Gould MPX-32™

Release 3.3

Reference Manual

Volume III

Installation and System Administration

December 1986

Publication Order Number: 323-001553-300

GOULD
Electronics

HISTORY

The MPX-32 Release 3.0 Reference Manual, Publication Order Number **323-001550-000,** was printed June, 1982.

Publication Order Number **323-001553-100** (Revision 1, Release 3.2) was printed June, 1983.

Publication Order Number **323-001553-200** (Revision 2, Release 3.2B) was printed March, 1985.

Publication Order Number **323-001553-201** (Change 1 to Revision 2, Release 3.2C) was printed December, 1985.

Publication Order Number **323-001553-300** (Revision 3, Release 3.3) was printed December, 1986.

Note: For Release 3.3 of MPX-32, the reference material formerly included at the back of each MPX-32 Reference Manual volume was placed in a separate document and assigned Publication Order Number **323-001550-300.**

This manual contains the following pages:

Title page
Copyright page
iii/iv through xvi
1-1 through 1-2
2-1 through 2-14
3-1 through 3-5/3-6
4-1 through 4-7/4-8
5-1 through 5-3/5-4
6-1 through 6-4
7-1 through 7-37/7-38
8-1 through 8-30
9-1 through 9-10
10-1 through 10-44
11-1 through 11-13/11-14
12-1 through 12-16
13-1 through 13-14

CONTENTS

Page

CHAPTER 1 BUILDING AND MAINTAINING THE MPX-32 SYSTEM - AN OVERVIEW

1. Building and Maintaining the MPX-32 System - an Overview 1-1

CHAPTER 2 INSTALLING A STARTER SYSTEM

2.1 Hardware Configuration 2-1
2.2 The Master System Distribution Tape (SDT) 2-2
2.2.1 Magnetic Tape 2-2
2.2.2 Floppy Disc 2-2
2.2.3 Utility Tape 2-7
2.3 Booting a System from the Master SDT 2-8
2.3.1 Control Switches 2-12
2.4 Philosophy of Bootstrapping 2-13
2.5 The System Builder 2-13
2.6 Operating Under the Starter System 2-14
2.7 Restoring Utility Processors, Libraries, and Other Files 2-14
2.8 Example First Use of Master SDT on User System 2-15

CHAPTER 3 BUILDING AND TESTING A USER-CONFIGURED SYSTEM

3.1 Building the SYSGEN Input Files 3-1
3.1.1 Building the Directive Input File 3-1
3.1.2 Building the Object Input File 3-1
3.1.3 The COMPRESS Task 3-2
3.2 Running SYSGEN 3-3
3.3 Testing a SYSGENed System 3-3
3.4 Terminal Initialization and System Protection 3-4

CHAPTER 4 INSTALLING A USER-CONFIGURED SYSTEM

4.1 Creating a User System Distribution Tape (SDT) 4-1
4.2 SDT Directive 4-1
4.3 Installing a User System Distribution Tape (SDT) 4-3
4.4 Saving/Restoring System Processor and Utility Load Modules 4-3
4.5 Booting a System from a User SDT 4-4
4.5.1 Example 4-7

CHAPTER 5 ON-LINE RESTART

5.1 Precautions 5-1
5.2 Restart Syntax 5-3

CHAPTER 6 - RECOVERING THE SYSTEM

6.1 Recovery from Disc at the IOP Console 6-1
6.2 Errors During Start-up 6-1
6.3 System Halt Analysis 6-2
6.4 Automatic IPL 6-4

CHAPTER 7 SYSTEM GENERATION (SYSGEN)

7.1 General Description 7-1
7.2 SYSGEN Logical File Codes 7-2
7.2.1 LFC Summary 7-3
7.3 Options 7-3
7.4 Accessing SYSGEN 7-3
7.5 SYSGEN Directives 7-4
7.5.1 ACTIVATE Directive 7-8
7.5.2 ARTSIZE Directive 7-9
7.5.3 BATCHPRI Directive 7-9
7.5.4 /CHANNELS Directive 7-9
7.5.5 CDOTS Directive 7-9
7.5.6 CONTROLLER Directive 7-10
7.5.7 DBGFILE Directive 7-12
7.5.8 DEBUGTLC Directive 7-12
7.5.9 DELTA 7-13
7.5.10 DEVICE Directive 7-13
7.5.11 DISP Directive 7-16
7.5.12 DPTIMO Directive 7-16
7.5.13 DPTRY Directive 7-17
7.5.14 DTSAVE Directive 7-17
7.5.15 //END Directive 7-17
7.5.16 EXTDMPX Directive 7-17
7.5.17 /FILES Directive 7-18
7.5.18 FLTSIZE Directive 7-18
7.5.19 //HARDWARE Directive 7-18
7.5.20 /INTERRUPTS Directive 7-18
7.5.21 IPU Directive 7-18
7.5.22 ITIM Directive 7-19
7.5.23 ITLB Directive 7-19
7.5.24 JOBS Directive 7-19
7.5.25 KTIMO Directive 7-19
7.5.26 LOD Directive 7-20
7.5.27 MACHINE Directive 7-20
7.5.28 /MEMORY Directive 7-20
7.5.29 MDT Directive 7-20
7.5.30 MMSG Directive 7-21
7.5.31 MNWI Directive 7-21
7.5.32 MODE Directive 7-21
7.2.33 MODULE Directive 7-22
7.5.34 /MODULES Directive 7-22
7.5.35 MRUN Directive 7-23
7.5.36 MTIM Directive 7-23
7.5.37 NAME Directive 7-23
7.5.38 NOBASE Directive 7-23
7.5.39 NOCMS Directive 7-24

7.5.40 NTIM Directive 7-24
7.5.41 OTHERS Directive 7-24
7.5.42 /OVERRIDE Directive 7-24
7.5.43 OWNER Directive 7-25
7.5.44 /PARAMETERS Directive 7-25
7.5.45 /PARTITION Directive 7-25
7.5.46 PASSWORD Directive 7-25
7.5.47 PATCH Directive 7-26
7.5.48 PCHFILE Directive 7-26
7.5.49 POD Directive 7-26
7.5.50 POOL Directive 7-26
7.5.51 PRIORITY Directive 7-27
7.5.52 PROGRAM Directive 7-27
7.5.53 PROJECT Directive 7-28
7.5.54 /RMSTABLS Directive 7-28
7.5.55 RMTSIZE Directive 7-28
7.5.56 SEQUENCE Directive 7-29
7.5.57 SGOSIZE Directive 7-29
7.5.58 SHARE Directive 7-29
7.5.59 SID Directive 7-30
7.5.60 SIZE Directive 7-30
7.5.61 SMD Directive 7-31
7.5.62 //SOFTWARE Directive 7-32
7.5.63 SVC Directive 7-32
7.5.64 SWAPLIM Directive 7-32
7.5.65 SWAPSIZE Directive 7-32
7.5.66 SWP Directive 7-32
7.5.67 SYCSIZE Directive 7-33
7.5.68 SYMTAB Directive 7-33
7.5.69 /SYSDEVS Directive 7-33
7.5.70 SYSMOD Directive 7-33
7.5.71 SYSTEM Directive 7-34
7.5.72 SYSTRAP Directive 7-34
7.5.73 /TABLES Directive 7-34
7.5.74 TERMPRI Directive 7-34
7.5.75 TIMER Directive 7-35
7.5.76 TITLE Directive 7-35
7.5.77 TQFULL Directive 7-35
7.5.78 TQMIN Directive 7-35
7.5.79 TRACE Directive 7-35
7.5.80 /TRAPS Directive 7-36
7.5.81 TYPE Directive 7-36
7.5.82 USERPROG Directive 7-36
7.5.83 /VP Directive 7-36
7.5.84 VP Directive 7-36
7.5.85 VPID Directive 7-37

CHAPTER 8 SYSTEM DEBUGGER (H.DBUG)

8.1 Using the Debugger 8-1
8.1.1 Arithmetic and Special Operators 8-1
8.1.2 Special Functions 8-2
8.1.3 Execution Breakpoints 8-3
8.1.4 Debugger Bases 8-3
8.1.5 Base Characters 8-3

8.1.6 Operator Restrictions 8-4
8.1.7 Expressions 8-4
8.1.8 Registers 8-4
8.1.9 Indirection 8-4
8.2 Accessing the Debugger 8-5
8.3 Debugger Directives 8-5
8.3.1 AB (Absolute) Directive 8-8
8.3.2 AD (Address) Directive 8-8
8.3.3 AR (Arithmetic) Directive 8-8
8.3.4 AS (Assemble Instruction) 8-9
8.3.5 BA (Base) Directive 8-9
8.3.6 BR (Breakpoint) Directive 8-9
8.3.7 BY (Bye) Directive 8-10
8.3.8 CB (Change Base Register) Directive 8-10
8.3.9 CD (Display Command List) Directive 8-10
8.3.10 CE (Zero Command List) Directive 8-10
8.3.11 CH (Display Controller Definition Table Entry) Directive 8-10
8.3.12 CL (Terminate Build Command List Mode) Directive 8-11
8.3.13 CM (Change Memory) Directive 8-11
8.3.14 CO (Continue) Directive 8-11
8.3.15 CR (Change Register) Directive 8-11
8.3.16 CS (Build Command List) Directive 8-12
8.3.17 CT (Continue then Terminate) Directive 8-12
8.3.18 CX (Execute Directive List) Directive 8-12
8.3.19 DB (Display Base Register) Directive 8-12
8.3.20 DE (Delete) Directive 8-12
8.3.21 DI (Display Instruction) Directive 8-13
8.3.22 DM (Display Memory) Directive 8-13
8.3.23 DQ (Display Dispatch Queue Entry) Directive 8-13
8.3.24 DR (Display Register) Directive 8-14
8.3.25 DS (Display Symbolic) Directive 8-14
8.3.26 DT (Display Event Trace) Directive 8-14
8.3.27 DU (Dump) Directive 8-15
8.3.28 EC (Echo) Directive 8-15
8.3.29 ET (Enter Event Trace Point) Directive 8-15
8.3.30 GO (Go) Directive 8-16
8.3.31 HC (Display Dispatch Queue Head Cell) Directive 8-16
8.3.32 LB (List Breakpoint) Directive 8-16
8.3.33 LP (Line Printer) Directive 8-16
8.3.34 LT (List Mobile Event Trace Point) Directive 8-17
8.3.35 MR (Map Register) Directive 8-17
8.3.36 MS (Modify CPU Scratchpad Location) Directive 8-17
8.3.37 PD (Display Patch List) Directive 8-18
8.3.38 PE (Zero Patch List) Directive 8-18
8.3.39 PR (Terminate Build Patch List Mode) Directive 8-18
8.3.40 PS (Program Status) Directive 8-18
8.3.41 PT (Build Patch List) Directive 8-18
8.3.42 PX (Execute Patch List) Directive 8-19
8.3.43 RE (Remap) Directive 8-19
8.3.44 RT (Remove Event Trace Point) Directive 8-19
8.3.45 SE (Search Equivalent) Directive 8-20
8.3.46 SM (Set Mask) Directive 8-20
8.3.47 SP (Scratchpad Dump) Directive 8-20
8.3.48 SY (Symbolic) Directive 8-20

8.3.49 TB (Trace Back) Directive 8-21
8.3.50 TE (Terminate) Directive 8-21
8.3.51 TR (Trace) Directives 8-21
8.3.52 TS (Trace Stop) Directive 8-22
8.3.53 TY (Terminal) Directive 8-22
8.3.54 UD (Display Unit Definition Table Entry) Directive 8-22
8.4 System Debugger Practice Debug Session 8-22
8.4.1 Step One - Accessing the Debugger 8-23
8.4.2 Step Two - Task Debugging with the System Debugger 8-23
8.4.3 Using the System Debugger to Display Memory 8-23
8.4.4 Using Debug to Display a Program 8-26
8.4.5 Summary 8-28
8.5 Example of Directive List Use 8-28
8.6 Example of Patch List Use 8-29

CHAPTER 9 ON-LINE SYSTEM PATCH FACILITY (J.INIT)

9.1 General 9-1
9.1.1 Dedicated Names 9-2
9.1.2 Conventions 9-2
9.2 Patch Directives 9-2
9.2.1 /B (Define a Base Address) Directive 9-3
9.2.2 /C (Change the Contents of a Memory Location) Directive 9-3
9.2.3 /D (Define a Named Value) Directive 9-4
9.2.4 /E (Exit) Directive 9-4
9.2.5 /G (Go to the Patch Area from a Specified Memory Location) Directive 9-5
9.2.6 /O (Select Patch Options) Directive 9-5
9.2.7 /P (Define a Patch Area) Directive 9-6
9.2.8 /R (Return from the Patch Area) Directive 9-6
9.2.9 /F, /N, /T (Conditional) Directive 9-6
9.2.10 /$ (Enter a Value into the Patch Area) Directive 9-7
9.2.11 /; (Comments) Directive 9-8
9.3 Entry Conditions 9-8
9.4 Exit Conditions 9-8
9.5 External References 9-8
9.6 Examples 9-10
9.7 Automatic Mounting of Public Volumes 9-10

CHAPTER 10 SYSTEM ADMINISTRATOR SERVICES

10.1 M.KEY Editor (KEY) 10-2
10.1.1 Using KEY 10-3
10.1.2 Directives 10-3
10.1.3 Examples of Directive Usage 10-11
10.2 M.PRJCT File 10-11
10.2.1 Using the PROJECT Program 10-12
10.2.2 Directives 10-12
10.3 Examples of Directive Usage 10-14
10.4 M.ACCNT File 10-15
10.5 M.CNTRL File 10-17
10.6 M.ERR File and xx.ERR Files 10-17
10.6.1 Creating xx.ERR File 10-17
10.7 Terminal/ALIM/ASYNCH Initialization (INIT) 10-19
10.7.1 The LOGONFLE 10-19

10.7.2 ALIM Terminal Record Syntax and Defaults 10-21
10.7.3 ACM Controller Record Syntax and Defaults 10-28
10.7.3.1 True Full-Duplex Operation for the ACM 10-31
10.7.4 Sample LOGONFLE 10-31
10.8 Using INIT 10-32
10.9 INIT Errors 10-32
10.10 The System Console (Messages) 10-32
10.10.1 Information Messages 10-33
10.10.2 Action Messages 10-33
10.10.3 At the Terminal (Messages) 10-33
10.11 Floppy Disc Media Initialization (J.FORMF) 10-33
10.12 M.MOUNT File 10-34
10.13 Operator Intervention Inhibit 10-34
10.14 System Date/Time Backup Program (J.DTSAVE) 10-35
10.15 Swap Scheduler Control Options 10-35
10.16 Multiprocessor Recovery Task (J.UNLOCK) 10-38
10.17 Shadow Memory 10-38
10.18 Memory Disc 10-41
10.18.1 General Information 10-41
10.18.2 Memory Disc Configuration 10-42
10.18.3 Memory Disc Usage 10-42
10.18.4 Memory Disc Aborts and Errors 10-43
10.18.5 Memory Disc Save Task (J.MDSAVE) 10-43
10.18.6 Memory Disc Restore Task (J.MDREST) 10-44
10.19 Label ANSI Tape Utility (J.LABEL) 10-44

CHAPTER 11 DEVICE INITIALIZER/LOADER (DEVINITL)

11.1 Initialization Directive File 11-1
11.2 Firmware File 11-1
11.2.1 Loader Data Record Format 11-3
11.3 Using DEVINITL 11-4
11.4 DEVINITL Directives 11-4
11.4.1 DEV_CNTRL Directive 11-6
11.4.2 DEVICE Directive 11-7
11.4.3 IDENT Directive 11-7
11.4.4 OPTION Directive 11-8
11.4.5 REREAD Directive 11-9
11.4.6 RETRY Directive 11-9
11.4.7 WCS_FILE Directive 11-10
11.5 Example Initialization Directive File 11-10
11.6 Milestone Messages 11-11
11.7 Error Conditions and Messages 11-11
11.7.1 Initialization Directive File Errors 11-11
11.7.2 Device Loading Errors 11-11
11.7.3 Directive Parsing Errors 11-12

CHAPTER 12 ALTERABLE CONTROL STORE (ACS) LOAD AND DISPLAY UTILITY

12.1 LOADACS Directive File (M.ACS) 12-1
12.2 Firmware File 12-2
12.2.1 CONCEPT 32/67 Usage 12-2
12.2.2 Firmware File Record 12-2
12.3 LOADACS 12-3
12.4 DUMPACS 12-4
12.5 ACS Directives 12-4

12.5 ACS Directives 12-4
12.5.1 CHECKSUM Directive 12-5
12.5.2 COMPARE Directive 12-6
12.5.3 COPY Directive 12-6
12.5.4 DUMP Directive 12-7
12.5.5 ENABLE Directive 12-8
12.5.6 EXIT Directive 12-8
12.5.7 LOAD Directive 12-8
12.5.8 MODE Directive 12-10
12.5.9 PATCH Directive 12-10
12.5.10 REVISION Directive 12-11
12.5.11 VERIFY Directive 12-12
12.6 Sample M.ACS File 12-12
12.7 Error Conditions and Messages 12-13
12.7.1 LOADACS Error Conditions and Messages 12-13
12.7.2 DUMPACS Error Conditions and Messages 12-15

CHAPTER 13 VOLUME FORMATTER (J.VFMT)

13.1 General Description 13-1
13.2 Logical File Code Assignments 13-1
13.2.1 Audit Trail (SLO) 13-1
13.2.2 Directive Input (SYC) 13-2
13.3 Using the Volume Formatter 13-2
13.3.1 Directive Syntax Rules 13-3
13.3.1.1 Verb 13-3
13.3.1.2 Parameters 13-3
13.3.1.3 Options 13-3
13.3.2 Directive Line Continuations 13-3
13.4 Accessing the Volume Formatter 13-4
13.5 Volume Formatter Directive Verbs 13-4
13.5.1 EXIT Directive 13-4
13.5.2 FORMAT Directive 13-5
13.5.3 NEWBOOT Directive 13-7
13.5.4 REPLACE Directive 13-8
13.6 Volume Formatter Directive Parameters 13-8
13.6.1 DEVICE Parameter 13-8
13.6.2 VOLUME Parameter 13-9
13.6.3 BOOTFILE Parameter 13-9
13.7 Volume Formatter Directive Options 13-9
13.7.1 ACCESS Option 13-10
13.7.2 ALOC Option 13-10
13.7.3 CONFIRM Option 13-11
13.7.4 DETACH Option 13-11
13.7.5 ECC Option 13-11
13.7.6 IMAGE Option 13-12
13.7.7 ISIZE Option 13-12
13.7.8 MAXAU Option 13-12
13.7.9 MAXRES Option 13-12
13.7.10 MAXROOT Option 13-13
13.7.11 OWNER Option 13-13
13.7.12 PROJECTGROUP Option 13-13
13.8 Errors 13-13
13.9 Examples 13-14

FIGURES

Page

1-1 MPX-32 Installation/Configuration Overview 1-2

2-1 Master System Distribution Tape Format 2-7

3-1 Testing a User-Configured System 3-5

4-1 User System Distribution Tape Format 4-2

5-1 Establishing a New Default System 5-2

10-1 Shadow Memory Configuration with a Single Processor and a Single Region of Shadow Memory 10-38
10-2 Shadow Memory Configuration with Two Processors and a Single Region of Shadow Memory 10-39
10-3 Shadow Memory Configuration with Two Processors and Two Regions of Shadow Memory 10-39
10-4 Memory Disc 10-44

11-1 Initialization Directive File Format 11-2

12-1 M.ACS Format 12-1

TABLES

Page

2-1 Deliverable Software for MPX-32 2-3

7-1 MPX-32 Device Type Codes 7-11
7-2 Device Handlers 7-12
7-3 Disc Device Codes 7-16

11-1 Valid Control Record Types 11-3

13-1 Volume Formatter Logical File Code Assignments 13-2
13-2 Allocating Units 13-7

Documentation Conventions

Notation conventions used in directive syntax and message examples throughout this manual are described below.

lowercase letters

In directive syntax, lowercase letters identify a generic element that must be replaced with a value. For example,

```
!ACTIVATE taskname
```

means replace taskname with the name of a task, e.g.,

```
!ACTIVATE DOCCONV
```

In messages, lowercase letters identify a variable element. For example,

```
**BREAK** ON:taskname
```

means a break occurred on the specified task.

UPPERCASE LETTERS

In directive syntax, uppercase letters specify a keyword must be entered as shown for input, and is printed as shown in output. For example,

```
SAVE filename
```

means enter SAVE followed by a file name, e.g.,

```
SAVE DOCCONV
```

In messages, uppercase letters specify status or information. For example,

```
taskname,taskno ABORTED

*YOUR TASK IS IN HOLD. ENTER CONTINUE TO RESUME IT
```

Braces { }

Elements placed one under the other inside braces specify a required choice. You must enter one of the arguments from the specified group. For example,

$$\begin{Bmatrix}\text{counter}\\ \text{startbyte}\end{Bmatrix}$$

means enter the value for counter or startbyte.

Brackets [ ]

An element inside brackets is optional. For example,

```
[CURR]
```

means the term CURR is optional.

Items placed one under the other within brackets specify you can enter one of the group of options or none at all. For example,

$$\begin{bmatrix}\text{base name}\\ \text{progname}\end{bmatrix}$$

means enter the base name or the program name or neither.

Items in brackets within encompassing brackets specify one item is required only when the other item is used. For example,

```
TRACE [lower address [upper address] ]
```

means both the lower address and the upper address are optional, and the lower address can be used alone. However, if the upper address is used, the lower address must also be used.

Commas between multiple brackets within an encompassing set of brackets are semi-optional; that is, they are not required unless subsequent elements are selected. For example,

```
M.DFCB fcb,lfc [, [a] , [ b ] , [ c ] , [ d ] , [ e ] ]
```

could be coded as

```
M.DFCB FCB12,IN

   or

M.DFCB FCB12,IN,,ERRAD

   or

M.DFCB FCB13,OUT,,ERAD,,PCK
```

Horizontal Ellipsis ...

The horizontal ellipsis indicates the previous element can be repeated. For example,

```
name [,name]...
```

means one or more names separated by commas can be entered.

Vertical Ellipsis

```
.
.
.
```

The vertical ellipsis specifies directives, parameters, or instructions have been omitted. For example,

```
COLLECT 1
  .
  .
  .
LIST
```

means one or more directives have been omitted between the COLLECT and LIST commands.

Numbers and Special Characters

In a syntax statement, any number, symbol, or special character must be entered as shown. For example,

```
(value)
```

means enter the proper value enclosed in parentheses; e.g., (234).

Underscore

In syntax statements, underscoring specifies the letters, numbers or characters that may be typed by the user as an abbreviation. For example,

```
ACTIVATE  taskname
```

means spell out the directive verb ACTIVATE or abbreviate it to ACTI.

```
RESET
```

means type RESET or RST.

In examples, all terminal input is underscored; terminal output is not. For example,

```
TSM > EDIT
```

means TSM > was written to the terminal; EDIT is typed by the user.

CHAPTER 1

BUILDING AND MAINTAINING THE MPX-32 SYSTEM - AN OVERVIEW

MPX-32 uses system utilities such as the Volume Manager and Text Editor to provide mechanisms for building and maintaining resident operating systems. A resident system is configured by running the System Generator utility, SYSGEN. A tailored system is configured by modifying the Master System Distribution Tape (SDT) and SYSGEN file before installation. A System Debugger is supp ed that can debug a resident operating system or resident user-developed interrupt and device handlers.

This volume provides documentation on installation from a System Distribution Tape (SDT) and Utility Tape, SYSGEN, on-line and IOP console restart capability, the System Debugger. the System Patch facility, System Administrator Services, the Device Initializer/Loader, the Alterable Control Store Load and Display utility, and the Volume Formatter.

The MPX-32 operating system supports floppy disc usage. All references to the System Distribution Tape (SDT) apply whether the distribution medium is magnetic tape or floppy disc.

Figure 1-1 provides an overview of installation and configuration as described in Chapters 2 through 4.

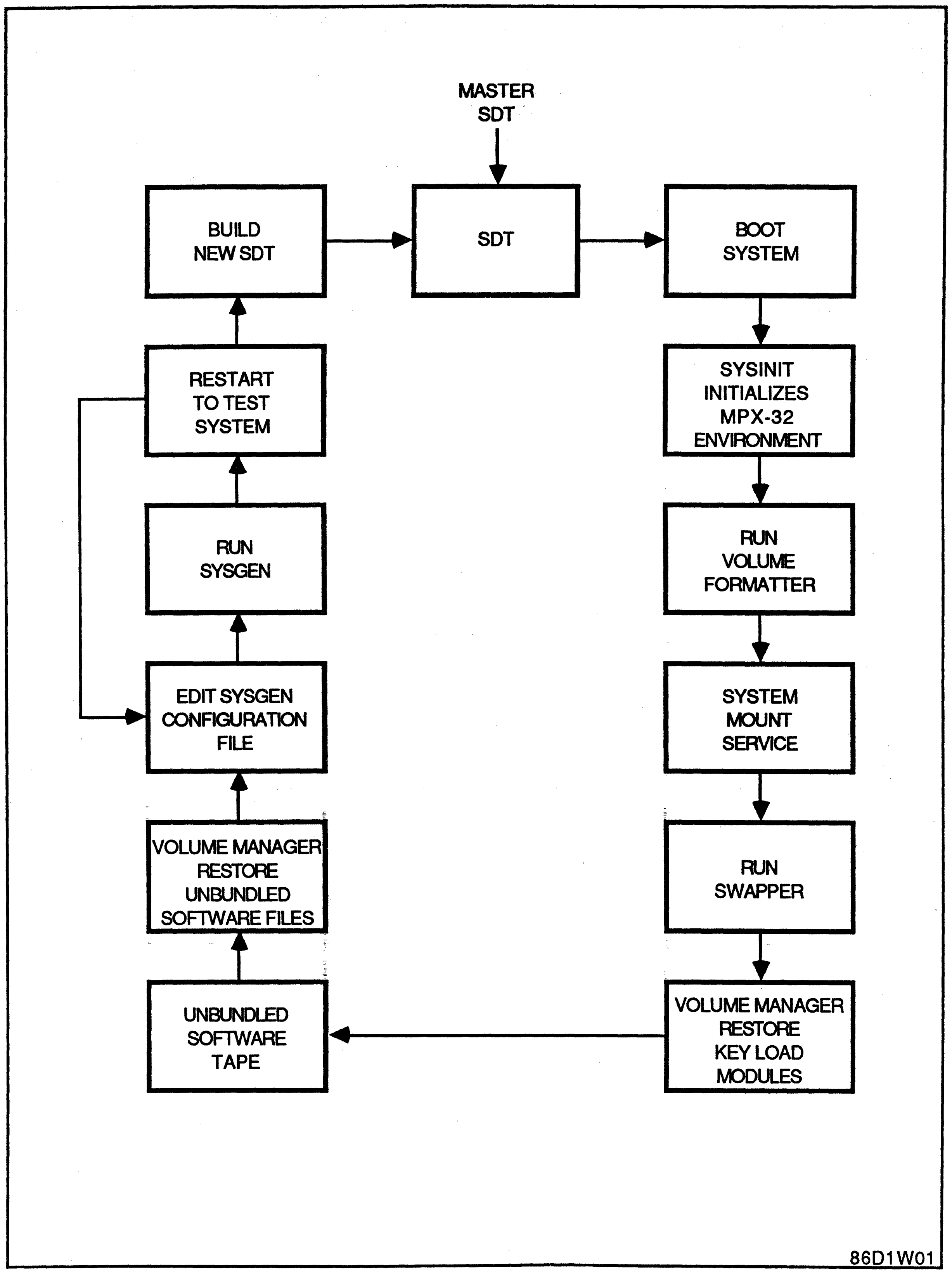

MASTER SDT
BUILD NEW SDT
SDT
BOOT SYSTEM
RESTART TO TEST SYSTEM
SYSINIT INITIALIZES MPX-32 ENVIRONMENT
RUN SYSGEN
RUN VOLUME FORMATTER
EDIT SYSGEN CONFIGURATION FILE
SYSTEM MOUNT SERVICE
VOLUME MANAGER RESTORE UNBUNDLED SOFTWARE FILES
RUN SWAPPER
UNBUNDLED SOFTWARE TAPE
VOLUME MANAGER RESTORE KEY LOAD MODULES
86D1W01

Figure 1-1. MPX-32 Installation/Configuration Overview

CHAPTER 2

INSTALLING A STARTER SYSTEM

Starter systems are supplied on the Master System Distribution Tape (SDT). This chapter descr'bes the minimum hardware configuration supported by the starter system and the format of the Master SDT. It also includes an example of booting the starter system.

2.1 Hardware Configuration

The following hardware and logical addresses are used for installing the starter system on a CONCEPT/32 computer:

	Software	Hardware	
		Channel	Subaddress
128KW Memory			
XIO Magnetic Tape (Class F) -or-	xxxx	xx	xx
IOP Floppy Disc	7EF0	7E	F0
IOP Console	7EFC	7E	FC
IOP Line Printer	7EF8	7E	F8
Disc Drive (XIO) (Class F)	xxxx	xx	xx
Disc Drive	7Exx	7E	xx

User-definable addresses are indicated by x.

Only the system console is configured in the starter system. User terminals are configured by the SYSGEN utility. Once terminals have been configured, they can be initialized by the system module J.TINIT. The following parameters must be specified until a terminal initialization file named LOGONFLE is created by the System Administrator:

Wakeup Character - ?
Baud Rate - 9600
Parity - EVEN
Character Size - 7
ALIM Only - HALF DUPLEX

For details on creating a LOGONFLE file, see Chapter 10.

The following discs are supported by MPX-32 and can be used when booting from the Master SDT:

Disc	Code
1.2MB Floppy Disc - Class F Device	FL001
40MB Moving Head Disc - Class F Device*	MH040
80MB Moving Head Disc - Class F Device*	MH080
160MB Moving Head Disc - Class F Device*	MH160

Disc	Code
300MB Moving Head Disc - Class F Device*	MH300
600MB Moving Head Disc - Class F Device	MH600
5MB Fixed Head Disc - Class F Device*	FH005
32MB Cartridge Module Disc - Class F Device*	ÇD032
Any Nonfloppy Disc - Class F Device*	ANY

* Disc Code applies to both IOP discs and disc processors.

If a disc code is specified and a mismatch with the drive occurs, an error results. If disc code ANY is specified with a nonfloppy disc, a mismatch error cannot occur since the relevant operating system table entries are modified to reflect the drive.

2.2 The Master System Distribution Tape (SDT)

Figure 2-1 shows the format of the Master SDT. Table 2-1 lists the software included on the Master SDT.

The system initialization (SYSINIT) process (see Section 2.5) distinguishes between a Master and User SDT by the system name stored in the communications region of the MPX-32 operating system (C.SYSTEM). Therefore, the following file names are reserved for the system:

Magnetic Tape	Floppy Disc
MSTR.27	FLOP.SYS
MSTR.75	
MSTR.87	

MSTR.75 is a null file included for compatibility purposes only.

When MSTR.27 or MSTR.87 is detected by SYSINIT, the processing of the system images for a master SDT boot is performed. Therefore, these files should not be modified.

Following the system images are a group of saved files. These files include all essential load modules to support a fully operational MPX-32 system. Also included are the object files required to SYSGEN an MPX-32 system tailored to individual hardware and software requirements.

2.2.1 Magnetic Tape

The format of the magnetic tape Master SDT is similiar to the User SDT. Both contain a tape boot loader followed by an MPX-32 image. The master tape, however, contains three MPX-32 images instead of one (one null image for the 32/7x computers, one image for the 32/27 computer, and one image for the 32/67, 32/87, and 32/97 computers). When building the Master SDT, the Volume Manager sets a flag enabling the tape boot loader to recognize a master boot and skip to the proper image for the appropriate CPU.

2.2.2 Floppy Disc

The format of the floppy disc Master SDT is the same as the User SDT. It contains a boot loader followed by the MPX-32 image. When building the Master SDT, the Volume Manager sets a flag enabling the boot loader to recognize a master boot for the CPU.

Table 2-1
Deliverable Software for MPX-32 (Page 1 of 4)

OPERATING SYSTEM MODULES

The following modules are memory resident:

Source File Name	Object File Name	Callable Name	Description
SH.ACBA	OH.ACBA	H.ACBA	Vector Processor SVC
SH.ALOC	OH.ALOC	H.ALOC	Resource Allocation (Compatible Interface)
SH.BKDM	OH.BKDM	H.BKDM	Blocked Data Management Module
SH.DBUG1	OH.DBUG1	H.DBUG1	Mapped Portion of System Debugger (for resident O.S.)
SH.DBUG2	OH.DBUG2	H.DBUG2	Unmapped portion of System Debugger (for resident O.S.)*
SH.DMPMT	OH.DMPMT	H.DMPMT	Stand-alone Dump Writer
SH.EXEC	OH.EXEC	H.EXEC	Executive (CPU Scheduler)
SH.EXEC2	OH.EXEC2	H.EXEC2	Optional Executive (CPU Scheduler)
SH.FISE	OH.FISE	H.FISE	File System (Compatible Interface)
SH.IOCS	OH.IOCS	H.IOCS	Input/Output Control System
SH.MDT	OH.MDT	H.MDT	Rapid File Access Module
SH.MEMM	OH.MEMM	H.MEMM	Memory Management Module
SH.MEMM2	OH.MEMM2	H.MEMM2	Optional Memory Management Module
SH.MONS	OH.MONS	H.MONS	Monitor Services (Compatible Interface)
SH.MVMT	OH.MVMT	H.MVMT	Multivolume Magnetic Tape Management
SH.REMM	OH.REMM	H.REMM	Resource Management Module
SH.REXS	OH.REXS	H.REXS	Resident Executive Services
SH.SINIT	OH.SINIT	H.SINIT	System Initializer
SH.SWAPR	OH.SWAPR	H.SWAPR	Swapper (Resident)
SH.TAMM	OH.TAMM	H.TAMM	Task Management Module
SH.TSM	OH.TSM	H.TSM	Terminal Service Manager
SH.VOMM	OH.VOMM	H.VOMM	Volume Management Module

* Remains physically memory resident but is not included in logical address space as part of the system map.

INTERRUPT AND TRAP HANDLERS

The following routines are memory resident:

Source File Name	Object File Name	Callable Name	Description
SH.CALM	OH.CALM	H.CALM	Optional Calm Replacement SVC Trap Processor
SH.CPU	OH.CPU	H.CPU	IPU To CPU Trap Processor (IPU Task Scheduler)
SH.CPU2	OH.CPU2	H.CPU2	Optional IPU to CPU Trap Processor (IPU Task Scheduler)
SH.ICP	OH.ICP	H.ICP	Indirectly Connected Interrupt Program
SH.IP00	OH.IP00	H.IP00	Power Fail Trap Processor
SH.IP02	OH.IP02	H.IP02	Memory Parity Trap Processor
SH.IP03	OH.IP03	H.IP03	Nonpresent Memory Trap Processor
SH.IP04	OH.IP04	H.IP04	Undefined Instruction Trap Processor
SH.IP05	OH.IP05	H.IP05	Privilege Violation Trap Processor
SH.IP06	OH.IP06	H.IP06	SVC Trap Processor
SH.IP07	OH.IP07	H.IP07	Machine Check Trap Processor
SH.IP08	OH.IP08	H.IP08	System Check Trap Processor
SH.IP09	OH.IP09	H.IP09	MAP Fault Trap Processor
SH.IP0C	OH.IP0C	H.IP0C	Address Specification Trap Processor (32/27, 32/87, 32/97)
SH.IP0F	OH.IP0F	H.IP0F	Arithmetic Exception Trap Processor
SH.IP10	OH.IP10	H.IP10	Cache Memory Parity Error Trap Processor
SH.IP13	OH.IP13	H.IP13	Attention Interrupt Processor
SH.IPAS	OH.IPAS	H.IPAS	System Auto-start Trap Processor
SH.IPCL	OH.IPCL	H.IPCL	Real-time Clock Interrupt Processor
SH.IPHT	OH.IPHT	H.IPHT	CPU Halt Trap Handler
SH.IPIT	OH.IPIT	H.IPIT	Interval Timer Interrupt Processor
SH.IPU	OH.IPU	H.IPU	IPU Executive Trap Processor
SH.IPUAS	OH.IPUAS	H.IPUAS	IPU Power Up Auto Start Trap Processor
SH.IPUIT	OH.IPUIT	H.IPUIT	IPU Accounting Interval Timer Processor
SH.IPVP	OH.IPVP	H.IPVP	Vector Processor Interrupt Handler

Table 2-1
Deliverable Software for MPX-32 (Page 2 of 4)

DEVICE HANDLERS

The following handlers are memory resident:

Source File Name	Object File Name	Load Module	Description
SH.ASMP	OH.ASMP	H.ASMP	ALIM (ASYNC) - GPMC
SH.BSMP	OH.BSMP	H.BSMP	BLIM (BI-SYNC) - GPMC
SH.CALM	OH.CALM	H.CALM	Calm Emulation Trap Handling
SH.CPMP	OH.CPMP	H.CPMP	Card Reader/Punch - GPMC
SH.CTXIO	OH.CTXIO	H.CTXIO	IOP Console Terminal
SH.DCXIO	OH.DCXIO	H.DCXIO	XIO Disc Handler
SH.F8XIO	OH.F8XIO	H.F8XIO	IOP 8-Line ASYNC (Full duplex support)
SH.GPMCS	OH.GPMCS	H.GPMCS	GPMC Subroutines
SH.HSDG	OH.HSDG	H.HSDG	Generic High Speed Data (HSD)
SH.IFXIO	OH.IFXIO	H.IFXIO	XIO Channel Interrupt Fielder
SH.LPXIO	OH.LPXIO	H.LPXIO	XIO Line Printer
SH.MDXIO	OH.MDXIO	H.MDXIO	XIO Memory Disc
SH.MTXIO	OH.MTXIO	H.MTXIO	XIO Magnetic Tape
SH.MUX0	OH.MUX0	H.MUX0	GPMC Multiplexor
SH.NUXIO	OH.NUXIO	H.NUXIO	XIO Null Device
SH.SLMP	OH.SLMP	H.SLMP	SLIM (Synchronous) - GPMC
SH.XIOS	OH.XIOS	H.XIOS	XIO Common Subroutines

NONRESIDENT ROUTINES

Source File Name	Object File Name	Load Module	Description
SJ.ACCNT	OJ.ACCNT	J.ACCNT	Accounting Utility
SJ.ADMNT	OJ.ADMNT	ADMOUNT	Dismount ANSI Labeled Tape Utility
SJ.AMOUNT	OJ.AMOUNT	AMOUNT	Mount ANSI Labeled Tape Utility
SJ.ASTAT	OJ.ASTAT	ASTAT	Display ANSI Labeled Tape Utility
SJ.ATAPE	OJ.ATAPE	J.ATAPE	ANSI Labeled Tape Processing Task
SJ.AUTO	OJ.AUTO	N/A	Auto Disc Geometry Subroutine
SJ.AVOL1	OJ.AVOL1	AVOLM	Log ANSI Labeled Tape Utility
SJ.AVOL2	OJ.AVOL2		
SJ.COMP1	OJ.COMP1	COMPRESS	Object Module Concatenation Utility
SJ.DECMP	OJ.DECMP	N/A	Compressed File Read Subroutine
SJ.DEVL	OJ.DEVL	DEVINITL	Write Control Storage (WCS) Initializer
SJ.DTSAVE	OJ.DTSAVE	J.DTSAVE	Auto Date and Time Update for Mounted Volumes
SJ.ERR	OJ.ERR	M.ERR	MPX-32 Abort Code Module
SJ.xx.ER	OJ.xx.ER	xx.ER	Individual Error Files for all unbundled products
SJ.FORMF	OJ.FORMF	J.FORMF	Format Floppy Formatter Program
SJ.FREAD	OJ.FREAD	N/A	Read Subroutine for Key and Project
SJ.INIT	OJ.INIT	J.INIT	System Initializer
SJ.KEY	OJ.KEY	KEY	M.Key File Editor
SJ.LABEL	OJ.LABEL	J.LABEL	Label ANSI Tape Utility
SJ.LIST	OJ.LIST	LIST	List File Utility
SJ.MDREST	OJ.MDREST	J.MDREST	Memory Disc Restore Task
SJ.MDSAVE	OJ.MDSAVE	J.MDSAVE	Memory Disc Save Task
SJ.MDTI	OJ.MDTI	J.MDTI	MDT Initialization Task
SJ.MOUNT	OJ.MOUNT	J.MOUNT	System Mount Service
SJ.OPCOM	OJ.OPCOM	OPCOM	Operator Communications
SJ.PROJ	OJ.PROJ	J.PRJCT	Project Accounting Utility
SJ.REST	OJ.REST	RESTART	On-line Restart
SJ.SOEX	OJ.SOEX	J.SOEX	Output Spooling Executive
SJ.SOUT	OJ.SOUT	J.SOUT	Output Spooler
SJ.SSIN	OJ.SSIN	J.SSIN1	Input Spooling - Files
SJ.SSIN	OJ.SSIN	J.SSIN2	Input Spooling - Devices
SJ.SWAPR1	OJ.SWAPR1	J.SWAPR	Nonresident Swapper
SJ.SWAPR2	OJ.SWAPR2		
SJ.TDEFI	OJ.TDEFI	J.TDEFI	Terminal Definition Initialization Task
SJ.TINIT	OJ.TINIT	J.TINIT	Terminal Initializer
SJ.TSET	OJ.TSET	J.TSET	Set Terminal Type Utility
SJ.TSM	OJ.TSM	J.TSM	Terminal Service Monitor
SJ.UNLCK	OJ.UNLCK	J.UNLOCK	Dual Port Unlock Utility
SJ.VFMT	OJ.VFMT	J.VFMT	Volume Formatter
SJ.VFPRE	N/A	N/A	Volume Formatter PRE File
SJ.VOLM	OJ.VOLM	VOLMGR	Volume Manager
SJ.VMPRE	N/A	N/A	Volume Manager PRE File

Table 2-1
Deliverable Software for MPX-32 (Page 3 of 4)

The following are SYSGEN load module components:

Source File Name	Object File Name	Description
SJ.FMT10	OJ.FMT10	Sysgen Formatter
SJ.OBUTL	OJ.OBUTL	Object Processor
SJ.PSCAN	OJ.SCAN	Sysgen Scanner
SJ.SDBUG	OJ.SDBUG	Sysgen Debugger
SJ.SEXEC	OJ.SEXEC	Executive (Root Segment)
SJ.SGINI	OJ.SGINI	Initialization Overlay
SJ.SPH01	OJ.SPH01	Phase 1
SJ.SPH02	OJ.SPH02	Phase 2
SJ.SPH03	OJ.SPH03	Phase 3
SJ.SPH04	OJ.SPH04	Phase 4
SJ.SSCAN	OJ.SSCAN	Keyword Scanner
SJ.STACK	OJ.STACK	Sysgen Stack
SJ.STBLS	OJ.STBLS	Device Type Table

The following are SYSGEN files:

File Name	Description
MSTR.27	System Image File for 32/27 Master
MSTR.27S	System Symbol Table File for 32/27 Master
DIR.27	Directive File for 32/27 Master
DIR.27F	Directive File for 32/27 Floppy Master
MSTR.87	System Image File for 32/67, 32/87 or 32/97 Master
MSTR.87S	System Symbol Table File for 32/67, 32/87, or 32/97 Master
DIR.87	Directive File for 32/67, 32/87, or 32/97 Master
MSTR.75	Dummy 7x System Image
DIR.87F	Directive File for 32/67,32/87, or 32/97 Floppy Master
OH.32	Compressed System Object
OH.32_E	Extended Compressed System Object
SG.32	Sample SYSGEN Macro/Directives
JH.32	COMPRESS Input Directives
JH.32_E	Extended COMPRESS Input Directives

The following are macro library files:

File Name	Description
M.MACLIB	MPX/RTM Macro Library
M.MPXMAC	MPX-32 Macro Library
SM.MPXMC	Source used to load M.MPXMAC (MPX-32 macros)
SM.RTMMC	Source used to load M.MACLIB (RTM compatible macro library)

The following are subroutine library files. These files are null until subroutines are inserted into the library.

