STATE TABLE FORMAT

This table contains the current operational state of each line. The table is maintained dynamically for each line. The algebraic sum of any of the items below is valid except where the states represented by the sum are incompatible (e.g., a line cannot be accepting input and have input

Table G-3. Line Control Table Items

Label	Size	Contents	Comments
STATE	Byte	Line State	Initialized by CSTART (described in Line State Table format below).
MODE	Byte	Line/Terminal mode	Initially supplied at assembly time. May be modified by Special Actions, by CSET routine or by user program (described in Terminal Mode Table format below).
COCTERMN	Byte	Terminal type	Initially supplied at assembly time. May be changed by CSET routine (described in Terminal Type Table below).
COCOT	Word	Location of Output Conversion Table	Supplied at assembly time and may be changed by user program (see below for standard table format). Note: these tables are indexed by terminal type code rather than line number.
COCIT	Word	Location of Input Conversion Table	
LINKO	Halfword	Input buffer address	Relative address in buffer pool of first buffer in chain. Initialized and maintained by COC support routines.
	Halfword	Output buffer address	
COCBAI	Word	Input character address	Byte address of next character to be transmitted or stored. Initialized and maintained by COC routines.
COCBAO	Word	Output character address	
COCMC	Halfword	Input character limit	Number of characters to be accepted from terminal during current input.
ARS	Halfword	Input character count	Count of converted input characters contained in current message.
COCENDI	Word	Input end-action	Contains FPT pointed to by the CAL executed when end-action interrupt is triggered. Initialized and maintained by COC support routines.
COCENDO	Word	Output interrupt-specification	
FLAG	Byte	Output sequence flag	Initialized and maintained by COC support routines. Controls sequence of output actions resulting from input.

waiting; therefore, the code 5 is invalid). The codes assigned are as follows:

Decimal Code	Mode of Terminal (One Byte Per Line)
00	Inactive.
1	Accepting input.
2	Transmitting output.
4	Input waiting; enable when output complete.
8	Input complete.
64	Error in message.
128	Out of service (i.e., receiver is not on).

TERMINAL MODE TABLE FORMAT

This table contains the current operating mode of the terminal connected to each line. It is specified at assembly time, and is normally unchanged during operation except for the break flag and the escape-sequence flag.

The codes assigned are as follows:

Decimal Code	Mode of Terminal (One Byte Per Line)
1	Simplex – input.
2	Simplex – output.
3	Full-duplex (i.e., independent simultaneous I/O).
4	Half-duplex.

Decimal Code	Mode of Terminal (One Byte Per Line)
16	No-translate mode; neither input nor output characters are to be translated.
32	Break flag; long space has been received.
64	Escape sequence (ESC character has been received; follower character is awaited.
128	Echo flag (Half-duplex mode should also be set).

TERMINAL TYPE TABLE (COCTERMN)

This table contains a code byte that specifies, for each line, the type of terminal connected. This code is used as an index into the COCIT and COCOT tables when translating input and output characters respectively. If the user wants to add new terminal types or introduce different modes of handling a single terminal, he can accomplish this by assigning unused type codes, adding new translation tables, and expanding the COCOT and COCIT tables accordingly.

The terminal type codes assigned are as follows:

Code	Terminal Type
0	Teletype, model 33.
1	Teletype, model 35.
2	Teletype, model 37.

BUFFER POOL

Input from, and output to, user terminals is buffered in linked chains of four-word blocks maintained by the COC Support Package. Each buffer block has the format shown and consists of a halfword link, followed by (up to) 14 data characters.

Buffer Format

word 0	Link		C_1	C_2
word 1	C_3	C_4	C_5	C_6
word 2	C_7	C_8	C_9	C_{10}
word 3	C_{11}	C_{12}	C_{13}	C_{14}

where

link contains the address of the next buffer in the chain (relative to COCBUF, the buffer pool base address).

C_i is the I/O message character.

The number of buffers in the pool is specified by the user in the assembly time parameter, COCNB. The buffers are initially chained to the head of buffer pool, COCHPB. That is, the location COCHPB contains the relative address of the first available four-word buffer in the pool. The link halfword of this buffer contains the relative address of the next available buffer, and so on. The end of the chain is signified by a zero link in the last buffer of the chain.

When a message is to be transmitted to a terminal, the message characters are translated to external code, and moved from the user area into the buffer pool. Buffers are removed from the chain of available buffers as needed, and assigned to the specific output line. The assigned buffers are linked together, and the relative address of the first buffer is stored in the LINKO Table in the entry position corresponding to the specified line number. After all characters contained in a buffer have been transmitted to the terminal, the buffer is released and linked back into the chain of available buffers.

Buffering of input messages is handled in exactly the same way. The relative address of the first buffer in the chain for each terminal is stored in the LINKI Table. The buffer pool may also be used for other dynamic storage requirements.

STANDARD INPUT/OUTPUT TRANSLATION TABLES

Tables G-4 and G-5 show the equivalence between the internal EBCDIC character set and the external codes generated/recognized by the standard Xerox Teletype. In the present implementation, one input and one output translation table serve for the TTY. Variations in control character handling for the terminal is resolved within the supplied special action routines described in the following tables.

INPUT SPECIAL ACTION TABLE

Translated input characters with values in the range $00\text{-}3F_{16}$ are handled by special routines that perform the actions listed below in Table G-6. Users may add new entries to the table in the range $21_{16}\text{-}3F_{16}$ (except for the parity error code, 38_{16}). The input translation table must be modified (or another table supplied) to cause conversion of input characters into the new special character codes. The parameter SPEC must also be altered to give the new limit to the Special Action Table.

Table G-4. Input Translation Table

ASCII†		EBCDIC		ASCII†		EBCDIC	
Code	Character	Code	Character	Code	Character	Code	Character
0	NULL	00		20	blank	40	blank
1	SOH	01		21	!	5A	!
2	STX	02		22	"	7F	"
3	ETX	03		23	#	7B	#
4	EOT	04		24	$	5B	$
5	ENQ	09		25	%	6C	%
6	ACK	06		26	&	50	&
7	BEL	07		27	'	7D	'
8	BS	08		28	(	4D	(
9	HT	05		29	)	5D	)
A	NL	15		2A	*	5C	*
B	VT	0B		2B	+	4E	+
C	FF	0C		2C	,	6B	,
D	CR	0D		2D	-	60	-
E	SO	0E		2E	.	4B	.
F	SI	0F		2F	/	61	/
10	DLE	10		30	0	F0	0
11	DC1(XON)	11		31	1	F1	1
12	DC2(TAPE)	12		32	2	F2	2
13	DC3(XOFF)	13		33	3	F3	3
14	DC4(TAPE)	14		34	4	F4	4
15	NAK	0A		35	5	F5	5
16	SYN	16		36	6	F6	6
17	ETB	17		37	7	F7	7
18	CAN	18		38	8	F8	8
19	EM	19		39	9	F9	9
1A	SS	1A		3A	:	7A	:
1B	ESCI	1B		3B	;	5E	;
1C	FS	1C		3C	<	4C	<
1D	GS	1D		3D	=	7E	=
1E	RS	1E		3E	>	6E	>
1F	US	1F		3F	?	6F	?

†The ASCII code shown is the 7-bit code without the parity bit setting. The Input Translation Table, as contained in the program, is coded for even parity ASCII. All odd parity positions in the table contain a hexadecimal 38, indicating a parity error to the Support Package.

Table G-5. Input Conversion

ASCII		EBCDIC		ASCII		EBCDIC	
Code	Character	Code	Character	Code	Character	Code	Character
40	@	7C	@	4A	J	D1	J
41	A	C1	A	4B	K	D2	K
42	B	C2	B	4C	L	D3	L
43	C	C3	C	4D	M	D4	M
44	D	C4	D	4E	N	D5	N
45	E	C5	E	4F	O	D6	O
46	F	C6	F	50	P	D7	P
47	G	C7	G	51	Q	D8	Q
48	H	C8	H	52	R	D9	R
49	I	C9	I	53	S	E2	S

Table G-5. Input Conversion (cont.)