File Name	Description
MPXDIR	MPX-32 System Subroutine Directory
MPXLIB	MPX-32 System Subroutine Library

The following are extended MPX-32 Modules (used with OH.32_E):

OH.MEMM	Extended H.MEMM
OH.REMM	Extended H.REMM
OH.REXS	Extended H.REXS
OH.TAMM	Extended H.TAMM
OH.VOMM	Extended H.VOMM

The following are PRE files:

File Name	Description
MPX_EXT	PRE File for assembly of Extended MPX-32 modules
MPX_NON	PRE File for assembly of Nonextended MPX-32 modules
MPXPRE	Default PRE File for macro assembler
S3227	32/27 PRE File

Table 2-1
Deliverable Software for MPX-32 (Page 4 of 4)

The following are Job Control Language (JCL) files:

File Name	Description
JJ.A.NON	JCL to assemble/catalog nonresident modules
JJ.A.RS1	JCL to assemble resident modules (Services/Processors)
JJ.A.RS2	JCL to assemble resident modules (Interrupts/Traps)
JJ.A.RS3	JCL to assemble resident modules (Device Handlers)
JJ.A.VOL	JCL to assemble/catalog VOLMGR
JJ.A.SGN	JCL to assemble/catalog SYSGEN
JJ.A.SWP	JCL to create swapper load module (J.SWAPR)
JJ.A.TDI	JCL to assembly/catalog J.TDEFI
JJ.B.LIB	JCL to create/build the null MPXLIB/MPXDIR
JJ.B.MAC	JCL to create/build macro libraries M.MPXMAC, M.MACLIB
JJ.COMPR	JCL to create/load COMPRESSed files Object Module
JJ.F.27	JCL to create 32/27 Floppy Master Image
JJ.F.87	JCL to create 32/67, 32/87, 32/97 Floppy Master Image
JJ.INDER	JCL to assemble/catalog all individual SJ.xx.ERR files
JJ.M.ERR	JCL to assemble/catalog M.ERR (MPX-32 Release 3.x Aborts)
JJ.MSTR	JCL to create Master System Images
JJ.XX.ERR	JCL to assemble/catalog SJ.xx.ER (User Aborts)
FLOPSDT	JCL to generate an SDT on floppy
MSTRSDT	JCL to build the Master SDT
MSTRSRCE	JCL to build the Master Source Tape

2.2.3 Utility Tape

The utilities are contained on a separate tape, as an unbundled product, as shown in Figure 2-1. After the SDT has been restored, restore the desired utilities before exiting the Volume Manager.

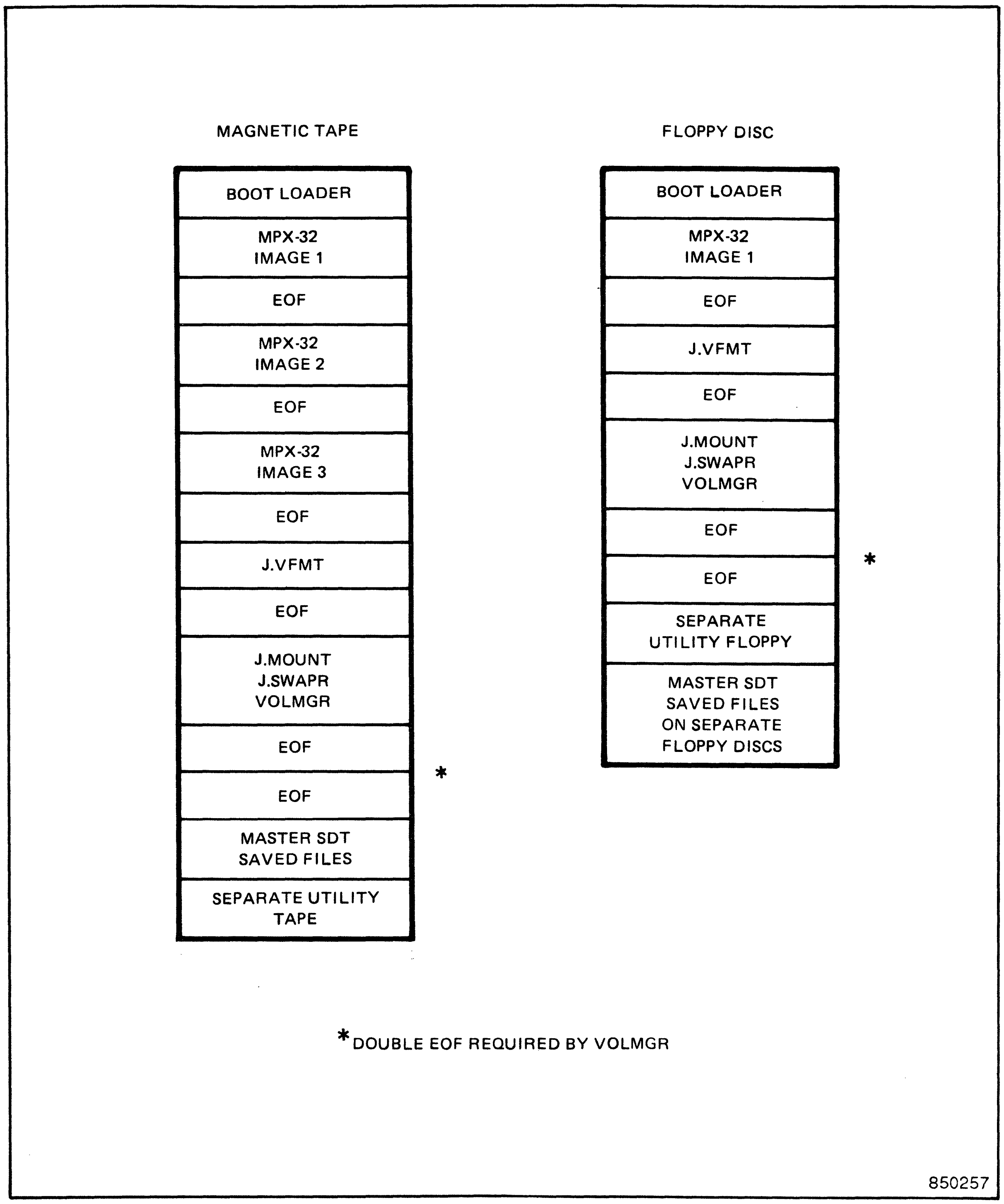


Figure 2-1. Master System Distribution Tape Format

2.3 Booting a System from the Master SDT

Mount the Master System Distribution Tape (SDT) on the tape drive. Turn on the drive and its controller.

Mount an initialized disc pack on the disc drive to be the system volume disc. Turn on the drive. Initialized refers to hardware initialization of the media and not to the software process of formatting a volume for use by the operating system. If the volume to be used is not initialized, the Media Verification Program must be run to verify sector addresses and ensure the quality of the pack surfaces. Refer to the following documentation in the CONCEPT/32 Diagnostic Facility Peripheral Descriptions manual:

- IOP Media Verification Disc Controller Program
- IOP Disc Controller Diagnostic
- RPU Disc Processor Media Verification Program
- RPU Disc Processor Diagnostic

If an IOP is to be configured, it must be assigned to channel X'7E'.

To boot a CONCEPT/32 computer, on the system console:

Enter Panel Mode by typing:	@@P
System Response:	//
Halt the System by typing:	//HALT
System Response:	//
Reset the System by typing:	//RST
System Response:	//
Load the System by typing:	//IPL=address of tape unit or floppy disc

The boot loader on the Master SDT begins execution. It selects and loads the proper system image for the CPU to be used, and passes control to SYSINIT, the system initialization program. The following prompts are displayed on the system console. See numbered comments on the following page for explanation.

```
1.   >>

2.   MPX-32 MASTER SDT FOR 32/XX SERIES COMPUTERS

3.   PLEASE ENTER THE 4 CHAR. DEVICE.....

             LOGICAL ADDRESS:

4.   IS THIS AN IOP OR AN XIO DISC CONTROLLER? (REPLY I OR X):

     MEMORY INITIALIZATION STARTED........

     MEMORY INITIALIZATION COMPLETE.......

5.   ENTER DATE AND TIME:

     TASK LOADING FROM TAPE STARTED.....
```

6. FMT >

7. ENTER SYSTEM VOLUME CHANNEL AND SUBADDRESS:

8. ENTER SWAP VOLUME CHANNEL AND SUBADDRESS
   (OR < CR > IF SYSTEM VOLUME):

9. VOL >

10. INITIALIZATION COMPLETE
    TERMINAL SETUP COMPLETE

    INITIALIZATION COMPLETE
    M.ALOC1 DENIAL, NO LOGON FILE, DEFAULT USED
    TERMINAL SETUP COMPLETE

11. PRESS ATTENTION FOR TSM

Comments:

1. This prompt is displayed if the system debugger is configured in the system. Enter TE to continue normal system operation. When upgrading to a new release of MPX-32, terminal initialization is inhibited at this point by responding to the system debugger as follows:

   ```
   >> CM 780, 10000000
   >> TE
   ```

   This sets control switch three.

2. XX is 27 for a 32/27, 67 for a 32/67, 87 for a 32/87 or 97 for a 32/97.

3. Enter the channel and subaddress of the desired drive (for example, 0800) as the logical address.

4. Enter the appropriate response for this disc prompt.

5. Enter the date and time using the following syntax:

   ```
   date,hh:mn:ss [, [D] [,TZ=num] ]
   ```

date	is the current date in one of the following formats: mm/dd/yy dd-mm-yy ddmmmyy where mm is the two-digit decimal month, dd is the two-digit decimal day, yy is the two-digit decimal year, and mmm is the three ASCII character month abbreviation.
hh	is the two-digit decimal hour (24 hour time)
mn	is the two-digit decimal minute
ss	is the two-digit decimal second
D	indicates daylight savings time. Specifying this field causes the internally stored binary time to be adjusted by one hour.

TZ=num	allows the internal binary time to be biased by num hours. The value of num can be positive or negative. This field allows file times to be kept according to a given standard while the displayed time (see OPCOM TIME directive) is the correct local time.

Examples:

```
06/16/81,08:45:00
14-06-81,13:00:00
03NOV81,09:25:00
08/29/81,10:33,D
15-04-81,19:15:00,D,TZ=-3
05MAY81,16:15,,TZ=10
```

6. This is the Volume Formatter prompt. The Volume Formatter builds the on-disc structures required by the MPX-32 system.

   If installing MPX-32 for the first time, follow item A. If upgrading MPX-32, follow item B.

   A. The minimum response required at this prompt is:

      FMT > FORMAT DEVICE=devmnc VOLUME=volname

devmnc	is a six-character device mnemonic (two-character device code, two-digit hexadecimal channel number, two-digit hexadecimal device subaddress); for example, DM0800 or DF0801.
volname	is a 1- to 16-character volume name. Valid characters are A through Z, 0 to 9, dot (.) and underscore (_).

      If desired, other format command parameters may also be included on the command line, for example, ACCESS=, CONFIRM=, MAXRES=. Exit the Volume Formatter and proceed to step 7.

   B. Do not reformat the volume at this point. Enter the following to the Volume Formatter prompt.

      FMT> REPLACE DEVICE=devmnc VOLUME=volname

devmnc	is a six character device mnemonic
volname	is the 1- to 16-character name of the system volume

      This causes the master image on the Master SDT to be the new default image. Exit the Volume Formatter.

7. Reenter the two-character channel number and two-character subaddress that was entered in step (6); for example, 0800 or 0801.

8. If the SYSTEM volume is the swap volume, enter a carriage return. If a different volume is to be the swap volume, enter the two-digit hexadecimal channel number and two-digit device subaddress of the desired volume; for example, 0802. The swap volume is the volume where disc space will be reserved for swapping eligible tasks during periods of heavy system activity. For more details on task swapping, see MPX-32 Reference Manual Volume 1, Chapter 2.

9. This is the Volume Manager prompt. When installing MPX-32 for the first time, a system directory must be created before files on the SDT can be restored. Only the system directory is required. OBJECT, SOURCE, and DEMO are optional and do not have to be created.

   When installing or upgrading MPX-32 with floppy discs, only the system image is on the boot floppy, all saved files are on additional discs.

   The minimum response required at the Volume Manager prompt is:

   VOL > <u>CREATE D SYSTEM ENTRIES=xx</u>

   If desired, other parameters such as ENTRIES=, OWNER=, PROJECTGROUP=, ACCESS= may also be included on the directive line.

   Saved files on the Master SDT can now be restored. Each RESTORE directive processes one save image. Repeat this command until all desired saved files on the SDT have been restored. Mount any unbundled software tapes on the tape drive and restore in the same manner. After all desired saved files are restored, type EXIT to exit the Volume Manager.

   ```
   VOL > RESTORE VOLUME=SYSTEM
     .
     .
     .
     .
   VOL > EXIT
   ```

   If using floppy discs, only the system image is on the boot floppy. Saved files and utilities are on additional floppy discs.

10. If J.TINIT is restored from the Master SDT, one of these initialization messages are displayed. The one that is displayed depends on whether the system file, LOGONFLE , exists on the currently mounted system volume. See Chapter 10 of this volume for details on how to build LOGONFLE.

11. The system is now fully operational. Upon entering @@A for attention, a prompt is issued to enter a one- to eight-character logon owner name and a one- to eight-character logon key. The following characters cannot be used in owner names or keys: blanks, commas, semicolons, equal signs, line feeds, dollar signs, exclamation points, percent signs, and left or right parentheses. After entering a valid owner name and key, the TSM prompt is displayed and any valid TSM directive can be entered.

Note: If the system does not install as described, check the hardware/firmware revisions of the system and contact the Gould CSD Field Representative.

2.3.1 Control Switches

While rebooting the system, various initialization processes can be inhibited or enabled by setting the appropriate control switches. The assignment of the 13 switches is:

Switch	Function if set
0	Inhibits volume clean-up by J.MOUNT
1	SYSINIT enters the system debugger before processing patches
2	Inhibits patch processing. See Chapter 9, Section 9.3.
3	Inhibits terminal initialization
4	Inhibits accounting functions including the M.KEY, M.PRJCT, M.ACCNT, and M.ERR files
5	Inhibits processing of the sequential task activation table at IPL time
6	If J.MOUNT encounters an invalid resource descriptor due to an invalid resource descriptor type field or space definition, it branches and links to the system debugger (if present) with register 2 pointing to the resource descriptor.
7	J.MOUNT prereads the file space bit map (SMAP) or the resource descriptor allocation bit map (DMAP). J.MOUNT will not perform file overlap protection.
8	Delete spooled output files instead of resubmitting them for processing
9	Inhibits activating LOADACS during IPL or RESTART operations
10	Enables faster memory initialization by checking only one location per map block to determine if that map block is present. It is not recommended that this switch be set on the first IPL after power up.
11-12	Reserved

The control switches can be accessed by the IOP console. The proper time to set the switches is while the system is waiting for the date and time to be entered. To set, for example, switch 0, the following must be entered:

```
ENTER DATE AND TIME: @@P
//CS=80000000 Initialization Inhibited
//@@C
<CR>
INVALID DATE FORMAT=MM/DD/XX
ENTER DATE AND TIME:
```

During power up, control switches are prezeroed if the proper firmware revision level has been installed. Power up without prezeroing can cause unexpected system responses due to incorrect control settings.

All control switch settings are preserved during system reboots not involving system power up (i.e., on-line restart and IPL).

2.4 Philosophy of Bootstrapping

The SDT process attempts to eliminate as much stand alone I/O from the startup code as possible due to the difficulty in maintaining stand alone capability for a number of different types of controllers. This philosophy is reflected in the operation of the system initialization program SYSINIT. For further details on SYSINIT, see Section 2.5.

Because there may be no information on the disc at IPL time, tasks cannot be activated from the disc. Therefore, the four needed tasks (Volume Formatter, Mount, J.SWAPR, and Volume Manager) are supplied in load module format on the Master SDT. SYSINIT performs tape activations on these four load modules to allow the System Builder to:

. Format the disc and build a bootable operating system (Volume Formatter)

. Mount the system volume (Mount)

. Activate the swapper (J.SWAPR)

. Restore the files needed from the Master SDT and the Utility tape to make the system function (Volume Manager).

2.5 The System Builder

Following the operating system in memory is a two part module called SYSINIT. This module is included by SYSGEN and is responsible for initializing the MPX-32 environment. The first part, Phase I, runs stand alone. That is, MPX-32 is not yet functioning. Phase I loads the CPU scratchpad, enables interrupt levels, resets I/O channels, and performs any other cleanup operations required. It then sets up Phase II to run as an MPX-32 task. When Phase I enables interrupts, a clock interrupt occurs and causes Phase II to begin execution as a task. Phase II activates the Volume Formatter, allowing the new volume to be formatted. Next, the volume mount service is activated and mounts the system volume. After the system volume is mounted, J.SWAPR is activated. After this, the Volume Manager is run. This utility restores any files that are required for system operation. These files must include all key load modules and may also include any desired files. When Phase II completes, the system is fully initialized and must be opened for user access. Phase II accomplishes this by activating J.INIT, J.TINIT and J.TSM.

Because there may be no information on the disc at IPL time, activations are made from the SDT, causing backward and forward movement of the tape.

The user interaction for the above procedure on a Master SDT is described in Section 2.3.

The user interaction for the above procedure on a User SDT is described in Chapter 4.

2.6 Operating Under the Starter System

The MPX-32 system is both disc and memory resident. TSM can be entered by entering @@A then the system owner name and no key. Valid owner names are created by the KEY utility.

2.7 Restoring Utility Processors, Libraries, and Other Files

The Volume Formatter (J.VFMT) and Volume Manager (VOLMGR) utilities are supplied on the Master SDT. All other utilities such as the Text Editor, Macro Assembler, and Media are an unbundled product. After all required files have been restored from the Master SDT, all desired unbundled software must be restored.

Load modules on the tapes are saved by using Volume Manager SAVE directives and can be restored from the tapes by using Volume Manager RESTORE directives. Each save directive allows up to 48 entries. An End-of-File (EOF) is written after each group saved.

When restoring files, use a restore directive for each group. Initially, all files should be restored from the Master SDT and the Utility tape using the Volume Manager RESTORE VOLUME directive. In subsequent interactions, selective restore capabilities of the Volume Manager can be used for system installation and maintenance.

The number of groups of files saved on the tapes can vary, and an equivalent number of restore directives are required to install them. A listing is provided with the Master SDT showing the groups in which the files were saved. Before continuing, check the listing. Restore all groups of libraries, system files, and utilities (one restore directive per group) using the listing as a guide.

2.8 Example First Use of Master SDT on User System

```
            //HALT
            //RST
            //IPL=1000 (if magnetic tape)
            //IPL=7EF0 (if floppy disc)
>>TE
MPX-32 MASTER SDT FOR 32/87 SERIES COMPUTERS
PLEASE ENTER THE 4 CHAR. DEVICE.....
            LOGICAL ADDRESS: 0800
IS THIS AN IOP OR AN XIO DISC CONTROLLER: (REPLY I OR X): X
MEMORY INITIALIZATION STARTED.....
MEMORY INITIALIZATION COMPLETE.....
ENTER DATE AND TIME: 04/23/81,13:30
TASK LOADING FROM TAPE STARTED
FMT > FORMAT DEVICE=DM0800 VOLUME=JONES MAXRES=3000
ENTER SYSTEM VOLUME CHANNEL AND SUBADDRESS: 0800
VOL > CREATE D SYSTEM ENTR=1000
VOL > RESTORE VOLUME=SYSTEM
            .
            .
            .
VOL > EXIT
ENTER SWAP VOLUME CHANNEL AND SUBADDRESS
            (OR < CR > IF SYSTEM VOLUME): <CR>
TERMINAL INITIALIZATION COMPLETE
TERMINAL SETUP COMPLETE
PRESS ATTENTION FOR TSM
```

CHAPTER 3

BUILDING AND TESTING A USER-CONFIGURED SYSTEM

This chapter describes how to SYSGEN and test a resident system geared to a specific installation. Once the starter system has been installed, modifications can be made to the input files used by the System Generation utility, SYSGEN, to configure a tailored system. The modified data is saved to tape. This tape then becomes the user SDT. The SYSGEN utility is described in Chapter 7. This chapter establishes SYSGEN in the cycle of building an MPX-32 system.

3.1 Building the SYSGEN Input Files

3.1.1 Building the Directive Input File

The directive input file for SYSGEN determines the configuration of the system. The input file specifies such items as: hardware to support, interrupts to connect, and devices to use for spooled system I/O. File SG.32, provided on the Master SDT, can be used as a model or as a working base for preparing a SYSGEN directive file for the CONCEPT/32. Either edit this file or build a directive input file from scratch using the Text Editor. The resulting file must be uncompressed.

The system debugger, DEBUG, can be included in the resident system by specifying a SYSGEN PROGRAM or USERPROG directive. This is recommended, particularly in an initial system. The system debugger adds approximately 300W to the size of the resident system. See Chapter 8 of this volume for further details on the system debugger.

The size of the target system image can be reduced by one 2KW map block by removing compatibility mode system modules H.ALOC, H.FISE, H.MONS, and H.CALM using the SYSGEN NOCMS directive. Because the system image end address is map block bounded, and the compatibility mode system modules total less than one map block, the size of the system image may not be affected if the compatibility mode modules are removed.

3.1.2 Building the Object Input File

Using the directive input file, SYSGEN determines which system modules, user modules, interrupt handlers, and trap handlers are needed to build the target system. SYSGEN reads the object code for these modules and handlers from the object input file. An object input file, OH.32, for the CONCEPT/32 is provided on the Master SDT.

User object modules can be added to the object input file. The names of these modules are added to the file JH.32 supplied on the master SDT. The file is then compressed using the COMPRESS task. This task is described in the next section.

3.1.3 The COMPRESS Task

The COMPRESS task builds a file containing any number of object files. Its input file must contain the ASCII names, one per record/line, of the object modules to copy to the output file. The logical file code for input to COMPRESS is IN. The default assignment of IN is to the system file JH.32. The logical file code for output from COMPRESS is OT. The default assignment of OT is to the system file OH.32.

COMPRESS writes a control stream naming the files copied and lists the number of records each contains. It also reports any allocation or read errors.

Syntax:

```
COMPRESS
```

The provided object input file used by SYSGEN is created using the COMPRESS task. The COMPRESS input file for the SYSGEN object input file is included on the Master SDT and can be a model or a working base for modifying the SYSGEN object input file.

Proper execution of the COMPRESS task depends on the order of the following modules in the input file:

- H.IP06 must be specified before H.CALM is specified.
- H.IOCS must be specified before H.MVMT or H.BKDM is specified.
- H.XIOS must be specified before H.IFXIO is specified.
- H.IFXIO must be specified before H.??XIO is specified, where ?? are any valid ASCII characters.
- H.MUX0 must be specified before H.??MP is specified, where ?? are any valid ASCII characters.
- H.SWAPR must be specified before H.SINIT is specified. H.SWAPR and H.SINIT must be the last two names in the input file, if the system debugger is not to be configured in the system image. When the system debugger is specified (USERPROG=DEBUG) the unmapped portion, H.DBUG2, must be the last name in the input file immediately following H.SINIT. The mapped portion of the system debugger, H.DBUG1, can be positioned anywhere in the input file before H.SWAPR.

H.IOCS, H.IFXIO, and H.MUX0 are only specified one time in the input file.

COMPRESS IN Assignment	COMPRESS OT Assignment/ SYSGEN OBJ Assignment
JH.32	OH.32

Example:
```
TSM > ASSIGN  IN TO JH.32 BLOCKED=Y
TSM > ASSIGN  OT TO OH.32 BLOCKED=Y
TSM > ASSIGN  LO TO LFC=UT
TSM > COMPRESS
COPIED - 2 RECORDS FROM PATH pathname
COPIED - 2 RECORDS FROM PATH pathname
                    •
                    •
                    •
TSM >
```

pathname is the name of the file from which the records were copied.

If a complete pathname is specified in the COMPRESS input file, the complete pathname is displayed. If only a file name is specified in the COMPRESS input file and the file is found on the current working volume and directory, only the file name is displayed. If only a file name is specified in the COMPRESS input file and the file is not found on the current working volume and directory, the system volume and directory are searched; if the file is found, the complete pathname is displayed. If a file name specified in the COMPRESS input file cannot be found, a message is displayed and the COMPRESS task continues execution.

3.2 Running SYSGEN

Interactive and batch access, required and default assignments, and other aspects of running SYSGEN are covered in Chapter 7. One simple path for configuring a system is described in this section.

The logical file code (LFC) for a SYSGEN directive file is DIR. A modified version of SG.32 can be assigned to DIR. The object input file, OH.32, can be assigned to OBJ, which is the logical file code for the SYSGEN object input file. A TSM ASSIGN directive can assign the input files as shown below.

```
TSM > ASSIGN OBJ TO OH.32 BLOCKED=Y
TSM > ASSIGN DIR TO SG.32 BLOCKED=Y
TSM > SYSGEN
```

Alternatively, a job file can run SYSGEN in the batch mode with the above file assignments.

The SYSGEN output file that contains the resident operating system is specified with the SYSTEM directive in the SYSGEN directives file. The output file is created automatically by SYSGEN. File space for the output file need not be created before running SYSGEN. The file name used with the system directive is also the name to use as the system load file when building an SDT by using the RESTART directive.

3.3 Testing a SYSGENed System

After the tailored system is configured, it can be tested using the RESTART directive. Initially, any owner can use this directive. The M.KEY file can prohibit the use of RESTART by any owner name.

During initial installation, the RESTART directive can be used for a one-shot on-line test of the new system before a tailored SDT is created. The system console is the only interactive device configured at this time.

A new system should not be tested while terminals are configured and activities are in progress. Use of the RESTART directive from the console under these conditions is described in the On-line Restart chapter.

The RESTART directive is issued to TSM:

```
    TSM > $RESTART sysfile
```

Sysfile is the name of the file specified with the SYSGEN SYSTEM directive. The SYSGENed configuration of the resident operating system will be booted from that disc file.

The RESTART command with no system file specified or IPL from the IOP console (see Chapter 5) causes the default system, in this case, the starter system, to return as the working resident system. System configuration development can continue by the editor, SYSGEN, and one-shot RESTART as needed. Figure 3-1 shows an overview of the on-line RESTART and test process.

3.4 Terminal Initialization and System Protection

Chapter 10 of this volume describes how to initialize terminals on MPX-32 and how to build an M.KEY file to provide authorized owners access to the system and implement privileges by owner.

The RESTART process initializes terminals and puts M.KEY privileges into effect if terminal initialization and M.KEY files have been established.

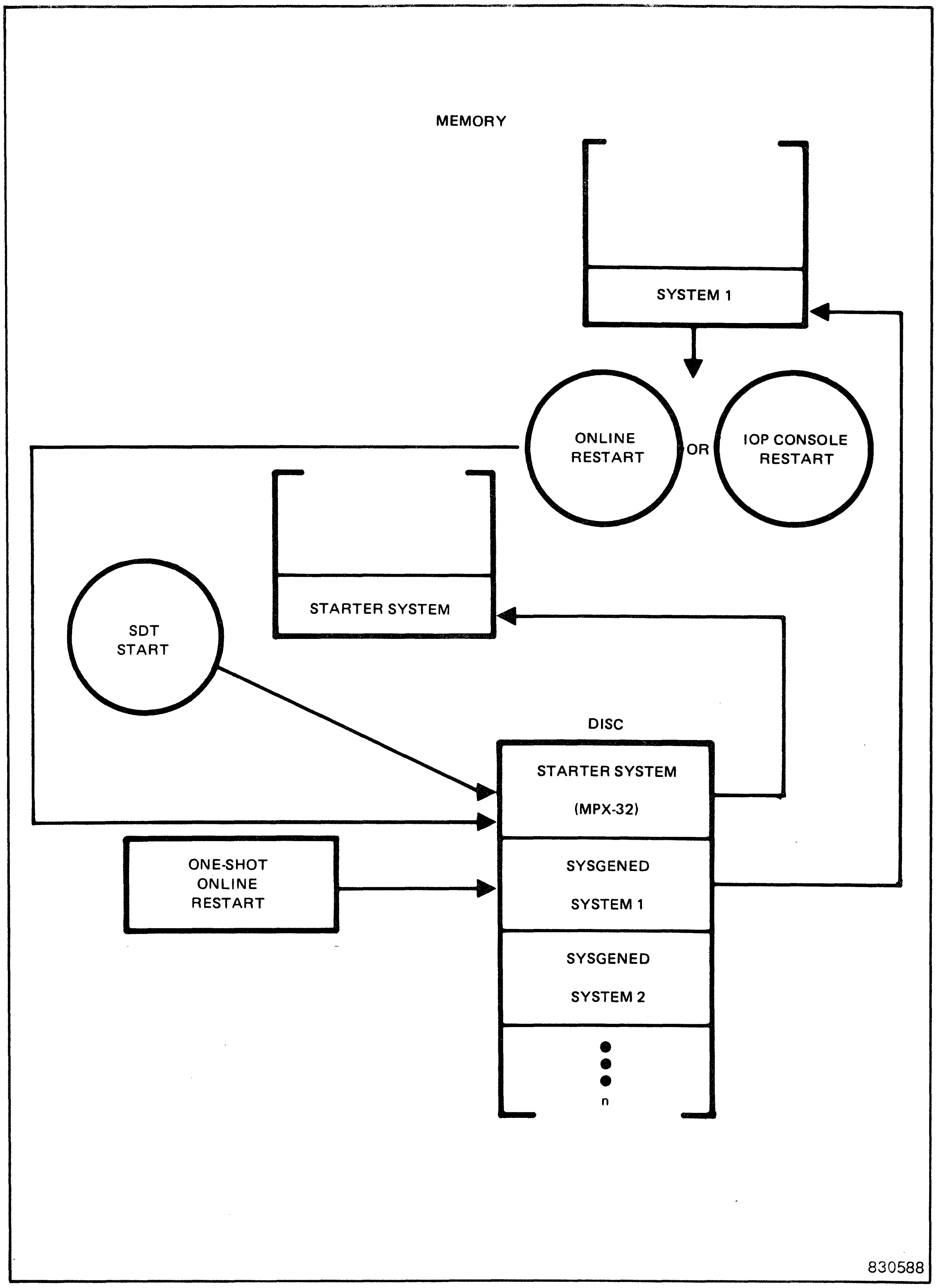


Figure 3-1. Testing a User-Configured System

CHAPTER 4

INSTALLING A USER-CONFIGURED SYSTEM

Once a configured system is tested and ready to install in place of the starter system, the Volume Manager builds a user System Distribution Tape (SDT).

4.1 Creating a User System Distribution Tape (SDT)

The user SDT is created by the Volume Manager and is similar to the Master SDT. Following the desired system image are four essential load modules that must be configured:

. Volume Formatter (J.VFMT)

. System Mount (J.MOUNT)

. Swapper (J.SWAPR)

. Volume Manager (VOLMGR)

These load modules are automatically activated from the SDT, but they are not restored to the disc since directory information has not been retained about them. Therefore, these four modules must be restored to disc before they can be saved onto the user SDT. These load modules must appear in this order for the system to be built properly. Figure 4-1 shows the layout of a user SDT.

Following the system image and essential load modules, another group of files required for system operation must be saved: OPCOM, J.INIT, and J.TSM. These files must also be restored to disc when the VOLMGR program is activated. The minimum required modules to restore and save are: OPCOM, VOLMGR, J.INIT, J.MOUNT, J.TSM, and J.SWAPR.

The recommended set of load modules for an SDT is: VOLMGR, J.MOUNT, J.VFMT, OPCOM, J.INIT, J.TSM, J.SOUT, J.SSIN1, J.SSIN2, J.TINIT, J.SOEX, and EDIT.

As boot programs cannot process multivolume headers, a user SDT should not be created on multivolume tape.

4.2 SDT Directive

The Volume Manager SDT directive generates a bootable System Distribution Tape to logical file code 'TAP'. A loader is written, followed by the desired system image and the three key load modules.

Syntax:

```
    SDT [PATH=] sysfile
```

sysfile	is the name of the file containing the desired resident operating system image. This name is the same as the name supplied in the SYSGEN SYSTEM directive. The four load modules J.VFMT, J.SWAPR, J.MOUNT, VOLMGR are automatically implied by the SDT directive and must not be specified.

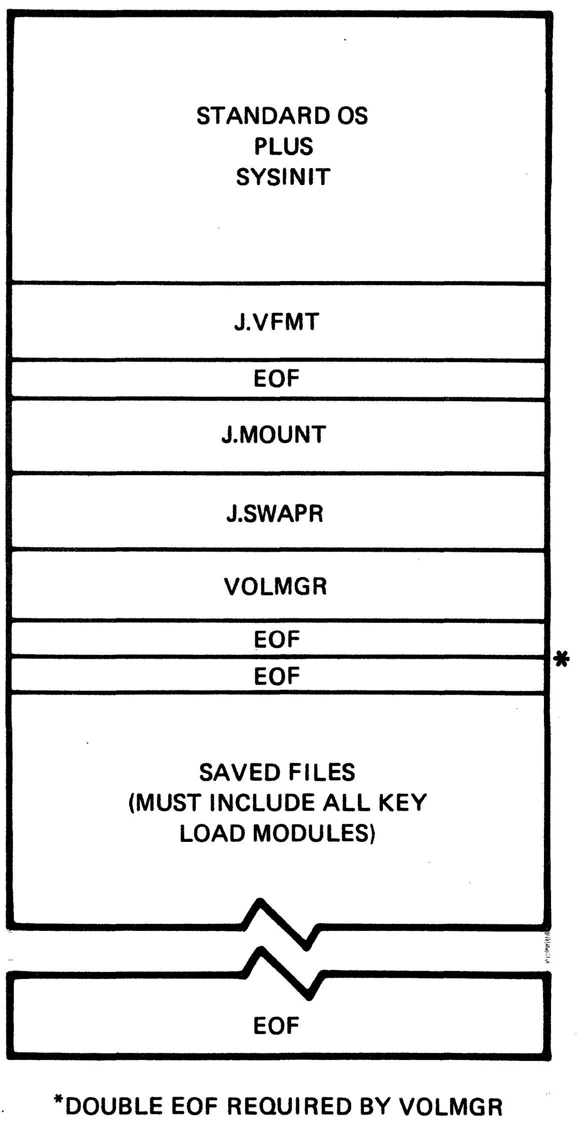


Figure 4-1. User System Distribution Tape Format

4.3 Installing a User System Distribution Tape (SDT)

All devices specified in the SYSGEN configuration should be connected.

The procedure to install a user SDT is similar to the procedure to install a master SDT as described in Section 2.3.

J.TINIT searches for a file in the system directory named LOGONFLE. The LOGONFLE can be built using the editor as documented in Chapter 10 of this volume. If LOGONFLE exists as a system file, J.TINIT uses the parameters contained in the file to initialize the terminals on the system. When terminal initialization is completed, the following message is displayed on the console:

```
INITIALIZATION COMPLETE
TERMINAL SETUP COMPLETE
SYSTEM READY... PRESS ATTENTION FOR TSM
```

If LOGONFLE does not exist as a system file, J.TINIT uses default parameters for all terminals configured in the current system image (default parameter values are described in Chapter 2 of this volume). Once terminal initialization is completed and default values are assumed, the following message is displayed on the console:

```
INITIALIZATION COMPLETE
M.ASSN DENIAL, NO LOGON FILE, DEFAULT USED
TERMINAL SETUP COMPLETE
SYSTEM READY... PRESS ATTENTION FOR TSM
```

4.4 Saving/Restoring System Processor and Utility Load Modules

Additional load modules, libraries, and files are saved and restored using Volume Manager save and restore directives.

4.5 Booting a System from a User SDT

To boot a CONCEPT/32 computer, on the system console:

Enter Panel Mode:	@@P
System Response:	//
Halt the System:	//HALT
System Response:	//
Reset the System:	//RST
System Response:	//
Load the System:	//IPL=address of tape unit or floppy disc

The following prompts are displayed on the system console:

1. >>

   MEMORY INITIALIZATION STARTED.....

   MEMORY INITIALIZATION COMPLETE.....

2. ENTER DATE AND TIME:

   TASK LOADING FROM TAPE STARTED.....

3. FMT >

4. ENTER SYSTEM VOLUME CHANNEL AND SUBADDRESS:

5. ENTER SWAP VOLUME CHANNEL AND SUBADDRESS
   (OR < CR > IF SYSTEM VOLUME):

6. VOL >

7. INITIALIZATION COMPLETE
   TERMINAL SETUP COMPLETE

   INITIALIZATION COMPLETE
   M.ALOC1 DENIAL, NO LOGON FILE, DEFAULT USED
   TERMINAL SETUP COMPLETE

8. PRESS ATTENTION FOR TSM

Comments:

1. This prompt is displayed if the system debugger is configured in the system. Enter TE to continue normal system operation.

2. Enter the date and time using the following syntax:

```
date,hh:mn:ss [ , [D] [,TZ=num] ]
```

date	is the current date in one of the following formats: mm/dd/yy dd-mm-yy ddmmmyy mm is the two-digit decimal month, dd is the two-digit decimal day, yy is the two-digit decimal year, and mmm is the three ASCII character month abbreviation.
hh	is the two-digit decimal hour in 24 hour time
mn	is the two-digit decimal minute
ss	is the two-digit decimal second
D	indicates daylight savings time is in effect. Specifying this field causes the internally stored binary time to be adjusted by one hour.
TZ=num	allows the internal binary time to be biased by num hours. The value of num can be positive or negative. This field allows file times to be kept according to a given standard while the displayed time by the OPCOM TIME directive is the correct local time.

Examples:

```
06/16/81,08:45:00
14-06-81,13:00:00
03NOV81,09:25:00
08/29/81,10:33,D
15-04-81,19:15:00,D,TZ=-3
05MAY81,16:15,,TZ=10
```

3. This is the Volume Formatter prompt. The Volume Formatter builds the on-disc structures required by the MPX-32 system. The minimum response to this prompt is:

```
FMT > FORMAT DEVICE=devmnc VOLUME=volname
```

devmnc	is a six-character device mnemonic (two-character device code, two-digit hexadecimal channel number, two-digit hexadecimal device subaddress), for example, DM0800 or DF0801
volname	is a 1- to 16-character volume name. Valid characters are A through Z, 0 to 9, dot (.) and underscore (-). Other FORMAT command parameters can also be included on the directive line, ACCESS=, CONFIRM=, MAXRES=.

4. Reenter the two-character channel number and two-character subaddress that was entered in step 3, for example, 0800 or 0801.

5. If the system volume is the swap volume, enter a carriage return. If a different volume is to be the swap volume, enter the two-digit hexadecimal channel number and two-digit device subaddress of the desired volume, for example, 0802.

6. This is the Volume Manager prompt. If a system directory larger than the default is required, a system directory can be created before files on the SDT are restored.

   VOL > <u>CREATE D SYSTEM ENTRIES=xx</u>

   Other parameters can also be included on the directive line, for example OWNER=, PROJECTGROUP=, ACCESS=.

   Saved files on the SDT can now be restored. Each RESTORE directive processes one save image. Repeat this directive until all desired saved files on the SDT have been restored. Mount any unbundled software tapes on the tape drive and restore in the same manner. After all desired saved files are restored, type EXIT to exit the Volume Manager.

   VOL > <u>RESTORE VOLUME=SYSTEM</u>

   .
   .
   .

   VOL > <u>EXIT</u>

   If using floppy discs, only the system image is on the boot floppy. Saved files are on additional floppy discs.

7. If J.TINIT is restored from the SDT, one of these initialization messages is displayed. Which one depends on whether the system file, LOGONFLE, exists on the currently mounted system volume. See Chapter 10 of this volume for details on how to build LOGONFLE.

8. The system is now fully operational. Upon entering @@A for attention, a prompt is issued to enter a one- to eight-character logon owner name and one- to eight-character logon key. The following characters cannot be used in owner names or keys: blanks, commas, semicolons, equal signs, line feeds, dollar signs, exclamation points, percent signs, and left or right parentheses. After entering a valid owner name and key, the TSM prompt is displayed and any valid TSM directive can be entered.

4.5.1 Example

```
        //HALT
        //RST
        //IPL=1000 (if magnetic tape)
        //IPL=7EF0 (if floppy disc)

>>TE
MEMORY INITIALIZATION STARTED.....
MEMORY INITIALIZATION COMPLETE.....
ENTER DATE AND TIME: 27APR81,08:00:00
TASK LOADING FROM TAPE STARTED.....
FMT > FORMAT DEVICE=DM0800 VOLUME=JONES MAXRES = 3000
ENTER SYSTEM VOLUME CHANNEL AND SUBADDRESS:0800
VOL > CREATE D SYSTEM ENTR=1000
VOL > RESTORE VOLUME=SYSTEM
      .
      .
      .
VOL > EXIT
ENTER SWAP VOLUME CHANNEL AND SUBADDRESS
  OR < CR > IF SYSTEM VOLUME: < CR >
PRESS ATTENTION FOR TSM
```

CHAPTER 5

ON-LINE RESTART

The TSM $RESTART directive is used to:

. one-shot test a new system

. restart the current default operating system after a test

. establish a new default system

An on-line restart does not alter the control switch settings.

Figure 5-1 illustrates the backup, or default, system concept implemented in MPX-32.

Note: The RESTART processor does not reinitialize the Analog/Digital Interface (ADI) board or its associated Real-Time Peripheral (RTP) Equipment. Therefore, system images containing the ADI handler should be brought up using IPL procedures.

5.1 Precautions

Before using the on-line RESTART directive, the following steps should be taken:

. Use the TSM $SIGNAL directive to notify all terminal users to stop interactive and batch activity and logoff.

. Use the OPCOM LIST directive to see if any user tasks remain active. Abort tasks by task number using the OPCOM ABORT directive.

. Check batch stream activity. It should wind down as users logoff.

. Independent tasks are killed during the restart process. All pending I/O is lost. Independent tasks must be reestablished or reactivated under the new system.

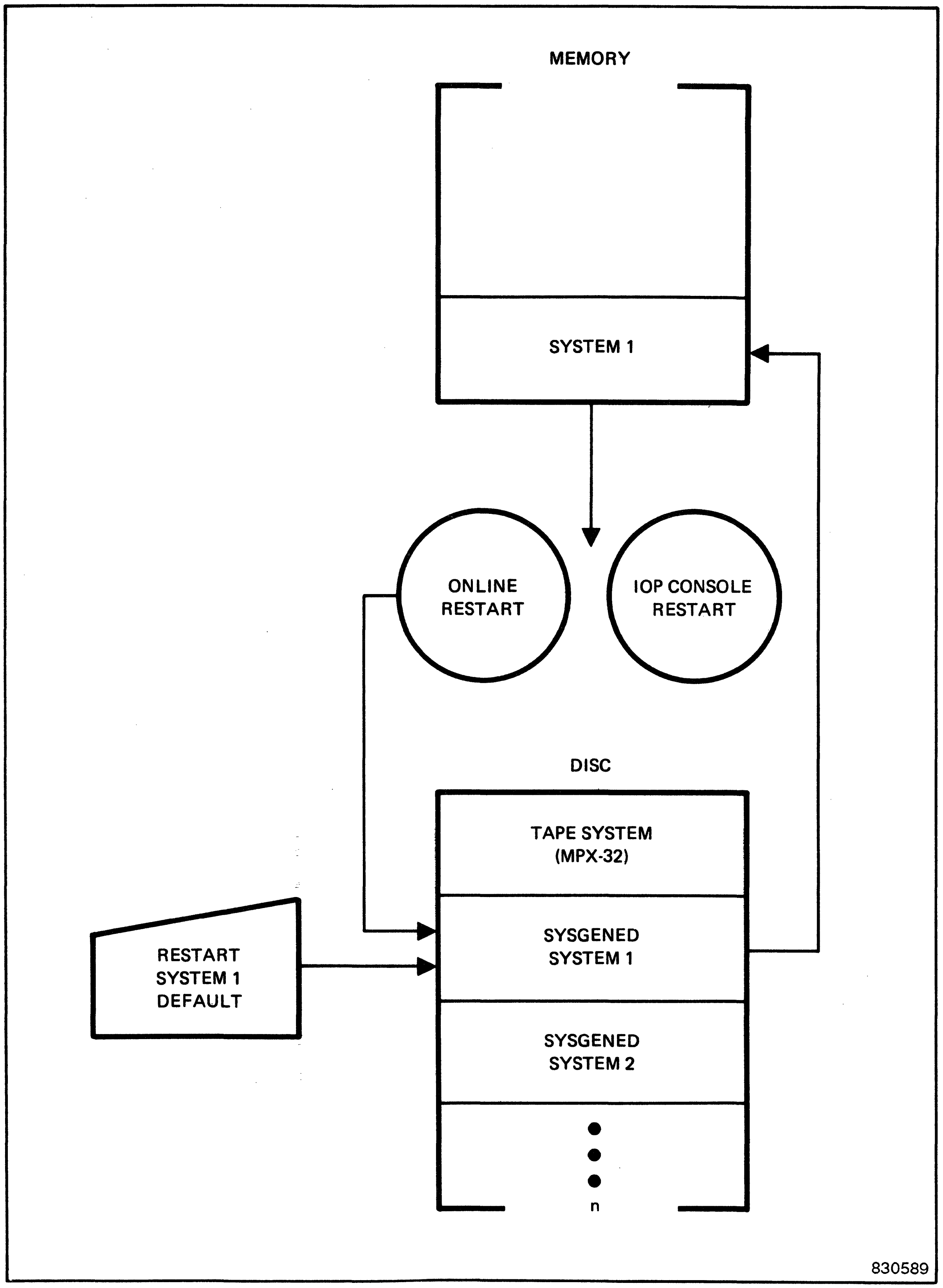


Figure 5-1. Establishing a New Default System

5.2 Restart Syntax

```
$RESTART [pathname]
```

pathname is the pathname of the file containing the operating system to be restarted. The file name supplied must be the same as the file name supplied with the SYSGEN SYSTEM directive. The following prompts are then displayed:

```
DO YOU WISH TO RESTART?  (Y OR N):

DO YOU WISH THIS TO BE YOUR DEFAULT IMAGE?  (Y OR N):
```

These prompts are independent of each other. The possible responses and resulting actions are as follows.

Restart Prompt	Default Image Prompt	Resulting Action
Y	Y	The initial program load, IPL, sequence is performed, and a new default image is established.
Y	N	The initial program load, IPL, sequence is performed, but the default image is not changed.
N	Y	The initial program load, IPL, sequence is not performed, but a new default image is established.
N	N	Nothing happens; the RESTART command is ignored by the operating system.

If a pathname is not specified, the current default system is loaded as the resident operating system and the following prompt is displayed:

```
DO YOU WISH TO RESTART? (Y OR N):
```

If Y is specified, the IPL sequence is performed. If N is specified, the IPL sequence is not performed.

Note: An on-line restart does not alter the control switch settings.

CHAPTER 6

RECOVERING THE SYSTEM

If the operating system halts, a fresh copy of the disc file containing the operating system image can usually be restored into memory from the IOP console. This is an initial program load (IPL) and restart operation, and goes through the restart cycle illustrated in the previous chapter.

6.1 Recovery from Disc at the IOP Console

To IPL and restart the default MPX-32 system from disc, enter the following at the IOP console:

```
    //RST
    //IPL xxxx
```

where xxxx is the address of the system disc

Restart logic is always located on the system disc, and directs processing to the most recent default system file regardless of its disc location.

6.2 Errors During Start-up

If the system halts at or near location X'6FC' during IPL from disc, enter GPR at the // prompt to read the contents of Register 1.

Register 1 contains one of the following error codes that will aid in debugging:

Error Code	Description
1	Checksum error detected on system image
2	System image was not located at address specified to bootstrap
3	Fatal I/O error reading system image
4	Error retry attempts exhausted
5	System does not match machine type. For example, system SYSGENed for a 32/27 and attempt made to boot 32/87

If the system halts during system initialization (SYSINIT) phase 1, register 5 contains abort code SY01. Register 1 contains one of the following error codes:

Error Code	Description
1	Unrecognizable IPL device indicator
2	Invalid machine type
3	No UDT associated with the IPL device
4	Target system image not compatible with CPU model
5	Cannot boot extended mode MPX-32 on a 32/27

If the system halts during SYSINIT phase 2, an abort message is displayed on the system console.

6.3 System Halt Analysis

If the operating system halts, the following procedure can be used on a CONCEPT/32 computer to determine what operations were in progress when a halt occurred.

1. Check the CPU front panel to determine if the halt was caused by a hardware failure. Check interrupt active, run, halt, and any other pertinent indicators.

2. Read PSW and Instruction

   //PSW

3. Read registers 0 - 7

   a. //GPR

   b. If R7=X'54524150', a trap, the registers contain the following information:

Register	Contents
0	PSD word 1
1	PSD word 2
2	Real address of the instruction causing the trap
3	Instruction causing the trap
4	Trap status word
5	ASCII trap type (e.g. MF01)
6	Address of registers saved at time of trap

   If R5 = X'564D3939' (a potential file overlap) and R7 = X'00000034' (a double allocation error) occur, the registers contain the following information:

   R0 = PSW
   R1 = VOMM FCB address
   R2 = MVTE address
   R3 = Requested size in number of allocation units
   R4 = Next available SMAP bit position
   R6 = Internal VOMM error code

If R5 = X'564D3939' (a potential file overlap) and R7 = X'00000000' (a double deallocation error) occur, the registers contain the following information:

R0 = PSW
R1 = VOMM FCB address
R2 = MVTE address
R3 = MVTE address
R4 = Start SMAP bit position to perform deallocation
R6 = SMAP start block number

c. The registers at the time of the trap are examined by entering:

//MA = X	X is the contents of register six

followed by seven returns. Each return displays the contents of the next register.

d. PSD2, real address, instruction, and status is examined by entering four returns, one return for each item

//	Press return key four times to access PSD2, real address, instruction, and status

4. Access system debugger, if configured

a. //RST	
b. //MA=B5C	C.DEBUG (address of system debugger or zero if not included in SYSGEN)
c. //PC=x	x is the address displayed as MD=x resulting from the MA=B5C step
d. //RUN	

5. In the system debugger

a. DT	Systems with event trace enabled
b. ABS	
c. ECHO	
d. REMAP	
e. 8E8	Determines current task
f. 8E8,+40	Dumps dispatch queue information
g. DU 0,nnnn	nnnn is the address of H.IP00, obtained from SYSGEN load map
h. DU N,2000N	User task TSA

6. Make a dump tape, if a crash dump routine is configured

a. Mount a tape on MT1000	
b. //HALT	
c. //RST	
d. //MA=C28	C.CRDUMP (address of crash dump routine or zero if H.DEBUG2 or H.DMPMT is not included in system)
e. //PC=x	x is the address displayed as MD=X resulting from the MA=C28 step
f. //RUN	

g. When the system is up, use ANALYZE to study the crash dump

6.4 Automatic IPL

If automatic IPL is hardware enabled and the system halts because of a power failure, the CPU initiates an IPL during the power up initialization. Automatic restart, the ability to restart or continue a software program that was interrupted by a power outage, is not supported. The software does, however, provide the required parameters for user implementation of the power up automatic restart feature.

If operator intervention is inhibited, all prompts normally displayed on the system console during system initialization are inhibited. The system provides default responses for the inhibited prompts. See Section 10.13.

CHAPTER 7

SYSTEM GENERATION (SYSGEN)

System Generation for an MPX-32 system involves supplying a set of configuration directives to the SYSGEN utility. Using these directives, the utility creates a permanent file containing the installation specific MPX-32 system in memory image absolute format.

7.1 General Description

SYSGEN is a privileged system utility that operates in a standard MPX-32 system. SYSGEN can be executed in batch or interactive mode. The system where SYSGEN is executed must have enough free memory to hold the generated system and SYSGEN itself. SYSGEN requires 16KW.

The resources that SYSGEN requires are:

. Directives
. System object modules
. File to contain the system resident image
. System symbol table file

The directives must be in card image format and can be supplied interactively or by using batch from magnetic tape, disc, or card reader. If supplied on a magnetic tape, the tape can include SYSGEN resident object modules and MPX-32 modules. The file for the resident system image and the system table file are permanent disc files created by SYSGEN.

System object modules include: interrupt and trap processors such as H.IP00 and H.IPIT, modules that form the MPX-32 nucleus such as H.EXEC and H.REMM, resident system tasks such as the swapper and the system debugger and the object for SYSINIT. These modules are provided as files to be restored from the master SDT after a starter system is installed. The file naming convention for SDT files that are designed to be part of the resident system is OH.module.

User object modules for interrupt handlers, resident system tasks, and user-callable modules defined with the MODULE directive must also be included in the object file for SYSGEN.

A task called COMPRESS, provided on the master SDT, concatenates object code. This task selects all required system module files for SYSGEN into the SYSGEN object input file. The COMPRESS input file can be modified to contain any modules, interrupt handlers, resident tasks, or device handlers needed to configure the resident operating system. See Volume III, Chapter 4.

Any handlers or tasks that are SYSGENed must be Assembler object modules that conform to a basic structure. The modules must begin with a Halfword Address Table (HAT), end with an initialization entry point, and use the following system macros: M.EIR, M.XIR, M.MODT, M.SVCT, and M.SVCP. A task that is incorporated in the resident operating system using the PROGRAM directive is illustrated at the end of this section.

System tables are constructed and linked to the resident system modules, handlers, and user-supplied resident modules and handlers as specified by SYSGEN directives. A resident system image is formed and subsequently written to the dynamically acquired disc file specified in the SYSTEM directive. Concurrent with this process, a listing of directives is built and a load map of the system is generated. The symbol table can be saved in a system symbol table file specified with the SYMTAB directive and used in patching the system.

Sysgen uses big blocking buffers for object processing. To inhibit use of these blocking buffers, set option 1 before activation.

7.2 SYSGEN Logical File Codes

The logical file codes associated with SYSGEN are: Directives File (DIR), Object Module File (OBJ), Base Object Module File (OBR), and Listed Output (SLO).

DIR Default and Optional Assignments

The file containing SYSGEN directives is assigned to logical file code DIR.

When in the interactive mode, the default assignment for DIR is to the terminal. When in the batch mode, the default assignment is to SYC.

The optional assignment for DIR is to SYC.

OBJ Default and Optional Assignments

The file containing system resident modules is assigned to logical file code OBJ.

The default assignment for OBJ is to OH.32.

There is one optional assignment for OBJ:

```
    $ASSIGN OBJ TO pathname BLOC=Y
```

pathname	is the name of a file containing system resident object files

OBR Default and Optional Assignments

The file containing extended system modules is assigned to logical file code OBR.

The default for OBR is to NU (null device).

There is one optional assignment for OBR:

```
    $ASSIGN OBR TO pathname BLOC=Y
```

pathname	is the name of the file containing extended system modules

SLO Default and Optional Assignments

SYSGEN listed output - the directives list, the load map, and the error list are assigned to logical file code SLO.

The default assignment for SLO is to SLO.

There are three optional assignments for SLO:

$$\text{\$ASSIGN SLO TO} \left\{\begin{array}{l}\text{pathname}\\ \text{DEV=devmnc}\\ \text{LFC=UT}\end{array}\right\}$$

pathname	is the name of the file to contain SYSGEN listed output
devmnc	is the device mnemonic of a device to contain SYSGEN listed output
UT	is the logical file code UT

7.2.1 LFC Summary

LFC	Default assignment	Optional assignment
DIR	UT (interactive) SYC (batch)	SYC
OBJ	OH.32	pathname
OBR	NU (null device)	pathname

SLO	SLO	pathname
		DEV=devmnc
		LFC=UT

7.3 Options

Option	Description
1	Big blocking buffers inhibited

The following options are processed when the C.DEV bit of C.BIT is set:

Option	Description
17	Each overlay module is debuggable
18	Enter SYSGEN debugger when initializing system modules
19	Enter SYSGEN debugger after a break or abort request is issued.

7.4 Accessing SYSGEN

SYSGEN is accessed from batch or TSM in one of two ways:

```
$SYSGEN
$EXECUTE SYSGEN
```

When directive input (DIR) is assigned to a terminal, a SYS> prompt is displayed.

To exit SYS and return to TSM, use CTRLC.

7.5 SYSGEN Directives

There are three major types of SYSGEN directives:

. Section directives - begin with // and indicate the beginning of the three major sections: //HARDWARE, //SOFTWARE, and //END.

. Subsection directives - begin with / and indicate the subsections within major sections.

. Keyword directives - have no slash, and are part of a subsection within a major section.

SYSGEN directives begin in byte one of the record and contain no embedded blanks. Numeric values are represented by decimal numbers unless otherwise specified.

Directives, except TITLE, can be continued across more than one input line by placing a hyphen (-) as the last significant character on the line to be continued.

All SYSGEN directives are required unless specifically described as optional. In general, the order or presentation of SYSGEN directives is not critical within a directive subsection. However, if the order is critical, the individual directive discussion clarifies the proper order.

File SG.32 contains a starter SYSGEN directive file. It can be used as it exists or it can be modified.

SYSGEN directives are summarized below in the order they appear in the sample directive file (SG.32). Directives are described individually, in alphabetical order, on the following pages.

Directive	Description
TITLE	Permits identifying information to be printed on the listed output file
//HARDWARE	Indicates the beginning of the hardware section of directives
/PARAMETERS	Designates the parameters subsection of the //HARDWARE section
MACHINE	Specifies the type of computer for which the system is being configured
IPU	Specifies an Internal Processing Unit will be configured into the target system
/MEMORY	Designates the memory subsection of the //HARDWARE section
SIZE	Specifies the memory configuration of the target system
TYPE	See the SIZE directive

/CHANNELS	Designates the controller and device directives subsection of the //HARDWARE section
CONTROLLER	Specifies hardware channels to be configured. Nonpresent channels can be configured
DEVICE	Defines configured hardware I/O devices. Nonpresent devices can be configured. Nonconfigured discs, however, must be specified as off-line.
/INTERRUPTS	Designates the interrupt directives subsection of the //HARDWARE section
PRIORITY	Specifies the interrupt configuration and the interrupt processors to be used by the target system
/TRAPS	Designates the trap directives subsection of the //HARDWARE section
PROGRAM	Specifies the names of trap handlers to be configured on the system
SYSTRAP	Specifies the default trap handlers that are to be overridden. This directive can only be used if a PROGRAM directive is not specified.
USERPROG	Specifies the names of trap handlers and resident system modules to be configured on the system if a PROGRAM directive is not specified
/SYSDEVS	Designates the system device directives subsection of the //HARDWARE section
LOD	Specifies a system listed output device. Used as the default device in related OPCOM directives
POD	Specifies a system punched output device. Used as the default device in related OPCOM directives.
SID	Specifies a system input device (SID). Used as the default device in related OPCOM directives.
SWP	This directive is ignored
/VP	Designates the Vector Processor subsection of the //HARDWARE section
VP	Specifies the device characteristics of the Vector Processors to be configured
VPID	Specifies the unit-specific information for a Vector Processor
//SOFTWARE	Indicates the beginning of the software section of directives
/PARAMETERS	Designates the parameters subsection of the //SOFTWARE section
DELTA	Selects optional IPU/CPU scheduling algorithm

DISP	Determines the number of entries in the dispatch queue. One entry is required for each concurrently operating program. Each entry requires 58 words of resident image storage. A minimum of eight entries is required.
POOL	Specifies the size of the memory pool to reserve
MTIM	Specifies the number of real-time clock interrupts per second
NTIM	Specifies the number of real-time clock interrupts per time unit
ITIM	Specifies the time interval set for the interval timer on the RTOM module
SYSTEM	Specifies the name of the file to be used as storage for the generated MPX-32 resident image
ITLB	Generates an Indirectly Connected Task Linkage Block
MMSG	Specifies the maximum number of no-wait messages to be sent by a task
MRUN	Specifies the maximum number of no-wait run request to be sent by a task.
MNWI	Specifies the maximum number of no-wait I/O requests that can be concurrently outstanding for a task
SYMTAB	Specifies the name of the file to be used as storage for the system symbol table
PASSWORD	This directive is ignored
TQFULL	Specifies the largest time quantum a single user time-distribution task acquires before being relinked to the bottom of the priority list at its base execution priority.
TQMIN	Specifies the smallest time quantum a single user task acquires before preemption by a higher priority user time-distribution task.
BATCHPRI	Specifies the execution base priority for batch jobs
TERMPRI	Specifies the execution priority level for all tasks activated in the interactive terminal environment
PATCH	Specifies a patch area to append to the MPX-32 resident image
MODE	Requests special system operations
SVC	Increases the size of the SVC table
FLTSIZE	This directive is ignored
RMTSIZE	Increases the size of the Resourcemark Table

ACTIVATE	Specifies the names of load modules to be activated after the system has been booted. Status is not checked.
SEQUENCE	Specifies the names of load modules to be activated after the system has been booted. Status is checked.
TRACE	Allows initialization of the system trace flag word C.TRACE
DEBUGTLC	Allows specification of the console address for the System Debugger's stand-alone I/O
PCHFILE	Provides the name of the file to be used as the patch file for the generated system
DBGFILE	Provides the name of the file containing the default task debugger load module for the generated system
DPTIMO	Specifies a default time-out value to be applied to resource delays encountered when attempting to access a multiprocessor, shared-volume resource
DPTRY	Specifies the decimal number of tries to obtain a multiprocessor resource before issuing a denial
KTIMO	Specifies the number of seconds the kill directive will attempt to abort a task before it kills it
DTSAVE	Specifies the elapsed time before the date/time backup program resumes
SWAPSIZE	Specifies the initial swap file size
SWAPLIM	Specifies the minimum partial swap quantum
EXTDMPX	Specifies the location for extended memory MPX-32
/MODULES	Designates the modules subsection of the //SOFTWARE section
MODULE	Defines the name of an optional user module to be included in the MPX-32 resident image, the module number to be associated with the module, and the number of SVC callable entry points in that module.
/OVERRIDE	Designates the override subsection of the //SOFTWARE section
SYSMOD	Replaces a system module with another module, or removes modules H.ALOC, H.FISE, and H.MONS
NOBASE	Automatically excludes support for the execution of tasks utilizing base register addressing from the system image
NOCMS	Automatically excludes support for MPX-32 Rev. 1 compatibility mode services (modules H.ALOC, H.FISE, H.MONS, and H.CALM) from the system image
/PARTITION	Designates the partition subsection of the //SOFTWARE section

NAME	Defines Datapool or Global Common memory partitions
OWNER	Specifies an owner name and access rights that apply to a memory partition defined by a NAME directive
PROJECT	Specifies a project group name and access rights which apply to a memory partition defined by a NAME directive
OTHERS	Specifies access rights that apply to users of a memory partition defined by a NAME directive who are not the owner or members of the associated project group
/TABLES	Designates the tables subsection of the //SOFTWARE section
CDOTS	Specifies the size of the user CDOT array
JOBS	Specifies the maximum number of batch jobs that can be active concurrently
MDT	Enables rapid file allocation through a Memory Resident Descriptor Table (MDT)
SHARE	Specifies the number of entries in the shared memory table
TIMER	Specifies the number of timer entries to be generated in the MPX-32 resident image
/RMSTABLS	Designates the Resource Management System Tables subsection of the //SOFTWARE section
ARTSIZE	Specifies the size of the Allocated Resource Table (ART)
/FILES	Designates the system files subsection of the //SOFTWARE section
SMD	This directive is ignored
SYCSIZE	Included for compatibility only. This directive is accepted by SYSGEN, but the results are not used in job processing.
SGOSIZE	Included for compatibility only. This directive is accepted by SYSGEN, but the results are not used in job processing.
//END	Required as the last SYSGEN directive

7.5.1 ACTIVATE Directive

The ACTIVATE directive names load modules to be activated by SYSINIT immediately after the target system is booted. Status checks are not performed on these tasks. This directive is optional.

Syntax:

```
ACTIVATE=(name1,...name7)
```

name1, ...name7	are the one- to eight-character ASCII load module names to be activated, separated by commas. A maximum of seven names can be entered per directive.

7.5.2 ARTSIZE Directive

The ARTSIZE directive specifies the size of the Allocated Resource Table (ART). This directive is optional. The number of entries in the ART determines the maximum number of system resources that can be allocated in the system at one time.

Syntax:

```
ARTSIZE=entries
```

entries	is the number of entries to be reserved for the Allocated Resource Table (ART). If not specified the default is 100.

7.5.3 BATCHPRI Directive

The BATCHPRI directive specifies the execution priority level of all batch jobs.

Syntax:

```
BATCHPRI=nn
```

nn	is the two-digit decimal time-distribution priority level, 55-64, to use for batch jobs. If not specified, the default is priority 61.

7.5.4 /CHANNELS Directive

The /CHANNELS directive designates the controller and device subsection of the //HARDWARE section. This directive is required.

Syntax:

```
/CHANNELS
```

7.5.5 CDOTS Directive

The CDOTS directive specifies the size of the user CDOT array which follows the fixed communications area in lower memory. The user array can be read by any task but can only be written into by privileged tasks. C.USER is the starting point of the user array.

Syntax:

```
CDOTS=number
```

number	is the decimal number of words in the user communications variable array

7.5.6 CONTROLLER Directive

The CONTROLLER directive represents one hardware channel to be configured in the generated system. This directive is required and must be repeated for each channel to be configured. The handler key word processing permits specification of reentrancy and Controller Definition Table (CDT) generation. The type of reentrancy specified the first time a handler name appears in a CONTROLLER or DEVICE statement is used for the entire system. The CDT per Unit Definition Table (UDT) specification applies until another HANDLER keyword or CONTROLLER statement is processed. Nonpresent channels can be configured.

Syntax:

```
                                                       ┌            ( name                ) ┐
    CONTROLLER=ttcc, PRIORITY=intlev, CLASS=class      │,HANDLER= < (name ┌,I┐[,C] )      > │
                                                       │            (     │  │            ) │
    [,MUX=type] [,SUBCH=aa] [,CACHE]                   └            (     └ S┘            ) ┘
```

tt	is the two-character ASCII device code (See Table 7-1)
cc	is the two-digit hexadecimal channel number
intlev	is the hexadecimal interrupt level. If the interrupt is on an IOP device, the priority must be the same as other devices on that IOP.

class	is the device class
	
	D = 16MB Addressable E Class
	F = Extended I/O
	M = Memory Disc

name	is the one- to eight-character handler name. If not specified, the following defaults are used: H.IFXIO (XIO or IOP controllers), H.MUX0 (GPMC controllers), and H.NUXIO (null device).
I	specifies interrupt priority level reentrancy, one copy per channel
S	specifies system level reentrancy, one copy per system
C	specifies one CDT for each UDT
type	is the type of multiplex controller being configured: GPMC - General Purpose Multiplex Controller XIO - Extended I/O IOP - Input/Output Processor
aa	is the IOP subchannel the controller is connected to. This subchannel will be used to verify proper device address specifications on subsequent device directives. For example, the subchannel should match the first device address digit.
CACHE	specifies a cache controller. All devices under a cache controller are also cache.

Table 7-1

MPX-32 Device Type Codes

Dev Type Code	Device	Device Description
00	CT	Operator console (not assignable)
01	DC	Any disc unit except memory disc
02	DM	Any moving head or memory disc
03	DF	Any fixed head disc
04	MT	Any magnetic tape unit
05	M9	Any 9-track magnetic tape unit*
06	M7	Any 7-track magnetic tape unit*
08	CR	Any card reader
0A	LP	Any line printer
0B	PT	Any paper tape reader-punch
0C	TY	Any teletypewriter (other than console)
0D	CT	Operator console (assignable)
0E	FL	Floppy disc
0F	NU	Null device
10	CA	Communications adapter (binary synchronous/ asynchronous)
11	U0	Available for user-defined applications
12	U1	Available for user-defined applications
13	U2	Available for user-defined applications
14	U3	Available for user-defined applications
15	U4	Available for user-defined applications
16	U5	Available for user-defined applications
17	U6	Available for user-defined applications
18	U7	Available for user-defined applications
19	U8	Available for user-defined applications
1A	U9	Available for user-defined applications
1B	LF	Line printer/floppy controller (used only with SYSGEN)
N/A	ANY	Any nonfloppy disc except memory disc

* When both 7- and 9-track magnetic tape units are configured, the designation must be 7-track.

Notes:

Multiple controller directives are invalid on a single channel, even if the devices configured on the channel have mixed device types.

Extended I/O handlers default to system reentrant handlers. The extended I/O interrupt fielder (H.IFXIO) is channel reentrant. One copy should be specified for each extended I/O channel configured by using the I parameter.

GPMC device handlers default to system reentrant handlers.

If a line printer and a floppy disc are configured on the same IOP channel, only one CONTROLLER directive should be used. Multiple DEVICE directives specify the device and handler.

The XIO common subroutines (H.XIOS) are not named within the SYSGEN directive file. They are automatically included during SYSGEN if MUX=XIO or MUX=IOP is specified.

The GPMC subroutines (H.GPMCS) are not named within the SYSGEN directive file. They are automatically included during SYSGEN if MUX=GPMC is specified.

Memory discs must be configured on channel 00 as DM00.

Table 7-2
Device Handlers

Handler Module Name	Description
H.ASMP	GPMC Async Comm. for Terminals
H.A8XIO	IOP 8-Line Asynchronous Handler
H.BSMP	Bisync GPMC
H.CTXIO	IOP Console
H.DCXIO	XIO Disc*
H.F8XIO	IOP 8-Line Full Duplex Handler
H.HSDG	High Speed Data Handler
H.LPXIO	XIO Line Printer
H.MDXIO	Memory Disc Handler
H.MTXIO	XIO Magnetic Tape**
H.NUXIO	NULL Device Handler
H.SLMP	SLIM Handler

* For use with all F-class discs including floppy discs.
** For use with low speed and high speed F-class magnetic tape processors.

7.5.7 DBGFILE Directive

The DBGFILE directive names the permanent file containing the default task debugger load module for the system being generated. This directive is optional. If not specified, the default filename is MPXDB.

Syntax:

```
DBGFILE=filename
```

filename is the one- to eight-character ASCII file name of the file containing the default task debugger load module

7.5.8 DEBUGTLC Directive

The DEBUGTLC directive specifies the console address for the System Debugger's stand-alone I/O. This directive is optional.

Syntax:

```
DEBUGTLC=cc
```

cc is a two-digit hexadecimal channel number. If not specified, the default is X'7E'.

7.5.9 DELTA Directive

The DELTA directive selects the optional IPU/CPU scheduler. When DELTA is specified, H.EXEC2 replaces H.EXEC, and H.CPU2 replaces H.CPU. See Volume I, Section 2.5.

Syntax:

```
    DELTA = cc
```

cc	is a two-digit decimal number between 1 and 54 specifying the IPU bias task boost value

Note:

The IPU directive must be present to perform the substitution. If it is not specified, the DELTA directive is ignored.

Do not include H.CPU2 or H.EXEC2 in the PROGRAM directive, as it is automatically selected.

7.5.10 DEVICE Directive

The DEVICE directive defines the configured hardware I/O devices. This directive is required. The handler key word processing permits specification of reentrancy and of Controller Definition Table (CDT) generation. The type of reentrancy specified the first time a handler name appears in a CONTROLLER or DEVICE statement is used for the entire system. The CDT per Unit Definition Table (UDT) specification applies until another HANDLER key word or CONTROLLER statement is processed. Nonpresent devices can be configured. Then, when the devices are added, a warm start with the devices marked on-line allows them to be used.

The null device, NU, must be included in every configuration.

Syntax:

```
                             ┌        ( mdsize            ) ┐
                             │        (mdsize    [,D] )   │
            { aa        }    │         devcode              │
  DEVICE=   { (aa,n,inc)}    │ ,DISC= (devcode  ┌,M┐ )     │ [,SHR] [,DTC=tt]
                             │                  │ P│        │
                             └                  └ D┘        ┘

          [,LINSIZ=x]          [,PAGE=y] [,SPOOL=(code,code,...) ] [,FEOP]

          ┌            ( name                   ) ┐
          │ ,HANDLER=  {           ┌,I┐          } │ [,PHYSA=ccaa] [,OFF] [,IOQ=mode]
          │            (name       └,S┘  [,C] ) ) │
          └                                       ┘
          [,CACHE] [,QITD] [,DEAL] [,START=start] [,ANSI] [,FULL] [,SLPR]
```

aa	is the two-digit hexadecimal device subaddress. The first digit is the controller address; the second digit is the device address.
n	is an optional parameter specifying the decimal number of devices starting at the subaddress
inc	is an optional parameter specifying the hexadecimal address increment for each additional device. If not specified, the default is one.
mdsize	specifies the size in KBs if the device is a memory disc

devcode	is the five-character device code for disc storage devices. It is not required for devices other than disc. See Table 7-3.
M	specifies the device is a multiport disc that is compatible with an MPX-32 Release 3.3 or later system
P	specifies the device is a multiport disc that is compatible with an MPX-32 Release 3.2C or earlier system
D	specifies the device is a dual-ported disc (for compatibility only)
SHR	specifies the device is a shared device
tt	is a two-character device code from Table 7-1. If not specified, the device code specified on the associated CONTROLLER directive is used.
x	specifies characters per line for TSM devices
y	specifies lines per screen for TSM devices or number of lines per page for listed output devices. Zero prevents ENTER CR FOR MORE message displays.
code	indicates the device is available for automatic selection as the destination device for spooled printed and punched output. The specification consists of two-character codes separated by commas. The codes and their use for output are:

BL	batch	SLO
BB	batch	SBO
RL	real-time	SLO
RB	real-time	SBO

FEOP	specifies that J.SOUT inhibits the normal initial form feed and terminates all printing with an additional form feed, if specified for LP device. FEOP is applicable only on electrostatic printers and laser printers that retain the last page of output until a form feed (EJECT) is received.
name	is the one- to eight-character handler name. If not specified, the following defaults are used: H.NUXIO (null device), H.DCXIO (any XIO or IOP disc including floppy disc), H.MTXIO (magnetic tape), H.CTXIO (operator console), H.LPXIO (line printer), H.F8XIO (XIO and IOP terminals), and H.ASMP (GPMC terminals). This parameter must be specified for a memory disc (H.MDXIO). See Table 7-2 for handler module names.
I	specifies interrupt priority level reentrancy, one copy per channel
S	specifies system level reentrancy, one copy per system
C	specifies one CDT for each UDT
PHYSA	specifies the physical (bus) channel address and device subaddress for devices. Used when logical channel and subaddress do not match the physical channel and subaddress.
cc	is the two-digit hexadecimal channel address
OFF	specifies that the device or devices described by this directive are to be SYSGENed in an off-line state
mode	is used by the IOP and GPMC to indicate the I/O queue entries are to be linked from the UDT (IOQ=DEV) or from the CDT (IOQ=CONT). The default mode is IOQ=CONT. However, most handlers provide their own initialization

	for this parameter and ignore any value specified by SYSGEN. Therefore, this parameter should not be used for any standard MPX-32 handler supplied by Gould CSD unless such a requirement is specified in the installation instructions for that handler.
CACHE	specifies a cache device
QITD	specifies a quarter-inch tape drive
DEAL	specifies that memory for the memory disc is not allocated at system initialization. DEAL is valid only for single-ported memory disc. If DEAL is supplied with a dual-ported memory disc, it is ignored.
start	specifies the decimal start map block number of the memory disc. If not specified, default is the highest possible address in memory.
FULL	allows full-duplex mode of operation if FULL and NOECHO are specified in LOGONFILE. This causes a UDT to be created for both the read and write subaddresses even though only the read subaddress is specified. Valid only for 8-line asynch devices.
ANSI	specifies this drive is used only for ANSI tape processing. See the ANSI chapter for details. This parameter does not restrict the system administrator.
SLPR	indicates the device is a serial printer. See Notes.

Notes:
The SHR parameter does not apply to TSM devices.
Extended I/O handlers default to system reentrant handlers.
GPMC device handlers default to system level reentrant handlers.
A CDT is generated each time a HANDLER key word appears, except on the first DEVICE statement following a CONTROLLER statement with a HANDLER specification.

For XIO devices, the subaddress specified as the aa parameter is two hexadecimal digits. The first digit is the controller address and the second digit is the device address. For IOP devices, the digit specified for the controller address must be equal to the digit specified in the CONTROLLER directive SUBCH parameter. Except for floppy discs, all F-class discs must specify an even device address. The device address is determined by the unit address plug installed in the drive or by switches in the drive.

The device address is the unit address multiplied by two and converted to its hexadecimal equivalent (for example, unit address one is device address two, unit address seven is device address E). If more than one disc is configured using one DEVICE directive, the device address and the increment (inc) must be even numbers. F-class cartridge module drives adhere to the same conventions. SYSGEN automatically configures the captive media portion of the drive at the next sequential odd device address. Once configured, cartridge module drives are treated as individual devices. The removable media portion is treated as a moving head disc at the even device address, and the captive media portion is treated as a fixed head disc at the next sequential odd device address.

The CACHE parameter can be applied to ACM devices (H.F8XIO), to quadruple the normal I/O time out. This is intended for slow spooling devices with large internal buffers using XON/XOFF (WXON) flow control.

A memory disc cannot be used as a cache device. An error occurs if both memory disc and cache are specified.

When SPLR is specified: the HANDLER parameter must equal H.F8XIO; the LINESIZE and PAGE directives must be used (suggested settings LINSIZ = 133 and PAGE = 60); the DEVICE FULL parameter must not be specified. If the printer has a large buffer, the DEVICE CACHE parameter should be specified to extend the normal time-out. This will account for long I/O wait times (XOFF) while the buffer is purging.