ASCII		EBCDIC		ASCII		EBCDIC	
Code	Character	Code	Character	Code	Character	Code	Character
54	T	E3	T	67	g	87	g
55	U	E4	U	68	h	88	h
56	V	E5	V	69	i	89	i
57	W	E6	W	6A	j	91	j
58	X	E7	X	6B	k	92	k
59	Y	E8	Y	6C	l	93	l
5A	Z	E9	Z	6D	m	94	m
5B	[[	4F	|	6E	n	95	n
5C		4A	¢	6F	o	96	o
5D	]--[	5F	¬	70	p	97	p
5E	^ ^ ↑	6A	^	71	q	98	q
5F	- - ←	6D	-	72	r	99	r
	↑ ↑ ↑ most TTYs			73	s	A2	s
	| XDS 7015			74	t	A3	t
	68 ASCII			75	u	A4	u
				76	v	A5	v
				77	w	A6	w
				78	x	A7	x
60		7C	@	79	y	A8	y
61	a	81	a	7A	z	A9	z
62	b	82	b	7B	{	42	{
63	c	83	c	7C	¦ ¬	5F	¬
64	d	84	d	7D	}	52	}
65	e	85	e	7E	~	4F	~
66	f	86	f	7F	RUBOUT	20	|

Table G-6. Special Action Table

Hex Value	EBCDIC Name	End-of-Message	Special Actions
00	Null	---	Ignore.
01	SOH	000	Process normally.
02	STX	---	Process normally.
03	ETX	---	Process normally.
04	EOT	---	Ignore.
05	HT	---	Process normally.
06	ACK	---	Process normally.
07	BEL	---	Process normally.
08	BS	---	Process normally.
09	ENQ	---	Process normally.
0A	NAK	Yes	Do input-complete processing and store EOT character in input buffer.
0B	VT	---	Process normally.

Table G-6. Special Action Table (cont.)

Hex Value	EBCDIC Name	End-of-Message	Special Actions
0C	FF	---	Process normally.
0D	CR	Yes	Echo LF, put X'0D' in input buffer, do input-complete processing.
0E	SO	---	Process normally.
0F	SI	---	Process normally.
10	DLE	---	Process normally.
11	DC1	---	Process normally.
12	DC2	---	Process normally.
13	DC3	---	Process normally.
14	DC4	---	Process normally.
15	LF(NL)	Yes	Echo CR. Put X'15' in buffer: do input-complete processing.
16	SYN	---	Process normally.
17	ETB	---	Process normally.
18	CAN	---	Delete line from input buffer. Send "back arrow" character then LF, CR.
19	EM	---	Process normally.
1A	SUB	---	Process normally.
1B	ESC	---	Set escape flag in MODE byte. Next input character will be treated as a follower in 2-character ESC sequence.
1C	FS	---	Process normally.
1D	GS	---	Process normally.
1E	RS	---	Process normally.
1F	US	---	Toggle Echo Mode; i.e., turn off if now on, and vice versa.
20	DEL	---	Erase last character from input buffer. Send "backslash" to TTY.
38	Parity error	---	Send "#" to terminal. Put 2F in buffer.

Note: All translated characters in the range 21_{16}-$3F_{16}$ will be treated as a parity error. The lower limit is defined by the value of SPEC, which the user must change when adding to the Special Action Tables.

ESCAPE SEQUENCE TABLES

The COC Support Package includes the provision for handling 2-character ESCAPE sequence by use of routines supplied by the user. In such a sequence, the first of two characters is usually an ESC character. Alternatively, the user may supply a translation table that can cause any desired character to be translated into the ESC code ($1B_{16}$). The second ESCAPE sequence is called a follower character. The ESCAPE sequences are handled by two tables.

FOLLOWER CHARACTER TABLE

This table has one byte for each designated follower character that contains the EBCDIC code for the character. The characters do not have to be stored in numeric sequence.

Follower Character Table

CESCNR

$\frac{N+1}{4}$ words

N	C_1	C_2	C_3
C_4	C_5	C_6	C_7
⋮	⋮	⋮	⋮

where

N is the number of entries.

C_i is the EBCDIC code for one follower character.

ESCAPE BRANCH TABLE

This table has one word for each designated follower character that contains a branch instruction to the routine that performs the desired functions. The position of an entry in the follower character table defines the position of the corresponding branch instruction in the ESCAPE Branch Table.

ESCAPE Branch Table

CESCNR

N words

Branch instruction to routine that handles follower character C1.
Branch to routine that handles character C2.
⋮
Branch to routine that handles character CN.

The COC Support Package is supplied with a set of dummy entries. The user may replace these, delete them, or expand the table, as necessary.

OUTPUT CONVERSION (EBCDIC-USASCII)

In Table G-7 all empty squares are coded F1, and no character is sent to the terminal.

The upper character in each square represents the graphic(s) or the name of the character and the lower characters represent the hexadecimal value of the character to be transmitted, minus the even-parity bit which is included in the supplied table.

In special cases, the output conversion table contains an F, followed by the code number of a special handling routine.

The usable F codes are those with bad parity (F1, 2, 4, 7, 8, B, D, and E), since other F codes represent correct parity-checked characters.

The assigned codes are:

F1	Null. No character is placed in the output buffer.
F2	CR and LF are placed in the buffer.
F4	Output message is complete.

Table G-7. Output Conversion (EBCDIC - USASCII)

Least Significant Digits (of EBCDIC)	Most Significant Digits (of EBCDIC) 0	1			4	5	6	7	8	9	A	B	C	D	E	F
0	NUL 00	DLE 10			SP 20	& 26	- 2D									0 30
1	SOH 01	DC1 11					/ 2F		a 61	j 6A		\ 5C	A 41	J 4A		1 31
2	STX 02	DC2 12							b 62	k 6B	s 73	{ 7B	B 42	K 4B	S 53	2 32
3	ETX 03	DC3 13							c 63	l 6C	t 74	} 7D	C 43	L 4C	T 54	3 33
4	EOT 04	DC4 14							d 64	m 6D	u 75	[≠ 5B	D 44	M 4D	U 55	4 34
5	HT 09	NL LF 0A							e 65	n 6E	v 76	] ! 5D	E 45	N 4E	V 56	5 35
6	ACK 06	SYN 16							f 66	o 6F	w 77		F 46	O 4F	W 57	6 36
7	BEL 07	ETB 17							g 67	p 70	x 78		G 47	P 50	X 58	7 37
8	EOM BS 08	CAN 18							h 68	q 71	y 79		H 48	Q 51	Y 59	8 38
9	ENQ 05	EM 19							i 69	r 72	z 7A		I 49	R 52	Z 5A	9 39
A	NAK 15	SUB 1A			¢ 60	! | 21	¬ ^ 5E	: 3A								
B	VT 0B	ESC 1B			. 2E	$ 24	, 2C	# 23								
C	FF 0C	FS 1C			< 3C	* 2A	% 25	@ 40								
D	CR 0D	GS 1D			(28	) 29	_ = 5E	' 27								
E	SO 0E	RS 1E			+ 2B	; 3B	> 3E	= 3D								
F	SI 0F	US 1F			| ¦ 7C	¬ ~ 7E	? 3F	" 22								DEL 7F

APPENDIX H. JOB MANAGEMENT

CONCEPT OF A JOB

A job is a collection of related primary or secondary tasks wherein each task is a body of procedural code, and the job is the unit of work that is scheduled and initialized. Tasks are the subunits of work that are executed. Each job has a system Job Control Block associated with it containing the data that is sharable or common across all tasks within the job. The Job Control Block resides in system space.

Externally, a job is identified by a unique eight-character EBCDIC name. Internally, a job is identified by a unique number assigned by the system. There are two permanent jobs in CP-R; the CP-R job (which is the system itself), and the BKG (background) job. Additionally, there may be any number of foreground user jobs active at a given time, up to the SYSGEN set limit on the number of jobs allowed.

The foreground programs of existing CP-R installations will run as tasks in the CP-R job. User jobs may contain any number of tasks.

An important concept of a job in CP-R is that all resources, files, devices, or user defined resources are job-related and can be used by any task in the job. Devices may be designated at SYSGEN as sharable (across jobs) or allocatable (nonsharable across jobs).

To create a foreground job, the CP-R service SJOB is used. It is available either as a key-in or as a CAL. SJOB does not initiate any task in the job; instead, it sets up job controls and builds a Job Control Block in the system area of memory. A RUN key-in or CAL, or an INIT key-in or CAL may be issued to start tasks in a job as follows:

- A RUN key-in initiates a task in the CP-R job.
- A RUN CAL initiates a task in the CP-R job.
- An INIT key-in or CAL initiates a task in the specified job.

To terminate a foreground job, the CP-R service KJOB is used. KJOB is available as a key-in or as a CAL. It terminates the specified job by terminating every task within the job and deleting the job controls.

To facilitate the initiation of tasks in a job, a job level table called the Job Program Table (JPT) is maintained in the Job Control Block. The JPT allows the user to specify the name of a load module to be used for execution of a task. It is composed of doubleword pairs of EBCDIC task name/load module name equivalences. Table look-up on the task name determines which load module will be loaded as the task specified on a RUN or INIT request. JPT entries are made via the SETNAME system function call. SETNAME specifies a task name/load module name pair that are entered in the JPT of the specified job. Task initiation uses the JPT to determine which load module to load. If no JPT entry is found for the specified task name, the task name is used for the load module name. SETNAME may also be used to delete previously established JPT entries.