Table 7-3
Disc Device Codes

Disc	Disc Code
Reserved	FE004
1.2MB Floppy Disc - Class F Device	FL001
40MB Moving Head Disc - Class F Device	MH040
80MB Moving Head Disc - Class F Device	MH080
160MB Moving Head Disc - Class F Device	MH160
300MB Moving Head Disc - Class F Device	MH300
340MB (Winchester) Moving Head Disc - Class F Device	MH340
600MB Moving Head Disc - Class F Device	MH600
5MB Fixed Head Disc - Class F Device	FH005
32MB Cartridge Module Disc - Class F Device	CD032
Any Nonfloppy Disc - Class F Device	ANY

Memory discs must be configured on the even subaddresses of channel 00. The subaddress of the memory disc cannot be the same as the null device subaddress.

7.5.11 DISP Directive

The DISP directive determines the number of entries in the dispatch queue. One entry is used for each concurrently operating task.

Syntax:

```
DISP=entries
```

entries specifies the decimal number of entries in the dispatch queue. This value cannot exceed 255 and must be at least eight. If not specified, the default is ten.

7.5.12 DPTIMO Directive

The DPTIMO directive specifies a time-out value for resource delays encountered when attempting to access a multiprocessor, shared-volume resource. This directive is optional.

Syntax:

```
DPTIMO=num
```

num is the decimal number of timer units to wait. If not specified, the system assigns a sufficient number of timer units to result in a one-second delay.

7.5.13 DPTRY Directive

The DPTRY directive specifies the decimal number of times that a system tries to obtain a multiprocessor resource before returning a denial.

Syntax:

```
DPTRY=num
```

num	is the decimal number of tries to obtain a multiprocessor resource. If not specified, the default is zero which causes the system to try until the resource is obtained.

7.5.14 DTSAVE Directive

The DTSAVE directive specifies the amount of time to elapse before J.DTSAVE is resumed. This directive is optional.

Syntax:

```
DTSAVE=time
```

time	is the number of minutes to elapse before J.DTSAVE resumes. If not specified, the default is five minutes.

7.5.15 //END Directive

The //END directive is required as the last directive in the SYSGEN directives file.

Syntax:

```
//END
```

7.5.16 EXTDMPX Directive

The EXTDMPX directive designates the location of extended memory MPX-32. If no parameter is specified for EXTDMPX, the default is MINADDR.

Syntax:

$$\text{EXTDMPX}\ \begin{Bmatrix}\text{MINADDR}\\ \text{MAXADDR}\\ \text{startmap}\end{Bmatrix}$$

MINADDR	positions extended memory MPX-32 within the task's TSA
MAXADDR	positions extended memory MPX-32 at the top of the task's logical memory
startmap	is the decimal starting map where extended MPX-32 will be positioned

7.5.17 /FILES Directive

The /FILES directive designates the system files subsection of the //SOFTWARE section. This section is included for compatibility only.

Syntax:

```
/FILES
```

7.5.18 FLTSIZE Directive

The FLTSIZE directive is included for compatibility and is ignored by SYSGEN. Items following this directive on the same line are ignored.

Syntax:

```
FLTSIZE
```

7.5.19 //HARDWARE Directive

The //HARDWARE directive indicates the beginning of the hardware configuration section of directives. This directive is required.

Syntax:

```
//HARDWARE
```

7.5.20 /INTERRUPTS Directive

The /INTERRUPTS directive designates the interrupt subsection of the //HARDWARE section. This directive is required.

Syntax:

```
/INTERRUPTS
```

7.5.21 IPU Directive

The IPU directive specifies an IPU will be configured into the target system. This directive is optional.

Syntax:

```
IPU
```

7.5.22 ITIM Directive

The ITIM directive provides the expiration time interval of the RTOM interval timer.

Syntax:

```
ITIM=value
```

value	is the quantity expressed in tenths of microseconds of an RTOM interval-timer time quantum. For example, 38.4 microseconds is represented as 384. If not specified, the default is 38.4 microseconds.

7.5.23 ITLB Directive

The ITLB directive generates an Indirectly Connected Task Linkage Block. One ITLB directive is required for each indirectly linked task concurrently active in the system.

Syntax:

```
ITLB=intlevel
```

intlevel	is the two-character hexadecimal interrupt priority level where the task will be indirectly connected

7.5.24 JOBS Directive

The JOBS directive specifies the maximum number of batch jobs that can be concurrently active. Any number is valid. However, a large number can cause an increase in system response time due to increased system overhead associated with J.TSM processing. Therefore, the following numbers are suggested: 3 on a 32/27 and 10 on all other CONCEPT/32 computers.

Syntax:

```
JOBS=number
```

number	is the number of entries in the job table. If not specified, the default is one.

7.5.25 KTIMO Directive

The KTIMO directive specifies the number of seconds the kill directive will attempt to abort a task before it kills it. The default is ten seconds. If zero is specified, an immediate kill is performed.

Syntax:

```
KTIMO=number
```

number	is the decimal number of seconds to attempt an abort

7.5.26 LOD Directive

The LOD directive specifies a system listed output device that is the default device for SLO. This directive is required if a line printer is configured.

Syntax:

```
LOD=devmnc [,IBP]
```

devmnc	is the two-character device code, the two-digit hexadecimal channel number, and the two-digit hexadecimal device subaddress. See Table 7-1.
IBP	inhibits banner page produced on the SLO device

7.5.27 MACHINE Directive

The MACHINE directive indicates the type of computer for which the resultant system is being configured. This directive is optional. If specified, it must be the first directive in the /PARAMETERS subsection.

Syntax:

```
MACHINE=type
```

type	is the machine type 3227, 3267, 3287, or 3297. If 3227 is specified, the cache parity trap handler H.IP10 is not included in the system as a default trap handler. If not specified, any CONCEPT/32 machine can be used.

7.5.28 /MEMORY Directive

The /MEMORY directive designates the memory subsection of the //HARDWARE section. This directive is required.

Syntax:

```
/MEMORY
```

7.5.29 MDT Directive

The MDT directive enables rapid file allocation through a Memory Resident Descriptor Table (MDT). The MDT's parameters are defined by this directive, and module H.MDT is included in the resident operating system.

Syntax:

```
MDT = n [,BLOC=b]
```

n	specifies the decimal number of MDT entries. MDT entries are 192 words in length. To accommodate collision resolution, the actual number allocated is 25% greater than the specified number.
b	specifies the decimal starting physical map block number. If not specified, the default is the highest contiguous memory available.

If the specified starting map block is not available, initialization processing halts and an abort code is displayed at system initialization.

7.5.30 MMSG Directive

The MMSG directive specifies the maximum number of no-wait messages to be sent by a task.

Syntax:

```
MMSG=num
```

num	is the maximum number of no-wait messages to be sent by a task. If not specified, the default is five.

7.5.31 MNWI Directive

The MNWI directive specifies the maximum number of no-wait I/O requests that can be concurrently outstanding for a task.

Syntax:

```
MNWI=num
```

num	is the maximum number of no-wait I/O requests that can be concurrently outstanding for a task. If not specified, the default is five.

7.5.32 MODE Directive

The MODE directive requests the following special system operations:

- Continuous Batch - Batch stream input from SID is processed until the job control statement $$$ is encountered. All $$ job control statements are ignored.
- Inhibit Banner Page - Suppresses the banner page produced by system output tasks when processing SLO files.
- Inhibit Mount Message - Suppresses the mount message produced by J.MOUNT. This specification is not valid with multivolume magnetic tape operations.
- Dump - Performs a dump if an independent task aborts.
- Scratchpad Locations - Does not zero unused CPU scratchpad locations at IPL.
- Inhibit Operator Intervention - Suppresses all prompts normally displayed on the system console during system initialization.

Syntax:

$$\text{MODE=}\left\{\begin{array}{l}\text{SCBT}\\ \text{SIBP}\\ \text{SIMM}\\ \text{DUMP}\\ \text{LSPA}\\ \text{SNOP}\end{array}\right\}$$

SCBT	sets continuous batch
SIBP	sets inhibit banner page
SIMM	sets inhibit mount messages for nonmultivolume magnetic tape operations
DUMP	sets dump request for aborting real-time tasks
LSPA	inhibits zeroing of unused scratchpad locations during IPL processing. If not specified, the IPL process zeroes all unused CPU scratchpad locations.
SNOP	sets inhibit operator intervention

7.5.33 MODULE Directive

The MODULE directive names optional user modules to be included in the MPX-32 resident image. One MODULE directive is required for each user module to be included.

Syntax:

```
    MODULE=(name,module,entpoints)
```

name	is the one- to eight-character ASCII module name. The name can not contain a comma, an equal sign, or a left or right parenthesis. The module name N.ACXRF is reserved for use only as module number 12.
module	is a two-digit decimal number representing the internal identification number of the module. Modules 00 through 12 are reserved for MPX-32 modules. A module number of 12 is allowed only if the module name is N.ACXRF.
entpoints	is the hexadecimal number of entry points contained in this module. The last entry point of each user-supplied module is an initialization entry point called by SYSGEN during construction of the MPX-32 image and is overlaid following execution. This entry point should not be included in the entpoints value supplied on the directive.

Module initialization must include the system macros M.EIR, M.XIR, M.MODT, and M.SVCT.

7.5.34 /MODULES Directive

The /MODULES directive designates the modules subsection of the //SOFTWARE section. This directive is optional.

Syntax:

```
    /MODULES
```

7.5.35 MRUN Directive

The MRUN directive specifies the maximum number of no-wait run requests to be sent by a task.

Syntax:

```
MRUN=num
```

num	is the maximum number of run requests that can be sent by a task. If not specified, the default is five.

7.5.36 MTIM Directive

The MTIM directive provides the number of real-time clock interrupts per second.

Syntax:

```
MTIM=number
```

number	is the number of real-time clock interrupts per second. If not specified, the default is 60.

7.5.37 NAME Directive

The NAME directive defines static Datapool or Global Common memory partitions. If multiple static partitions are defined within a map block, only one partition can be included in the task's logical address space at a given time. One NAME directive is required for each memory partition desired.

Syntax:

```
NAME=name,SIZE=np,STRTPG=sp,MAP=pm
```

name	is the one- to eight-character partition name
np	is the decimal number of pages for the partition. The maximum number of pages is 304 (152KW).
sp	is the logical hexadecimal starting protection granule for the partition
pm	is the starting physical decimal map block number. A map block is 2KW.

7.5.38 NOBASE Directive

The NOBASE directive automatically excludes support in H.EXEC, H.IP00, H.IP04, H.IP0F, and H.TAMM for the execution of tasks utilizing base mode addressing from the system image.

Syntax:

```
NOBASE
```

7.5.39 NOCMS Directive

The NOCMS directive automatically excludes support for MPX-1.x compatibility mode services (modules H.ALOC, H.FISE, H.MONS, and H.CALM) from the system image. Software that calls a removed compatibility mode system module is aborted with an SV02 abort code.

Syntax:

```
NOCMS
```

7.5.40 NTIM Directive

The NTIM directive provides the number of real-time clock interrupts per time unit.

Syntax:

```
NTIM=number
```

number	is the number of clock interrupts per time unit. If not specified, the default is 60.

7.5.41 OTHERS Directive

The OTHERS directive specifies access rights allowed to users who are not the owner or members of the project group associated with a partition previously defined with a NAME directive. If more than one NAME directive has been specified, the OTHERS directive only applies to the last one specified. This directive is optional. If not specified, default is OTHERS=(R).

If both read and write access are desired, a comma must be used as a delimiter, for example, OTHERS=(R,W).

The OTHERS directive must always be specified last when OWNER and/or PROJECT directives are also specified. The order is OWNER, PROJECT, OTHERS.

Syntax:

```
OTHERS=( [R] [,W] [,N] )
```

R	specifies read access is allowed
W	specifies write access is allowed
N	specifies no access is allowed

7.5.42 /OVERRIDE Directive

The /OVERRIDE directive designates the override subsection of the //SOFTWARE section. This directive is optional.

Syntax:

```
/OVERRIDE
```

7.5.43 OWNER Directive

The OWNER directive specifies the owner name and associated access rights that apply to the partition previously defined in a NAME directive. If more than one NAME directive has been specified, the OWNER directive only applies to the last one specified. This directive is optional. If not specified, default is OWNER=(SYSTEM,R,W).

Syntax:

```
OWNER=(name [,R] [,W] [,N] )
```

name	is the one- to eight-character owner name to be associated with the partition
R	specifies read access is allowed
W	specifies write access is allowed
N	specifies no access is allowed

7.5.44 /PARAMETERS Directive

The /PARAMETERS directive designates beginning of the parameters subsection of the //HARDWARE and //SOFTWARE sections. This directive is required and it must be the first subsection of the //HARDWARE and //SOFTWARE sections.

Syntax:

```
/PARAMETERS
```

7.5.45 /PARTITION Directive

The /PARTITION directive designates the memory partition and/or global common subsection of the //SOFTWARE section. This directive is optional.

Syntax:

```
/PARTITION
```

7.5.46 PASSWORD Directive

The PASSWORD directive is included for compatibility and is ignored by SYSGEN. Items following this directive on the same line are ignored.

Syntax:

```
PASSWORD
```

7.5.47 PATCH Directive

The PATCH directive specifies a system patch area to append to the MPX-32 resident image. This directive is optional. If not specified, a system patch area is not reserved.

Syntax:

```
PATCH=number
```

number	specifies the hexadecimal number of bytes to be added to the resident MPX-32 system as a system patch area. If the extended mode was activated at SYSGEN this specifies the hexadecimal number of bytes to be added in the first 16KW of memory. Note: the system patch area is restricted by SYSGEN to the first 16KW of memory. SYSGEN also restricts the area from exceeding the first 16KW of memory.

7.5.48 PCHFILE Directive

The PCHFILE directive names the permanent file which the generated system uses as its patch file. SYSGEN does not create the file; it must be supplied if patches are to be generated. This directive is optional.

Syntax:

```
PCHFILE=filename
```

filename	is the one- to eight-character ASCII file name of the file to contain patches for the generated system. If not specified, the default is M.PATCH.

7.5.49 POD Directive

The POD directive specifies the system punched output (SPO) device to use as the default device for SBO.

Syntax:

```
POD=devmnc
```

devmnc	is the two-character device code, the two-digit hexadecimal channel number, and the two-digit hexadecimal device subaddress. See Table 7-1.

7.5.50 POOL Directive

The POOL directive specifies the size of the memory pool to be reserved at SYSGEN time. The specified size of the memory pool is rounded up to the end of the current map block.

Syntax:

```
POOL=words
```

words	is the decimal number of words to be reserved. If not specified, the default is 1000 words.

7.5.51 PRIORITY Directive

The PRIORITY directive specifies the interrupt configuration and interrupt processors to use for the target system. This directive is optional. The interrupt processors provided are: Attention (H.IP13), Real-time Clock (H.IPCL), and CPU Scheduler (H.IPIT). The CPU scheduler is directly connected to the lowest interrupt priority level in the system. If a machine directive is not specified, the lowest interrupt level is 5F.

If an IPU is SYSGENed into the system, an IPU accounting processor (H.IPUIT) is available. This processor is directly connected to the IPU accounting interval timer at any priority. The lowest available priority is recommended.

I/O channel interrupt levels on a CONCEPT/32 are 04 to 13.

Syntax:

```
PRIORITY=intlev,RTOM=(channel,subaddress) [,PROGRAM=name] [,INTV]
```

intlev	is the two-digit hexadecimal interrupt level. The lowest levels are 5F on a 32/27, and 6F on all other CONCEPT/32 computers.
channel	is the two-digit hexadecimal RTOM board or IOP RTOM function channel address
subaddress	is the two-digit hexadecimal RTOM board or IOP RTOM function subaddress, which is the one's complement of the RTOM relative physical priority for the interrupt level.
name	is the one- to eight-character name of program located in the file assigned to LFC OBJ. If the program is to be directly connected to this interrupt level and SYSGENed with the resident operating system, the name must be supplied. If tasks are to be indirectly connected to the interrupt level, no program name should be supplied. An indirectly connected task linkage block (ITLB) is defined using the ITLB directive.
INTV	indicates the level is an RTOM interval timer. A device entry, whose address is equal to the interrupt priority level, is being built in the scratchpad.

7.5.52 PROGRAM Directive

The PROGRAM directive specifies the program names of the trap processors to be configured on the resident system. This directive is optional. If specified, no defaults are in effect and all trap handlers to be included on the system must be listed. If not specified, the following defaults are used:

H.IP00	H.IP09
H.IPAS	H.IP0C
H.IP02	H.IP0F
H.IP03	H.IP10 (32/27 excluded)
H.IP04	H.IP13
H.IP05	H.IPHT
H.IP06	H.IPU
H.IP07	H.IPUAS (if an IPU is configured)
H.IP08	H.CPU

If the defaults are required along with other optional system modules, the USERPROG directive should be used instead of the PROGRAM directive.

Syntax:

```
     PROGRAM=(name1, . . . name7)
```

name1, ...name7	are one- to eight-character ASCII program names of the trap handlers assigned to LFC OBJ, separated by commas. A maximum of seven names are entered per directive.

Note: Do not include H.CPU2 or H.EXEC2 in this directive, because the DELTA directive automatically selects them.

7.5.53 PROJECT Directive

The PROJECT directive specifies the project group name and associated access rights which apply to the partition previously defined with a NAME directive. If more than one NAME directive has been specified, the PROJECT directive only applies to the last one specified. This directive is optional. If not specified, the default is PROJECT= (SYSTEM,R,W).

The OWNER directive must precede PROJECT and OTHERS directives if specified. The order is OWNER, PROJECT, OTHERS.

Syntax:

```
     PROJECT=(name [,R] [,W] [,N] )
```

name	is the one- to eight-character project group name associated with the partition
R	specifies read access is allowed
W	specifies write access is allowed
N	specifies no access is allowed

7.5.54 /RMSTABLS Directive

The /RMSTABLS directive designates the Resource Management System Tables subsection of the //SOFTWARE section. This directive is required if the ARTSIZE directive is specified.

Syntax:

```
     /RMSTABLS
```

7.5.55 RMTSIZE Directive

The RMTSIZE directive increases the size of the Resourcemark Table. This directive is optional.

Syntax:

```
     RMTSIZE=num
```

num	is a decimal number between 64 and 1000. If not specified, the default is 64.

7.5.56 SEQUENCE Directive

The SEQUENCE directive names load modules to be activated in a sequential manner by SYSINIT immediately after the target system is booted. Tasks activated with this directive must complete and exit before the next task is activated. This directive is optional.

The system console cannot be used as a TSM terminal until all specified tasks are complete.

SYSINIT checks the completion status of each task as it exits and displays a message on the system console if abnormal status was returned. A maximum of 70 bytes of call back information (in ASCII) can be sent using the Receiver Exit Block (RXB) when exiting a run receiver task. This information, if present, is displayed by SYSINIT on the system console. The user status byte, if not zero, is also displayed in decimal on the system console.

If control switch 5 is set before booting the system, task names listed in the SEQUENCE directive will not be activated as part of the target system boot (load) process.

Syntax:

```
SEQUENCE=(name1,...name7)
```

name1,...name7	are the one- to eight-character ASCII load module names to be activated, separated by commas. A maximum of seven names are entered per directive.

7.5.57 SGOSIZE Directive

The SGOSIZE directive is included for compatibility only. This directive is accepted by SYSGEN, but the results are not used in job processing.

Syntax:

```
SGOSIZE=blocks
```

blocks	is the number of 192-word blocks of disc space to be allocated for each SGO file

7.5.58 SHARE Directive

The SHARE directive specifies the number of entries in the shared memory table. Each entry defines a shared memory area; for example, CSECT, Global Common, Datapool, or Shared Image. This directive is optional. A minimum of 2 must be specified to use the Text Editor and Volume Manager.

Syntax:

```
SHARE=number
```

number	is the number of entries in the shared memory table. It must be sufficient to define all static and dynamic partitions. If not specified, the default is 0.

7.5.59 SID Directive

The SID directive specifies a System Input Device (SID) to use as the default device in related OPCOM commands.

Syntax:

```
SID=devmnc [,DENSITY=density] [,PARITY=parity]
```

devmnc	is the two-character device code, the two-digit hexadecimal channel number, and the two-digit hexadecimal device subaddress. See Table 7-1.
density	is used only for 7-track magnetic tape. H is for high density. L is for low density.
parity	is used only for 7-track magnetic tape. E for even parity. O for odd parity.

7.5.60 SIZE Directive

The SIZE directive specifies the memory configuration of the target system. This directive is required. Using multiple SIZE directives, memory types must be specified in order of configuration in memory; that is, from low address to high address.

Syntax:

```
SIZE=nn,TYPE=c,CLASS=x [,RESERVED] [,MULTI [,MS2] ]
```

nn	is the memory size in decimal number of map blocks. Maximum value is 2048.
c	is memory type E, H, S, N, H1, H2, or H3. E, H, and S are memory types as described in Volume I. Absent memory is indicated by N. H1, H2, and H3 are shadow memory types.
x	is memory class C or S: C is core memory S is semiconductor memory
RESERVED	reserves shadow memory for requesting tasks. Valid only with types H1, H2, or H3.
MULTI	specifies the multiprocessor memory flag is set indicating this memory is not to be initialized during the startup procedure
MS2	identifies the shared memory configured as Multiprocessor Shared Memory System $(MS)^2$ rather than the MBC/MBA Multiport Memory System. Multiprocessor Shared Memory System $(MS)^2$ is required for any dual-ported memory disc.

Example:

This is a multiprocessor shared memory example for two systems with 2MB of multiprocessor shared memory and 4MB of non-shared memory. The two systems are to share 128KW of memory. The balance of the 2MB is divided for use by the two systems.

System 1:

```
/MEMORY
SIZE=512,TYPE=S,CLASS=S                     ;4MB OF NON-SHARED
                                           ;MEMORY
SIZE=64,TYPE=S,CLASS=S,MULTI                ;128KW TO BE SHARED
SIZE=112,TYPE=S,CLASS=S,MULTI               ;224KW TO BE USED BY
                                           ;SYSTEM 1
SIZE=80,TYPE=N,CLASS=S,MULTI                ;160KW TO BE USED BY
                                           ;SYSTEM 2
.
.
/PARTITION
NAME=GLOBAL01,SIZE=256,STRTPG=100,MAP=512
.
.
```

System 2:

```
/MEMORY
SIZE=512,TYPE=S,CLASS=S                     ;4MB OF NON-SHARED
                                           ;MEMORY
SIZE=64,TYPE=S,CLASS=S,MULTI                ;128KW TO BE SHARED
SIZE=112,TYPE=N,CLASS=S,MULTI               ;224KW TO BE USED BY
                                           ;SYSTEM 1
SIZE=80,TYPE=S,CLASS=S,MULTI                ;160KW TO BE USED BY
                                           ;SYSTEM 2
.
.
/PARTITION
NAME=GLOBAL01,SIZE=256,STRTPG=100,MAP=512
.
.
```

Notes:

A system can have only one type of shared memory configured.

If shadow memory is not configured properly, the following SYSGEN directive error message is displayed:

```
*** SHADOW MEMORY CONFIGURATION VIOLATION
```

If RESERVED is used with nonshadow memory types, the following SYSGEN directive error message is displayed:

```
*** ONLY SHADOW MEMORY CAN BE RESERVED
```

For more information on shadow memory, refer to Volume III, Chapter 10.

7.5.61 SMD Directive

The SMD directive is included for compatibility and is ignored by SYSGEN. Items following this directive on the same line are ignored.

Syntax:

```
    SMD
```

7.5.62 //SOFTWARE Directive

The //SOFTWARE directive indicates the beginning of the software configuration section of directives. This directive is required.

Syntax:

```
//SOFTWARE
```

7.5.63 SVC Directive

The SVC directive increases the size of the SVC type 1 table. This directive is optional.

Syntax:

```
SVC=num
```

num	is the hexadecimal number, X'80' through X'FF' of entries allowed in the SVC Type 1 table. If not specified, the default is 7F.

7.5.64 SWAPLIM Directive

The SWAPLIM directive specifies the minimum partial swap quantum. This directive is optional. If not specified or zero is specified, and if memory allows, swapper swaps on demand one map block at a time. If a number n is supplied, swapper swaps out n map blocks of the task or the entire task, whichever is smaller. To achieve total task swapping for a 32/27 or a 32/87 n must be at least 128. To achieve total task swapping for a 32/67 or a 32/97 use 2048 for n. The value specified is dependent on system use. When heavy swapping is anticipated and small tasks are running, a value between 7 and 15 is recommended.

Syntax:

```
SWAPLIM = n
```

n	is the minimum partial swap quantum in map blocks

7.5.65 SWAPSIZE Directive

The SWAPSIZE directive specifies the initial swap file size. This directive is optional. If not specified, the default is two times the configured memory. If zero is specified, swapping is inhibited.

Syntax:

```
SWAPSIZE=size
```

size	is the initial swap file size in megabytes

7.5.66 SWP Directive

The SWP directive is included for compatibility and is ignored by SYSGEN. Items following this directive on the same line are ignored.

Syntax:

```
SWP
```

7.5.67 SYCSIZE Directive

The SYCSIZE directive is included for compatibility only. This directive is accepted by SYSGEN, but the results are not used in job processing.

Syntax:

```
SYCSIZE=blocks
```

blocks	is the number of 192-word blocks of disc space to be allocated for each SYC file

7.5.68 SYMTAB Directive

The SYMTAB directive names the permanent file where the symbol table of the generated system is written. If the file does not currently exist, SYSGEN creates the file. This directive is required.

Syntax:

```
SYMTAB=filename
```

filename	is the ASCII file name of the system symbol table file

7.5.69 /SYSDEVS Directive

The /SYSDEVS directive designates the system device subsection of the //HARDWARE section. This directive is required.

Syntax:

```
/SYSDEVS
```

7.5.70 SYSMOD Directive

The SYSMOD directive replaces system modules with other modules or removes the compatibilty mode system modules H.ALOC, H.FISE, and H.MONS. One SYSMOD directive is required for each system module to be replaced.

Syntax:

```
SYSMOD=name1,REPMOD=name2
```

name1	is the one- to eight-character ASCII name of the system module to be replaced or removed
name2	is the one- to eight-character ASCII name of the replacement module or NULL

7.5.71 SYSTEM Directive

The SYSTEM directive names the permanent file where the generated system is written. If the file does not currently exist, SYSGEN creates the file as a system file. This directive is required.

Syntax:

```
SYSTEM=sysfile
```

sysfile	is the one- to eight-character name of the file to contain the resident system image generated at SYSGEN. This file name must not be the same as the name of the current default image.

7.5.72 SYSTRAP Directive

The SYSTRAP directive selectively overrides any of the default trap handlers listed under the PROGRAM directive.

The SYSTRAP directive cannot be used if a PROGRAM directive is used.

Syntax:

```
SYSTRAP=name1,REPTRAP=name2
```

name1	is the default trap handler name to be overridden (e.g., H.IP10)
name2	is the one- to eight-character ASCII program name assigned LFC OBJ used in place of name1

7.5.73 /TABLES Directive

The /TABLES directive designates the tables subsection of the //SOFTWARE directives. This directive is required.

Syntax:

```
/TABLES
```

7.5.74 TERMPRI Directive

The TERMPRI directive specifies the execution priority level for all tasks activated in the interactive terminal environment.

Syntax:

```
TERMPRI=nn
```

nn	is the two-digit decimal time-distribution priority level, 55 to 64, for interactive processing. If not specified, the default is priority 60.

7.5.75 TIMER Directive

The TIMER directive specifies the number of timer table entries to be generated in the MPX-32 resident image.

Syntax:

```
TIMER=number
```

number	is the number of timer table entries to be generated

7.5.76 TITLE Directive

The TITLE directive attaches identifying information to the SYSGEN listed output. Optional. If specified, it must be the first directive in the directive stream.

Syntax:

```
TITLE=data
```

data	is 1 to 66 ASCII characters of information

7.5.77 TQFULL Directive

The TQFULL directive specifies the maximum time quantum that a time-distribution task can acquire prior to preemption by another time-distribution task at the same priority level. If TQFULL or TQMIN is supplied, then both are required. TQFULL must be greater than TQMIN. If not specified, TQFULL defaults to 600 milliseconds.

Syntax:

```
TQFULL=time
```

time	is the maximum number of milliseconds in the quantum

7.5.78 TQMIN Directive

The TQMIN directive specifies the minimum time quantum that a time-distribution task can acquire prior to preemption by another time-distribution task at a higher priority level. If TQMIN or TQFULL is supplied, then both are required. TQMIN must be less than TQFULL. If not specified, TQMIN defaults to 200 milliseconds.

Syntax:

```
TQMIN=time
```

time	is the minimum number of milliseconds in the time quantum

7.5.79 TRACE Directive

The TRACE directive initializes the system trace flag word C.TRACE. Bit indicators within C.TRACE are described in the MPX-32 Technical Manual, Chapter 7. This directive is optional. If not specified, C.TRACE defaults to X'FFFFFFFE'.

Syntax:

```
TRACE=num
```

num	is a zero- to eight-character hexadecimal number

7.5.80 /TRAPS Directive

The /TRAPS directive designates the trap descriptor subsection of the //HARDWARE section. This directive is required.

Syntax:

```
/TRAPS
```

7.5.81 TYPE Directive

See the SIZE directive.

7.5.82 USERPROG Directive

The USERPROG directive specifies system modules to be included in the system along with the default trap handlers specified in the PROGRAM directive. This directive is optional.

Syntax:

```
USERPROG=(name1, . . . name7)
```

name1,...name 7	are one- to eight-character ASCII program names of modules assigned to LFC OBJ, separated by commas. A maximum of seven names are entered per directive.

7.5.83 /VP Directive

The /VP directive designates the Vector Processor (VP) subsection of the //HARDWARE section. This directive is required for systems containing a VP3300 or VP6410.

Syntax:

```
/VP
```

7.5.84 VP Directive

The VP directive specifies the device characteristics of the Vector Processors to be configured. This directive is required for systems containing a VP3300 or VP6410 VP.

Syntax:

$$\text{VP=}\left\{\begin{array}{l}\text{(aa, [,number])}\\ \text{aa}\end{array}\right\}\text{ [,PROGRAM=module1]}$$

aa	is the two-digit hexadecimal subchannel number
number	is the number of VPs to be configured. If not specified, the default is one.
module1	is the name of the module used to include a VP memory partition into a task's logical address space. If not specified, the default is H.ACBA.

7.5.85 VPID Directive

The VPID directive specifies unit specific information for a VP. This directive is required for systems containing a VP3300 or VP6410. One VPID directive must be used for each Vector Processor specified in the VP directive.

Syntax:

```
VPID=aa, VPTYPE=tt, STARTBLK=blk, PRIORITY=intlev, INTRPT(cc,ss)

[,PROGRAM=module2] [,IPCA=ipsize] [,BUS0=b0size] [,BUS1=b1size]

[,BUS2=b2size] [,BUS3=b3size]
```

aa	is the two-digit hexadecimal subchannel number
tt	is the type of VP: 33 specifies a VP3300, and 64 specifies a VP6410
blk	is the decimal starting physical map block address where the VP memory partitions are to be allocated
intlev	is the two-digit hexadecimal interrupt level
cc	is the RTOM/IOP channel address
ss	is the RTOM/IOP two-digit hexadecimal interrupt subaddress.
module2	is the name of the interrupt handler to be used with this VP. If not specified, the default is H.IPVP.
ipsize	is the number of pages to be allocated for the Interrupt Processor Context. If not specified, the default is 16 pages.
b0size	is the number of pages allocated for BUS0. If not specified, the default is 16 pages. This parameter is valid for VP6410 only.
b1size	is the number of pages allocated for BUS1. If not specified, the default is 32 pages for a VP6410 and 48 pages for a VP3300.
b2size	is the number of pages allocated for BUS2. If not specified, the default is no pages. This parameter is valid for VP3300 only.
b3size	is the number of pages allocated for BUS3. If not specified, the default is no pages. This parameter is valid for VP3300 only.

CHAPTER 8

SYSTEM DEBUGGER (H.DBUG)

The System Debugger is provided on the master SDT. It debugs the resident operating system as well as SYSGENed user interrupt and I/O handlers. The debugger is SYSGENed as part of the resident MPX-32 starter system, and it can be included in any user configuration of the resident system by using the SYSGEN directive PROGRAM=DEBUG. The System Debugger is composed of two parts: a small portion runs mapped as part of the resident operating system and occupies approximately 600 words of system logical address space; the remaining portion runs unmapped and represents the bulk of the debugger. The unmapped portion permanently occupies approximately 8KW of physical memory, but does not increase the size of the logical address space devoted to the operating system.

The system debugger only operates in privileged mode because it is using stand-alone I/O. Therefore, even privileged instructions are executed. Invalid instructions are flagged with an asterisk immediately following the opcode text. The instructions that may alter the program counter in the pseudo-PSD are executed, such as LPSD, BRI, SVC, etc., and traced by the debugger.

Unusual instruction sequences (branch increment with positive register contents, double word instructions using an odd number register, etc.) are flagged with a question mark following the instruction text. These inform the user of possible problems.

The debugger always starts in symbolic mode to allow addressing by base name plus offset. The AB (Absolute) command switches to absolute addressing. The SY (Symbolic) command switches back to symbolic addressing. (Absolute or symbolic modes apply to all address displays.)

8.1 Using the Debugger

8.1.1 Arithmetic and Special Operators

The debugger recognizes the following characters as unique operators allowable in any command or expression:

Character	Usage
G thru Z	Symbolic base values
R	Register designator
parenthesis ( )	Contents of
Colon (:)	Current value of last displayed contents
Asterisk (*)	Indirect address if first character in a field
Slash (/)	Current total
Semicolon (;)	Task high address
Question Mark (?)	Task low address

Character	Usage
Dollar Sign ($)	Current PC contents from PSD
Plus Sign (+)	Field on left added to field on right
Minus Sign (-)	Field on right subtracted from field on left
Asterisk (*)	Field on left multiplied by field on right
Slash (/)	Field on left divided by field on right
Ampersand (&)	Field on left is ORed with field on right
At-sign (@)	Field on left is ANDed with field on right
Right Angle Bracket (>)	Field on left is shifted right by the count in the right side field
Left Angle Bracket (<)	Field on left is shifted left by the count in the right side field
Apostrophe (')	ASCII text delimiter
Backslash (\)	Subfield delimiter

8.1.2 Special Functions

The debugger expects to read a two-character directive and options; however, the following special functions may be input instead of the normal directives:

Function	Meaning
^	Display previous location
*	Display location indirect to current location
carriage return	Display next location
>	Right shift current location and display
<	Left shift current location and display
$	Display current location

These functions require only the one byte of input.

8.1.3 Execution Breakpoints

A breakpoint is an address where program execution can be interrupted. Eight fixed breakpoints can be defined. In addition, a one-shot breakpoint can be established for the directed execution directives GO or CT.

8.1.4 Debugger Bases

The debugger uses a range of one-character bases to provide symbolic references to memory. Once established, the base characters (characters G through Z) can be used in any directive or arithmetic expression. The following bases are initialized by the debugger when loaded with MPX-32:

Base	Module
G	H.TAMM
H	H.MEMM
I	H.EXEC
J	Available
K	Available
L	H.ALOC
M	H.FISE
N	TSA start (C.TSAD)
O	Available
P	Available
Q	IP06 (SVC)
R	H.SWAPR
S	X'78000' (User DEBUG)
T	H.TSM
U	Available
V	H.VOMM
W	H.MONS
X	H.REXS
Y	H.IOCS
Z	H.REMM

Note: 40I is equivalent to 40+I or I+40.

8.1.5 Base Characters

The characters 'G' through 'Z' can be set to any value by the BA (Base) directive. For example,

```
BA G 427F0
```

defines G as location 427F0. G appears in all address displays in place of 427F0. G can be used in any directive and it is processed as 427F0.

8.1.6 Operator Restrictions

Arithmetic expressions are evaluated left to right.

Operators can appear in any directive and are not restricted in length. For example:

```
AR D9C35FFF,@3800000, >14
```

This results in a hexadecimal 18.

```
AR *34595,<2, +Z, +3000
```

If location 34595 contained 500 (hex) and Z was set to 30000, the above expression has a value of 34400.

```
DM O + 34C2 -1>2
```

The location specified by base O was previously set to 30000, so the DM location is CD30 (30000 + 34C2 -1 and right-shifted 2 bits).

The above examples illustrate expressions that can be used in any appropriate directive.

8.1.7 Expressions

The debugger processes expressions from left to right. For example,

```
AR 2*32+G/2
```

breaks down to two times 32 plus the value of G. That total is divided by 2 and the answer is typed.

8.1.8 Registers

The contents of a register is specified in any expression by enclosing the register number in parentheses. For example,

```
DM (1)
```

displays the contents of R1.

The DR directive displays registers and the CR directive can modify the registers.

8.1.9 Indirection

Indirection must precede a field. For example,

```
DM *0
```

causes the value at location 0 to be obtained and displayed as an address.

8.2 Accessing the Debugger

If the system debugger is configured as part of a resident system, the debugger is automatically accessed by SYSINIT when the configured system is installed from the user SDT. (The installation process is described in Chapter 2.) The debugger is activated by SYSINIT. The debugger input prompt is a double angle bracket (>>).

Debugger directives can be issued from the system console and patches can be made in the memory resident image of the system. The debugger is terminated by a TE (Terminate) directive. It returns control to SYSINIT, which continues building the system as described in Chapter 2.

There are three other ways the debugger can be accessed:

. A privileged user task can code a branch and link through the communications region variable that points to the debugger (C.DEBUG):

```
BL  *C.DEBUG
```

. Branch and link can only be used in modules or tasks that do not reside in the extended execution area of MPX. For modules that reside in the extended execution area or vary between extended and nonextended placement, the MBR_DBG macro can be used to access the debugger. The MBR_DBG macro determines the operating mode of calling module and inserts the appropriate code for debugger access.

. An OPCOM DEBUG directive can be used.

The system debugger uses stand alone drivers to perform I/O. The debugger routes listings to the printer configured as LP7E and it gets directives from the terminal or teletype configured as TY7E (normally the system console). Although any terminal user can issue the OPCOM DEBUG directive, once the system debugger gains control, its prompt is displayed on the system console and it accepts directives only from that device.

8.3 Debugger Directives

When the debugger is entered, it initializes bases and sets symbolic mode. It is then ready to receive and process directives.

The directive syntax requires the first two characters be entered unless special operators are used. Certain directives contain one or more fields of additional information, separated by a comma.

Debugger directives are summarized below and described in detail in the following pages. The summary is divided into three sections: general debugging directives, directive list directives, and patch list directives.

Directive	Description
AB	Displays all subsequent addresses as numeric (absolute mode)
AD	Displays low and high limits of a task address space
AR	Evaluates an arithmetic expression and displays its value
AS	Converts an instruction into its hexadecimal equivalent

Directive	Description
BA	Creates, deletes, or modifies the definition of a user base and displays its addresses
BR	Sets a breakpoint at a specified address
BY or TE	Exits the debugger (only when entered by a branch and link)
CB	Changes the contents of a base register
CH	Displays Controller Definition Table (CDT) entries
CM	Changes the contents of memory to a new value
CO	Continues tracing or execution from a breakpoint set with a BR command
CR	Changes the value of a user register
CT	Continues processing, setting a one-shot breakpoint at a specified address which terminates the trace function
DB	Displays the base registers
DE	Deletes a breakpoint which was set with the BR command
DI	Displays memory locations in instruction format within a specified address
DM	Displays memory locations within a specified address range
DQ	Displays dispatch queue entries
DR	Displays all registers
DS	Displays memory locations in instruction format within a specified address range
DT	Dumps the Event Trace Table to the printer. This directive assumes memory partition at hex 78000 and requires reassembly of the entire resident source with the event table enabled
DU	Dumps output to the line printer
EC	Echoes terminal output to the line printer
ET	Places an event trace point at a specified address
GO	Resumes execution of a user program at a specified address or the last known user PSW. Optionally sets a one-shot trap at a specified address.
HC	Displays the state queue head cells
LB	Lists all breakpoints
LP	Sets line printer output mode
LT	Displays a list of the current mobile event trace points
MR	Allows inspection of one CPU map register pair on a 32/87
MS	Modifies scratchpad locations
PS	Displays last known user PSW and condition codes
RE	Remaps the debugger to the map associated with a specified dispatch queue entry number or the current program

Directive	Description
RT	Removes a mobile event trace point at a specified address
SE	Compares specified words, in the range set by the SM directive, to a specified value
SM	Sets the mask as a left-justified hexadecimal number for the SE directive
SP	Dumps CPU scratchpad RAM locations
SY	Displays subsequent addresses as displacements from bases. This is symbolic mode.
TB	Displays instructions that cause TSA stack push
TE	Same as BY directive
TR	Traces user programs and displays each instruction after execution
TS	Terminates trace initiated by the TR directive
TY	Sends output to a terminal and resets echo mode
UD	Displays Unit Definition Table (UDT) directive entries

The following directives construct a list of system debug directives and execute them at specified breakpoint(s). The directive list is also executed at the users request. The BR directive executes a directive list on a breakpoint.