CP-R interjob communication may occur either because it is not prevented, such as in the unrestricted use of memory by foreground tasks, or because system services are invoked. Permanent disk files may be shared by tasks regardless of job designation. Access to disk files is controlled via Open/Close requests. Public libraries are shared between all jobs. Foreground mailboxes may be used by foreground tasks for interjob communication.

CP-R services SIGNAL, POLL, and POST may be used to communicate between jobs. Services ENQUEUE and DEQUEUE may be used for system independent resource management between jobs.

APPENDIX I. TASK MANAGEMENT

Each centrally or directly connected interrupt in CP-R defines a new task. Each program loaded into memory by the Foreground or Background Loader defines a load module. In the case of foreground, a load module may consist of one or more tasks. The background load module consists of one task that runs when no hardware interrupts are active. Each secondary load module also defines a single task entry. Two additional permanent tasks exist, which are the CP-R Control Tasks.

The background activity is controlled by the JCP and runs a sequential stream of job steps within jobs.

Foreground primary and secondary activity is controlled by CP-R task management as directed by user service calls, key-ins, and control commands.

Upon receiving a request to run a primary load module, the Foreground Loader portion of the Control Task becomes active and loads the program into memory when the required core is available. The initialization required by the primary load module can be accomplished by operations performed when the program is given control at its entry point. This code runs as an extension of the CP-R Control Task and therefore blocks all Control Task functions until it EXITs or TERMs.

The tasks connected to interrupts in the foreground program become active when the interrupt becomes active and remain active until they EXIT or TERM. The interrupt can be triggered from external sources as an end-action to a previously requested service or by a user TRIGGER CAL.

Secondary tasks are created by use of the INIT service call. Upon receipt of the service call, a task and load module entry is created and the secondary task is loaded at the dispatcher level and software level specified in the INIT service call. Unlike primary load modules, secondary load modules are initialized at their own priority and therefore many such initializations may occur concurrently. There is thus no reason for special initialization code as is required for primary load modules.

DISCONNECTING PRIMARY TASKS FROM INTERRUPTS

The user can disconnect a primary task from an interrupt through the use of a DISCONNECT function call. The interrupt will be left disarmed and available for connection to another program prior to the termination of the currently connected program.

EVENT MANAGEMENT

Services requested by tasks via CALs are classified as immediate, synchronous, or asynchronous, as described below.

IMMEDIATE

All services are classified as immediate if they are fully satisfied by the time CP-R returns control to the user and require no waits or delays for an event to occur. The services that are immediate are as follows:

DEVICE	JTRAP	DEQUEUE
PC	STATUS	DISARM
DEBUG	ERASE	DISCONNECT
ERRSEND	RELPAGE	SLAVE
MEDIA	DISABLE	TEST
POST	ARM	EXTM
SCHED	CONNECT	SETNAME
INT	TEXIT	ALARM
TRIGGER	TRTY	RECALARM
ENABLE	EXIT	CALRTN
MASTER	ABORT	GETTIME
RLS	TRTN	SJOB
STOPIO	TERM	KJOB
STARTIO	EXDA	STOP
IOEX	START	DEACTIVATE
TIME	MODIFY	UNLOCK
TRAP	LOCK	

SYNCHRONOUS

Some services involve a wait while processing and become synchronous either implicitly (a wait option was not provided) or explicitly (user selected wait option in his FPT). These services initiate the action, wait for it to complete, execute retry paths as required, and finally return to the caller after posting the successful or unsuccessful type completion data. Services that are implicitly synchronous are as follows:

DEVICE	WAIT	ALLOT
OPEN	WAITALL	DELETE
CLOSE	WAITANY	TRUNCATE
CORRES	GETPAGE	PREFMODE
ACTIVATE	TEXT	JOB

Services that may be explicitly made synchronous by specifying a wait option are as follows:

REW	CHECK (no busy address)
WEOF	PRINT
UNLOAD	TYPE
READ	SEGLOAD
WRITE	STIMER
PFIL	DELFPT (no busy address)
PREC	STDLB
ENQUEUE	POLL
INIT	SIGNAL

ASYNCHRONOUS

Some services involve a wait while processing and become asynchronous when the user specifies 'no-wait' in his FPT.

Upon execution of the service call, CP-R initiates the service and returns to the caller. Services that may be requested asynchronously are as follows:

REW	CHECK (with busy address)
WEOF	PRINT
UNLOAD	TYPE
READ	SEGLOAD
WRITE	STIMER
PFIL	DELFPT (with busy address)
PREC	STDLB
ENQUEUE	POLL
INIT	SIGNAL

In order to control asynchronous and synchronous services, CP-R creates temporary Event Control Blocks (ECBs) in which it retains information such as type of service, priority of the caller, calling program's identification, caller's FPT or DCB address from the original service request, end-action requested, and type completion when the service is done. These ECBs are automatically deleted by CP-R prior to returning to the calling task if a synchronous service request was made. If the request was asynchronous, the ECB will not be deleted until a service checking call is issued by the user, resulting in the posting of type completion and exit to the user via the normal, error, or abnormal return.

Therefore, when an asynchronous service request has been made, the user must ensure that this temporary control information is deleted by issuing a CHECK, DELFPT, WAITANY, WAITALL, or TEST service call until the FPT/DCB is posted as complete.

Some service calls have individual characteristics that vary from those described above. These are as follows:

RUN

The RUN service call is implicitly asynchronous in that the RUN request is initiated when the RUN call is executed but program loading is actually performed at some later time by the Foreground Loader. The foreground user can optionally request status posting into a signal address and/or an end-action interrupt. No event controls are created and CHECK class calls do not address themselves to previously requested RUNS.

SEGLOAD

A SEGLOAD service call from the background has implicit wait regardless of FPT options. A SEGLOAD service call from the foreground can be with or without wait. A CHECK on a SEGLOAD call that contains the OVLOAD Table address corresponding to the read FPT for the segment loaded may be used. If no check is issued, the system will automatically delete the event controls for the user.

APPENDIX J. RESOURCE MANAGEMENT

All CP-R resources are job-related and can be used by any task in the job, within the restrictions governing the specific type of resource. Resources include files and devices.

CP-R allows the user to control his file and device resources by not placing unnecessary system use restrictions on them and by providing the system services with which the user may superimpose his own controls.

Permanent files may be shared by tasks from different jobs without restriction. Within a job the only restriction is that a user cannot read sequentially from a file that was previously OPEN.

DCB-file correspondence for tasks in foreground jobs can be made via load time ASSIGNs, user DCB manipulation, and the CP-R services DEVICE and STDLB.

File service routines are provided to dynamically define (ALLOT), delete (DELETE), or truncate (TRUNCATE) permanent disk files. Service calls are provided to dynamically modify DCB parameters (DEVICE), inspect DCB parameters (DEVICE), acquire or release a file via its operational label, and alter an operational label assignment (STDLB). These service routines provide job-level file resource management under user control.

In CP-R, devices may be acquired for a job via explicit OPEN requests, implicit OPENs resulting from READ requests, or explicit STDLB service requests. As for files, DCB-device correspondence can be made via load time ASSIGNs, user DCB manipulations, and DEVICE and STDLB services.

Device use is specified at SYSGEN and via a STOPIO service call as available for foreground, all callers, no callers (in which case user requests are queued by the system), and diagnostic (down). The device is further designated at SYSGEN as sharable or for exclusive use only. An OPEN request on a device designated for exclusive use will result in the device being acquired by the job for exclusive use if the device has not already been acquired by a previous job.

Dynamic device assignment may be made via the service functions DEVICE and STDLB.

Memory is considered a resource and is managed for tasks in foreground jobs and the background job. Segments containing reentrant code (such as Public Libraries) may be considered as resources and shared by foreground and background tasks.

The background job may use all the CP-R services for file and device management. Background may be excluded from use of specific devices dedicated to foreground use. The background job takes the lowest priority in the system but may not be preempted from use of a device for which it has obtained exclusive use.

A background job may specify DCB-file and DCB-device correspondence via run-time ASSIGN, load-time ASSIGN, user manipulation of DCBs, and CP-R services DEVICE and STDLB.

The STDLB system function call allows tasks to acquire or release exclusive use of an allocatable device in addition to STDLB operational label assignment. The STDLB service is provided to acquire, release, or reassign a device or file and to reassign an operational label. STDLB operates on the operational label table in the Job Control Block (JCB) of the job containing the requesting task. If specified and if possible, it also reserves the device for the job.

At job creation, all jobs get the CP-R job (system permanent values) operational label assignments. Tasks in the job may use STDLB calls to alter these assignments for their own job only. These alterations apply for the duration of the job for all tasks in the job or until a task in the job issues another STDLB request. STDLB calls may be used as often as required by the job.

The !STDLB control command applies to background use only and is processed by the JCP.

The STDLB key-in is accepted by the key-in processor and processed by the STDLB CAL. A STDLB request generated by a key-in or by a task in the CP-R job is recognized by the STDLB CAL. When so recognized, the CAL processor changes the operational label assignment in the CP-R job, and changes the operational label assignment in all jobs currently using the CP-R default values.

The STDLB CAL processor uses standard OPEN/CLOSE calls to set up assignments for files and sharable devices and to reserve exclusive devices for STDLB requests.

CLOSE checks the operational label assignments in a requesting job to ensure that a file or device is not released prematurely. A STDLB request to release a file or device will alter the job entry before requesting a close so that CLOSE will not find the file or device and will thus allow the resource to be released.

The use of STDLB does not preclude use of OPEN/CLOSE by tasks, but provides assurance that if STDLB is used to acquire a resource for a job, either a STDLB is required to release the resource or it will not be released until job

termination. If STDLB were not used to acquire a resource for a job, then OPEN/CLOSE pairs would acquire/release the resource as available.

File or device acquisition is accomplished by specifying the operational label and resource name (operational label, numeric zero, disk file name, device name) in the FPT. If the enqueue option was not specified, the resource will be acquired exclusively for the job if the resource is "allocatable", and shared by the job if the resource is "sharable". An error completion code will be returned if it was not possible to acquire the resource. If the enqueue option was specified and the resource was not available, STDLB will return to the user with the FPT BUSY bit set. In any case, the user must CHECK at some point to obtain completion status and force completion of the STDLB request. The standard rules for using CHECK apply. If a time was specified in the FPT, the enqueue will be timed out; otherwise, no time-out will be performed and the request will not disappear unless a specific delete FPT request is made, the job terminates, or the resource is acquired.

Release of a resource (one previously acquired by an STDLB call) occurs when the release option is specified.

APPENDIX K. PERIODIC SCHEDULING

The periodic scheduling feature of CP-R allows the user to specify a task which CP-R should initialize at user requested intervals. The user may establish such a request via the monitor service call or key-in. The request may be deleted using either service call or key-in.

The periodic scheduler executes as a control task overlay and keeps its requests for initialization in the file SCHED in the SP area. The SCHED file is automatically alloted as an extensible file, when the first request for the periodic services is received. The user may override the automatic allotment by alloting the file before executing a periodic scheduler request. This allows the user to set an upper limit on the number of such requests outstanding. The SCHED file must be unblocked and have a granule size of 256 words. Each granule holds up to 18 periodic scheduler requests.

The start time specified by the request will be rounded up to the next five second start time. Any interval specification will be rounded up to the next five second quantity. If the period specified proves to be less than the time required for task execution, the number of cycles missed will be logged when the task is next initialized.

The entry will be automatically deleted for any error received on the INIT. Exceptions are if the task already has been INITed or if there is not enough table space. Requests with zero intervals will, of course, be deleted upon their singular INIT.

A full description of all variations of periodic scheduling exists under the SCHED monitor service call and SCHED key-in.

APPENDIX L. ERROR LOGGING

Error logging is provided as an CP-R option that may be invoked by using the ERRORLOG option on the :MONITOR card during SYSGEN.

As error and other log records are produced, temporary space is obtained for each record and the record filled in. Pointers to each record are pushed into a stack and a count of the records incremented. The Control Task is then triggered to copy the records out to the ER oplabel. As the records are written out, the temporary space is returned to the common pool and the pointer is pulled from the stack. The records are 16 words long and may be written to a blocked or unblocked file, a magnetic tape, a card punch, or NULL, depending upon the ER oplabel assignment.

If a log record is lost for any reason (i.e., no more room in the stack or no temporary space available), a count of lost logs will be incremented.

The following error messages may be output by the Control Task during error logging:

```
!!ERROR LOG FILE IS FULL
!!UNRECOVERABLE ERROR ON ERROR LOG FILE
!!UNABLE TO OPEN ERROR LOG FILE
```

The number of I/O accesses, errors, the error rate by device and the error log counters may be interrogated using the ESUM key-in (see Chapter 3).

Error logging may be turned off or on by the use of the following key-in:

ELOG {ON / OFF} ↵

Turning error logging off this way saves the majority of the execution time overhead associated with error logging but retains the maintenance of I/O statistics displayed by ESUM key-in.

Text entries may be put in the error log by using the ERRSEND key-in.

Date/time key-in causes a series of startup, time stamp and configuration entries to be placed in the error log.

If the error log is maintained on a file, it can be emptied and reinitialized with the time-stamp and configuration entries by the following key-in:

```
ELOG PURGE
```

The purge will not take effect if the error log file is being used by any process other than error logging (such as ELLA, the Error Log Listing and Analysis program) at the time of the command.

APPENDIX M. SYSTEM ALARM PROCEDURES

CP-R has a number of software checks that will indicate when an irrecoverable software or hardware fault has occurred. When such a fault is detected, an immediate and controlled system shutdown will be forced and the sequence of events listed below will follow:

1. All inhibits will be set and all registers saved.

2. A 1-second wait will occur, during which a descending tone will be output on the console speaker to permit current I/O to run down.

3. All I/O devices will be HIOed through their primary device address.

4. The 1-Khz alarm will be turned on.

5. An alarm message in the form of "!!ALARM text" will be output on OC.

6. If there is a CK area present in the system, memory will be saved on the CK area until either the end of memory or end of area is reached. 1024-byte blocks will be used and a "CORE SAVED" message will be output on OC when end of memory or CK area is reached.

7. The alarm will be turned OFF.

8. One of three final actions will then be done based on the content of the system alarm receiver control word ALARMREC.

   a. The system will hang in a one instruction loop with all interrupts inhibited if the ALARMREC cell is zero. This is the normal default case.

   b. The system will reboot from the system disk if the ALARMREC cell is negative. This may be set as a default by use of the REBOOT option on the :MONITOR card during SYSGEN. It may also be set by a foreground user using the ALARMREC CAL.

   c. The system will branch to a user alarm receiver located in real memory. The system state will be as if a system reset had occurred and the user's alarm receiver will be entered in Master-unmapped mode. All interrupts will be disarmed and disabled and all inhibits will be set. All I/O will have been stopped. No CALs should be done in the user alarm receiver.

      This option is selected by a user task setting ALARMREC to the address of his alarm receiver by use of the ALARMREC CAL.

The core saved on the CK area may be interrogated by using the CKD, CRS, or CRD key-ins when the system is brought back up (see Chapter 3).

Note: The first ROLL-OUT operation will overwrite the CK area and the core image will be destroyed.

APPENDIX N. CP-R SERVICE CALLS

Requests for CP-R services are made via CAL1 instructions that cause a Monitor trap, thus allowing CP-R to perform the requested service†. CAL2, 3, and 4 may be used similarly for user-dependent services.

CAL1 instructions may be immediate, as in the form

```
CAL1,x      value
```

or contain an FPT (Function Parameter Table) address, as in the form

```
CAL1,x      [*] address[,index]
```

An FPT contains one or more words with parameters to further define the exact CP-R function to be performed.

CAL1 calls may be executed out of the registers, via the EXU instruction, or cascading EXU instructions. The FPT also may reside in the registers, as may any directly or indirectly referenced parameter within the FPT (unless otherwise specified in the CAL description).

---

†Refer to the inside front cover of this manual for a list of the available CP-R services.

APPENDIX O. ERROR AND ABNORMAL CODES

TYPE COMPLETIONS (TYC)

A Type Completion Code (TYC) is an indicator that shows the type of completion for a service call. The routine that processes the service request or an associated routine detects normal or abnormal completion of the service and notifies the user by posting the TYC in the TYC field of the user's FPT or DCB. The Type Completion Codes are listed in Table O-1.

ERROR CODES

An error code is returned to the user via the provided error or abnormal return address if normal completion of the requested service is not possible. At entry to the user's Error/Abnormal routine, the error code is contained in byte 0 of register 10. For I/O requests, the DCB address is contained in the address field (low order 17 bits) of register 10. For non-I/O requests, the FPT address is contained in the address field of register 10. Register 8 contains the address of the location following the CAL1 instruction. The previous contents of registers 8 and 10 are lost. The Error Codes are listed in Table O-2.