Directive	Description
CD	Displays the contents of the directive list
CE	Zeros the command list
CL	Turns off the directive list building mode
CS	Enters the directive list building mode
CX	Executes the directive list

The following group of directives can be used during the debug phase of IPL. These directives build a patch list of debug directives to apply to the system image being IPLed. The patch list is automatically written to disc as part of the system image at the return to SYSINIT from the system debugger. Execution of the patches is automatic at the beginning of the debug phase of IPL.

Directive	Description
PD	Displays the contents of the patch list
PE	Zeros the patch list
PR	Turns off the patch list building mode
PT	Enters the patch list building mode
PX	Executes the patch list

8.3.1 AB (Absolute) Directive

The AB directive displays all subsequent addresses as numeric (absolute mode).

Syntax:

```
AB
```

8.3.2 AD (Address) Directive

The AD directive displays the low and high limits of a task's address space.

Syntax:

```
AD
```

8.3.3 AR (Arithmetic) Directive

The AR directive evaluates an arithmetic expression. The debugger processes the expression in its left to right order and displays the value.

Syntax:

```
AR expr
```

expr	is the arithmetic expression to be evaluated

8.3.4 AS (Assemble Instruction) Directive

The AS directive converts an assembly language instruction to its 8-digit hexadecimal equivalent.

Instruction groups supported are memory-reference and register-register.

I/O instruction group is not supported.

Also, the Assembler mnemonics used in the syntax for this directive are abbreviated to only four characters rather than five characters. Use the DI directive to display the valid four-character mnemonics available for use with the AS directive.

Syntax:

```
AS [B] opcode [,reg] [,offset] [,index] [ (Breg) ]
```

B	activates base mode operation
opcode	is the four-character Assembler mnemonic
reg	is a number from zero to seven
offset	is a hexadecimal number or an expression
index	is the number one, two or three
Breg	is the base register (B0-B7) that is used for base mode instructions. If this parameter is specified, base mode must be activated (B parameter).

8.3.5 BA (Base) Directive

The BA directive defines a user base by adding its name to the internal base definition table, deletes a user base name from the base table, or redefines a user base by changing the value specified in the base name's definition.

A maximum of twenty bases are definable.

Syntax:

```
BA base [addr]
```

base	is a one-character alphanumeric base name. See Section 8.1.4.
addr	is the logical address for the base. If not supplied, the specified base name is deleted. If addr is supplied and base is already defined, base is redefined to represent addr.

8.3.6 BR (Breakpoint) Directive

The BR directive sets a breakpoint at a specified address. A breakpoint remains in effect until cleared with the DE directive. Upon execution of the breakpoint, BRK@addr is displayed along with the contents of the registers to indicate which breakpoint was executed. A maximum of eight breakpoints can be set. Breakpoints and event trace points (see ET command) cannot be set at the same address location.

Syntax:

```
BR addr [,C] [,B]
```

addr	is the address at which the breakpoint is set
C	specifies execution of a debug directive list at this address
B	specifies that the breakpoint resides in extended MPX

8.3.7 BY (Bye) Directive

The BY directive exits the debugger when entry was made using a branch and link instruction. The debugger returns control to the calling program. See also the TE directive.

Syntax:

    BY

8.3.8 CB (Change Base Register) Directive

The CB directive modifies the contents of one or more user base registers.

Syntax:

    CB reg, value [,value]...

reg	is a user register, B0-B7
value	is the 32-bit value to be stored in the specified register. Successive values are stored in consecutive user registers. Two successive commas with no intervening value skips the user register corresponding to the missing value, leaving its contents unchanged. If register seven has been altered or skipped and an unused value remains, it is ignored.

8.3.9 CD (Display Command List) Directive

The CD directive displays the contents of the system debugger directive list.

Syntax:

    CD

8.3.10 CE (Zero Command List) Directive

The CE directive zeros the system debugger directive list. Once zeroed, the list can be replaced or left zeroed to remove the directive list function.

Syntax:

    CE

8.3.11 CH (Display Controller Definition Table Entry) Directive

The CH directive displays one or all CDT entries.

Syntax:

    CH [index]

index	is the index of the CDT entry to be displayed. If not specified, all CDT entries are displayed.

8.3.12 CL (Terminate Build Directive List Mode) Directive

The CL directive terminates the system debugger directive list building mode.

Syntax:

```
CL
```

8.3.13 CM (Change Memory) Directive

The CM directive changes the contents of one or more words to a new 32-bit value. The specified address is changed to either a right-justified hexadecimal value or a left-justified blank-filled ASCII text word.

Syntax:

```
CM addr,value [,value]...
```

addr	is the address of the first or only word to be changed
value	is the 32-bit value to be stored at the specified address. Successive values are stored in consecutive words beginning at addr. Two consecutive commas with no intervening value can be used to skip the memory address corresponding to the missing value, leaving its contents unchanged.

8.3.14 CO (Continue) Directive

The CO directive continues tracing or execution from a breakpoint. If the user is in trace mode (entered by a TR directive), tracing continues and the debugger does not exit while TR is active. If the user is not in trace mode, execution continues. (See GO directive).

Syntax:

```
CO [addr [,stop] ]
```

addr	is the address where program execution with optional trace continues. If not specified, the program resumes from the last known location.
stop	is the address where execution and tracing terminates

8.3.15 CR (Change Register) Directive

The CR directive modifies the contents of one or more user registers.

Syntax:

```
CR reg,value [,value]...
```

reg	is a user register, R0-R7
value	is the 32-bit value to be stored in the specified register. Successive values are stored in consecutive user registers. Two consecutive commas with no intervening value skips the user register corresponding to the missing value, leaving its contents unchanged. If user register seven has been altered or skipped and an unused value remains, it is ignored.

8.3.16 CS (Build Directive List) Directive

The CS directive puts the system debugger in the directive list build mode. The following directives are not valid once CS is specified: CD, CE, CL, CX, or another CS directive.

Syntax:

```
CS
```

8.3.17 CT (Continue then Terminate) Directive

The CT directive continues processing from a specified dollar sign ($) location, setting a one-shot breakpoint at a specified address which terminates the trace function.

Syntax:

```
CT [addr]
```

addr	is the address where an optional one-shot breakpoint is set. Tracing is terminated and the users program reentered. If not specified, return to the user's program is at the last known program counter value.

8.3.18 CX (Execute Directive List) Directive

The CX directive executes the system debugger directive list.

Syntax:

```
CX
```

8.3.19 DB (Display Base Register) Directive

The DB directive displays one or more user base registers.

Syntax:

```
DB [Breg]
```

Breg	is a user base register, 0-7. If no register is specified, the default is all user base registers are displayed.

8.3.20 DE (Delete) Directive

The DE directive deletes a breakpoint that was set with a BR directive and restores user instructions to their original locations.

Syntax:

```
DE addr
```

addr	is the address where the breakpoint is deleted and the user's instruction is restored to its original location

8.3.21 DI (Display Instruction) Directive

The DI directive displays any memory locations within a specified address range, in instruction format, one line at a time (same as DS directive).

Syntax:

```
DI addl [,add2][,B]
```

addl	is the address where the display starts
add2	is the address where the display ends. If not specified, the display continues until a character other than a carriage return is entered at the end of a line.
B	specifies base mode instruction display

8.3.22 DM (Display Memory) Directive

The DM directive displays all memory locations within the specified address range. If DM is used with no parameters, memory locations are displayed 24 lines at a time. A carriage return continues the display; any other character entered terminates the display.

Syntax:

```
DM [addl [,add2] [,number] ]
```

addl	is the address where the display starts
add2	is the address where the display ends. If not specified, the ending address defaults to the starting address. For example, addl is the only location displayed.
number	is the number of words to be displayed per line on the console (minimum one, maximum eight). If not specified, the default is four.

8.3.23 DQ (Display Dispatch Queue Entry) Directive

The DQ directive displays one or all DQE entries.

Syntax:

```
DQ [index]
```

index	is the index of the DQE entry to be displayed. If not specified, all DQE entries are displayed.

8.3.24 DR (Display Register) Directive

The DR directive displays the memory locations of registers.

Syntax:

```
DR [reg]
```

reg	is a user register (R0-R7). If not specified, all registers are displayed.

8.3.25 DS (Display Symbolic) Directive

The DS directive displays any memory locations within a specified address range, in instruction format, one line at a time (same as DI directive).

Syntax:

```
DS add1[,add2] [,B]
```

add1	is the address where the display starts
add2	is the address where the display ends. If not specified, the display continues until a character other than a carriage return is entered at the end of a line.
B	specifies base mode instruction display

8.3.26 DT (Display Event Trace) Directive

The DT directive dumps the Event Trace Table to the line printer. Event trace is disabled until printing is completed.

Caution: This directive assumes memory partition at hexadecimal 78000 and requires reassembly of the entire resident source with event trace enabled.

Syntax:

```
DT
```

8.3.27 DU (Dump) Directive

The DU directive writes a range of memory to a line printer. ASCII format is used for the right-hand side of the memory display.

Syntax:

```
DU [start [,stop] ]
```

start	is the memory address where the dump starts
stop	is the memory address where the dump stops. If not specified, the default is the end of the operating system.

If no parameters are specified, the entire operating system area is output.

8.3.28 EC (Echo) Directive

The EC directive generates hard copy output when a CRT is the terminal device. Output is listed, line by line, on the line printer for every carriage return on the terminal. To terminate the EC directive, specify the TY directive.

Syntax:

```
EC
```

8.3.29 ET (Enter Event Trace Point) Directive

The ET directive places an event trace point within either resident or nonresident code without requiring reassembly of a program. To turn on the trace, this directive resets bits 21 and 22 of C.TRACE.

Prerequisites are:

1. The system must have been SYSGENed with a static partition for the Event Trace Table.

2. Both H.DBUG (System Debugger) and H.IP06 (SVC Trap Handler) must have been assembled with TRACE set true.

A maximum of eight event trace points can be set. Event trace points and breakpoints (see BR directive) cannot be set at the same address location.

Syntax:

```
ET addr [,B]
```

addr	is an absolute or relative address with a symbolic base
B	specifies that the event trace point resides in extended MPX-32

8.3.30 GO (Go) Directive

The GO directive transfers control to the user task, optionally setting a one-shot trap at a specified address.

Syntax:

```
GO [start [,stop] ] [,B]
```

start	is the address where the program starts executing
stop	is the address where the program stops executing. When specified, a breakpoint is set at this address before execution begins at the start address. If not specified, the program is reactivated at the start address.
B	specifies base mode execution

If no parameters are specified, the program is entered at the last known user PSW.

8.3.31 HC (Display Dispatch Queue Head Cell) Directive

The HC directive produces a three column display of the MPX-32 state queue head cells. Each entry is three words long.

Syntax:

```
HC
```

8.3.32 LB (List Breakpoint) Directive

The LB directive lists all active fixed breakpoints set using a BR directive

Syntax:

```
LB
```

8.3.33 LP (Line Printer) Directive

The LP directive directs output to the line printer. If the user is tracing, no request is made for input after each instruction. A breakpoint must be set to terminate the trace.

Syntax:

```
LP
```

8.3.34 LT (List Mobile Event Trace Point) Directive

The LT directive lists current mobile event trace points within resident or non-resident code.

Prerequisites are:

1. The system must have been SYSGENed with a static partition for the Event Trace Table.

2. Both H.DBUG (system debugger), and H.IP06 (SVC trap handler), must have been assembled with TRACE set true.

Syntax:

```
LT
```

8.3.35 MR (Map Register) Directive

The MR directive allows inspection of one CPU map register pair. This directive is available only on the 32/87 computer.

Syntax:

```
MR  index
```

index	is the number of the map register from 0 to FF

8.3.36 MS (Modify CPU Scratchpad Location) Directive

The MS directive modifies any location within the CPU scratchpad random access memory. Because the scratchpad is used by the CPU to process I/O and interrupt requests, incorrect modification of scratchpad locations could cause undesireable results. This directive should be used with extreme caution.

The scratchpad address specified in this directive can be within two ranges. The first range is a number between 0 and FF corresponding to the actual scratchpad address as defined in hardware. The second range is a number between 300 and 6FC corresponding to a scratchpad location obtained by using the SP directive to display all scratchpad locations.

If an invalid address is specified or an attempt is made to write past the end of scratchpad, an error message is displayed.

Syntax:

```
MS  loc,value [,value]...
```

loc	is the scratchpad address 0 to FF, or 300 to 6FC
value	is the 32-bit value stored in the specified location. Successive values are stored in successive locations.

8.3.37 PD (Display Patch List) Directive

The PD directive displays the contents of the system debugger patch list. The display list is either the previous patch list written to disc or the current edited version.

Syntax:

```
PD
```

8.3.38 PE (Zero Patch List) Directive

The PE directive zeros the system debugger patch list. Once zeroed, the list can be replaced or left zeroed to remove the patch function.

Syntax:

```
PE
```

8.3.39 PR (Terminate Build Patch List Mode) Directive

The PR directive terminates the system debugger patch list building mode.

Syntax:

```
PR
```

8.3.40 PS (Program Status) Directive

The PS directive displays the user Program Status Word (PSW) and condition codes.

Syntax:

```
PS
```

8.3.41 PT (Build Patch List) Directive

The PT directive causes the system debugger to enter the patch list build mode. The following directives are not valid after PT has been specified: PD, PE, PR, PX, or another PT directive.

Syntax:

```
PT
```

8.3.42 PX (Execute Patch List) Directive

The PX directive executes the system debugger patch list.

Syntax:

```
PX
```

8.3.43 RE (Remap) Directive

The RE directive remaps the debugger to the map associated with a program.

Syntax:

```
RE [number]
```

number	is the dispatch queue number in the range 0 to FF where remapping is associated. If not specified, remapping defaults to the dispatch queue number in C.CURR.

8.3.44 RT (Remove Event Trace Point) Directive

The RT directive removes a mobile event trace point within resident or nonresident code.

Prerequisites are:

1. The system must have been SYSGENed with a static partition for the Event Trace Table.

2. Both H.DBUG (system debugger), and H.IP06 (SVC trap handler), must have been assembled with TRACE set true.

Syntax:

```
RT addr
```

addr	is an absolute or relative address with a symbolic base

8.3.45 SE (Search Equivalent) Directive

The SE directive compares specified words to a specified value. Each word is ANDed with the Search Mask (see SM directive) before being compared to the value. Each bit in the range is listed.

Syntax:

```
SE value, start, stop
```

value	is the value where each word is compared
start	is the address where the search begins.
stop	is the address where the search ends.

8.3.46 SM (Set Mask) Directive

The SM directive sets the mask for the SE directive. The mask parameter is interpreted as a left-justified hexadecimal number or right-justified ASCII character string.

Syntax:

```
SM [mask]
```

mask	is a new mask value. If not specified, the default is the previously entered value; if none, X'FFFFFFFF'.

8.3.47 SP (Scratchpad Dump) Directive

The SP directive outputs to the terminal the contents of the scratchpad locations. Twenty-three lines of text are output at a time. A carriage return continues output, entering any other character terminates output.

Syntax:

```
SP [loc]
```

loc	is the CPU scratchpad address 0 to FF. If not specified, all scratchpad locations are displayed.

8.3.48 SY (Symbolic) Directive

The SY directive displays all subsequent addresses as displacements from bases (symbolic mode).

Syntax:

```
SY
```

8.3.49 TB (Trace Back) Directive

The TB directive displays the instructions that caused each TSA stack push. TB is only valid if there is a current task which is not at the base level of its stack. Use caution to ensure the debugger is mapped with the correct task space. The instructions are displayed in logical order, for example, the first instruction shown is the instruction executed by the task (usually a SVC) that caused the first push. Some operating system routines allocate extra stack frames as work space, therefore some displays can contain apparently invalid information.

Syntax:

```
TB
```

8.3.50 TE (Terminate) Directive

The TE directive exits the debugger when entry was made using a branch and link. Control is returned to the calling program. See the BY directive.

The TE directive should not exit from a breakpoint. See the CO and CT directives.

Syntax:

```
TE
```

8.3.51 TR (Trace) Directives

The TR directive executes and displays results of user instructions one at a time. Addresses are displayed as a base character plus offset value.

The last instruction executed is displayed and the cursor is held at the end of the line awaiting a user directive. A carriage return or line feed causes the next instruction to be executed and displayed, an up arrow (↑) causes the previous instruction to be redisplayed, and an equal sign (=) causes the hexadecimal equivalent of the instruction just executed to be displayed. Any other character causes the debugger to prompt for a directive.

Symbols established by the BA directive are used for display purposes if the SY directive was set.

Syntax:

```
TR [start [,stop] ] [,B]
```

start	is the address of the first user instruction to be traced. If not specified, the default is $ (current PSD value).
stop	is the address of the last user instruction to be traced. If not specified, tracing continues without bounds.
B	specifies base mode display and execution for tracing through extended MPX-32.

8.3.52 TS (Trace Stop) Directive

The TS directive exits the trace mode initiated by the TR directive. All further I/O is directed to the user's terminal.

Syntax:

```
TS
```

8.3.53 TY (Terminal) Directive

The TY directive directs output to a terminal and to reset EC (Echo) mode.

Syntax:

```
TY
```

8.3.54 UD (Display Unit Definition Table Entry) Directive

The UD directive displays one or all UDT entries.

Syntax:

```
UD [index]
```

index	is the index of the UDT entry to be displayed. If not specified, all UDT entries are displayed.

8.4 System Debugger Practice Debug Session

The following example is for first time users of the system debugger. It is recommended that the user try the following directives on the system console of an MPX-32 system. This should only be done on a stand-alone system (with no other users on the system), because it is very easy for the inexperienced user to crash the system. This example does not attempt to explain every directive or even the complete syntax of the directives. It is intended to help use the system debugger. While using this example, please refer to the directives descriptions and read each description before proceeding.

8.4.1 Step One - Accessing the Debugger

As described in Section 8.2, the debugger is generally entered using the instruction:

```
     BL      *C.DEBUG
```

This is the case whether it is entered by a user written program or through the OPCOM DEBUG directive. The calling program must be privileged to make this call. Also, the macro 'M.EQUS' or 'M.COMM.' must be included in the program to allow the symbol 'C.DEBUG' to be properly assembled.

To exit the debugger, when entered by a branch and link, use the TE directive.

At the operator's console, invoke the debugger using OPCOM.

```
     TSM>OPCOM              TSM>OPCOM DEBUG          TSM>!DEBUG
     ??DEBUG        or      >>                    or >>
     >>
```

Then exit.

```
     >>TE                   >>TE                     >>TE
       ??                     TSM>                     TSM>
```

Notice you return to the process you left.

8.4.2 Step Two - Task Debugging with the System Debugger

While it is normally best to debug a task using the task debugger, there are cases where the system debugger is the better tool, because the system debugger allows the user to trace through system calls.

8.4.3 Using the System Debugger to Display Memory

At the console, perform the following:

```
     TSM>!DEBUG             Enter the debugger (OPCOM is the task)
     >>
```

Because Assembler generated listings have the program relative addresses listed, it is necessary to set a base at the start of the task to be debugged. This allows the use of the offsets given by the Assembler. To do this, it is necessary to know where the task starts. This information is contained within the Task Service Area (TSA) represented by the predefined debug base N. See Section 8.1.4. To set a user base, first select the character desired as the base. In this example, use P. Using the macro expansion of the TSA variables, the start of the task is at T.BIAS which equates to hexadecimal '704'. To set the base, use the following debug directive.

```
     >>BA P *704N
```

This sets base P to equal the address stored at hexadecimal '704' relative to the TSA. The * indicates the address specified is indirect. When referencing locations that consist of an offset and a base, as in the 704N above, the + operator is implied.

Now, use the base to look at information contained within the task.

To find where the debug call is made, use the DI (Display Instruction) directive to list the instructions with their program relative offsets. The current task program counter is pointing one word past the call to the debugger, so the following instruction displays the debug call:

```
>>DI $-4
P+5C40          BL   0,X1
P+5C44          LF   R0,P+20
```

The $ indicates the current program counter location. In this case, the debugger is entered by an indexed branch to allow OPCOM to test for the presence of the debugger before attempting to branch to it.

To look at more instructions, repeated carriage returns display successive memory locations in instruction format.

To display noninstruction memory contents, use the DM (Display Memory) directive. This directive is the default directive. If only a number is entered, the data contained at the specified address is displayed on the terminal.

For example:

```
>>DM 50P
 P+50  46554C20                  *FUL *

>>50P
 P+50  46554C20                  *FUL *
```

The DM directive is not always defaultable. If the memory location to be displayed is ambiguous with a debug directive, then DM must be entered.

```
>>EC04              Would be interpreted by the debugger as the ECHO directive
                    must be entered

>>DM EC04
```

Other potentially ambiguous directives are:

- AB - Set Absolute Addressing Mode
- AD - Display Task Address Limits
- BA - Define Base
- DE - Delete Breakpoint
- EC - Echo Terminal Output to Line Printer

To display the registers at any point, two directives are used. The first is the DR (Display Register) directive. DR lists any or all of the eight general purpose registers.

```
>>DR
 A1037C44   00021F8C   00000000   0000002F   *.. D.........../*
 0000000D   00000004   44544255   20202020   *........DEBU    *

>>DR R5
 00000004                                  *....*
```

The second directive is the PS (Program Status) directive. PS lists the Program Status Word, the condition codes, and the general purpose registers.

```
>>PS
  PSW    A0037C44    CC00

 A1037C44    00021F8C   00000000    0000002F    *.. D........../*
 0000000D    00000004   44544255    20202020    *........DEBU    *
```

Often, calls to system services require data structure addresses be passed in registers. An example is the FCB address in almost all I/O service calls. If examination of such a structure is required, two methods can be employed.

The first is to display the registers, or a particular register using the DR directive. Then the memory locations pointed to by the register containing the data can be examined by the DM (Display Memory) directive.

The second involves using the DM directive with a special operator, the parentheses, indicating 'contents of a register'. A number in the range of zero to seven is enclosed in parentheses and indicates the contents of a register. To look at a 16 word FCB, use the following directive.

```
>>DM (1,(1+40)
    or
>>(1,(1+40)
```

By using the debug directives to their maximum potential, a great deal of information about a task is made readily available to the user. Two examples of such directive use are given below.

1. <u>Displaying the Dispatch Queue Entry for the Current Task</u>

   In general, have an MPX-32 Technical Manual available so that system data structures can be analyzed. For the debugger to display the actual relative addresses displayed in the diagrams, and not the absolute addresses, set a debug base. Q is an appropriate base to use for the dispatch queue. Set the base as follows.

   ```
   >>BA Q *8E8
   ```

   Because the * indicates indirect, base Q is set to the address stored at X'8E8'. The number X'8E8' was determined by looking at a C-dot cross reference for C.CURR. This is the dispatch queue head cell for the current task. To examine any part of the queue entry, refer to the proper offset followed by Q.

   ```
   >>Q,20Q
   Q+0   000008E8   000008E8   3C3C3C02   17000015    *........<<<.....*
   Q+10  53595354   454D2020   4F50434F   4D202020    *SYSTEM  OPCOM   *
   ```

The state queue linkage, the task number, the owner name, and the task name for the current task are shown.

2. Traversing a Linked Queue

If the integrity of a linked queue is in doubt, the debugger easily traces through the entries as shown below:

```
>>Q
 Q+0   000008E8                                         *....*
>>*
 8E8   00003D30                                         *..=0*
>>*
 Q+0   000008E8                                         *....*
```

Because the * indicates indirect, it refers to the address displayed on the previous line. In this manner, linked queues of any length can be traversed to ensure they contain all required entries, are connected, or whatever else may be desired.

8.4.4 Using Debug to Display a Program

Assume a program is failing in one particular subroutine. By using the breakpoint capabilities of debug, let the program run at machine speed until the trouble spot is encountered. To set a breakpoint, use the following directive.

```
>>BR addr
```

To let the program run until reaching the breakpoint, do the following:

```
>>TE          If the debugger was entered by BL *C.DEBUG
      (or)
>>BY          If the debugger was entered by BL *C.DEBUG
      (or)
>>CO          If the debugger was entered from a breakpoint
      (or)
>>CT          If the debugger was entered from a breakpoint
```

The four different debug exit directives are explained below.

TE or BY: either of these two directives can exit the debugger when it is entered by a branch and link. These are the only directives that are acceptable. Use of the CO or CT directives produce errors later on.

CO: continues the last mode of execution. If the user was tracing, then terminated trace mode using a noncarriage return input and now wishes to resume tracing, either CO or TR accomplishes it. If the program was running normally and was stopped by a debug breakpoint, CO will continue execution of the program in run mode. This directive should be used only if the most recent entry into the debugger was through a breakpoint.

CT: continues execution of the task in run mode regardless of the previous mode (trace or run). Like the CO directive, CT should only be used when the most recent entry into the debugger was through a breakpoint.

Upon reaching the breakpoint, the following is displayed:

```
>>BR 5C48P
>>TE
  BRK   P+5C48     CC 0100

89032C08   FFFFFFF0 00000000    0000002F   *..,............/*
0000000D   00000004   44454255   20202020   *.......DEBU    *
```

Warning: Breakpoints can not be set on SVC instructions. An error message is displayed should the user attempt this. The trace mode should only be used when the debugger is invoked by a breakpoint. If entry was accomplished by a branch and link, a breakpoint should be set, TE executed, and the trace started after entering the breakpoint.

When the area of code to be debugged has been reached, a helpful directive is TR (Trace). This directive executes instructions, one at a time, and displays the disassembled instruction, along with any pertinent registers or memory locations, on the console.

```
>>TR
```

After tracing several instructions, a bug is found. The instruction at xxxxP is:

```
  LD        R4,100,X2
```

what should have been there is:

```
  LD        R4,100,X3
```

In order to continue the debugging process, the instruction must be modified and re-executed. The CM (Change Memory) directive does this.

```
>>CM addr,value
```

This directive requires the correct hexadecimal value for the new instruction. The debugger assembles memory reference and register-register instructions. The result is displayed in hexadecimal. To find the value needed above use:

```
>> AS LD,4,100,3
   AE600102
```

Then to patch the bad instruction:

```
>>CM addr,AE600102
```

and to reexecute it:

```
>>TR addr
```

If it is desired, registers can be modified by using the CR directive.

This method for patching a program works for single line errors. That is, a bad memory address, an incorrect register, or any other error requiring modification of only one line of code. For more major problems that require the insertion of more instructions, the following technique can be employed.

Because several low memory areas (for example, X'D0' to X'FF') are not used after the system is booted, we can use these locations as a scratch area for building patches. MPX-32 has a section of memory reserved for patching by J.INIT that can build temporary debug patches. It is found by examining C.MPAC (determine actual address by assembling M.EQUS). This word contains the next available address within the MPX-32 patch area. The patch area end address is contained in C.MPAH.

To use the available memory locations, first, select the instruction where code is to be added. Using the area mentioned above, replace the instruction with a branch to X'D0'. At location X'D0', build the desired instructions ending with a branch back to the proper return location. This is generally one word past the branch to X'D0'. The instructions are, for the most part, generated by the debug AS directive, placed by the CM directive, and verified before being executed by the DI directive.

8.4.5 Summary

The best way to learn the debugger is to use it. Reread Section 8.3 for complete descriptions of the directives because this section does not cover all of the directives or make full use of the ones covered. Type carefully because the debugger executes exactly as requested. If you leave a base off of an address, it references the absolute location.

8.5 Example of Directive List Use

Problem:	Log contents of FCB after repeated I/O operations performed by new resident operating system module without continuous user intervention.
Solution:	Place a breakpoint at 63C relative to new module. Start with the directive list option and create a directive list to dump the FCB and continue.
Implementation:	Build Directive List

1. Access the system debugger

```
TSM>!DEBUG
```

2. Clear the directive list

```
>>CE
```

3. Set directive list build mode

```
>>CS
```

4. Construct directive list to dump FCB

```
>>DU (1,(1+40  Dump from address in R1 for 40 hexadecimal locations
>>CO           Continue
```

5. Terminate directive list build mode

```
>>CL
```

6. Display directive list

```
>>CD
```

Resultant CRT display:

```
CL ——→DU (1,(1+40
CL ——→CO
```

7. Set breakpoint

```
>>BR 63CX,C
```

8. Return to TSM

```
>>TE
```

8.6 Example of Patch List Use

Problem: System is generated containing user created operating system modules. The new module contains the patchable coding error

```
LW          R1,0W,R3
   (should be)
LA          R1,0W,R3
```

Solution: System debugger patch list containing a Change Memory (CM) directive

Implementation: Build Patch List

1. IPL system. When the debug prompt appears, locate erroneous instruction (for example, location 43C relative).

2. Use the Assemble directive to construct new instruction

```
>>AS LA,1,0,3
```

Result: 34E00000

3. Clear patch list

```
>>PE
```

4. Set patch list build mode

```
>>PT
```

5. Build patch list

```
>>CM  43C(relative),34E00000          Replace instruction at 43C relative
>>TE                                  Return to SYSINIT
```

6. Terminate patch list build mode

```
>>PR
```

7. Display patch list

```
>>PD
```

Result:

```
PL ———→ CM 43C(relative),34E00000
PL ———→ TE
```

8. Execute patch list

```
>>PX
```

Result:

Location 43C (relative) changed to 34E00000 and control returns to SYSINIT when the patch list is written to disc.

CHAPTER 9

ON-LINE SYSTEM PATCH FACILITY (J.INIT)

J.INIT provides for temporary or permanent patching of the MPX-32 resident image. In addition, J.INIT performs the Alterable Control Store/Writable Control Store (ACS/WCS) loading function. See Chapter 12.

J.INIT also initiates mount requests for any public volumes to be mounted at IPL time. The information for these requests is provided in the system file, M.MOUNT. See Section 9.7.

9.1 General

J.INIT processes patch directives from the patch file.

The patch file name is supplied in the SYSGEN PCHFILE directive. This file should contain valid patch directives as described in Section 9.2. The patch file is maintained by the Text Editor and should be a stored as a blocked, uncompressed file. An associated symbol table file is built by SYSGEN when the SYMTAB directive is specified. The SYSGEN PATCH directive should define the size of the patch area.

Only one patch area is used for extended and nonbase modules. If the extended mode is activated at SYSGEN, the patch area must be located in the first 16KW of memory. Otherwise, the patch area can be located anywhere in the resident MPX-32 system.

Patches for any modules that are located in extended memory must be constructed by using the appropriate base register instructions. J.INIT is unable to perform opcode decoding or substitution for extended modules. Memory reference instructions must be encoded with the base register field equal to zero. If these instructions are designated relocatable, J.INIT inserts the appropriate base register and offsets into the address field. J.INIT generates an error if an attempt is made to enter a relocatable base register instruction with a nonbase zero field.

Entry of a patch that invokes a branch between extended and nonbase modules is not supported. Because the adapter code addresses are not available to J.INIT, and because J.INIT cannot build adapter sequences, the user must locate the required adapter and enter the branch in absolute mode.

Patch processing terminates when the Exit (/E) directive is encountered in the patch input file. At this time, an audit trail of all patches specified is written to the SLO file. The audit trail listing is suppressed by specifying the NPR option with the option (/O) directive.

The patch program accepts directives to control processing. The general format of a directive is:

/d f_2 f_3 f_4 f_n

/d	is the directive name that must be followed by one or more spaces. This is called field 1 for error messages.

f_2 - f_n	are fields containing the names, values, and special symbols processed by the directive. Fields must be separated by one or more spaces or a comma.

For a patch directive to reference an extended module's DSECT data, SYSGEN generates system symbol table entries containing the DSECT base addresses. The name for these entries in the system symbol table is the module name with the first character replaced by "$." For example, $.REMM indicates the address of H.REMM's DSECT data. These names are used on relocatable patches that are DSECT relative.

WARNING: One character module names (A,B,C,etc.) or module names with only the first character unique (AMOD, BMOD, CMOD, etc.) cause the generation of multiply defined DSECT symbol table entries ($,$MOD). This generates an error in J.INIT.

9.1.1 Dedicated Names

Dedicated names used by J.INIT are:

Dedicated Name	Description
$	Equivalent to the address of the next free patch area location
R	Indicates a relative address in the positive direction
-R	Indicates a relative address in the negative direction
;	Delimits fields to be processed and comments

9.1.2 Conventions

All field entries on patch directives must conform to the following conventions:

name	one to eight ASCII characters, one of which must be nonnumeric
value	one to eight hexadecimal digits; leading zeros need not be specified. Only whole words are generated.
address	one to five hexadecimal digits; leading zeros need not be specified. Must be word resolution (bits 30-31 equal 0).

9.2 Patch Directives

Patch directives are summarized below and described in detail on the following pages.

Directive	Description
/B	Define a base address
/C	Change the contents of a memory location
/D	Define a named value
/E	Exit

Directive	Description
/G	Go to the patch area from a specified memory location
/O	Select patching options
/P	Define a patch area
/R	Return from the patch area
/F, /N, /T	Process patch directives conditionally
/$	Enter a value into the patch area
/;	Comment only

9.2.1 /B (Define a Base Address) Directive

The /B directive allows a name to be equated to a base address. This definition is inserted in the internal symbol table and can be referenced by subsequent directives.

Syntax:

```
/B name address
```

name is equated to address

(or)

/B name_1 name_2

name_1 is equated to the address value of name_2

9.2.2 /C (Change the Contents of a Memory Location) Directive

The /C directive changes the value of any location in memory.

Syntax:

```
/C address value
```

the specified value is inserted at the specified address

(or)

```
/C address value R
```

the address field of the value parameter is added to the address parameter to form the address field of value. The specified value is inserted at the calculated address.

(or)

```
/C address value -R
```

The address field of the value parameter is subtracted from the address parameter to form the address field of value. The specified value is inserted at the calculated address.

(or)

    /C name address value

The value of the name parameter is added to the address parameter to form the actual address. The specified value is inserted at the actual address.

(or)

    /C name address value R

the specified value is inserted at the specified address, relative to name. The value of name is added to value to form the actual address.

(or)

    /C name address value -R

the specified value is inserted at the specified address, relative to name. The address field of value is subtracted from name to form the address portion of value.

(or)

/C name_1 address value name_2

the value of name_2 is added to the value parameter and inserted at the actual address. The value of name_1 is added to the address parameter to form the actual address.

9.2.3 /D (Define a Named Value) Directive

The /D directive equates a name to the address of a value. The value is stored in the next free location of the patch area. The definition is inserted in the internal symbol table and can be referenced by subsequent directives.

Syntax:

    /D name value

name is equated to the address of value in the patch area

(or)

/D name_1 name_2

name_1 is equated to the address of the value of name_2 in the patch area

9.2.4 Exit (/E)

The /E directive terminates directive processing.

Syntax:

    /E

9.2.5 /G (Go to the Patch Area from a Specified Memory Location) Directive

The /G directive inserts an unconditional branch to the patch area at any memory location. The location branched to will be the next free location of the patch area plus, optionally, an offset.

Syntax:

    /G address

an unconditional branch to the specified address of the patch area is inserted

(or)

    /G name address

an unconditional branch to the specified address of the patch area relative to name is inserted. The value of name is added to address to determine the actual address of the branch in the patch area.

(or)

    /G name address value

an unconditional branch to the patch area at address, relative to name is inserted. The branch is to the next free patch location plus value number of bytes (word resolution). The offset area is thus reserved. The basic form, with no name field, can also be used.

9.2.6 /O (Select Patch Options) Directive

The /O directive specifies options for controlling the processing of subsequent directives.

Syntax:

/O $name_1$ $name_2$ $name_n$

$name_1$, $name_2$ and $name_n$ are the option names. A single /O directive can contain more than one name parameter. There can be any number of /O directives.

The available options are:

name	Use
NAM	Informs J.INIT that the definitions in the patch file should be merged into the internal symbol table. This option should be specified once per single run.
NHE	Informs J.INIT not to halt if patch errors are detected. If not specified, any patch error causes a halt. Entering //RUN continues processing.
NPR	Informs J.INIT that a patch listing is not to be produced.
SYM	Directs J.INIT to produce a listing of the internal symbol table. This specification should not be made in the first directive of a patch deck.

9.2.7 /P (Define a Patch Area) Directive

The /P directive defines a temporary patch area or appends patches to the patch area defined by SYSGEN. It should be used during debugging, but never when saving patches. See the /O directive.

Syntax:

    /P address value

a patch area is defined starting at location address, value (word resolution) bytes long. If the extended mode is activated in SYSGEN, this address must be in the first 16KW of memory. Since no attempt is made to protect this area, it should be some area of the resident image not used during debug operations.

(or)

    /P modname

modname signals J.INIT to determine the operating mode of a module and to generate a temporary patch area address according to the mode. This allows a patch area to be situated in the extended partition.

(or)

    /P CUR

subsequent patches are added to those entered during a previous patch run

9.2.8 /R (Return from the Patch Area) Directive

The /R directive allows an unconditional branch back to the instruction plus one word produced by the last /G directive encountered. An offset can be specified to reserve a number of patch locations immediately following the branch back.

Syntax:

    /R

an unconditional branch is inserted to the location plus one word containing the last branch generated by a /G directive

(or)

    /R value

an unconditional branch is inserted to the location plus one word containing the last branch generated by a /G directive value number of bytes of the patch area (word resolution) are reserved. This reserved area follows the generated branch.

9.2.9 /F, /N, /T (Conditional) Directive

The /F, /N, and /T directives allow skipping the processing of other directives based upon the presence or absence of a specified name in the symbol table. This creates a general patch file that attempts to modify only those modules included in the resident image.

Syntax:

/F name_1 name_2

if name_1 is false or not defined in the symbol table, discontinue directive processing until a /N directive containing name_2 is encountered

(or)

/T name_1 name_2

if name_1 is true or defined in the symbol table, discontinue directive processing until a /N directive containing name_2 is encountered

(or)

/T EXTDMPX modname label

(or)

/F EXTDMPX modname label

allows subsequent patches to be incorporated or skipped based on J.INIT's determination of the operation mode of modname. This directive supports the inclusion of extended module in the extended MPX-32 memory partition. This directive is active until a /N label is encountered.

(or)

/N name

the /F or /T skip sequences are terminated

9.2.10 /$ (Enter a Value into the Patch Area) Directive

The /$ directive inserts a value into the next free location of the patch area.

Syntax:

/$ value

value is inserted into the next free patch location

(or)

/$ value R

value is inserted, relative at the actual location. The value of $ is added to the address field of the value parameter for the actual location.

(or)

/$ value -R

the address field of value is subtracted from $ to form the actual address field of value. Value is then stored at the next free patch location.

(or)

    /$ name value

value is inserted, relative to name. The value of name is added to the value parameter and placed into the next free patch location.

9.2.11 /; (Comments) Directive

The /; directive can be included on any patch directive as a delimiter. The total directive can be designated as a comment by the use of this directive.

Syntax:

    /; text

9.3 Entry Conditions

Calling Sequence:

J.INIT is activated by SYSINIT at start-up. If the system is running, J.INIT can also be activated from TSM or OPCOM.

Patch processing is inhibited by setting control switch two.

9.4 Exit Conditions

Return Sequence:

M.EXIT	Exit to MPX-32

Registers:

None

9.5 External References

Abort Cases:

Halt if error detected and the NHE option has not been specified.

Output Messages:

J.INIT produces an audit trail of all patches made unless the NPR option is specified. The information produced includes a source image of each patch, the actual location patched, the actual value stored, and the previous contents of the location. The number of remaining free patch locations is also listed.

J.INIT can include error messages along with the audit trail listing. All error messages are preceded and/or followed by asterisks (*******). Possible error messages are:

ERROR IN PREVIOUS PATCH-FIELD-n

> n = number of the field containing the error (/d = field 1)

BASE TABLE OVERFLOW

> An attempt has been made to insert too many names in the internal symbol table (limit = 215_{10}).

PATCH AREA OVERFLOW

> An attempt has been made to insert too many patches in the area defined during SYSGEN or on the /P directive.

DUPLICATE NAME - name

> An attempt has been made to insert the displayed name in the internal symbol table and it is currently in the table.

END OF FILE ON patchfilename

> An attempt has been made to save the source image of a patch on the patch file and it is full (limit = 900 images).

UNABLE TO ALLOCATE patchfilename

> If the patch file does not exist, J.PATCH attempts to create it. This message indicates that sufficient disc space was unavailable. 100 blocks are required.

UNABLE TO ALLOCATE INPUT DEVICE

> The request to allocate the patch input device has been denied.

SAVE AND ADD PATCHES SPECIFIED

> Indicates that an /O SAV option was followed by a /P CUR directive.

PATCH ERRORS DETECTED

> Output at the end of the audit trail if any patch errors were detected. Also output to the console teletypewriter.