If an error condition is detected in the servicing of a request and the user has not provided an error return address in the request FPT/DCB, processing will terminate and one of the following will occur:

1. If the Abort Override bit is set in the FPT, register 8 and 10 are modified as if an error return address was provided and control is returned to the user at the instruction following the CAL1.

2. If the Abort Override bit is not set, control will be turned over to the TRAP handler (a pseudotrap, TRAP50, occurs). The user's trap handler will be entered if present; otherwise, the user will be aborted.

If an abnormal condition is detected in the servicing of a request and an abnormal return address has not been provided, registers 8 and 10 will not be altered and the CAL processor will return to the instruction following the CAL1. In this case the contents of the TYC field will be unpredictable.

Note that in order to handle an FPT abnormal condition the user's FPT must contain an abnormal return address; to handle an FPT error condition the user's FPT must contain an error return address. Similarly, if the user wishes to handle DCB abnormal conditions, the DCB must contain an abnormal return address; in order to handle DCB error conditions, the DCB must contain an error return address.

Abnormal conditions (DCB or FPT) apply only to I/O service requests.

Table O-3 illustrates the settings of TYC, BUSY, and R10 for various types of calls and error conditions.

Table O-1. Type Completion Codes

Service Category	TYC (Hex)	Description	R10, byte 0 Settings From Table N-2
All	00	If Busy = 0, immediate error occurred on an I/O request.	One of the DCB error or abnormal codes.
		If Busy = 1, service is in process.	Unchanged
All	01	Normal completion.	Unchanged
I/O	02	Lost data (user's buffer was shorter than data record read).	07
I/O	03	Beginning of tape.	1D
I/O	04	Foreground I/O request would have required operator intervention to complete. Instead, the request was errored.	30
I/O	05	Physical end of tape, area, or file space.	1C
I/O	06	End of data (IEOD).	05

Table O-1. Type Completion Codes (cont.)

Service Category	TYC (Hex)	Description	R10, byte 0 Settings From Table N-2
I/O	07	End of file (tape mark on tape).	06
I/O	08	Irrecoverable I/O error.	41
I/O	09	Reserved.	-
I/O	0A	Write protection violation.	42
I/O	0B-0F	Reserved.	-
I/O	10	Line printer loss of position.	4B
I/O	11	Inconsistent status or position detected during I/O.	4C
I/O	12	Request aborted.	4D
I/O	13-18	Reserved.	-
I/O	19-60	Reserved.	-
Non-I/O	61-FF	See Table N-2 for meanings.	Same as TYC

Table O-2. Monitor Error and Abnormal Codes

Service Category	R10 Hex Code	Description	Return Address
I/O	01	A DCB has been opened with incorrect parameters.	DCB Abnormal
I/O	03	The assigned disk file does not exist or the assigned device is down or unavailable.	DCB Abnormal
I/O	05	An end-of-data or end-of-file (tapemark) has been encountered (IEOD).	FPT Abnormal
I/O	06	The end of input has been encountered (i.e., a control command has been read on the C device).	FPT Abnormal
I/O	07	The buffer specified is smaller than the data read.	FPT Abnormal
I/O	0A	An attempt has been made to close a DCB that is already closed.	DCB Abnormal
I/O	1C	The end-of-tape has been encountered.	FPT Abnormal
I/O	1D	The beginning-of-tape has been encountered.	FPT Abnormal
I/O	2E	An attempt has been made to open a DCB that is already open.	DCB Abnormal
I/O	2F	DED DPndd, R key-in in effect.	DCB Abnormal
I/O	30	The request resulted in a condition which the operator can correct if the proper message has been output on OC.	FPT Abnormal

Table O-2. Monitor Error and Abnormal Codes (cont.)

Service Category	R10 Hex Code	Description	Return address
I/O	40	A request has been made to read an output device.	DCB Error
I/O	41	An irrecoverable error has occurred.	FPT Error
I/O	42	A RAD write protection violation has occurred.	FPT Error
I/O	44	A request has been made to write on an input device.	DCB Error
I/O	46	The DCB contains insufficient information to open a closed DCB on a read operation.	DCB Error
I/O	47	The DCB contains insufficient information to open a closed DCB on a write operation.	DCB Error
I/O	48	A non-real-time request was made on a busy DCB.	DCB Error
I/O	4A	The user buffer address or byte count is invalid.	DCB Error
I/O	4B	Hard copy or format error on BDP Printer. Paper position is indeterminate.	FPT Error
I/O	4C	Inconsistent status or position detected during I/O.	FPT Error
I/O	4D	The request has been aborted by the system or the user.	FPT Error
I/O	4E	An error has occurred from which the cooperative cannot recover.	FPT Error
I/O	54	More than one attempt has been made to read a control message from the C device through the same DCB.	DCB Error
I/O	55	The DCB cannot be opened because the RFT is full, the RAD is down, or no buffer could be found for the directory search.	DCB Error
I/O	58	A foreground request was made to the C device.	DCB Error
I/O	59	DCB has changed since being OPENed.	DCB Error
I/O	5B	Illegal job identification for a RAD file close request.	DCB Error
I/O	60	Input request on a shared device or file.	DCB Error
All	61	Interrupt Label is undefined. Interrupt Address is outside valid range or is not valid for call.	FPT Error
All	62	Task Name illegal or invalid.	FPT Error
All	63	Job Name illegal or invalid.	FPT Error
All	64	End Action used is not legal for caller.	FPT Error
All	65	Conflict with lower priority task.	FPT Error
All	66	Space was not available in a table whose size was set at SYSGEN in TSPACE.	FPT Error
All	67	The operation timed-out (non-I/O time-out).	FPT Error

Table O-2. Monitor Error and Abnormal Codes (cont.)

Service Category	R10 Hex Code	Description	Return Address
All	68	The responsible task terminated without completing a service requested.	FPT Error
All	69	A service request was deleted by the responsible task or a third party.	FPT Error
All	6A	The service requested is not valid for the caller because of his Job/Task Type.	FPT Error
All	6B	Wait not legal for primary task.	FPT Error
All	6C	Clock value missing or invalid.	FPT Error
All	6D	Signalling task terminated.	FPT Error
All	6E	Signalling task aborted.	FPT Error
All	6F	The responsible task terminated abnormally without completing the service.	FPT Error
All	70	Illegal or invalid RAD area name.	FPT Error
All	71	The file name is illegal or undefined.	FPT Error
All	72	Space was not available on disk.	FPT Error
All	73	Debug initialization while in DEBUG.	FPT Error
All	74	Oplabel invalid.	FPT Error
All	75	Debug request while terming.	FPT Error
All	76	Resource invalid.	FPT Error
All	77	Illegal file format.	FPT Error
All	78	Two tasks attempting to act on one item simultaneously.	FPT Error
All	79	Illegal combination of parameters.	FPT Error
Symbionts	7A	Illegal JOB card format or parameter encountered by symbionts.	FPT Error
SJOB	7B	Illegal account/user name.	FPT Error
I/O	7C	File directory reentrancy (used internally).	Not Applicable
ASSIGN	7D	DCB is too small for assignment requested.	FPT Error
All	82	Task already idle on stop request.	FPT Error
All	83	Immediate service request cannot be satisfied.	FPT Error
All	84	Resource never acquired originally on DEQ.	FPT Error
All	86	Identification not valid.	FPT Error

Table O-2. Monitor Error and Abnormal Codes (cont.)

Service Category	R10 Hex Code	Description	Return Address
All	8A	CAL already connected.	FPT Error
All	8B	Address parameter not valid.	FPT Error
All	8C	Invalid status flags.	FPT Error
All	8D	Data area invalid.	FPT Error
All	8E	List task is not active.	FPT Error
All	90	Pages not in a defined partition.	FPT Error
All	91	Pages not in a preferred partition.	FPT Error
All	92	VPNL/VPNH not in same preferred partition or segment.	FPT Error
All	93	Preferred page allocation conflict.	FPT Error
All	94	—	
All	95	Preferred page not assigned to requestor.	FPT Error
All	96	Invalid segment number.	FPT Error
All	97	VPNL/VPNH error.	FPT Error
All	98	Page allocation conflict.	FPT Error
All	99	On-going I/O in one or more pages.	FPT Error
All	9A	Inactive segment is not in "defined" state.	FPT Error
All	9B	Operation on inactive segment.	FPT Error
All	9C	Unlock on segment that is not locked.	FPT Error
All	9D	Erase on segment that is not active.	FPT Error
All	9E	Lock on segment that is already locked.	FPT Error
All	9F	CAL or FPT address reference to inactive segment.	FPT Error
All	A0	Activate to active segment.	FPT Error
All	A1	A read error has occurred.	FPT Error
All	A2	Segment length equals zero.	FPT Error
All	A3	Simplified Memory Management Memory size exceeded.	FPT Error
All	A4	Real memory not available.	FPT Error
All	AF	Memory management control error.	FPT Error
All	B0	Task being initiated is a PUBLIB.	FPT Error
All	B1	Secondary task being initiated is primary, or vice versa.	FPT Error

Table O-2. Monitor Error and Abnormal Codes (cont.)

Service Category	R10 Hex Code	Description	Return Address
All	B2	Secondary program exceeds memory limits.	FPT Error
All	B3	Too many segments for secondary task.	FPT Error
All	B4	Load module being initiated uses a PUBLIB that names a nonpublib load module.	FPT Error
All	B5	Secondary task uses a primary PUBLIB, or vice versa.	FPT Error
All	B6	I/O error when trying to initiate a background program.	FPT Error
All	B7	Background program initiated into foreground or vice versa.	FPT Error

Table O-3. TYC, BUSY and R10, Byte 0 Settings

Service Type	Service Started Normally	Unable to Start Service	Service Started and Completed Normally	Service Started Normally, Completed Abnormally
Asynchronous Service with wait	–	Busy = 0; TYC = 00 if I/O, = Error Code if non-I/O; R10 from Table N-2, I/O: DCB Error or Abnormal non-I/O: R10 ≥ 61	TYC = 01; Busy = 0; R10 = Entry Value	TYC from Table N-1; Busy 0; R10 from Table N-2, FPT Error or Abnormal
Asynchronous Service	TYC = 00; Busy = 1; R10 = Entry Value	Busy = 0; TYC = 00 if I/O, = Error Code if non-I/O; R10 from Table N-2, I/O: DCB Error or Abnormal non-I/O: R10 ≥ 61	–	–
CHECK of a previous service	TYC = 00; Busy = 1; R10 = Entry Value (Busy exit)	(NOP)	TYC = 01; Busy = 0; R10 = Entry Value	TYC from Table N-1; R10 from Table N-2, FPT Error or Abnormal
Immediate Service Request	–	–	TYC = 01; Busy = 0; R10 = Entry Value	TYC from Table N-1; Busy = 0; R10 from Table N-2, FPT Error or Abnormal
DELFPT a Service	TYC = 00; Busy = 1; R10 = Entry Value (Busy exit)	(NOP)	TYC = 01; Busy = 0; R10 = Entry Value (successfully deleted or normally completed)	TYC from Table N-1; R10 from Table N-2, FPT Error or Abnormal (service deleted abnormally or completed with errors)

APPENDIX P. VOLUME INITIALIZATION

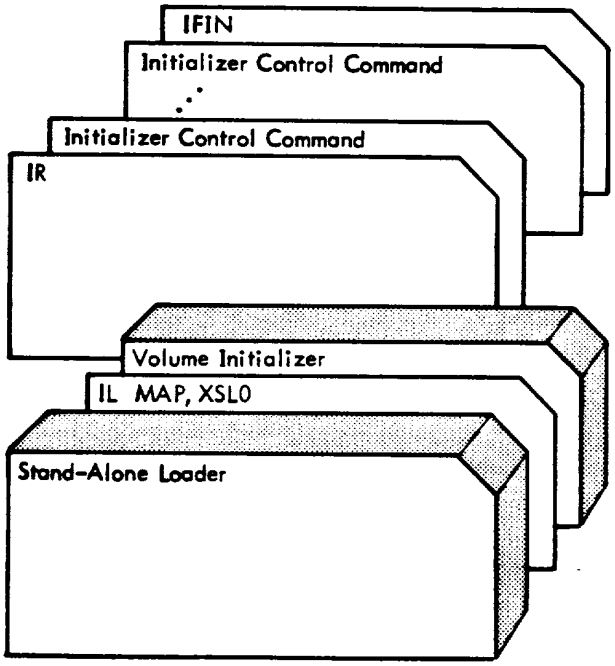
INTRODUCTION

There is a special stand-alone processor called VOLINIT that initializes disk packs. Any number of disk packs can be initialized in a single run. VOLINIT establishes serial numbers and ownerships, writes headers and other system information in selected areas of the volumes, and tests the surface of the disks and selects alternate tracks to be used in place of flawed tracks, if necessary. The following information is a summary of VOLINIT operations procedures. For more detailed information refer to Publication 706226-11D00.

LOADING VOLINIT

To get the Volume Initializer loaded into the system in preparation for reading control commands, use the following loading procedure:

1. Form the following deck:



2. Set Sense Switch 1 and boot in from the card reader. Type NO or YES as desired in reply to a request to make any hardware assignment changes.

3. When the deck is read in, key-in :SYST C, CRA03.

4. Reset Sense Switch 1. The program will now be loaded and executed.

Note: If loading VOLINIT on a Sigma 9 with less than 128K of memory, the following additional steps must be taken:

- Place an address stop (stop on instruction) at location 10C5 before performing Step 2, above.
- When the program stops at location 10C5 (prior to Step 3), place the machine in instruction hold and step once. Then release the instruction hold and continue.

VOLINIT COMMANDS

Any number of disk pack devices can be initialized by one VOLINIT job. If a processor control command is lengthy and must be continued on another card, the continuation card must be concluded with a semicolon (;) character. The processor command has the form

DPndd, mmmm[,(option), . . . (option)]

where

ndd specifies the address of the device containing the volume to be initialized.

mmmm specifies the model number of the device.

The options are

PUBLIC, sn specifies the serial number for the volume being initialized is public or private. The serial number, sn, parameter is the one- to eight-byte EBCDIC serial number of the volume.

FORMAT, $\begin{Bmatrix} adr_1\,[-adr_2], \ldots \\ NONE \end{Bmatrix}$ specifies areas of the the device to be formatted. Formatting consists of the writing of sector headers and test patterns to verify the recording capability of a given area of a volume. If the FORMAT option is omitted, the entire device will be formatted by default. If the option (FORMAT, NONE) is specified, formatting will not be performed. Otherwise, only those tracks specified in the range(s) $adr_1 - adr_2$ will be formatted. Each parameter 'adr' specifies a cylinder/track address, with a group of tracks

specified by giving the addresses of the first and last tracks separated by a hyphen. Thus, for example, the specification (FORMAT, 0/0, 100/0-202/19) will cause the first track and the 100-202 cylinders of a disk pack volume to be formatted.

An acceptable maintenance procedure is to format new disk packs and any volume being initialized that has not been formatted in the preceding two months.

FLAW,adr_1 [-adr_2],... specifies areas on the device that will be unconditionally flawed (see "Flawing", below). The parameter adr_1 [-adr_2] specifies a single cylinder/track address as in FORMAT. Those tracks that are found to be bad while formatting are automatically flawed; consequently, the FLAW option is necessary only for deleting tracks that are marginal or troublesome.

The VOLINIT error summary should be maintained for each volume to keep a record of bad tracks.

NO TEST specifies that surface testing will be inhibited. Areas specified by FORMAT are automatically surface tested unless NO TEST is specified. A surface test consists of writing preselected patterns on the device.

If a processor command contains only the device name, DPndd and model number, with no options are specified, the VOLINIT processor logs the contents of the volume (serial number; date; public/private indicator; home address, if public; account number, if private; and the number of available cylinders, if private) and does not write on the volume. If options have been specified, the previous contents of the volume are invalid.

It is important to note that the operator must make certain that the correct disk packs are mounted on the correct spindles before the VOLINIT commands are processed.

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.