EXTD/NONEXTD ADDRESS CONFLICT

Indicates an extended patch exceeds the first 16KW of memory or an attempt was made to: place a nonbase patch in the extended area or place an extended patch in a nonbase area.

9.6 Examples

Changing locations in a resident module

```
/O NAM                            Get module description
/C H.IOCS 2154 CA803331           See Note 1
/C H.IOCS 574 EC001003 R          See Note 2
/E                                End of patches
```

Notes:

1. Changes location 2154 of H.IOCS to a LI R5,X'3331'.
2. Changes location 574 of H.IOCS to a BU to location 1003 of H.IOCS.

Inserting into the patch area

```
/O NPR NHE NAM                    See Note 1
/G H.REXS 100                     See Note 2
/$ CB050001                       See Note 3
/$ H.REXS F20005E5                See Note 4
/$ EE000009 R
/R
/$ 0000 0000
/O SYM                            List symbol table
/E                                End of patches
```

Notes:

1. No print, no halt on error, get module definitions, save patches.
2. Branch to the patch area from location 100 of H.REXS.
3. CI R6, 1 inserted in the next free location of patch area.
4. BCF EQ,5E5 of H.REXS inserted.

9.7 Automatic Mounting of Public Volumes

The last function of J.INIT before exiting is the mounting of public disc volumes. The mount requests are read from the system file M.MOUNT which must be provided by the user. The directive input format is the same as the OPCOM MOUNT directive.

Use the editor to build a list of mount directives to be processed and store them unnumbered into a system file named M.MOUNT. J.INIT processes the file on each subsequent IPL.

If a nonpublic volume is to be mounted, use the OPCOM MOUNT directive, instead of the M.MOUNT file.

See Volume III, Chapter 10 for more information.

CHAPTER 10

SYSTEM ADMINISTRATOR SERVICES

The System Administrator (SA) implements and controls the system and resource protection facilities provided by MPX-32. System protection is concerned primarily with user access to the system. The system protection facilities are implemented by the M.KEY and M.PRJCT file.

While not a direct function of the M.KEY and M.PRJCT files, resource protection facilities are not fully utilized unless these files are present and set up properly.

An individual can be designated to have the responsibilities of initializing terminals, responding to system console messages as necessary for system operation, and initializing floppy disc media.

Within a task, a privileged function can be performed for an unprivileged user with an SVC call. At the command processor level, this is done by allowing users to execute tasks that have the System Administrator (SA) attribute. This attribute is established when the load module or executable image is created.

The System Administrator attribute implies the following:

- Resource access without authorization checking.
- Use of the M.CALL macros (SVC 0).
- Authorization to mount public volumes.
- Authorization to update the current system date and time.
- Authorization to invoke the dual-processor shared-volume recovery task (J.UNLOCK).

The SA attribute is honored only if the load module or executable image file resides on the system volume and the owner of the file is SYSTEM.

Any task executed with OWNER=SYSTEM also has the SA attribute.

A task cataloged with SA can also specify the NODEBUG attribute to prevent unauthorized users from modifying the code.

M.CALL macros can be used by SA tasks that are not privileged. Therefore, programs for such tasks should be written only by users with a thorough knowledge of the operating system.

The operating system also provides various optional services to control access to the system, accounting services, and user diagnostics. These services are invoked through the following system files:

- M.KEY - Defines the set of authorized users of the system. Absence of this file prevents control of access to the system. This file is managed by the KEY program.
- M.PRJCT - Defines the set of authorized project group names for resource access and accounting. This file is managed by J.PRJCT.
- M.ACCNT - Records accounting information for terminal sessions and jobs.
- M.CNTRL - Provides automatic command file processing at logon.
- M.ERR - Contains system error explanations. This file can be modified to include user error messages.
- M.MOUNT - Provides automatic mounting of specified volumes during IPL.

If the M.KEY, M.PRJCT, M.ACCNT, or M.ERR file is recreated, the new file is immediately effective. If control switch four is set at IPL, these four files are ignored while the system is running.

10.1 M.KEY Editor (KEY)

The KEY program controls access to the system. When an M.KEY file exists, each user must be authorized using the KEY program before using the system.

Each user can have the following attributes and/or restrictions:

- key - a one to eight-character owner key required for logon
- default project group name
- default volume name
- default working directory
- default tab settings
- flagwords - a series of directives that determine access to various parts of the system

After the KEY program has been run, only those owners and users established in the M.KEY file are allowed to logon the system and access files.

10.1.1 Using KEY

The System Administrator attribute is required to run the KEY program. The primary file required to use KEY is an input file containing M.KEY information for each owner.

M.KEY is supplied on the Master SDT with owner name SYSTEM.

If the M.KEY file is recreated, the new file is immediately effective. If control switch four is set at IPL, the M.KEY file is ignored while the system is running.

The input file is prepared using the Edit STORE command. This file must be assigned to logical file code INP. Output is unblocked and automatically assigned to the system file M.KEY.

When recreating the M.KEY file, valid owner names must be specified before logging off the system so that valid names exist for future logons.

```
TSM>$ASSIGN INP TO FILENAME
TSM>$KEY
```

10.1.2 Directives

Key directives have one or more associated parameters. Some parameters can be preceded by a keyword and equal sign. Multiple parameters can be separated by TSM delimiters: spaces, commas, equal sign, and parentheses.

Continuation of the directive to the next physical line is specified by a comma following the last parameter on the line.

KEY directives are summarized below and described individually on the following pages.

Directive	Description
ADD	Authorizes a new user access to the system
CHANGE	Changes the attributes of an existing owner
DEFAULTS	Resets system defaults and establishes defaults for subsequent ADD commands
DELETE	Removes an existing owner from the M.KEY file
LOG	Lists existing owners and all attributes except their key
NEWFILE	Clears the M.KEY file of all existing entries
X	Indicates end of file and exits

ADD Directive

The ADD directive authorizes new users access to the system and specifies any restrictions applicable to them.

Syntax:

```
ADD [OWNER=]name  ┌ KEY=name                        ┐
                  │ VOLUME=name                     │
                  │ DIRECTORY=name                  │
                  │ PROJECT=name                    │
                  │ TABS=T [,T] [,T] ...            │
                  │            ⎧ALL            ⎫    │
                  │ [FLAGS=]   ⎨NONE           ⎬    │
                  │            ⎪ALLEXCEPT flist⎪    │
                  └            ⎩flist          ⎭    ┘
```

OWNER=	is the one- to eight-character owner name
KEY=	is the one- to eight-character key associated with this owner name. If not specified, the default is no key.
VOLUME=	is the one- to 16-character default volume name associated with this owner name. If not specified, the default is SYSTEM.
DIRECTORY=	is the one- to 16-character default directory name associated with this owner name. If not specified, the default is SYSTEM.
PROJECT=	is the one- to eight-character default project group name associated with this owner name. If not specified, the default is SYSTEM.
TABS=	sets all eight-decimal tab positions to be associated with this owner name. If not specified, default tabs are set at 10, 20, and 36.
FLAGS=ALL	specifies all restrictions apply to this owner
=NONE	specifies no restrictions apply to this owner
=ALLEXCEPT	specifies all restrictions except the following listed key words apply to this owner
=flist	specifies the operations associated with following listed key words cannot be performed by this owner

The following key words disable the corresponding OPCOM directives:

ABORT	MODE
ACTIVATE	MODIFY
BATCH	MOUNT
BREAK	OFFLINE
CONNECT	ONLINE
CONTINUE	PURGEAC
DELETETIMER	REDIRECT
DEPRINT	REPRINT
DEPUNCH	REPUNCH
DISABLE	REQUEST
DISCONNECT	RESUME
DISMOUNT	SEARCH
DUMP	SEND
ENABLE	SETTIMER
ESTABLISH	SNAP
HOLD	STATUS
KILL	SYSASSIGN
LIST	TIME

When the following key words are specified, the corresponding restrictions apply:

Key word	Restriction
CATPRIV	Cannot catalog privileged programs
CHANGEDEF	Cannot change working directory or project group
MDT	Cannot use the J.MDTI utility
OWNERACC	Cannot access tasks with different owner name
PRIORITY	Cannot use the TSM $URGENT directive
PRIVILEGE	Cannot run privileged programs other than OPCOM
REMOVE	Cannot use TSM $REMOVE directive
RESTART	Cannot use the TSM $RESTART directive
USERFLAGS (n)	Decimal number in range 0 to 255 for use by user for security. User programs may examine this number in bits 56 to 63 of T.ACCESS.

When the following keywords are specified, the corresponding defaults apply:

Key word	Default
NOCOMMAND	OPTION NOCOMMAND in TSM
SEQUENTIAL	If more than one job is submitted by a TSM $SUBMIT directive, the jobs execute sequentially. If sequential execution is required for jobs run using a BATCH or RUN directive, S must be specified on the $JOB directive line.

CHANGE Directive

The CHANGE directive changes the attributes of an existing owner on the system.

Syntax:

```
    CHANGE   [OWNER=] name ┌ KEY=name                          ┐
                           │ VOLUME=name                       │
                           │ DIRECTORY=name                    │
                           │ PROJECT=name                      │
                           │ TABS=T [,T] [,T] ...              │
                           │              ( ALL            )   │
                           │ [FLAGS=]     ) NONE           (   │
                           │              ) ALLEXCEPT flist(   │
                           └              ( flist          )   ┘
```

OWNER=	is the one- to eight-character owner name
KEY=	is the one- to eight-character key associated with this owner name. If not specified, the default is no key.
VOLUME=	is the one- to 16-character default volume name associated with this owner name. If not specified, the default is SYSTEM.
DIRECTORY=	is the one- to 16-character default directory name associated with this owner name. If not specified, the default is SYSTEM.
PROJECT=	is the one- to eight-character default project group name associated with this owner name. If not specified, the default is SYSTEM.
TABS=	sets all eight-decimal tab positions to be associated with this owner name. If not specified, the default tabs are set at 10, 20, and 36.
FLAGS=ALL	specifies all restrictions apply to this owner
=NONE	specifies no restrictions apply to this owner
=ALLEXCEPT	specifies all restrictions except the following listed key words apply to this owner
=flist	specifies the operations associated with following listed key words cannot be performed by this owner

The following key words disable the corresponding OPCOM directives:

ABORT	MODE
ACTIVATE	MODIFY
BATCH	MOUNT
BREAK	OFFLINE
CONNECT	ONLINE
CONTINUE	PURGEAC
DELETETIMER	REDIRECT
DEPRINT	REPRINT
DEPUNCH	REPUNCH
DISABLE	REQUEST
DISCONNECT	RESUME
DISMOUNT	SEARCH
DUMP	SEND
ENABLE	SETTIMER
ESTABLISH	SNAP
HOLD	STATUS
KILL	SYSASSIGN
LIST	TIME

When the following key words are specified, the corresponding restrictions apply:

Key word	Restrictions
CATPRIV	Cannot catalog privileged programs
CHANGEDEF	Cannot change working directory or project group
OWNERACC	Cannot access tasks with different owner name
PRIORITY	Cannot use the TSM $URGENT directive
PRIVILEGE	Cannot run privileged programs other than OPCOM
REMOVE	Cannot use TSM $REMOVE directive
RESTART	Cannot use the TSM $RESTART directive
USERFLAGS (n)	Decimal number in the range of 0 to 15 for use by user for security. User programs examine this number in bits 60 to 63 of T.ACCESS.

When the following key words are specified, the corresponding defaults apply:

Key word	Default
NOCOMMAND	OPTION NOCOMMAND in TSM.
SEQUENTIAL	If more than one job is submitted by a TSM $SUBMIT directive, the jobs execute sequentially. If sequential execution is required for jobs run using a BATCH or RUN directive, S must be specified on the $JOB directive line.

DEFAULTS Directive

The DEFAULTS directive establishes defaults for subsequent ADD directives.

Each DEFAULTS remains in effect until the next DEFAULTS is specified. When the next DEFAULTS is specified, the default for any keyword not specified is the SYSTEM default. When more than one keyword default is to be defined for a group of users, all defaults must be specified with one DEFAULTS directive as shown in the example of directive usage.

Syntax:

```
DEFAULTS [ KEY=name                                ]
         [ VOLUME=name                             ]
         [ DIRECTORY=name                          ]
         [ PROJECT=name                            ]
         [ TABS=T [,T] [,T] ...                    ]
         [            { ALL                 }      ]
         [ [FLAGS=]   { NONE                }      ]
         [            { ALLEXCEPT flist     }      ]
         [            { flist               }      ]
```

KEY=	is the one- to eight-character key associated with this owner name
VOLUME=	is the one- to 16-character default volume name associated with this owner name. If not specified, the default is SYSTEM.
DIRECTORY=	is the one- to 16-character default directory name associated with this owner name. If not specified, the default is SYSTEM.
PROJECT=	is the one- to eight-character default project group name associated with this owner name. If not specified, the default is SYSTEM.
TABS=	sets as many as eight-decimal tab positions to be associated with this owner name. If not specified, the default tabs are set at 10, 20, and 36.
FLAGS=ALL	specifies all restrictions apply to this owner
=NONE	specifies no restrictions apply to this owner
=ALLEXCEPT	specifies all restrictions the following listed key words apply to this owner
=flist	specifies the operations associated with following listed key words cannot be performed by this owner

The following key words disable the corresponding OPCOM directives:

ABORT	MODE
ACTIVATE	MODIFY
BATCH	MOUNT
BREAK	OFFLINE
CONNECT	ONLINE
CONTINUE	PURGEAC
DELETETIMER	REDIRECT
DEPRINT	REPRINT
DEPUNCH	REPUNCH
DISABLE	REQUEST
DISCONNECT	RESUME
DISMOUNT	SEARCH
DUMP	SEND
ENABLE	SETTIMER
ESTABLISH	SNAP
HOLD	STATUS
KILL	SYSASSIGN
LIST	TIME

When the following keywords are specified, the corresponding restrictions apply:

Key word	Restriction
CATPRIV	Cannot catalog privileged programs
CHANGEDEF	Cannot change working directory or project group
OWNERACC	Cannot access tasks with different owner name
PRIORITY	Cannot use the TSM $URGENT directive
PRIVILEGE	Cannot run privileged programs other than OPCOM
REMOVE	Cannot use TSM $REMOVE directive
RESTART	Cannot use the TSM $RESTART directive
USERFLAGS (n)	Decimal number in the range of 0 to 15 for use by user for security. User programs examine this number in bits 60 to 63 of T.ACCESS.

When the following keywords are specified, the corresponding defaults apply:

Key word	Default
NOCOMMAND	OPTION NOCOMMAND in TSM
SEQUENTIAL	If more than one job is submitted by a TSM $SUBMIT directive, the jobs execute sequentially. If sequential execution is required for jobs run using a BATCH or RUN directive, S must be specified on the $JOB directive line.

DELETE Directive

The DELETE directive removes an existing owner from the M.KEY file.

Syntax:

```
DELETE  [OWNER=] name
```

OWNER=	is the one- to eight-character owner name

LOG Directive

The LOG directive lists existing owners and all attributes except their key.

Syntax:

```
LOG  [[OWNER=] name]
```

OWNER=	is the one- to eight-character owner name whose attributes are to be displayed. If not specified, all owners attributes are displayed.

NEWFILE Directive

The NEWFILE directive clears the M.KEY file of all existing entries. If present, it must be the first directive specified.

Syntax:

```
NEWFILE
```

X Directive

The X directive indicates end-of-file and exit.

Syntax:

```
X
```

10.1.3 Examples of Directive Usage

```
NEWFILE
ADD OWNER=SYSTEM VOLUME=SYSTEM DIRECTORY=SYSTEM
DEFAULTS PROJECT=WORK TABS=7,12,24 FLAGS=NONE
ADD OWNER=OWN1 VOLUME=VOL1 DIRECTORY=DIR1
ADD OWNER=OWN2 VOLUME=VOL2 DIRECTORY=DIR2
ADD OWNER=OWN3
DEFAULTS VOLUME=TEST PROJECT=EXAM
ADD OWNER=OWN4 KEY=0416 TABS=10,25,50
CHANGE OWNER=OWN3 VOLUME=VOL3
ADD OWNER=OWN5 FLAGS=ALLEXCEPT MOUNT,SEARCH,PRIVILEGE,SEQUENTIAL
ADD OWNER=OWN6
DEFAULTS VOLUME=TEST DIRECTORY=DIR3 PROJECT=UTIL FLAGS=ABORT,HOLD,MOUNT,
MODIFY,PRIVILEGE,PRIORITY,SEQUENTIAL TABS=9,18,36
ADD OWNER=OWN7 KEY=1205
ADD OWNER=OWN8 KEY=1103
ADD OWNER=OWN9 KEY=0613
DELETE OWN3
LOG
X
```

10.2 M.PRJCT File

The M.PRJCT file contains the project group names that are valid for use with the TSM CHANGE PROJECT directive and that are used by the accounting utility M.ACCNT. If a M.PRJCT file does not exist, any project group name is valid.

When using the accounting utility, project group names can be established for each owner name on the system to gain access to TSM in one of two ways:

1. Using the Volume Manager, an M.PRJCT file can be created under the owner SYSTEM as a nonextendible file that is not accessible by PROJECTGROUP or OTHER. The file should be zeroed. As many as 32 entries can be placed in this file from an editor-created user file using the PROJECT program J.PRJCT. The output from the PROJECT program does not need to be assigned; output automatically goes to M.PRJCT unblocked. Only users with the System Administrator attribute can use the PROJECT program.

   Example:

   ```
   ENTER YOUR OWNERNAME:SYSTEM
   TSM>VOLMGR
   VOL>CREATE @SYSTEM(SYSTEM)M.PRJCT AUTO=N ZERO=Y-
   VOL>ACCESS=PROJECTGROUP ()-
   VOL>ACCESS=OTHER ( ) SIZE=4
   VOL>X

   TSM>$ASSIGN INP TO PFILE
   TSM>PROJECT
   ```

2. Default project group names can be established for each owner name using the KEY utility.

Project group names can be changed by a user with the $CHANGE PROJECT directive under TSM. When a project group name is changed, the account utility is terminated for the previous group name and initiated for the new group name. The change remains in effect until changed again or the user logs off the system. When the user logs on the system, default project group names are reestablished.

10.2.1 Using the PROJECT Program

The primary file required to use PROJECT is an input file containing M.PRJCT information for each owner.

If the M.PRJCT file is recreated, the new file does not become effective until the system is rebooted. If control switch four is set at IPL, the M.PRJCT file is ignored while the system is running.

The input file is prepared using the Edit STORE directive. It must be assigned to the logical file code INP. Output is unblocked and automatically assigned to the system file M.PRJCT.

Example:

```
TSM>$ASSIGN INP TO FILENAME
TSM>$PROJECT
```

Only owners with the System Administrator attribute can run the PROJECT program.

10.2.2 Directives

Input to the PROJECT program is a series of directives that are summarized below and described in detail on the following pages.

PROJECT directives have one or more associated parameters. Some parameters can be preceded by a keyword and equal sign. Multiple parameters can be separated by the normal TSM delimiters: spaces, commas, equal sign, and parentheses.

Continuation of a directive to the next physical line is specified by a comma following the last parameter on the line.

PROJECT directives are summarized below and described individually on the following pages.

Directive	Description
ADD	Authorizes a new project group name
CHANGE	Changes a key associated with a project group name
DELETE	Removes a project group name from the PROJECT file
LOG	Lists existing project group names
NEWFILE	Clears the M.PRJCT file of all existing entries
X	Indicates end-of-file and exits

ADD Directive

The ADD directive adds new project group names.

Syntax:

```
    ADD  [PROJECT=] pname   [KEY=key]
```

pname=	is the one- to eight-character project group name to be added
key=	is the one- to eight-character key to be associated with this project group name

CHANGE Directive

The CHANGE directive changes an existing key associated with a project group name or establishes a key with an existing project group name that does not have an associated key. A key cannot be deleted with this directive. If a key is to be deleted, the DELETE directive must be used to delete the project name; the ADD directive must then be used to reestablish the project name without a key.

Syntax:

```
    CHANGE   [PROJECT=] pname   [KEY=key]
```

pname=	is the one- to eight-character project group name
key=	is the new one- to eight-character key to be associated with this project group name

DELETE Directive

The DELETE directive removes an existing project group name from the PROJECT file.

Syntax:

```
    DELETE    [PROJECT=] pname
```

pname=	is the one- to eight-character project group name to be deleted

LOG Directive

The LOG directive lists existing project group names.

Syntax:

```
    LOG  [ [PROJECT=] pname]
```

pname=	is the one- to eight-character project group name whose authorization is to be listed. If not specified, all project group names are displayed.

NEWFILE Directive

The NEWFILE directive clears the M.PRJCT file of all existing entries. If present, it must be the first directive specified.

Syntax:

```
NEWFILE
```

X Directive

The X directive indicates end-of-file and exit.

Syntax:

```
X
```

10.3 Examples of Directive Usage

```
NEWFILE
ADD PROJECT=SYSTEM
ADD PROJECT=WORK
ADD PROJECT=EXAM
ADD PROJECT=UTIL
ADD TEST
ADD LANG KEY
ADD DOC KEY=JM
CHANGE PROJECT=LANG KEY=DR
CHANGE TEST KEY=CASE
DELETE PROJECT=DOC
ADD DOC
LOG
X
```

10.4 M.ACCNT File

The job accounting program indicates elapsed time, CPU time, and IPU time for all jobs. The M.ACCNT file must be created by the Volume Manager under the owner name SYSTEM as a nonextendible zeroed file with read access by PROJECTGROUP and OTHER. Its size is determined by the need of the individual site. Its format is 16 word entries in an unblocked 192W physical record. This file is where job accounting information is collected for use by TSM.

For example:

```
ENTER YOUR OWNERNAME:SYSTEM
TSM>VOLMGR
VOL>CREATE @SYSTEM(SYSTEM)M.ACCNT AUTO=N ZERO=Y SIZE=100-
VOL>ACCESS=PROJECTGROUP(R)-
VOL>ACCESS=OTHER(R)
```

The M.ACCNT file is created as a nonextendible file on volume SYSTEM and directory SYSTEM. PROJECTGROUP and OTHER have read access. As many as 1200 entries can be specified.

If the M.ACCNT file is recreated, the new file does not become effective until the system is rebooted. If control switch four is set at IPL, the M.ACCNT file is ignored while the system is running.

Default project group names for accounting purposes can be established with the M.KEY utility.

Data collected by the accounting program can be retrieved with the OPCOM LIST command. Refer to OPCOM utility documentation.

The format of M.ACCNT file entries follows.

Word	0 31
0 1	Owner name (1- to 8-character ASCII). See Note 1.
2 3	Project (1- to 8-character ASCII). See Note 2.
4 5	Date (mm/dd/yy) (ASCII). See Note 3.
6 7	Logon time (hh/mm/ss) (ASCII). See Note 4.
8 9	Elapsed time (hh:mm:ss) (ASCII). See Note 5.
10	Raw CPU time (Binary). See Note 6.
11	Raw IPU time (Binary). See Note 7.
12 13	Origin (1- to 8-character ASCII). See Note 8.
14 15	Reserved

Notes:

1. Owner name - The one- to eight-character ASCII owner name associated with the job.

2. Project - The one- to eight-character ASCII alphanumeric project name or number associated with the job.

3. Date - The ASCII numeric date associated with the job.

4. Logon time - The ASCII numeric time of day on the 24-hour clock the user signed on the system.

5. Elapsed time - The total time (24-hour clock) the user was signed on the system in ASCII.

6. Raw CPU time - The actual number of 38.4 microsecond intervals of CPU time for the job (unformatted equivalent of CPU time) in binary.

7. Raw IPU time - The actual number of 38.4 microsecond intervals of IPU time for the job (unformatted equivalent of IPU time) in binary.

8. Origin - The ASCII task pseudonym for the accounting session.

10.5 M.CNTRL File

The M.CNTRL file is a TSM command file selected by J.TSM automatically when a user logs on. If the M.CNTRL file exists, it must be located in the system directory.

As the M.CNTRL file can contain TSM directives and comments, it can establish defaults, send messages, and further restrict access to the TSM environment.

Examples:

```
      EDT>COL
      1.  NOTE LOG ON AT
      2.  !TIME
      3.  <cr>
      EDT>STO M.CNTRL SYS
```

This causes the time and date to be automatically displayed when a user logs on the system. NOTE indicates the line is a comment line.

Notes:

Attempts to break out of the M.CNTRL file are ignored.

Errors in M.CNTRL file processing result in normal error processing for control files.

10.6 M.ERR File and xx.ERR Files

The M.ERR file contains system abort codes and messages, and should not be modified. The xx.ERR files contain messages for unbundled products and user abort codes. For example, FT.ERR contains FORTRAN 77+ abort messages.

The abort code format is:

```
      xxnn
```

x	is an alphabetic character
n	is a numeral

The file SJ.XX.ER is provided on the SDT and can be modified with other abort messages by using the Text Editor (EDIT). When an abort code cannot be found in the M.ERR file, J.TSM appends .ERR to the abort code's prefix. Using the resulting name, J.TSM attempts to allocate a file in the system directory. If allocation occurs, the file is searched for the error code.

10.6.1 Creating xx.ERR File

Modify SJ.XX.ERR as follows:

1. Enter EDIT.

2. Use file SJ.XX.ER.

3. Modify the prefix at label STARTX.

   Example:

   ```
   STARTX PREFIX C'MD' MEDIA ERRORS
   ```

4. Remove the example messages.

5. Add messages using 'C' strings.

6. After each message, call macro MSG xx,nn where xx is the prefix specified at STARTX; nn is the abort number in ascending order. Example (starts in the second column):

   ```
   DATA C'ERROR ENCOUNTERED READING SYC FILE'
   MSG LM,01
   DATA C'ERROR ENCOUNTERED WRITING TO SLO'
   MSG LM,03
   ```

7. Save the completed file. Example:

   ```
   EDT>SAVE @SYSTEM(SOURCE)SJ.xxER
   (xx is the prefix specified at STARTX)
   ```

8. Exit EDIT.

9. Submit the following job stream to install the new error file:

   ```
   $JOB NEWERR OWNER
   $ASSIGN SI TO SJ.xx.ER BLOC=Y               (SJ.xx.ER is the file previously
                                               saved in Step 7 by the Text
                                               Editor (EDIT))

   $ASSIGN BO TO OJ.xxER BLOCK=Y
   $OPTION 1 4
   $ASSEMBLE
   $AS SGO TO OJ.xx.ER BLOCK=Y

   $IFF  PATH=@SYSTEM  ^(SYSTEM)xx.ERR  GO     (xx is the prefix entered in
                                               Step 3. This text ensures that
                                               an existing abort file code is
                                               not destroyed).

   $NOTE ERROR - ATTEMPT MADE TO WRITE OVER EXISTING ABORT FILE
   $GOTO END
   $DEFNAME GO
   $CATALOG
   CATALOG  xx.ERR                             (xx is the prefix entered in
                                               Step 3)

   $DEFNAME END
   $EOJ
   $$
   ```

   Abort messages are immediately in effect without rebooting the system.

10.7 Terminal/ALIM/ASYNCH Initialization (INIT)

Terminal/ALIM/ASYNCH initialization is the process of defining hardware characteristics of TSM and non-TSM devices. For purposes of this section, TSM devices are defined as terminals, device type code X'0C' TY, connected through model 9110 ALIM or 8510, 8511, or 8512 Eight-line Asynchronous Communications controllers. Non-TSM devices are defined as devices other than terminals; the device type code is not TY, but the device can be connected through these same controllers.

Hardware characteristics are typically defined by a user-created system file named LOGONFLE, or in its absence, by a set of system supplied defaults. Characteristics include baud rate, parity, and half or full duplex. The wakeup (ring) character for all terminals is also defined and a hardware characteristic.

Initialization of a TSM device is handled automatically when a system is installed or restarted, regardless of the existence of a LOGONFLE. Initialization of a non-TSM device requires a LOGONFLE entry defining the characteristics of the non-TSM device and the keyword NOTSM.

Although the IOP console is normally treated as a TSM device, it is not initialized by INIT.

10.7.1 The LOGONFLE

The LOGONFLE must contain the logon records in blocked, uncompressed format. The Editor STORE directive can create this file.

The form of a LOGONFLE is shown below. LOGONFLE must contain a record for the wakeup character definition and one record for each device definition. Only characters 1 to 72 of each record are interpreted. An asterisk (*) in column 1, from Record 2 on, indicates the line is a comment line. A semicolon (;) or an exclamation point (!) in any position of a line, from Record 2 on, permits comments to follow on the line.

```
Record 1: wakeup

Record 2: ccaa field field  ...
            .
            .
            .
            EOF
```

wakeup	is a two-digit hexadecimal number defining the character which must be typed at interactive terminals to start a logon sequence. If LOGONFLE was not created, system default is X'3F', question mark.
ccaa	is the channel number and subaddress of the device to initialize. Must be supplied in LOGONFLE for each device. If LOGONFLE was not created, the system default is to initialize all addresses defined at SYSGEN as device type TY with the system default parameters.
field	is a one- to eight-character keyword, (the first four characters are significant,) describing the characteristics of a device. Fields can be entered in any order and can be duplicated. Fields are evaluated left to right; later entries overrule earlier entries.

Category	Keyword	ALIM	8-Line Async
Baud Rates	19200	Not used	19200 bps
	9600	9600 bps	9600 bps
	7200	7200 bps	7200 bps
	4800	4800 bps	4800 bps
	3600	3600 bps	3600 bps
	2400	2400 bps	2400 bps
	2000	Not used	2000 bps
	1800	1800 bps	1800 bps
	1200	1200 bps	1200 bps
	900	900 bps	Not used
	600	600 bps	600 bps
	300	300 bps	300 bps
	150	150 bps	150 bps
	134	134.5 bps	134.5 bps
	110	110 bps	110 bps
	75	75 bps	75 bps
	50	50 bps	50 bps
	EXT	External rate	Not used
Parity	ODD	Odd parity	Odd parity
	EVEN	Even parity	Even parity
	NONE	No parity	No parity
Duplex	HALF	Half duplex	Half duplex
	FULL Full duplex bit set in UDT	Full duplex	Full duplex
Stop Bits	S1	1 stop bit	1 stop bit
	S1.5	1.5 stop bits	1.5 stop bits
	S2	2 stop bits	2 stop bits
Character Size	8	8-bit characters	8-bit characters
	7	7-bit characters	7-bit characters
	6	6-bit characters	6-bit characters
	5	5-bit characters	5-bit characters

Category	Keyword	ALIM	8-Line Async
Miscellaneous	GRAP	Graphic device	Graphic device
	MODEM	Sets Modem bit in UDT	Sets Modem bit in UDT
	REMOTE	Sets dial-up bit in UDT	Sets dial-up bit in UDT
	NOTSM	Not a TSM device	Not a TSM device
	INIT	Initialization data present	Initialization data present

10.7.2 ALIM Terminal Record Syntax and Defaults

Syntax:

```
ccaa [baud]   ┌HALF┐┌ECHO  ┐ [parity][charsize][stopbits][GRAP][MODEM]
              └FULL┘└NOECHO┘ [REMOTE][NOTSM][INIT value]
```

ccaa	is the channel and subaddress of the device.
baud	is the baud rate: 9600, 7200, 4800, 3600, 2400, 1800, 1200, 900, 600, 300, 150, 134, 110, 75, 50, or EXT. Note: If EXT is entered, the baud rate is set externally. The default is 9600 baud.
HALF,FULL	is the half or full duplex operation. The default is HALF.
ECHO,NOECHO	specifies characters are or are not to be echoed by the computer as they are received. If not specified, the default is ECHO for FULL duplex or NOECHO for HALF duplex.
parity	is ODD, EVEN, or NONE. The default is EVEN. If NONE is specified, the serial character is smaller due to the absence of the parity bit.
charsize	is the character size: 5, 6, 7, or 8. If not specified, the default is seven.
stopbits	is the number of stop bits: S1, S1.5, or S2. If not specified, the default is S1.
GRAP	specifies graphic device. If not specified, the default is not graphic device.
MODEM	sets modem bit in UDT. If not specified, the default is not set.
REMOTE	sets dial-up bit in UDT. If not specified, the default is not set. Also, sets switched mode. The default is private mode.
NOTSM	specifies a non-TSM device. If not specified, the default is TSM device.
INIT value	indicates the presence of hexadecimal initialization data. Five words of hexadecimal value in the ALIM initialization format as follows.

The following table shows the ALIM terminal configuration possible and their resulting operations.

SYSGEN Device Directive, FULL Option Specified	LOGONFLE FULL/HALF Option Specified	LOGONFLE ECHO/NOECHO Option Specified	Resulting Terminal Operation
Yes	Full	Echo	Half, Echo
Yes	Full	Noecho	Full, Noecho
Yes	Half	Echo	Half, Echo
Yes	Half	Noecho	Half, Noecho
No	Full	Echo	Half, Echo
No	Full	Noecho	Half, Noecho
No	Half	Echo	Half, Echo
No	Half	Noecho	Half, Noecho

ALIM Initialization Format

Fields containing an asterisk (*) are required; the remaining fields are optional. If not specified, the default is OFF.

Word	0 – 7	8 – 15	16 – 23	24 – 31
1	Channel time-out value*. See Note 1.			
2	Mode *. See Note 2.	Baud *. See Note 3.	Format. See Note 4.	Zero
3	Code. See Note 5.	STX. See Note 6	EXT. See Note 7.	CHAR A. See Note 8.
4	CHAR B. See Note 9.	CHAR C. See Note 10.	CHAR D (ring). See Note 11.	S1. See Note 12.
5	S2. See Note 13.	Zero		

Notes:

1. Bits 0-31 specify the channel time-out value in seconds. For example, X'0000005A' gives a time-out value of 90 seconds.

2. Mode bits are defined as follows:

Bits	Description
0-1	Stop bits (SB) interpreted as follows:

Bit 0	Bit 1	SB
0	0	Invalid
0	1	1 stop bit
1	0	1.5 stop bits
1	1	2 stop bits

2	Parity selection (PS) interpreted as follows:

Bit Setting	Description
0	Odd parity
1	Even parity

3	Parity enable (PE) interpreted as follows:

Bit Setting	Description
0	Disable parity
1	Enable parity

4-5	Character length (CL) interpreted as follows:

Bit 4	Bit 5	CL
0	0	5 bits
0	1	6 bits
1	0	7 bits
1	1	8 bits

6-7	Baud rate factor (BRF) interpreted as follows:

Bit 6	Bit 7	BRF
0	0	Synch mode
0	1	1X
1	0	16X
1	1	64X

3. Baud rate bits are defined as follows:

Bits	Description
8-11	Zero
12-15	With 16X selected in bits 6-7, baud rate is interpreted as follows:

Bit 12	Bit 13	Bit 14	Bit 15	Baud Rate
0	0	0	1	50 baud
0	0	1	0	75 baud
0	0	1	1	110 baud
0	1	0	0	134.5 baud
0	1	0	1	150 baud
0	1	1	0	300 baud
0	1	1	1	600 baud
1	0	0	0	900 baud
1	0	0	1	1200 baud
1	0	1	0	1800 baud
1	0	1	1	2400 baud
1	1	0	0	3600 baud
1	1	0	1	4800 baud
1	1	1	0	7200 baud
1	1	1	1	9600 baud
0	0	0	0	EXT

4. Format bits are defined as follows:

Bits	Description
16	Enable hardware ring detection
17	Enable break detection
18	Inhibit 1ms RTS off delay
19	Full or half duplex interpreted as follows:

Bit Setting	Description
0	Half duplex (HDX)
1	Full duplex (FDX)

Bits	Description
20	Switch indicator interpreted as follows:

Bit Setting	Description
0	Private
1	Switched line

Bits	Description
21	Must be set for types 103 and 212 modems

Bit Setting	Description
0	Switched
1	Constant

Bits	Description
22-23	Zero

5. Code bits are defined as follows:

Bits	Description
0-3	Zero
4-7	Code bits (four-bit hexadecimal value) interpreted as follows:

Value	Description
0	No action
1	Terminate on CHAR A
2	Terminate on CHAR A or B
3	Terminate on CHAR A, B, or C
4	Terminate on CHAR A* followed by B*
5	Terminate on CHAR A* followed by B* followed by C*

* These characters must be different from STX, ETX, ring, or strip characters

6. Start of text character (STX) bits are defined as follows:

Bits	
8	Action bit interpreted as follows:

Bit Setting	Description
0	Recognize character
1	Inhibit character recognition

9-15	Start of text character (STX) to be recognized (seven-bit ASCII character). The character must be different from ETX, strip, ring, and terminate characters.

Upon recognition, unblind mode is set causing this character and subsequent characters to be passed to memory. Blind mode, which deletes characters or blocks of characters, is set by either a receive blind order or the recognition of an ETX character during a receive blind order.

7. End of text character (ETX) bits are defined as follows:

Bit	Description
16	Action bit interpreted as follows:

Bit Setting	Description
0	Recognize character
1	Inhibit character recognition

17-23	End of text character (EXT) to be recognized (seven-bit ASCII character). The character must be different from STX, strip, ring, and terminate characters.

This character is recognized only during a receive blind order. Upon recognition, blind mode is set which blocks characters from being passed to memory. However, character recognition is still in effect for the STX and the ABC characters.

8. Termination CHAR A bits are defined as follows:

Bits	Description
24	Zero
25-31	Termination CHAR A (seven-bit ASCII character) which must be different from STX, ETX, ring, and strip characters.

Recognition of CHAR A causes channel end if the character is recognized in the manner specified in bits 0-7.

9. Termination CHAR B bits are defined as follows:

Bits	Description
0	Zero
1-7	Termination CHAR B (seven-bit ASCII character) which must be different from STX, ETX, ring, and strip characters.

Recognition of CHAR B causes channel end if the character is recognized in the manner specified in Word 3, bits 0-7.

10. Termination CHAR C bits are defined as follows:

Bits	Description
8	Zero
9-15	Termination CHAR C (seven-bit ASCII character) which must be different from STX, ETX, ring, and strip characters.

Recognition of CHAR C causes channel end if the character is recognized in the manner specified in Word 3, bits 0-7.

11. Ring interrupt CHAR D bits are defined as follows:

Bits	Description
16	Action bit interpreted as follows:

Bit Setting	Description
0	Recognize character
1	Inhibit character recognition

Bits	Description
17-23	Ring interrupt CHAR D bits (seven-bit ASCII character) which must be different from STX, ETX, strip, and terminate characters. When recognized, and no I/O is in progress, a ring interrupt is generated. Ring character reporting is enabled and disabled in the same manner as breaks.

12. Strip character (S1) bits are defined as follows:

Bits	Description
24	Action bit interpreted as follows:

Bit Setting	Description
0	Recognize character
1	Inhibit character recognition

Bits	Description
25-31	Strip character (S1) bits (seven-bit ASCII character) which must be different from STX, ETX, ring, and terminate characters.

The S1 character is to be stripped from an incoming data stream. The character is unconditionally removed from the input buffer. The character cannot have been previously specified for a control function.

13. Strip character (S2) bits are defined as follows:

Bits	Description
0	Action bit interpreted as follows:
1-7	Strip character (S2) bits (seven-bit ASCII character) which must be different from STX, ETX, ring, and terminate characters.

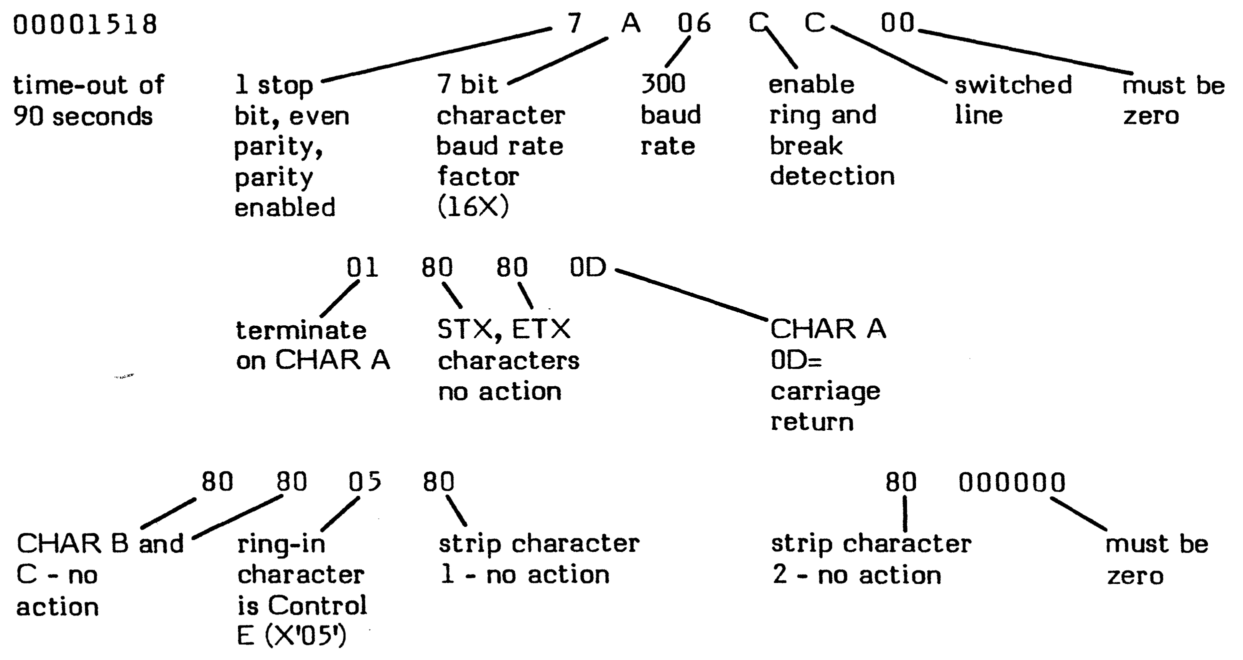
Bit Setting	Description
0	Recognize character
1	Inhibit character recognition

The S2 character is to be stripped from an incoming data stream. The character is unconditionally removed from the input buffer. The character cannot have been previously specified for a control function.

Example

The following example shows a dial-in type 103 modem interface to an ALIM with one stop bit, even parity, parity enabled, seven-bit character length, baud rate factor of 16X, and data rate of 300 baud.

```
2003 MODEM REMOTE INIT 00001518 7A06CC00 0180800D 80800580 80000000
```



00001518
time-out of 90 seconds
7
1 stop bit, even parity, parity enabled
A
7 bit character baud rate factor (16X)
06
300 baud rate
C
enable ring and break detection
C
switched line
00
must be zero
01
terminate on CHAR A
80 80
STX, ETX characters no action
0D
CHAR A 0D= carriage return
80 80
CHAR B and C - no action
05
ring-in character is Control E (X'05')
80
strip character 1 - no action
80
strip character 2 - no action
000000
must be zero

10.7.3 ACM Controller Record Syntax and Defaults

Syntax:

$$\text{ccaa}\quad[\text{baud}]\quad\begin{bmatrix}\text{HALF}\\\text{FULL}\end{bmatrix}\begin{bmatrix}\text{ECHO}\\\text{NOECHO}\end{bmatrix}\quad[\text{ parity }]\ [\text{ charsize }]\ [\text{ stopbits}]\,[\text{REMOTE}]$$

[MODEM] [GRAP] [NOTSM] [RXON] [WXON] [RHWF] [WHWF] [RDTR] [RRTS] [INIT value] [CXR=n]

ccaa	is the channel and subaddress of the device
baud	is baud rate: 19200, 9600, 7200, 4800, 3600, 2400, 2000, 1800, 1200, 600, 300, 150, 134, 110, 75, or 50. If not specified, the default is 9600 baud.
HALF,FULL	specifies half- or full-duplex operation. If not specified, the default is HALF. Refer to the next section.
ECHO,NOECHO	specifies characters are or are not echoed by the computer as they are received. If not specified, the default is ECHO for full-duplex or NOECHO for half-duplex.
parity	is ODD, EVEN, or NONE. If not specified, the default is EVEN. If none is specified, the serial character is smaller due to the absence of the parity bit.
charsize	is character size: 5, 6, 7, or 8. If not specified, the default is seven.
stopbits	is the number of stop bits: S1, S1.5, or S2. If not specified, the default is S1.
REMOTE	sets dial-up bit in UDT. If not specified, the default is not set.
MODEM	sets modem bit in UDT. If not specified, the default is not set. Also sets modem bit for ACE parameters. The default is no modem.
GRAP	specifies graphic device. If not specified, the default is not graphic device.
NOTSM	specifies a non-TSM device. If not specified, the default is TSM device.
RXON	specifies software read flow control (XON/XOFF). If not specified, the default is no software read flow control. If this option is selected for a terminal, NOECHO should be specified so the terminal can be used in local echo mode. This avoids line contention caused by echoplex.
WXON	specifies software write flow control (XON/XOFF). If not specified, the default is no software write flow control.
RHWF	specifies hardware read flow control (DTR). This parameter cannot be specified if either RXON or WXON has been specified. If not specified, the default is no hardware read flow control. If this option is selected for a terminal, NOECHO should be specified. This avoids line contention caused by echoplex.
WHWF	specifies hardware write flow control (DTR). This parameter cannot be specified if either RXON or WXON has been specified. If not specified, the default is no hardware write flow control. If this option is selected for a terminal, the NOECHO parameter should be specified. This avoids line contention caused by Echoplex.

RDTR	specifies the DTR line is used if RXON has been requested. This is the default, and it overrides RRTS if both are specified.
RRTS	specifies the RTS line is used if RXON has been requested.
INIT value	indicates the presence of hexadecimal initialization data. Value is a one-word hexadecimal value in the eight-line asynch initialization format.
CXR=n	specifies the number of seconds to wait for carrier before J.TSM resets DTR, causing the modem to go on hook. Valid values for n are 1-255. The value specified by n must include delays introduced by the modem. The use of this option will also cause a 2 second delay at logon time before transmitting the logon banner.

<u>ACM Initialization Format</u>

	0	1	2	3	4	5	6 7	8 9 10	11	12 15	16 23	24 31
Word 1	Duplex	MR	FP	PS	PE	SB	CL	Zero	DL	Baud rate	Wake-up character	Zero

Bit	Description
0	Full or half duplex bit defined as follows: 0 Full duplex 1 Half duplex
1	Modem ring (MR) bit defined as follows: 0 Disable 1 Enable
2	Forced parity (FP) bit defined as follows: 0 Normal parity as defined 1 Force parity to one if odd; force parity to zero if even
3	Parity selection (PS) bit defined as follows: 0 Odd parity 1 Even parity
4	Parity enabled (PE) bit defined as follows: 0 Disable parity 1 Enable parity

5 Stop bit (SB) defined as follows:

0	1 stop bit
1	2 stop bits or 1.5 stop bits for five character length

6-7 Character length (CL) defined as follows:

Bit 6	Bit 7	CL
0	0	5 bits
0	1	6 bits
1	0	7 bits
1	1	8 bits

Bits 8-10 Zero

Bit 11 Diagnostic loop (DL) bit defined as follows:

0	Reset diagnostic loop
1	Set diagnostic loop

Bits 12-15 Baud rate bits defined as follows:

Bit 12	Bit 13	Bit 14	Bit 15	Baud rate
0	0	0	0	50 baud
0	0	0	1	75 baud
0	0	1	0	110 baud
0	0	1	1	134.5 baud
0	1	0	0	150 baud
0	1	0	1	300 baud
0	1	1	0	600 baud
0	1	1	1	1200 baud
1	0	0	0	1800 baud
1	0	0	1	2000 baud
1	0	1	0	2400 baud
1	0	1	1	3600 baud
1	1	0	0	4800 baud
1	1	0	1	7200 baud
1	1	1	0	9600 baud
1	1	1	1	19200 baud

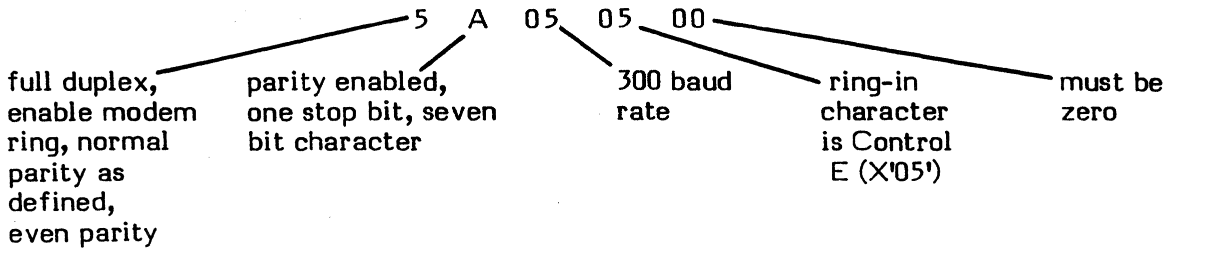
Bits 16-23 Eight-bit wake-up character

Bits 24-31 Must be zero

Example

The following example shows a dial-in modem interface to an eight-line asynch with one stop bit, even parity, parity enabled, full duplex, seven-bit character length, and modem ring enabled.

```
7EC0 MODEM REMOTE INIT 5A050500
```


5 A 05 05 00
full duplex, enable modem ring, normal parity as defined, even parity
parity enabled, one stop bit, seven bit character
300 baud rate
ring-in character is Control E (X'05')
must be zero

10.7.3.1 True Full-Duplex Operation for the ACM

The ACM device can be used for true full-duplex operation. The device must have been SYSGENed using the device directive's FULL parameter. This creates two UDT's per full-duplex subchannel: one for the read sub-channel and one for the write subchannel. The write subchannel subaddress is the read subchannel address plus X'8'; for example, a subchannel address of 7E00 derives a write subchannel address of 7E08. The device must be initialized as NOECHO (local echoplex) to avoid I/O contention with the write subchannel. After the device is initialized, it is in full-duplex, single-channel mode; all read and write I/O is performed through the read subchannel.

To set the device to full-duplex, dual-channel mode, the TSM OPTION UNQUIET directive must be used. Dual-channel I/O reads are performed on the read subchannel; writes are performed on the write subchannel. Dual-channel mode is also entered by assigning the read and write subchannels separately, as if OPTION UNQUIET were in effect, and issuing an M.RELP (SVC 1,X'27') before any I/O is performed. I/O is then performed through the FCB connected to the appropriate subchannel. To reset to the single-channel mode, use the M.RESP (SVC 1,X'26') service.

10.7.4 Sample LOGONFLE

```
05                                              WAKEUP CHARACTER WRITE
7EA0  19200  FULL                               ! DEFAULTS FOR OTHER VALUES
7EA1  9600   FULL NONE 8 S1 WXON REMO NOTSM     ! SAMPLE SERIAL PRINTER
7EA7  HALF   MODEM  REMOTE EVEN      300        ! DIAL-UP LINE
2000  9600   FULL   EVEN   7                    ! ALIM
```

REMOTE	sets dial-up bit in UDT. The default is not set.
MODEM	sets modem bit in UDT. The default is not set. sets modem bit for ACE parameters. The default is no modem.
EVEN	even parity is used.
7	7 bit data is used.
FULL	full duplex operation.
NOTSM	specifies a non-TSM device. The default is TSM device.
HALF	half duplex operation.
8	8 bit data is used.
S1	specifies 1 stop bit is required.
WXON	device uses software write flow control (XON/XOFF).

The following table shows the ACM configurations possible and their resulting operations.

SYSGEN DEVICE Directive, FULL Option Specified	LOGONFLE FULL/HALF Option Specified	LOGONFLE ECHO/NOECHO Option Specified	Resulting Terminal Operation
Yes	Full	Echo	Half, Echo
Yes	Full	Noecho	Full, Noecho
Yes	Half	Echo	Half, Echo
Yes	Half	Noecho	Half, Echo
No	Full	Echo	Half, Echo
No	Full	Noecho	Half, Noecho
No	Half	Echo	Half, Echo
No	Half	Noecho	Half, Noecho

10.8 Using INIT

INIT is a TSM command. LOGONFLE is assigned for input by default.

```
     TSM > $INIT [ccaa]
```

INIT	with no channel and subaddress uses the current version of LOGONFLE to reinitialize all devices that are currently free to allocate.
ccaa	a specific device can be reinitialized by supplying the appropriate channel and subaddress (hexadecimal). The record from LOGONFLE that matches the channel and subaddress reinitializes the device.

10.9 INIT Errors

INIT generates the following error messages:

DEVICE NOT PRESENT ADDR=ccaa
Specified device not plugged into the CPU.

DEVICE NOT TERMINATED ADDR=ccaa
Specified device not plugged in on device end of line.

M.ASSN DENIAL,NO LOGON FILE DEFAULT USED
A file named LOGONFLE is not on the system. Default parameters have been set. Non-TSM devices are not initialized.

M.ASSN DENIAL,ADDR=ccaa
Device at the specified channel and subaddress is in use and cannot be initialized.

NO UDT ENTRY FOR ADDR=ccaa
Specified device not SYSGENed.

ON ADDRESS ccaa, FIELD UNIDENTIFIED: xxxxxxxx
The string xxxxxxxx is not a valid key-word for a characteristic.

NON TSM DEVICE WITHOUT NOTSM IN LOGONFLE, ADDR=ccaa
A non-TSM device is specified in LOGONFLE without the NOTSM keyword.

ATTEMPT TO MAKE TSM DEVICE A NON TSM DEVICE, ADDR=ccaa
NOTSM keyword is specified in LOGONFLE for a TSM device.

INVALID NON TSM DEVICE TYPE CODE, ADDR=ccaa
Valid non-TSM device type codes are X'07' (CD) through X'0B' (PT) and X'0D' (CT) through X'1A' (U9).

TERMINAL SET-UP COMPLETE
Initialization complete.

10.10 The System Console (Messages)

Some system messages and operations pertain to the system console, and not to terminals running OPCOM, for example:

- mount and dismount messages for magnetic tape
- abort messages and codes for tasks running in the real-time environment
- I/O error conditions that can either be corrected off-line with I/O resumed or aborted

Displaying these messages on the console assumes a central location of configured system hardware, including the system console, the CPU, printers, and tape units. User tasks can also send messages to the system console.

10.10.1 Information Messages

Commands or responses to prompts at the system console always take precedence over messages sent to the console. The maximum uninterruptible output string for a message is 72 characters.

10.10.2 Action Messages

If a system message requires operator intervention and reply, the condition noted must be addressed. The reply is then entered using a carriage return as the terminator. For example,

```
    *CR7800 INOP : R,A? R
```

A card reader is not operating. The operator fixes it and responds R and a carriage return for resume. Input continues.

10.10.3 At the Terminal (Messages)

The terminal user receives only the OPCOM messages that pertain to directives issued from that terminal.

10.11 Floppy Disc Media Initialization (J.FORMF)

All IOP floppy discs must be properly formatted before they can be used on the operating system.

To format a floppy disc, type J.FORMF at the TSM prompt. The following message is displayed:

```
    J.FORMF-SPECIFY FORMAT:MODE0/MODE1/QUIT (REPLY 0, 1, OR Q):
```

If 0 (MODE0) is entered, the floppy disc is formatted as double density, 256 bytes per sector. The first physical sector of all tracks starts at hexadecimal 0 and ends at hexadecimal 19.

If 1 (MODE1) is entered, the floppy disc is formatted as double density, 256 bytes per sector. The first physical sector of all tracks starts at hexadecimal 1 and ends at hexadecimal 1A. MODE1 formatting allows data files to be transported to other computer systems that expect the physical sectors to begin at hexadecimal 1 and end at hexadecimal 1A. MODE1 is only supported on systems containing a Model 8031 Line Printer/Floppy Disc Controller.

If Q (QUIT) is entered, control returns to TSM and the floppy disc is not formatted.

The default mode selection should be MODE0 and must be MODE0 if data files are to be transported to another Gould CSD computer containing either a Model 8030 or 8031 Line Printer/Floppy Disc Controller.

The default device for J.FORMF is device address FL7EF0. To override the default, specify the following assign directive at the TSM prompt before J.FORMF is activated:

```
    $ASSIGN FL TO DEV=FLxxxx BLOCKED=N
```

where xxxx is the drive device address to be used by J.FORMF.

10.12 M.MOUNT File

The M.MOUNT file automatically mounts volumes at IPL.

The input file directive syntax is the same as the OPCOM MOUNT directive. Refer to the OPCOM utility documentation. The file must be stored unnumbered on the SYSTEM volume.

Examples of Usage:

```
EDT>COL
1.   MOUNT ATLAS04 ON DM0804 OPTION=PUBL
2.   MOUNT TB02 ON DM0802 OPTION=PUBL
          .
          .
          .
5.   <cr>
EDT> STO M.MOUNT SYS UNN
```

If desired, an asterisk (*) can be used in the volume name field. This causes the volume on the specified drive to be mounted, regardless of its name.

At IPL, a mount message is automatically displayed for each volume in the file. If a NO is entered to any message, an attempt is made to MOUNT any remaining volumes.

The OPTION=PUBL parameter must be specified for each volume. If it is not specified, the volume is dismounted when J.INIT exits because the volume's use count is zero. Volumes to be mounted as nonpublic should be mounted using the TSM MOUNT directive.

10.13 Operator Intervention Inhibit

When operator intervention inhibit is in effect, all prompts normally displayed on the system console are suppressed and the following defaults are in effect:

Prompt	Default
System date/time	More recent date/time recorded by J.DTSAVE or last system volume mount
Swap volume channel and subaddress	System volume
Mount messages - formatted volumes	Ready
Dual-port volume cleanup confirmation*	No
Wait I/O request retry	Abort (DPO)
Invalid port ID specification*	Port zero
Dual-port volume mount request verification for an allocated port*	Continue

The prompts marked by an asterisk (*) apply only when a dual-processor shared volume is mounted. If the default responses to these prompts are not the desired responses, unreliable file system management can result.

After system initialization, wait I/O request retry prompts remain inhibited. When a wait I/O retry is requested, a message is displayed on the system console specifying the inoperable device and the request is aborted.

Mount messages are not inhibited after system initialization. The SYSGEN or OPCOM MODE directive inhibits mount messages after system initialization. If mount messages are inhibited and a device is not ready when J.MOUNT issues a mount request, the request is repeated approximately 60 seconds later. If the device is still not ready after the second request, the mount request is aborted.

When operator intervention is inhibited and an attempt is made to boot a master system image, operator intervention inhibit is overridden and an operator must respond to the prompts.

10.14 System Date/Time Backup Program (J.DTSAVE)

The System Date/Time Backup Program (J.DTSAVE) is a privileged, nonresident system program. J.DTSAVE records the current date and time to the system volume at regular intervals. The intervals are established by the SYSGEN directive DTSAVE. If operator intervention is inhibited, the system date and time are initialized to either the last date and time recorded by J.DTSAVE or the date and time of the last system volume mount, whichever is more recent. Entering a chronologically accurate date increases the reliability of file system management. J.DTSAVE is activated by the SYSGEN and OPCOM ACTIVATE directives.

10.15 Swap Scheduler Control Options

The swap scheduler processes entries in the Memory Request Queue (MRQ). It provides memory allocation and swap scheduling for individual memory requests. So that the swap scheduler can function according to the requirements of a system, the following swap scheduler options can be set by the system administrator:

1) Swapper Algorithms
2) Wait State Ordering
3) Wait State Swap-on Priority Only
4) Call Back Swap-on Priority Only
5) User Set Swap-on Priority Only Flag
6) User Set Swap Inhibit Flag
7) Swap Thrash Control

To set or reset an option, edit the Swap Option File (SJ.SWAPR2). SJ.SWAPR2 contains instructions on how to edit the file. After the file is edited, it must be resaved as SJ.SWAPR2. Then, the JCL file, JJ.A.SWP, must be run to catalog the swapper load module (J.SWAPR). After J.SWAPR is cataloged, the system administrator must restart the system to incorporate the modified swap scheduler into the operating system.

If SJ.SWAPR2 is not modified, the swap scheduler options are defaulted so that J.SWAPR performs similar to MPX-32 Release 3.2C. To revert to swapping that is similar to MPX-32 Release 3.2C after the swap option file has been edited, re-edit the SJ.SWAPR2 file and set all options to their default settings.

Swap-on priority only (SOPO) means that a task will be outswapped only when memory is required for an equal or higher priority task, regardless of the task's state. If swap-on priority only is not in effect, a higher priority task in any wait state can be outswapped for a lower priority task that is ready to run. If the outswapped higher priority task changes to a ready-to-run state too soon, the task is inswapped again causing delays and reducing system performance.

1) Swapper Algorithms

There are two swap scheduler alogorithms:

Algorithm 1 - determines if sufficient swappable memory is available for the memory requestor before selecting an outswap candidate. If Algorithm 1 is set and the requested memory is not available, the swap scheduler suspends until the next memory scheduler event.

Algorithm 2 - outswaps a task without determining if sufficient swappable memory will be available for the memory requestor. If no memory is outswapped for one second, the outswapped tasks are eligible to be inswapped. When there has not been an inswap for one second, the cycle can repeat. This algorithm assumes that the highest priority task on the memory request queue will receive its required memory.

To set this option, edit the Method section of SJ.SWAPR2.

2) Wait State Ordering

The Wait State Ordering option determines the order that the swap scheduler searches the queues for swap candidates. The Swap Option File (SJ.SWAPR2) contains a list of queues that can be arranged in any order. If this list is not modified, the swap scheduler searches the list as follows:

Queue	Order
HOLD	First
SUSP	
RUNW	.
SWDV	
SWDC	.
SWSR	
SWSM	.
SWLO	
SWFI	.
MRQ	
ANYW	.
SWGQ	
SWTI	.
SWIO	
SWMP	Last

To change the list, edit the Wait State Search section of SJ.SWAPR2.

WARNING: All 15 wait states must appear in the list.

3) Wait State Swap-on Priority Only (SOPO)

The Wait State Swap-on Priority Only (SOPO) option allows any of the 15 wait states to be Swap-on Priority Only. This option allows all tasks in the designated wait state to be SOPO. If this option is not set, tasks are not designated SOPO because of their wait state.

To set this option, edit the Wait State Search section of SJ.SWAPR2.

4) Call Back Swap-on Priority Only (CB.SOPO)

The Call Back Swap-on Priority Only (CB.SOPO) option determines whether a task is SOPO when callback is pending for a no-wait message request or a no-wait run request. If this option is not set, the task is not SOPO.

To set this option, edit the CB.SOPO section of SJ.SWAPR2.

5) User Set Swap-on Priority Only Flag (US.SOPO)

The User Set Swap-on Priority Only Flag (US.SOPO) option allows a privileged task to set a flag bit in the DQE (DQE.USPO). This allows the task to be SOPO until the bit is reset. If this option is not set, DQE.USPO is ignored.

To set this option, edit the US.SOPO section of SJ.SWAPR2.

6) User Set Swap Inhibit Flag (US.SWIF)

The User Set Swap Inhibit Flag (US.SWIF) option allows a privileged task to set a flag bit in the DQE (DQE.USWI). This specifies the task as ineligible for outswapping until the bit is reset. If this option is not set, the swap inhibit bit (DQE.USWI) is ignored.

To set this option, edit the US.SWIF section of SJ.SWAPR2.

7) Swap Thrash Control

Swap thrashing may occur when tasks that are frequently changing between the ready-to-run and wait states are competing for memory. Thrashing causes these tasks to be swapped between memory and the swap file on the disc as their state changes. To reduce swap thrashing and improve system performance, the user can set SWAPDUTY or SWAPPERI.

SWAPDUTY is a value between 0 and 100 that indicates the percentage of time that the swap scheduler can actively swap tasks. If the swap scheduler activity is greater than SWAPDUTY, J.SWAPR processes only swap-on priority only tasks. If the SWAPDUTY is 100, there is no thrash control and all tasks are swapped according to wait state. If the SWAPDUTY is zero, only SOPO tasks are swapped. If most of the tasks are SOPO and the swap scheduler is very active, thrashing may occur.

SWAPPERI is a value between 2 and 64 seconds. If set to a value less than two, SWAPPERI defaults to two seconds. If set to a value greater than 64, SWAPPERI defaults to 64 seconds.

The percentage of the swap activity is constantly being monitored by the system. If the swap activity is greater than the SWAPDUTY, only SOPO tasks are swapped. If the percentage of swap activity is less than the SWAPDUTY, all tasks are swapped on a wait state basis.

If this option is not set, the system default is no thrash control and SWAPDUTY is measured every two seconds per megabyte of memory.

To set this option, edit the Thrash Control section of SJ.SWAPR2.

10.16 Multiprocessor Recovery Task (J.UNLOCK)

J.UNLOCK is the multiprocessor recovery task. In a multiprocessor system, J.UNLOCK allows a processor to recover and continue processing when another processor goes off-line. Resource locks, assign counts, and user counts owned by the off-line processor are removed from the shared volumes by the on-line processor.

J.UNLOCK is a resident, privileged task. It is activated by an OPCOM UNLOCK directive or by J.MOUNT when a multiprocessor shared volume is mounted. When the last multiprocessor shared volume is dismounted from the system, J.UNLOCK deactivates.

10.17 Shadow Memory

After the shadow memory board is installed and shadow memory is configured, the run-time performance improves for a task that uses shadow memory.

MPX-32 supports the following shadow memory configurations:

. a single processor (IPU or CPU) with a single region of shadow memory. See Figure 10-1.

. two processors (IPU and CPU) with a single region of shadow memory. The regions must have the same starting and ending addresses and be shadowed by an equal number of shadow memory boards on both processors. See Figure 10-2.

. two processors (IPU and CPU) with two nonoverlapping shadow memory regions. The starting addresses of both regions are different, and the number of shadow memory boards for each processor can be unequal. See Figure 10-3.

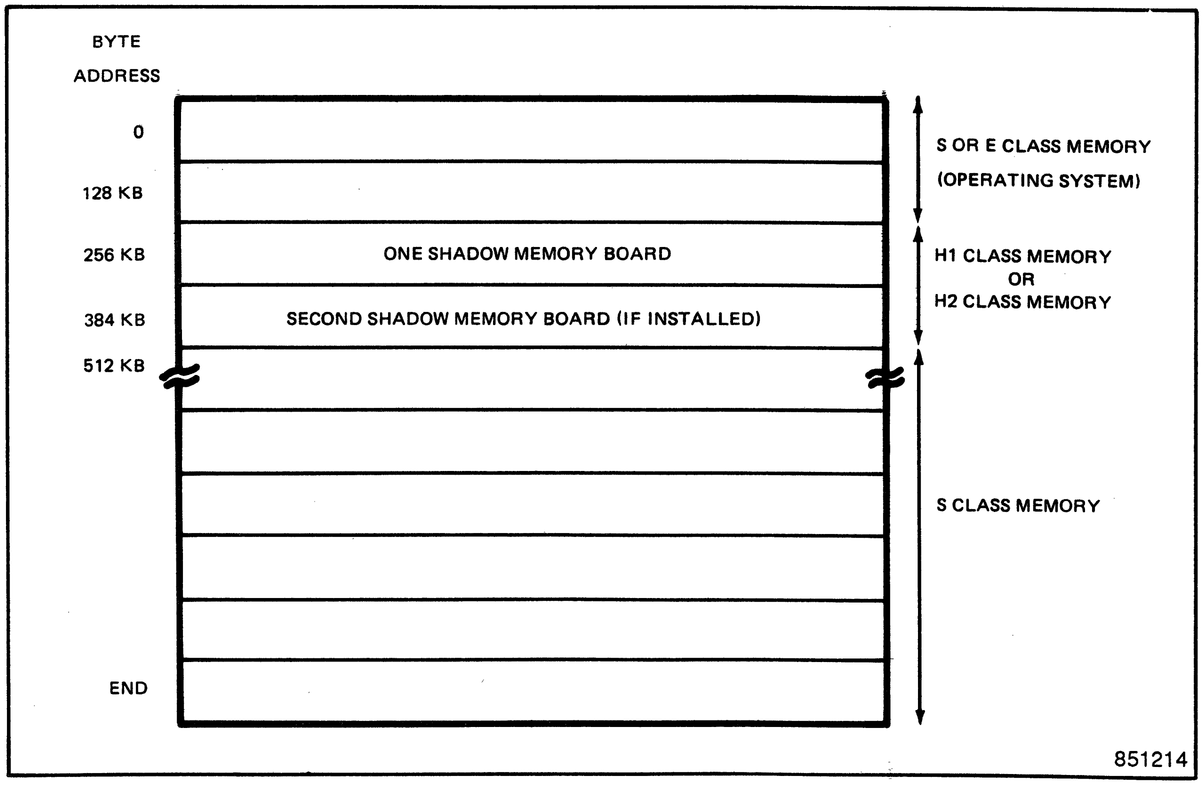


Figure 10-1. Shadow Memory Configuration with a Single Processor and a Single Region of Shadow Memory

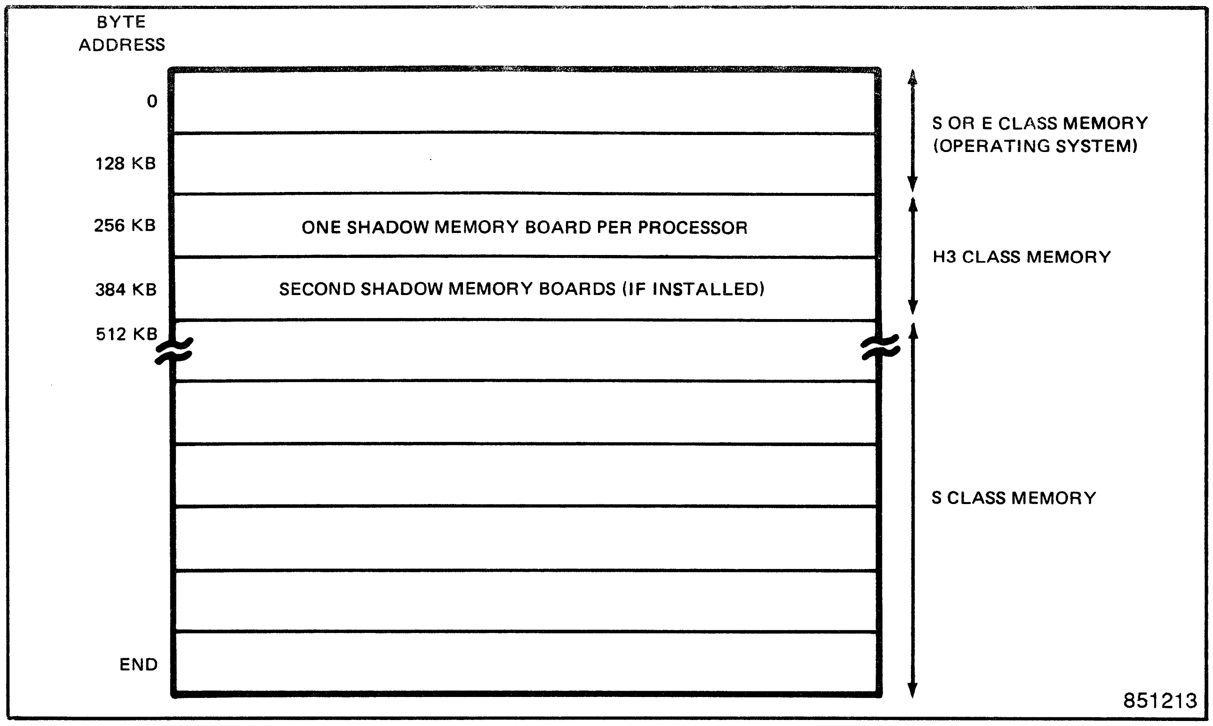

BYTE ADDRESS
0
128 KB
256 KB
384 KB
512 KB
END
ONE SHADOW MEMORY BOARD PER PROCESSOR
SECOND SHADOW MEMORY BOARDS (IF INSTALLED)
S OR E CLASS MEMORY (OPERATING SYSTEM)
H3 CLASS MEMORY
S CLASS MEMORY
851213

Figure 10-2. Shadow Memory Configuration with Two Processors and a Single Region of Shadow Memory

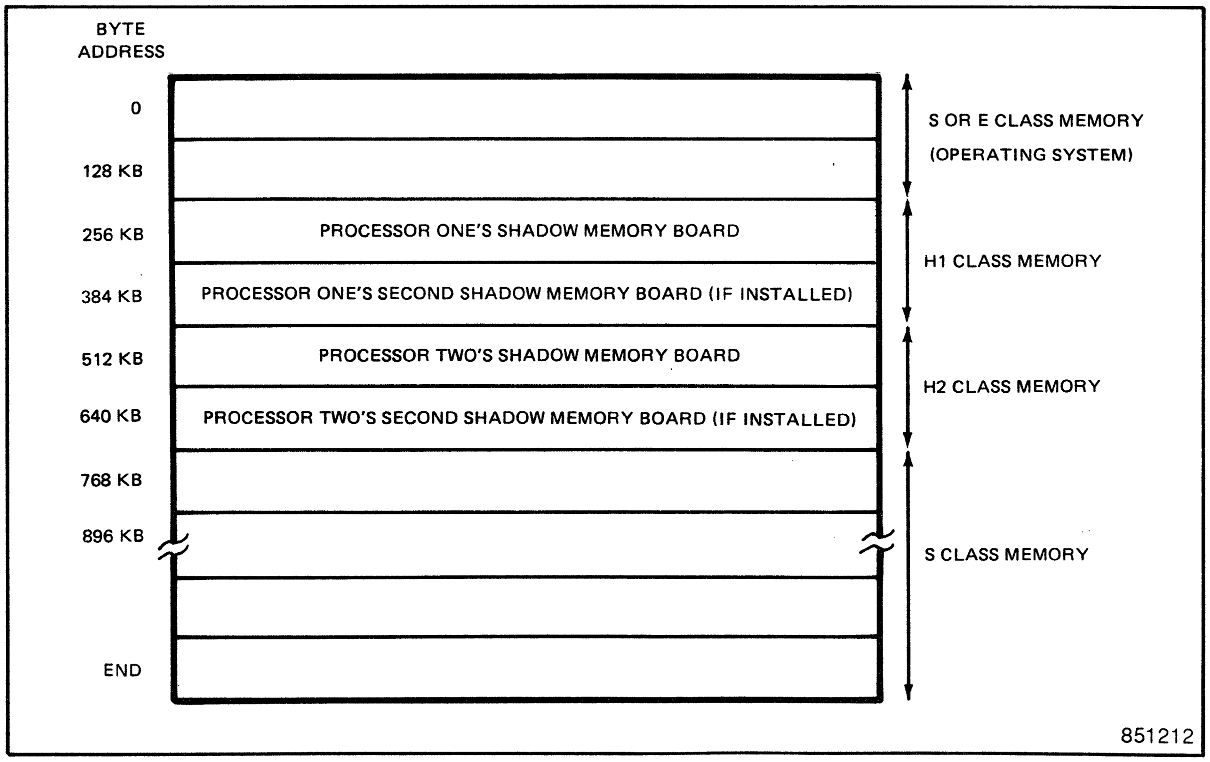

BYTE ADDRESS
0
128 KB
256 KB
384 KB
512 KB
640 KB
768 KB
896 KB
END
PROCESSOR ONE'S SHADOW MEMORY BOARD
PROCESSOR ONE'S SECOND SHADOW MEMORY BOARD (IF INSTALLED)
PROCESSOR TWO'S SHADOW MEMORY BOARD
PROCESSOR TWO'S SECOND SHADOW MEMORY BOARD (IF INSTALLED)
S OR E CLASS MEMORY (OPERATING SYSTEM)
H1 CLASS MEMORY
H2 CLASS MEMORY
S CLASS MEMORY
851212

Figure 10-3. Shadow Memory Configuration with Two Processors and Two Regions of Shadow Memory

Configurations not supported are:

- processors with multiple shadow boards that are not jumpered over a contiguous region of physical memory
- two processors with multiple shadow memory boards that have shadow memory regions that are overlapping, but do not have the same starting and ending addresses.

A shadow memory board can be jumpered to physical address zero which causes the resident portion of MPX-32 to be shadowed.

A processor's shadow memory is assigned to a contiguous region in physical memory; shadow memory is not supported for noncontiguous regions of physical memory.

There are three shadow memory classes:

H1	Shadow memory installed for processor one.
H2	Shadow memory installed for processor two.
H3	Processor one and two have equal amounts of shadow memory, and are both jumpered to the same addresses.

These classes specify the memory types used in the SYSGEN TYPE directive. The TYPE directive can reserve shadow memory for tasks that request shadow memory.

Notes:

The Processor Select Switch determines which processor is the CPU and which is the IPU. MPX-32 matches the processors to the CPU or IPU and determines if shadow memory is present.

If shadow memory is not reserved, tasks that require S class memory can allocate shadow memory. This can cause tasks that require the shadow memory to wait for shadow memory. If a task requests shadow memory and shadow memory has been allocated, J.SWAPR attempts to allocate shadow memory according to the task's state and priority.

The difference between H class memory and the shadow memory classes is that normally, if H class memory is not available, E class can be allocated. With the shadow memory classes, E class memory is never requested for the shadow memory area. H class memory cannot be used with H1, H2, or H3 class memory.

After a task has been configured, there are three ways that a task can access shadow memory:

J.SHAD	This utility permanently specifies the portions of a task's logical address to be located in shadow memory. For more information on J.SHAD, refer to Volume II.
$SHADOW	This directive specifies the portions of a task's logical address to be located in shadow memory. For more information on $SHADOW, refer to Volume II.

M.PTSK This service activates a task. When used with a type 11 RRS (assign to shadow memory), it specifies the portions of a task's logical address to be located in shadow memory. Nonshadow memory is specified as E or S class in word 0, byte 3 of the parameter block. This service is only for privileged users. For more information on M.PTSK, refer to Volume I.

10.18 Memory Disc

10.18.1 General Information

The MPX-32 memory disc is a partition of main memory that simulates a moving head disc with the following exceptions:

. Memory disc seeks and accesses data at a faster rate.

. Memory disc has a lower capacity.

. Memory disc is not bootable.

. Memory disc cannot be used as a physical cache disc.

. Memory disc can only be single or dual-ported.

If configured, the memory disc partition is allocated during system initialization, unless the system administrator requests otherwise at system generation.

There are two system tasks for memory disc usage -- J.MDSAVE and J.MDREST. J.MDSAVE dumps the contents of memory disc to a system file. J.MDREST restores any unmounted memory discs saved by J.MDSAVE. See the J.MDSAVE and J.MDREST sections in this chapter for more information.

Memory disc can be marked on-line and off-line by the OPCOM ONLINE and OFFLINE directives. These directives also allow the allocation and deallocation of the memory disc's memory. See OPCOM ONLINE and OFFLINE for more information.

The memory disc handler is named H.MDXIO. Refer to the MPX-32 Technical Manual for more information.

Notes:

Dual-ported memory discs can be located only in a multiprocessor shared memory area. Single-ported memory discs can be located in any memory area except the multiprocessor shared memory area.

Up to eight memory discs can be configured as long as the subaddress is not the same as the null device. Memory discs can be configured on channel 00, subaddress 0 and any following even subaddress. Memory allocated to a memory disc must be contiguous.

If the reloading of all memory discs is required at restart or IPL, use the SYSGEN SEQUENCE directive to activate J.MDREST. The M.MOUNT directive cannot reload a memory disc and should not be used to mount any memory discs that will be reloaded.

High priority real-time tasks that relinquish control only while I/O is in progress take an unfair amount of CPU time unless their I/O priorities are modified.

10.18.2 Memory Disc Configuration

The configuration size of a memory disc must be greater than 12KB. The memory disc size is specified at SYSGEN. To configure a memory disc, the following steps are necessary:

1. Add a SYSGEN CONTROLLER directive for each memory disc (channel 00)
2. Add a SYSGEN DEVICE directive for each memory disc.
3. Add a SYSGEN SEQUENCE directive for J.MDREST, if required. For more information on this option, refer to the J.MDREST system task in this chapter.
4. Perform a SYSGEN.
5. Restart the system.

Refer to the appropriate directive for specific memory disc information.

10.18.3 Memory Disc Usage

<u>Formatting and Mounting a Memory Disc</u>

After a system restart, a memory disc is formatted using the Volume Formatter (J.VFMT). A memory disc can then be mounted using the OPCOM or TSM MOUNT directive.

<u>Accessing a Memory Disc</u>

After a memory disc has been formatted and mounted, then it can be accessed like a hard disc; for example, through Volume Manager and the Text Editor.

<u>Dismounting a Memory Disc</u>

If a memory disc is mounted with the OPCOM MOUNT directive, it must be dismounted with the OPCOM DISMOUNT directive. The same is true for the TSM MOUNT and DISMOUNT directives. Failure to specify the corresponding directive results in an error message.

10.18.4 Memory Disc Aborts and Errors

Abort Cases

The following is the memory disc abort code and its description.

Code	Description
MM01	Request for memory disc I/O to a location outside the memory disc boundaries

Aborts during memory disc installation occur:

. If DM00 is not specified in the CONTROLLER directive for a memory disc. SYSGEN aborts, and an error message is generated in the SYSGEN listed output file.

. When reloading a memory disc that has been publicly mounted with M.MOUNT

. During an attempt to mount or format a memory disc for which no partition is present

Other possible aborts are the same as for other types of disc drives.