7244 pack copy, 270
7251/52 disk area definition, 270
7254 cartridge
  copies, 272
  initialization, 270
  mounting, 271
  removal, 271

A

A align specified columns, 209
A assign command, 129
abnormal
  address, 54
  codes, 322
  conditions, 54
ABORT function call, 96
access protection, 5, 110
account inventory file, 4, 228
accounting log, 22
ACTIVATE function call, 115
ALARM call, 106
alarm messages (SYSGEN and SYSLOAD), 262-264
ALARM receiver call, 106
ALL map, 154
ALL option, 261
ALLOBT command (JCP), 19
ALLOBT command (SYSGEN), 256
allocating SP area, 262
ALLOT command (RADEDIT), 170
ALLOT function call, 60
ANSCII control-character
  translation table, 236, 237
  7-bit communication codes, 231
AP examples, 184
ARM function call, 96, 79
assembly language, 152
ASSIGN
  control command, 13, 44, 149
  function call, 66
  service function, 44
asynchronous operation control, 121
asynchronous services, 315
ATTEND command, 15
ATTENTION key, 31

B

B branch command, 131
background, 1
  devices, 21
  job organization, 7
  processing, 12
  program area, 3
  programs, 75
  temp area, 4, 11, 44, 247
  temp area storage requirement, 270
  temp files, 19
BATCH command, 19, 226
batch processing, 3
BDSECTOR command, 178
binary object modules, 145
blank COMMON, 165
  control command, 146, 165
  name, 166
BOOT26, 238
BP command, 199
BREAK, 224
  function, 198
  signal handling, 305
buffer pool, 307

C

C control segments command, 129
C copy current line, 209
CAL handling, 78
CAL-parameter groups, 274-288
CALI instruction, 44, 54
CALRTN function call, 99
CANCEL command, 227
card punch messages/key-ins, 42
card reader messages/key-ins, 42
CATALOG command, 176
CC control command, 17
CHAN command, 252
channel designation codes, 8
character sets, 229
check completion, 121
check correspondence of DCB assignments, 66
CHECK function call, 121, 44, 71
checkpoint area, 4
CL change string-search column limits, 207
CLEAR command (patching), 296
CLEAR command (RADEDIT), 173
CLOSE function call, 55
cluster designation codes, 8
CM insert commentary, 205
COC command (SYSGEN), 257
COC support package, 297
combined key-ins, 40
COMMENT command, 259
common allocation, 165
COMMON control command, 146
CONNECT function call, 96, 79
connecting primary tasks, 79

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.

console interrupt, 290
CONTINUE command, 228
control codes, 229
control commands, 12
Control X, 223
Control Y, 224
cooperatives, 291
COPY command, 171
CORRES function call, 66
CP-R
  control task, 5,31
  disk file management, 271
  function calls, 321
  functions, 3
  interrupting, 224
  lowest-cost, minimum configuration, 266
  memory management, 107
  messages, 26,105
  processes, 1
  procedures system, 273
  responses, 26
  structure, 238
  system memory, 107,108
  temp stack, 77
  terms, 1
CTINT command, 258

D

D delete string, 207
D dump command, 130
DAL command, 22
data control block (DCB), 2,44,49,151
  assignment, 50
  creation, 49
  format, 50
DCBTAB, 151
DCT, 44
DE delete current record, 209
DE delete records, 202
DEACTIVATE function call, 116
DEBUG, 126
  command, 227
  command syntax, 128
  commands, 128
  error messages, 132
  function call, 126
  functions, 127
  input, 127
  output, 127
  snapshot, 131
  trap control, 128
DEFREF file, 152
DELETE command, 173
DELETE function call, 62
DELFPT function call, 122
DEQ function call, 93
device access, 45
DEVICE command, 253
device
  control, 41
  control messages, 31
  control table (DCT), 248
  designation codes, 8
  preemption, 48
DEVICE/file mode function calls, 65
direct access, 45
direct I/O execution (IOEX), 48
DISABLE function call, 99,79
DISARM function call, 96,79
DISCONNECT function call, 96,79
disconnecting primary tasks, 79,314
disk access methods, 45
disk allocation, 246,247
disk areas, 3,246
  codes, 9
  default sizes, 247
  definition, 270
  protection, 170
disk data protection, 43
  device model numbers and parameters, 255
  file account option, 44
  file identifiers, 10
  pack files, 45
  pack messages/key-ins, 42
  restoration messages, 179,182
  volume initialization, 328
  write protection, 6
DISPLAY format, 40
DPCOPY command, 172
DSECT allocation example, 147
DT key-in, 31
DUMP command, 176

E

E execution control command, 130
E overwrite and extend blanks, 208
EBCDIC 8-bit computer codes, 230
EBCDIC file, 152
echoing characters, 222
EDIT, 197,7,220
  command, 199
  command structure, 198
  command summary, 213
  commands, 198
  input/output conventions, 197
  messages, 211
ENABLE function call, 99,79
END command (EDIT), 199
END command (patching), 296
END command (RADEDIT), 179
end-action, 79
end-action processing, 304
ENQ function call, 92
entering multiline records, 223

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.

entry address, 152
EOD command, 20
ERASE function call, 116
erasing characters, 223
erasing current input line, 223
error
  address, 54
  codes, 322
  conditions, 54
  detecting, 224
  logging, 319
  reporting, 224
  response, 198
ERRSEND function call, 105
event management, 314
EXCLUDE command, 146
EXDA function call, 95
executing background programs, 75
EXIT command, 227
EXIT function call, 95
exit lower case interpret mode, 224
extensible files, 46
external definitions, 151
EXTM command, 228
EXTM function call, 87

F

F follow string, 208
FAST option, 261
FD find and delete records, 204
file (also see disks)
  blocked, 45
  closing, 55
  commands, 199
  compressed, 45
  identifiers, 197
  opening, 54
  organization, 45
  permanent, 44
  temporary, 44
  unblocked, 45
FIN command (JCP), 20
FIN command (SYSGEN), 259
foreground, 1
  blocking buffers, 107
  job examples, 193
  job organization, 7
  mailbox, 107
  preferred memory, 5, 107
  private memory, 5, 107
  program, 17
  program area, 3
FORTRAN interface, 165
FORTRAN job examples, 187
FS find and type sequence number, 204
FT find and type record and sequence number, 204
function parameter table (FPT), 2,44,273
GDSECTOR command, 179
GETASN function call, 67
GETPAGE function call, 116
GETTIME function call, 104
GO command, 200,228
GO file, 4,23
GO, OV, X1-X9 default sizes, 248

H

half duplex paper tape reading mode, 225
hardware configuration guidelines, 266
hardware interrupt requirements, 266
hardware options, 272

I

I insert command, 129
I/O
  cancelling, 223
  cleanup, 46
  device type codes, 8
  end action, 47
  interrupt, 290
  interrupt routines, 304
  key-in format, 41
  medium name, 11
  operation, 44
  queue table (IOQ), 248
  queueing, 46
  request, 44
  specifications, 8
  start, 46
  system, 44
  system calls, 54
  translation tables, 307
immediate services, 314
IN insert new records, 201
in-line FPT form, 273
INCLUDE control command, 145
INIT command, 18,227
INIT function call, 84
input
  control commands, 20
  conversion, 308
  interrupt processing, 304
  options, 294
  parameters, 249
  special action table, 307
  symbiont area, 4
  translation table, 308
INT function call, 87

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.

interpret upper case as lower case, 223
interrupt
  control, 79
  inhibits, 290
  label table (INTLB), 249
  switch, 31
interrupt-structure summary, 268
INTLB command, 258
intrarecord editing commands, 205
IOEX access area, 4
IOEX calls, 69
IOEX function status returns, 70
IS insert new records, 202

J

JCP loader, 15,293
JCP messages, 24
JOB, 1,7,80,81
  accounting, 4
  command, 13,227
  concept, 313
  control processor, 12
  function call, 63
  management, 313
  resource management, 316
JTRAP function call, 94
JU jump, 210

K

K:BACKBG, 115
K:BCKEND, 23,115
K:BPEND, 23,115
K:CCBUF, 23
K:PAGE, 23
keyboard-printer edited I/O, 49
KJOB function call, 80

L

LCOMMON command, 147
L look command, 131
L shift left, 208
labeled COMMON, 166
LIB control command, 145
libraries, 152
  constructing, 153
  file sizes, 169
  maintaining, 153
  protection, 153
  public, 4,150,153
LIMIT control command, 17
line editor (see EDIT)
line printer messages/key-ins, 43
linking a program, 138
  memory management aspects, 110-115
linking CP-R system processors, 265
LIST command, 16
LMAP command, 175
LMHDR command, 150
LOAD command, 15
load module, 1
load module inventory (LMI), 248
load-time assigns, 165
loader-generated items, 151
loading
  background programs, 75
  foreground secondary tasks, 74
  primary foreground programs, 75
  programs, 74
  system processors, 262
  user programs, 262
  VOLINIT, 328
LOCK function call, 118
logical device, 10,49

M

M modify command, 129
magnetic tape
  files, 21
  messages/key-ins, 43
main program name and entry, 166
main storage (memory) requirements, 268
MAP command, 174
mapped secondary task, 113,119
master directory, 248
MASTER function call, 102
master mode, 77
MD move and delete records, 203
MDEF command, 256
MEDIA command, 226
  control specification, 136
  conventions, 137
  key-ins, 136,39
  messages, 137
  processing, 134
  service call, 134
memory
  allocation, 238
  layout after SYSGEN, 238
  layout after SYSLOAD, 246
  layout of a program, 167
  management system calls, 115
  organization, 108
  partition, 107
  protection, 5,110
  requirements for CP-R, 268,269
MESSAGE command, 15,226
MK move and keep records, 203
MOD command, 257
MODIFY control command, 148
MODIFY function call, 100

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.

MODIFY patch area, 295
MODIR file, 152
MODULE file, 152
MONITOR command, 250
monitor error and abnormal codes, 323
multiline records, 198
MUST command, 228,220

N

N continue if no, 206
N name command, 130
NO make no change, 210
null device, 10

O

O overwrite, 208
OFF command, 226
on-line
  sessions, 222
  user processors, 220
OPEN system call, 54,55
operating system, 1
operational label table (OPLBS), 249
operational labels, 11,49
operator communications, 26
operator key-in, 31,32
output
  conversion (EBCDIC - USASCII), 311
  interrupt processing, 304
  message formats, 31
  symbiont area, 4
OV file, 4
overlay
  example, 144
  inventory (OVI), 248
  load modules, 138
  loader, 6,138
  loader control commands, 140
  loader diagnostics, 157
  loader examples, 191
  loading, 290
  restrictions, 139
  segment operations, 77
  segments, 151,166
  structure, 7,139
overlays, 5
OVLOAD table, 151

P

P patch command, 130
P precede string, 207
pack initialization, 270
paper tape input, 225
patch
  command formats, 295
  CP-R monitor, 295
  quick, 294
  simulation routine, 295
  system overlay or JCP, 295
  system tables, 295
PAUSE command, 16
PC function call, 87
periodic scheduling, 318
peripheral equipment requirements, 272
peripheral equipment supported, 254,255
PFIL command, 21
PFIL function call, 59
physical device names, 9
PMD command, 20
POLL function call, 90
POST function call, 91
power on/off, 272
PREC command, 21
PREC function call, 59
PREFMODE function call, 119
primary programs, 17,75
primary task control block, 75
primary task memory allocation, 109
PRINT function call, 64
PROC command, 250
processor
  availability, 268
  control commands, 22
  interface with CP-R, 23
program, 1
  control block (PCB), 2,76,151
  deck, 184
  file, 150
  map, 154,155
prompt characters, 222
PUBLIB control command, 149
PUNCH command, 259

Q

Q quit command, 131
queueing primary foreground programs, 74
QUIT command, 228

R

R remove command, 131
R shift right, 208
RAD file table (RFT), 44,55,248
RADEDIT, 6,168
  calling, 170
  commands, 170
  error messages, 179
READ function call, 55,56
real memory allocation, 107
real-time performance data, 290

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.

real-time processing, 2
RECALARM function call, 106
record editing commands, 201
reentrant subroutine, 2
register blocks, 272
registers 8 and 10 (error return), 54,327
releasing
  primary foreground programs, 75
  public library, 153
RELPAGE function call, 117
removable disk packs and cartridges, 270
RES control command, 146
RESERVE command (SYSGEN), 251
reserved pages, 109
reserving I/O devices, 48
resource management, 316
response to interrupts, 290
RESTORE command, 178
restricting input to upper case, 223
RET command, 200
return functions, 78
retyping the current line, 223
REWIND
  command, 21
  function call, 58
  manual command, 22
RF reverse blank preservation mode, 211
RFT (see RAD file table)
RLS function call, 84
RN renumber record, 204
roll-in, 2,119
roll-out, 2,119,120
ROOT control command, 142
root segments, 150
ROV command, 17
RUN command (JCP), 18
RUN command (TEL), 227
RUN function call, 83,315
run-time assigns, 165

S

S command, 131
S substitute string, 207
SAVE command (EDIT), 199
SAVE command (RADEDIT), 178
SCHED function call, 85
scheduling programs, 74
SE set intrarecord mode, 205
secondary
  storage devices, 269,255
  storage requirements, 269,270
  storage utilization, 3
  task, 107,113,119
  task dispatch time, 290
  task memory, 5,109
  task structures, 113
SEG control command, 143
SEGLOAD function call, 102,315
segment
  access protection, 113
  activity, 111
  states, 111,112
SEQ command, 199
sequence numbers, 197
sequential access, 45
SETNAME command, 227
SETNAME function call, 82
setting the tab relative mode, 223
SFIL command, 21
sharing
  DCBs, 46
  disk files, 47
  I/O devices, 47
  segments, 112
SIGNAL function call, 88
simplified memory management (SMM), 115
simulating
  tab characters, 223
  tab stops, 223
SITE command, 259
SJOB command, 18
SJOB function call, 81
skipping bad sectors, 168
SLAVE function call, 102
slave mode, 77
SMAP command, 174
software checks, 320
software segmentation, 110
special code properties, 229
SQUEEZE command, 173
SS set and step, 205
ST set, step, and type record, 205
standard symbols and codes, 229
START command, 228
START function call, 99
STARTIO function call, 72
STATUS function call, 100
STDLB command (JCP), 17
STDLB command (SYSGEN), 258
STDLB command (TEL), 226
STDLB function call, 71
STIMER function call, 104
STOP command, 228
STOP function call, 100
STOPIO function call, 72
subject file format, 197
symbiont file
  allot, 63
  delete, 63
symbiont key-ins, 38
symbionts, 5,291
symbol-code correspondences, 232-235
synchronous services, 314
SYSGEN, 238
SYSGEN control commands, 250
SYSGEN map example, 239-245
SYSGENLOAD, 238

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.

SYSLD command, 259
SYSLOAD, 238,260
system
  alarm procedures, 320
  control commands, 13
  CP-R, 273
  CP-R errors, 274
  CP-R procedure references, 273
  DCBs, 50
  function call, 80
  generation, 238
  initialization, 294
  job inventory (SJI), 249
  library files, 169
  memory, 5
  messages, 31
  operational labels, 9
  patching, 294,296
  processors, 23
  program area, 3
  task inventory (STI), 249
  virtual memory, 109

T

TABS command, 226
task, 1
  control block (TCB), 2,75
  management, 314
  scheduling and operation, 74
  status format, 41
  virtual memory, 109
TC type compressed, 202
TEL commands, 226
Teletype services (summary), 225
Teletype terminal keyboard, 221
temp stacks, 2,151
TERM function call, 95
terminal executive language (TEL), 220,225
terminal job entry (TJE), 220
  account maintenance, 228
  escape characters, 225
  key-ins, 37
  system, 2
terminal operations, 220
terminating foreground secondary tasks, 74
terminating lines, 223
TEST function call, 124
TEXIT function call, 95
TIME function call, 103
trace command formats, 295
TRAP function call, 94
trap handler messages, 26
trap handling, 78
TRIGGER function call, 99
triggering of interrupts, 79
TRTN function call, 95
TRTY function call, 95
TRUNCATE command, 174
TRUNCATE function call, 62
TS type record, without sequence number, 203,210
TY type record, with sequence number, 202,210
type completion codes (TYC), 322-327
TYPE function call, 64
Type I request, 44,46,56
Type II requests, 44,56
typing ahead, 223
typing commands, 224
typing error correction, 31
typing lines, 222

U

unit designation codes, 8
UNLOAD
  command, 22
  function call, 58
UNLOCK function call, 118
user
  areas, 3
  library files, 169
  temp stack, 77
utility control commands, 21

V

vertical format control (VFC), 52
virtual memory allocation, 109
VOLINIT commands, 328
volume initialization, 328

W

W command, 131
WAIT function call, 103
WAITALL function call, 123
WAITANY function call, 124
WEOF command, 22
WEOF function, 58
WRITE function call, 56
write-key, 5
write-lock, 5

X

XDMP command, 177
Xerox standard compressed language, 289

Y

Y continue if yes, 206

XEROX

Reader Comment Form

We would appreciate your comments and suggestions for improving this publication

Publication No.	Rev. Letter	Title	Current Date

How did you use this publication?

☐ Learning ☐ Installing ☐ Sales
☐ Reference ☐ Maintaining ☐ Operating

Is the material presented effectively?

☐ Fully Covered ☐ Well Illustrated ☐ Well organized ☐ Clear

What is your overall rating of this publication?

☐ Very Good ☐ Fair ☐ Very Poor
☐ Good ☐ Poor

What is your occupation?

Your other comments may be entered here. Please be specific and give page, column, and line number references where applicable. To report errors, please use the Xerox Software Improvement or Difficulty Report (1188) instead of this form.

Your name & Return Address

Thank You For Your Interest. (fold & fasten as shown on back, no postage needed if mailed in U.S.A.)

PLEASE FOLD AND TAPE –
NOTE: U. S. Postal Service will not deliver stapled forms

First Class
Permit No. 59153
Los Angeles, CA

BUSINESS REPLY MAIL
No postage stamp necessary if mailed in the United States

Postage will be paid by

Honeywell Information Systems
5250 W. Century Boulevard
Los Angeles, CA 90045

Attn: Programming Publications

Fold