Error Cases

Error cases can arise from improper configuration and will trap out at assign time. The following are memory disc errors and result in messages displayed on the operator's console:

. Insufficient contiguous memory of the correct type

. Requests for OPCOM to mark on-line a deallocated memory disc that cannot be fulfilled because there is no available contiguous memory

. Attempts to deallocate a dual-ported memory disc's memory

All other errors like end-of-medium and full disc are a result of improper disc usage, and display the same error messages as other types of disc device errors.

10.18.5 Memory Disc Save Task (J.MDSAVE)

The Memory Disc Save Task (J.MDSAVE) saves the contents of a memory disc to a system file. To activate J.MDSAVE, logon as the system administrator, then type:

```
J.MDSAVE DM00nn
```

nn   is the subchannel of the memory disc

The contents of the specified memory disc are saved to a system file named @SYSTEM(SYSTEM)DM00nn. See Figure 10-1.

10.18.6 Memory Disc Restore Task (J.MDREST)

The Memory Disc Restore Task (J.MDREST) restores all unmounted memory discs that have been saved by the J.MDSAVE system task. Before using J.MDREST, the J.MDSAVE system task must have been used. See Figure 10-4. To activate J.MDREST, logon as the system administrator, then type:

```
J.MDREST
```

When the SYSGEN SEQUENCE directive is entered as follows, J.MDREST is activated at system initialization:

```
SEQUENCE=J.MDREST
```

This reloads any unmounted memory discs from the system disc files saved by J.MDSAVE (@SYSTEM(SYSTEM)DM00nn). It is important to activate J.MDREST with the SYSGEN SEQUENCE directive and not the SYSGEN ACTIVATE directive. If ACTIVATE is used, an error message is displayed.

The M.MOUNT file should not contain entries for the memory discs to be reloaded. If the M.MOUNT file contains entries for memory discs to be reloaded, the mount messages for these entries should be aborted. If MODE=SNOP is specified in the SYSGEN MODE directive, no operator messages are displayed and the discs are mounted.

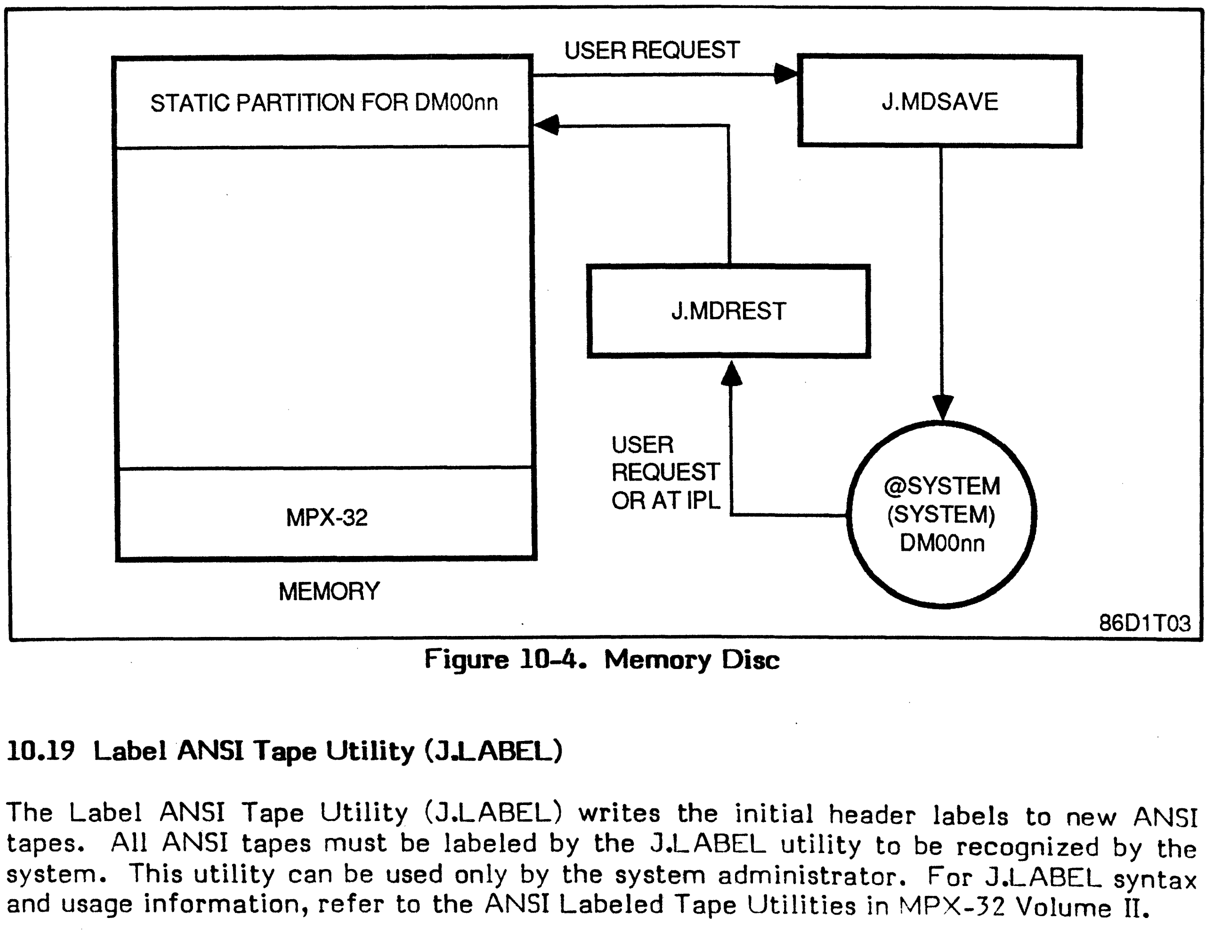


Figure 10-4. Memory Disc

10.19 Label ANSI Tape Utility (J.LABEL)

The Label ANSI Tape Utility (J.LABEL) writes the initial header labels to new ANSI tapes. All ANSI tapes must be labeled by the J.LABEL utility to be recognized by the system. This utility can be used only by the system administrator. For J.LABEL syntax and usage information, refer to the ANSI Labeled Tape Utilities in MPX-32 Volume II.

CHAPTER 11

DEVICE INITIALIZER/LOADER (DEVINITL)

The Device Initializer/Loader (DEVINITL) initializes devices by using an initialization directive file. DEVINITL also loads microcode onto devices having Writable Control Storage (WCS) capabilities by using a firmware file in conjunction with the initialization directive file.

11.1 Initialization Directive File

DEVINITL directives are in a file named DEV_INIT located on the system volume and system directory. This file contains default directives used when DEVINITL is activated by TSM. The DEV_INIT file contains the following three types of directives:

. Option Specification (optional) -- overrides default parameters for file matching and message displays.

. Device Load (required) -- specifies devices to be initialized and optionally specifies files that contain microcode to be loaded.

. Device Initialization (optional) -- specifies a directive byte and/or data required to initialize a device.

Based on these directives and their defaults, if any, DEVINITL initializes devices and/or loads microcode, checking only record type, length, and checksum. Messages are displayed indicating successful device initialization and/or loading or the errors which occurred preventing successful initialization and/or loading.

The format of the initialization directive file is shown in Figure 11-1.

11.2 Firmware File

Microcode is loaded by a single Execute Channel Program call from a single buffer that is extracted from a firmware file on disc. When using TSM, the loading process is overridden by specifying OPTION 9 before activating DEVINITL, in which case only the initialization process is performed.

As the microcode can be of any type, DEVINITL can load any WCS within the scope of its recognized record types.

The firmware file contains two types of records: data and control. All data record types are valid. Five control record types are valid: logical end-of-record, header record, set address-field length, set checksum-field length, and terminate load record. These fields are further described in Table 11-1.

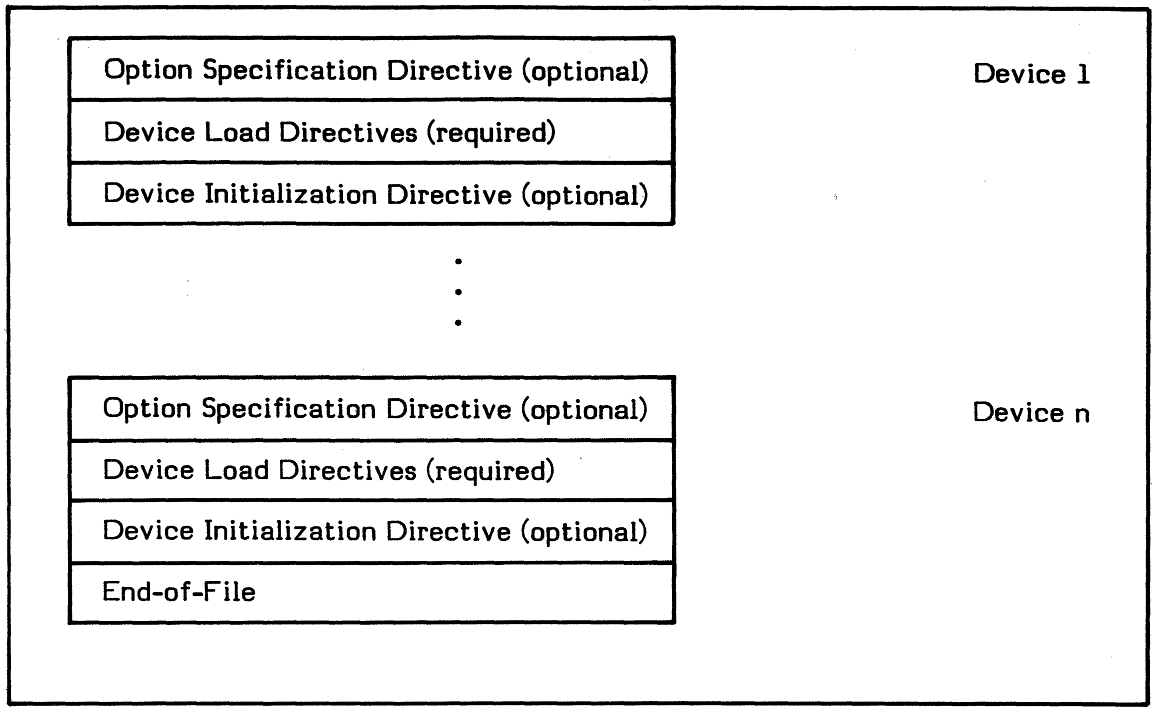

Option Specification Directive (optional)
Device Load Directives (required)
Device Initialization Directive (optional)
Device 1
Option Specification Directive (optional)
Device Load Directives (required)
Device Initialization Directive (optional)
End-of-File
Device n

Figure 11-1. Initialization Directive File Format

Table 11-1

Valid Control Record Types

Control Byte Value		
Decimal	Hex	Loader Control Record Description
0	0	Logical end-of-record - Remaining data in the current operating system record is ignored.
241	F1	Header record - The second byte of the record contains the number of data bytes in the record. The record can contain any file identification desired.
242	F2	Set address-field length - The next byte in the record defines the length of the address field in bytes for all subsequent loader data records.
243	F3	Set checksum-field length - The next byte in the record defines the length of the checksum field in bytes for all subsequent loader data records.
244-254	F4-FE	Reserved
255	FF	Terminate load record - This record indicates the end of the load. The address portion of this record specifies the initial transfer address for the control microcode previously read.

11.2.1 Loader Data Record Format

Loader data records define the data to be loaded into the device control memory (WCS). Each data record contains a byte count field, a microcode address field, a microcode data field, and a checksum field in the following format:

Record 0 n

Byte count See Note 1.	Firmware address. See Note 2.	Microcode data. See Note 3.	Checksum field. See Note 4.

Notes:

1. The byte count field defines the number of data bytes in the microcode data portion of the record. It also identifies the record as a loader data record as opposed to a loader control record. Valid byte counts are X'01' through X'F0'.

2. The firmware address field is a series of bytes defining the portion of the control memory where the associated data will be loaded. The default length of this field is two-bytes; however, it can be changed with a Set Address-Field Length control record. The length specified remains in effect until another Set Address-Field Length control record is specified or end-of-file is encountered.

3. The microcode data field is a series of bytes containing the data to be loaded at the address specified, and in the format expected by the device. This format is the responsibility of the user; DEVINITL only checks record length and checksum.

4. The checksum field is a series of bytes containing the checksum value for the microcode data field. The default length of this field is one byte; however, it can be changed with a Set Checksum-Field Length control record. The length specified remains in effect until another Set Checksum-Field Length control record is specified or end-of-file is encountered.

11.3 Using DEVINITL

DEVINITL can be activated at IPL, in batch mode or in interactive mode, by TSM.

To activate DEVINITL at IPL time, use the SYSGEN ACTIVATE directive. Refer to the SYSGEN Utility documentation. DEVINITL is activated immediately upon successful loading of the system image. When DEVINITL is activated at IPL time or in batch mode, milestone and error messages are displayed on the system console.

When DEVINITL is activated interactively, milestone and error messages are displayed on the terminal, and directives are passed to DEVINITL from the terminal. The first input required is a DEVICE directive. The initialization directive file is searched for the corresponding device entry.

If the device entry is found, any other valid directive can be entered. Any directive entered overrides the corresponding directive in that device's section. Any directive not entered defaults to that directive entry in the device's section. Terminal input only overrides initialization directive file directives; it does not permanently change them. A carriage return ends directive entry at the terminal.

If the device entry is not found, DEVINITL displays an appropriate message.

11.4 DEVINITL Directives

DEVINITL directives are summarized below, by command type. Each directive is described in detail in the remainder of this section, arranged alphabetically by directive name. Valid abbreviations are shown by underlining. In the initialization directive file, comments can be included in a directive line by specifying an exclamation point (!) anywhere on the line except within quoted ASCII strings. The remainder of the line is ignored.

All directives in an initialization directive file must be on separate lines.

When directives are entered through TSM, DEV_CNTRL is the only directive that must be on a separate line. All other directives can be continued across as many lines as necessary by placing a hyphen (-) as the last significant character on the line.

Option Specification Directive

OPTION	Specifies whether parameter defaults for file matching and message displays are to be overridden for subsequent device entries in the initialization directive file. This directive is valid in the initialization directive file only.

Device Load Directives

DEVICE	Specifies the device mnemonic, channel, and subchannel addresses for a device to be initialized and/or loaded. This directive is required for each device to be initialized and/or loaded. It must be the first command specified in the Device Load Directive section.
IDENT	Specifies a header record ASCII string that must be matched if OPTION WCSMATCH=Y is specified. This directive is not required if OPTION WCSMATCH=N.
REREAD	Specifies the number of read attempts to be performed on a firmware file if a read error occurs. The number specified is also used as the number of read attempts to be performed when the number of load attempts specified by a RETRY directive expires.
RETRY	Specifies the number of load attempts to be performed on a device if a load error occurs.
WCS_FILE	Specifies the pathname of a file containing microcode to be loaded onto a device. This directive is required if OPTION WCSMATCH=Y is specified or if microcode is to be loaded onto a device.

Device Initialization Directive

DEV_CNTRL	Specifies device initialization is required.

11.4.1 DEV_CNTRL Directive

The DEV_CNTRL directive initializes a device. If this directive is not specified for a device, that device is not initialized. Special directives or data needed to perform the initialization can be specified.

When this directive is encountered, DEVINITL creates an Input/Output Command Doubleword (IOCD) using any data specified. The IOCD is then passed to the Execute Channel Program for execution.

More than one DEV_CNTRL directive can be specified. However, only one DEV_CNTRL directive can be specified per input line.

Syntax:

```
DEV_CNTRL  [CMD=cc]  ┌data    ┐
                     │'string'│
                     └        ┘
```

CMD=cc	specifies the value of a valid hexadecimal IOCD directive byte. If not specified, default is X'FF' to initialize controller.
data	is any valid hexadecimal data. Each byte of data must be separated by a blank or a comma. Data cannot be carried over one input line.
'string'	is any ASCII data string enclosed in single quotes. Data cannot be carried over one input line. Its format is not checked.

If no parameters are specified, the device is initialized using system defaults.

Examples

```
DEV_CNTRL CMD=F3 'STARTUP'
```

The ASCII string STARTUP is passed to a device with a byte count of seven with the initialization directive byte X'F3'.

```
DEV_CNTRL F0 81 C2 F7 B1 77 FF
```

Data F081C2F7B177FF is passed to a device with a byte count of seven with the default initialization command byte X'FF'.

11.4.2 DEVICE Directive

The DEVICE directive specifies a device that will be initialized and/or loaded. This is required in each device's device load directive section in the initialization directive file (see Figure 11-1), and must be the first directive in the section.

The device to be initialized and/or loaded must have been specified as a device at SYSGEN. If this directive is not specified or if an invalid device is specified, a fatal error occurs, ending the loading process using the initialization directive file.

Syntax:

```
DEVICE=mmccss
```

mm	is a two-character device mnemonic
cc	is a two-character device channel address
ss	is a two-character device subchannel address

Example

```
DEVICE=CR2008
```

This initializes and/or loads firmware onto the card reader (CR) on channel 20, subchannel 08.

11.4.3 IDENT Directive

The IDENT directive causes the header record of a firmware file to be searched for a specified ASCII string.

This directive is required if OPTION WCSMATCH=Y is specified. If a match is found, firmware loading proceeds. If a match is not found, an error is reported and DEVINITL proceeds to the next device entry in the initialization directive file.

Syntax:

```
IDENT='string'
```

'string'	is any ASCII data string enclosed in single quotes. Data cannot be carried over one input line. Format is not checked.

Example

```
IDENT='FIRMWARE HEADER TY2.1'
```

This causes the header record of a firmware file to be searched for the string FIRMWARE HEADER TY2.1.

11.4.4 OPTION Directive

The OPTION directive overrides DEVINITL defaults for header record searches and output message displays for subsequent device entries in the initialization directive file. This directive remains in effect until another OPTION directive is specified or the initialization directive file terminates.

This directive is optional and is valid in the initialization directive file only. Although both the WCSMATCH and MILESTONE parameters are optional, one of them must be specified if the OPTION directive is used. Both of them can be specified in one directive. Values yes (Y) and true (T) are synonymous. Values no (N) and false (F) are synonymous.

Syntax:

$$\underline{\text{OPTION}} \left[\text{WCSMATCH=}\begin{Bmatrix}\text{Y}\\\text{N}\end{Bmatrix}\right] \left[\text{MILESTONE=}\begin{Bmatrix}\text{Y}\\\text{N}\end{Bmatrix}\right]$$

WCSMATCH	specifies if subsequent header records in the firmware file must match an ASCII string which is specified by an IDENT directive. If =Y and a match is not found, firmware loading is aborted. If =N is specified, a match is not required and firmware loading proceeds. If not specified, the default is N.
MILESTONE	specifies whether milestone messages indicating successful operations are to be displayed or inhibited during subsequent device initialization and firmware loading. If =Y is specified, messages are displayed. If =N is specified, milestone messages are not displayed (but error messages are). If nothing is specified, default is Y. See Section 11.6 for milestone messages.

Note: If this directive is specified, at least one of the parameters must be used.

Example

```
OPTION WCSMATCH=Y MILESTONE=N
```

This requires subsequent device entries in the initialization directive file to match the ASCII string specified with an IDENT directive. Milestone messages are not displayed.

11.4.5 REREAD Directive

The REREAD directive specifies the number of read attempts performed on a firmware file when a read error occurs. When the specified number is exhausted, an error is reported.

The number specified by this directive is also used independently by the RETRY directive.

Syntax:

```
    REREAD=num
```

num is any decimal number 0 through 99999999. If not specified, the default is one.

Example

```
    REREAD=3
```

This specifies three read attempts are made on the firmware file if a read error occurs.

11.4.6 RETRY Directive

The RETRY directive specifies the number of load attempts made on a device when a load error occurs. When the specified number is exhausted, DEVINITL automatically rewinds the firmware file and then attempts a read. The number of read attempts is the number specified in a REREAD directive.

When the RETRY and REREAD numbers are exhausted, an error is reported and DEVINITL proceeds to the next device entry in the initialization directive file.

Syntax:

```
    RETRY=num
```

num is a decimal number 0 through 99999999. If not specified, the default is three.

Example

```
    RETRY=5
```

This specifies that five load attempts are to be made on a device if a load error occurs. If loading is not successful after the fifth attempt, the firmware file automatically rewinds and read attempts are made, decrementing the REREAD directive specification.

11.4.7 WCS_FILE Directive

The WCS_FILE directive specifies the pathname of a file containing microcode to be loaded onto a device.

This directive is required if OPTION WCSMATCH=Y is specified or if microcode is to be loaded onto a device. If an invalid pathname is specified or DEVINITL is denied access to the specified file, an abort occurs and an error is reported.

Syntax:

```
    WCS_FILE=pathname
```

pathname	is the pathname of a file containing microcode

Example

```
    WCS_FILE=@SYSTEM(FIRMWARE)WCS.1
```

This loads the microcode file WCS.1 located on the system volume and FIRMWARE directory onto the specified device.

11.5 Example Initialization Directive File

In the following initialization directive file, all directives are on separate lines.

```
OPTION WCSMATCH=Y MILESTONE=N                    ! DEVICE 1
DEVICE=CR2008
ID='CARD READER FIRMWARE'
REREAD=0
RETRY=1
WCS_FILE=@SYSTEM(FIRMWARE)WCSCR
OPTION WCSMATCH=N                                ! DEVICE 2
DEVICE=TY7EB3
DEV_CNTRL CMD=F3 1B 1C
DEVICE=TY7EB4                                    ! DEVICE 3
DEV_CNTRL CMD=F3 1E
DEVICE=TY7EB5                                    ! DEVICE 4
DEV_CNTRL
OPTION WCSMATCH=N MILESTONE=Y                    ! DEVICE 5
DEVICE=DC7EC0
REREAD=3
RETRY=5
WCS_FILE=@SYSTEM(FIRMWARE)WCSDISC
```

In the following example, directives are entered from TSM. Directives are continued across lines except for DEV_CNTRL, which is on a separate line. In this case, the hyphen signifies continuation of input for the same device.

```
TSM> DEVINITL DEVICE=CR2008 ID='CARD READER FIRMWARE'-
DEV> REREAD=0 RETRY=1-
DEV> DEV_CNTRL
```

11.6 Milestone Messages

DEVICE xxxxxx INITIALIZED

This message is displayed when a device is successfully initialized.

DEVICE xxxxxx HAS BEEN LOADED WITH THE FIRMWARE
CONTAINED IN WCS FILE xxxxxxxx

This message is displayed when firmware is successfully loaded onto a device.

11.7 Error Conditions and Messages

11.7.1 Initialization Directive File Errors

The following message is displayed when DEVINITL is activated by TSM and a corresponding device entry is not found in the initialization command file.

NO DEFAULT DATA IS AVAILABLE FOR DEVICE xxxxxx

The following message is displayed when action is not taken on a device entry within the initialization directive file. Action is not taken because one of the following occurred: OPTION 9 is set, a DEV_CNTRL command is not specified for the device entry, or a WCS_FILE directive is not specified for the device entry.

** WARNING ** NO ACTION HAS BEEN CARRIED OUT FOR DEVICE xxxxxx

The following message is displayed when the initialization directive file does not contain sufficient directives for processing a device before end-of-file detection.

** ERROR ** A PREMATURE END-OF-FILE HAS BEEN DETECTED ON THE CONTROL FILE

The following message is displayed when a nondecimal number is specified with a REREAD directive.

** ERROR ** INVALID REREAD PARAMETER SPECIFIED xxxxxxxx

The following message is displayed when a nondecimal number is specified with a RETRY directive.

** ERROR ** INVALID RETRY PARAMETER SPECIFIED xxxxxxxx

11.7.2 Device Loading Errors

When a device cannot be successfully loaded, the following message is displayed along with the appropriate reason.

** ERROR ** UNABLE TO LOAD DEVICE xxxxxx FOR THE FOLLOWING REASON:

DEVICE CHANNEL MUST BE HEXADECIMAL.

DEVICE TYPE NOT CONFIGURED ON HOST SYSTEM. FOUND xx

```
UNABLE TO ALLOCATE DEVICE xxxxxx M.ASSN ERROR CODE: xx

UNABLE TO ALLOCATE FILE xxxxxxxx M.ASSN ERROR CODE: xx

WCS FILE xxxxxxxx IS INVALID - CONTAINS NO TERMINATION RECORD.

WCS FILE xxxxxxxx IS INVALID - CONTAINS AN ERRONEOUS BYTE
COUNT. OFFENDING RECORD BYTE COUNT: xx
FIRMWARE ADDRESS: xxxx
WCS FILE xxxxxxxx IS TOO LARGE FOR DEVINITL.
OFFENDING RECORD BYTE COUNT: xx
FIRMWARE ADDRESS: xxxx
```

Including microcode and data, maximum file size is 64K.

```
WCS FILE xxxxxxxx CONTAINS INVALID COMMAND: HEX xx

HARDWARE ERROR STATUS RECEIVED. STATUS AFTER LAST RETRY:
HEX xxxx

UNABLE TO OPEN DEVICE xxxxxx DEVICE MAY NOT BE PRESENT
```

11.7.3 Directive Parsing Errors

The following message is displayed when an invalid pathname is specified with the WCS_FILE directive. The string xxxxxxxx contains up to 80 characters.

```
** ERROR ** PATHNAME SYNTAX ERROR: xxxxxxxx
```

The following message is displayed when OPTION WCSMATCH=Y and an IDENT directive have been specified, but a match is not found in the header record of the file being searched.

```
** WARNING ** INCORRECT FIRMWARE REVISION FOR DEVICE xxxxxx
        REQUESTED xxxxxxxxxxxxxxxx
        ACTUAL    xxxxxxxxxxxxxxxx
```

The following message is displayed when the Execute Channel Program call fails after the specified number of RETRY and REREAD attempts. The number output as the reason is the contents of the third word of the FCB.

```
DEVICE xxxxxx INITIALIZATION FAILED.
        REASON xxxxxxxx
```

The following message is displayed when a nonhexadecimal digit is found during IOCD creation at initialization time. The initialization line is ignored.

```
** ERROR ** INVALID HEXADECIMAL CHARACTER: x
```

The following message is displayed when the two least significant bits of the IOCD operation code directive specified with DEV_CNTRL CMD=xx are not set.

```
** ERROR ** xx IS INVALID IOCL COMMAND
```

When a parse error occurs for a reason other than those described above, the following message is displayed along with the appropriate reason. Below this, the erroneous line is displayed with an exclamation point (!) below the item that caused the parse error.

```
** PARSE ERROR **
```

AN EQUAL SIGN IS MISSING FOLLOWING A KEYWORD

PREMATURE END OF INPUT STRING

> A directive is truncated.

DEVICE IDENTIFICATION IS MISSING

UNRECOGNIZABLE DIRECTIVE

A STRING IS MISSING FOLLOWING AN EQUAL SIGN

A QUOTED STRING IS IMPROPERLY TERMINATED

> An end quote is missing or attempt was made to carry a string past one input line.

AN ASSUMED OPTION LINE CONTAINS NO OPTION DIRECTIVES

> Parameter WCSMATCH and/or MILESTONE must be specified with an OPTION command.

A PREMATURE END-OF-STRING OCCURRED IN OPTION SPECIFICATION

> An OPTION directive cannot be carried past one input line.

THE SPECIFIED OPTION IS UNDEFINED

RETRY/REREAD IS NOT A POSITIVE DECIMAL

INVALID OPTION SENSE SPECIFIED OPTIONS=Y,N,T,F, OR BLANK

CHAPTER 12

ALTERABLE CONTROL STORE (ACS) LOAD AND DISPLAY UTILITY

The Alterable Control Store (ACS) Load and Display Utility performs two functions--LOADACS and DUMPACS. LOADACS writes firmware to the ACS or Writable Control Store (WCS). DUMPACS displays or compares firmware in Programmable Read Only Memory (PROM), ACS, and WCS. Through the SETCPU instruction, MPX-32 specifies whether the processor uses the PROM or ACS firmware.

With the exception of the CHECKSUM directive, the ACS Utility runs only on a CONCEPT 32/67 computer. The CHECKSUM directive can be used on any CONCEPT/32.

The numbers accepted and generated by LOADACS and DUMPACS are hexadecimal, except for decimal RMxx error codes.

12.1 LOADACS Directive File (M.ACS)

The LOADACS directives are in a file named M.ACS. M.ACS must be located on the system volume and the system directory. Use the Text Editor to create the file.

M.ACS consists of one or more groups of directives, each group for a particular processor. There can be more than one group of directives for any processor. The last directive in each group must be followed by a semicolon. See Figure 12-1 for the format of M.ACS.

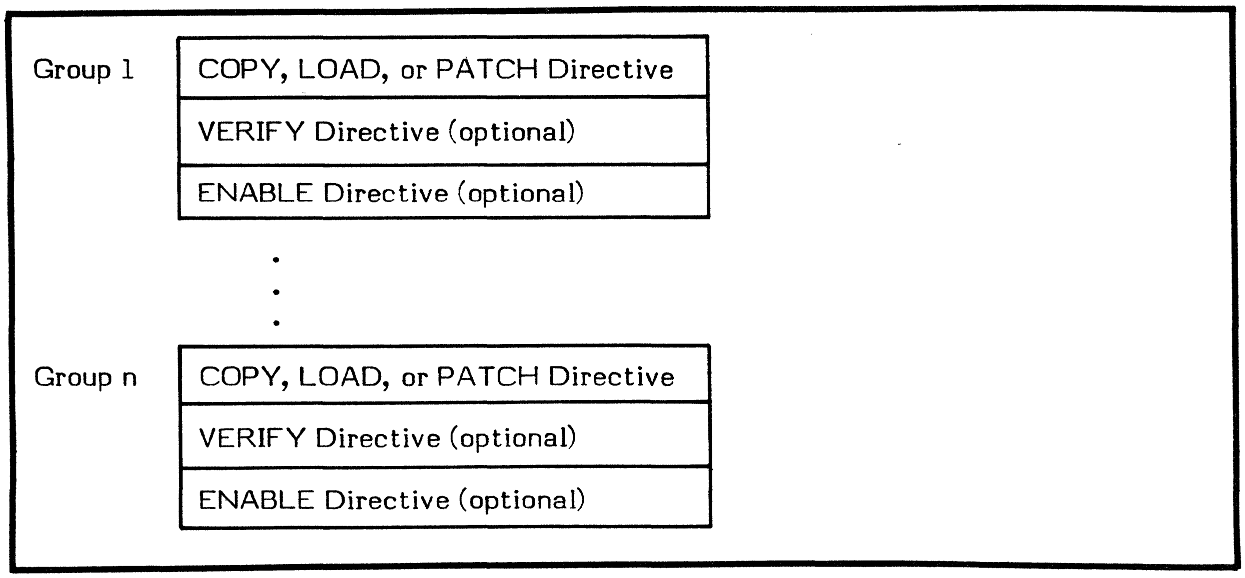

Group 1
COPY, LOAD, or PATCH Directive
VERIFY Directive (optional)
ENABLE Directive (optional)
Group n
COPY, LOAD, or PATCH Directive
VERIFY Directive (optional)
ENABLE Directive (optional)

Figure 12-1. M.ACS Format

12.2 Firmware File

The firmware file supplied must be restored as a system file using the Volume Manager.

The firmware file contains logical blocks of 256 bytes. The last byte of each logical block is the ones complement of the sum of all other bytes in the logical block. LOADACS uses this figure as a checksum during a LOAD.

If assigned, the firmware file must be assigned as unblocked.

12.2.1 CONCEPT 32/67 Usage

The CONCEPT 32/67 has two phases of processors, phase 1 and phase 2. The user must know which phase their system has so the M.ACS file can be properly constructed.

A properly constructed M.ACS directive loads or patches the proper firmware without user intervention regardless of the processor's phase. This maintains system portability from one phase processor to another.

The processor's phase is determined by the processor (CPU or IPU) PROM revision level. See the REVISION directive. If the last eight digits of the revision number are 2282000n, where n is any number, the processor is a phase 1.

Any other numbers indicate a phase 2 processor.

12.2.2 Firmware File Record

The maximum record length is one logical block; therefore, the maximum record size is 251 bytes. The maximum number of 64-bit microwords per record is 31.

The format of a record is:

0 7	8 23	24 n
Microword Count (Byte 0). See Note 1.	Start Address (Bytes 1 & 2). See Note 2.	Data (Bytes 3-250). See Note 3.

Notes:

1. Byte zero of a record contains the microword count for the record. A count of 00 means the record is the last in a logical block.

   If the last record of a logical block ends in the last three bytes of the block, a new record is not started with a 00 microword count.

   An FF microword count indicates an end-of-file.

2. Bytes one and two of a record contain the starting address of the record's data. Valid address ranges are:

   0000 to 0FFF for ACS or PROM
   1000 to 1FFF for 4K x 64 WCS
   1000 to 2FFF for 8K x 64 WCS

3. Bytes 3-250 of a record contain the data to be loaded.

Microword FFD always contains the ACS or PROM firmware revision. The revision's format is:

C67rnzzppppppzzzn

C	=	CONCEPT/32
67	=	two digit type of CONCEPT/32
r	=	firmware revision letter
n	=	firmware revision number
z	=	zero
p	=	six digit firmware part number

C67rn, with the variables filled, is the firmware source filename.

The following is an example of an ACS/PROM firmware revision.

C67A300322820003

The example's firmware source filename is C67A3. The firmware part number is 322820.

Microword FFE always contains the ACS or PROM data checksum. The checksum is the ones complement of the sum of the microwords. It is calculated by LOADACS during writes to ACS.

12.3 LOADACS

LOADACS is a subroutine in J.INIT that is executed during IPL and RESTART sequences. LOADACS receives commands from the M.ACS command file. If any of the following conditions are true, the firmware is not loaded and the load, patch, or copy message is not displayed:

. Control switch nine is set
. The machine is not a CONCEPT 32/67
. The sequence is not an IPL or RESTART
. M.ACS does not exist
. M.ACS is empty

LOADACS accepts different firmware and actions for the CPU and IPU. If the IPU is off-line, a load request causes an error message to be displayed.

LOADACS runs in the processor it is loading. Before loading ACS or WCS, LOADACS executes the SETCPU instruction to enable transfer in the PROM mode. LOADACS blocks external interrupts in the CPU before loading, and unblocks them afterward. In the IPU, LOADACS inhibits context switching.

To load new firmware in ACS or WCS, use the LOAD directive. To load PROM firmware with modifications, use the COPY directive followed by a LOAD or PATCH directive. To modify ACS or WCS, use a LOAD or PATCH directive. PROM cannot be loaded or modified.

All LOADACS output is displayed on the operator console and the LOD device. After a successful load, the following message is displayed:

```
        (LOADED  )
J.INIT:{PATCHED } {ACS}  FOR  {CPU}  FROM  {filename}
        (COPIED  ) {WCS}       {IPU}        {PROM    }

        REVISION = revnumber

     [ {CPU} IN {ACS }   MODE ]
     [ {IPU}    {PROM}        ]
```

The optional section (in brackets) is displayed for the ACS mode only.

12.4 DUMPACS

DUMPACS is a subroutine in J.INIT. When J.INIT is called from TSM and option 1 is set, DUMPACS is the only subroutine called. DUMPACS then displays a DMP prompt for directives.

```
TSM> OPTION 1
TSM> J.INIT
DMP>
```

Depending on the directive, the results are written to logical file code UT or the LOD device. These logical file codes cannot be reassigned.

DUMPACS dumps the contents of ACS, WCS, and PROM, compare the contents of these storage areas with a firmware file, and calculate checksums to be used by the LOADACS PATCH directive. DUMPACS also displays the revision microword and the mode of a processor.

12.5 ACS Directives

ACS directives are summarized below, arranged by function type. Each directive is described in detail in the remainder of this section, arranged alphabetically by directive name. Valid abbreviations are shown by underlining.

LOADACS Directives

Directive	Description
COPY	Copies contents of PROM to ACS
ENABLE	Sets processor in ACS or PROM mode. This directive is not valid for WCS.
LOAD	Loads firmware file into ACS or WCS
PATCH	Loads data from M.ACS to ACS or WCS
VERIFY	Compares ACS or WCS with the data that was written to ACS or WCS

DUMPACS Commands

Directive	Description
CHECKSUM	Provides the checksum used by the PATCH directive
COMPARE	Compares ACS, WCS, or PROM with a firmware file
DUMP	Dumps specified memory contents to the LOD device
EXIT	Exits DUMPACS and returns to TSM
MODE	Displays the mode of a processor
REVISION	Displays revision microword on LFC UT

12.5.1 CHECKSUM Directive

The CHECKSUM directive is used by DUMPACS to add values and compute the ones complement of the sum. The sum is then displayed on LFC UT.

When using LOADACS, the sum is the value entered in the optional SUM parameter for the PATCH directive.

The CHECKSUM directive can be continued across more than one input line by placing a hyphen (-) as the last significant character on the line to be continued. The hyphen must not be embedded in a value.

Syntax:

```
    CHECKSUM value [, value. . .]
```

value	is a value to be patched by LOADACS

Example

```
CHECKSUM 2433F55EC0024, 99FA5500, BB4302
```

This adds three values and computes the ones complement of the sum. The resulting value can be in the PATCH directive for the SUM parameter.

12.5.2 COMPARE Directive

The COMPARE directive is used by DUMPACS to compare the contents of the specified ACS, WCS, or PROM with a firmware file. When the firmware file and memory do not match, the memory location's address and contents plus the file's contents are written to the LOD device.

The firmware file must be formatted as described in Section 12.2.

Syntax:

$$\text{COMPARE} \begin{Bmatrix} \text{CPU} \\ \text{IPU} \end{Bmatrix} \begin{Bmatrix} \text{PROM} \\ \text{ACS} \\ \text{WCS} \end{Bmatrix} \text{pathname}$$

CPU	specifies the CPU is the processor that is used
IPU	specifies the IPU is the processor that is used
PROM	specifies the PROM is read
ACS	specifies the ACS is read
WCS	specifies the WCS is read
pathname	is the pathname of the firmware file. If the file does not exist, an error message is displayed.

Examples

COMPARE CPU ACS @SYSTEM(SYSTEM)FIRM

This compares the contents of the CPU's ACS to the system file named FIRM.

COMPARE IPU WCS FIRM

This compares the contents of the IPU's WCS to the user file named FIRM.

12.5.3 COPY Directive

The COPY directive is used by LOADACS to copy the contents of PROM to the CPU or IPU ACS.

Syntax:

$$\text{COPY} \begin{Bmatrix} \text{CPU} \\ \text{IPU} \end{Bmatrix}$$

CPU	specifies the CPU ACS is written
IPU	specifies the IPU ACS is written

12.5.4 DUMP Directive

The DUMP directive is used by DUMPACS to write the specified memory contents to the LOD device. If starting and ending addresses are not specified, the entire contents of the specified memory are written.

If an invalid address is specified, an error message is displayed on the terminal.

Syntax:

$$\text{DUMP} \begin{Bmatrix} \text{CPU} \\ \text{IPU} \end{Bmatrix} \begin{Bmatrix} \text{PROM} \\ \text{ACS} \\ \text{WCS} \end{Bmatrix} \text{[startaddr] [,endaddr]}$$

CPU	specifies the CPU is the processor that is used
IPU	specifies the IPU is the processor that is used
PROM	specifies the PROM is read
ACS	specifies the ACS is read
WCS	specifies the WCS is read
startaddr	is the starting address of the dump
endaddr	is the ending address of the dump

Example

```
DUMP CPU ACS
```

This writes the entire contents of the CPU's ACS to the LOD device.

12.5.5 ENABLE Directive

The ENABLE directive is used by LOADACS to set the processor which was just used for a COPY, LOAD, or PATCH, in the ACS or PROM mode. This directive is not valid for the WCS mode.

If the ACS mode is requested and a previous LOAD, COPY, PATCH, or VERIFY directive failed in ACS, the mode is not changed and an error message is written.

If the ENABLE operation is successful, the mode of the processor is reported to the console and the LOD device. When an ENABLE of IPU ACS causes the IPU to marked off-line due to incompatible firmware, the message IPU MARKED OFFLINE is displayed instead of the mode.

Syntax:

```
    ENABLE  { ACS  }
            { PROM }
```

ACS	specifies the processor uses the ACS firmware
PROM	specifies the processor uses the PROM firmware

12.5.6 EXIT Directive

The EXIT directive exits DUMPACS and returns control to TSM.

Syntax:

```
    EXIT
```

12.5.7 LOAD Directive

The LOAD directive is used by LOADACS to load a firmware file into ACS or WCS. If an ACS load is not successful, the processor used remains in the PROM mode.

The LOAD directive can be continued across more than one line by placing a hyphen (-) as the last significant character on the line to be continued. The hyphen must not be embedded in a value and must directly precede the ACSMATCH or WCSMATCH keyword.

Syntax:

```
LOAD {CPU} {ACS} FILE=p1file [,p2file] [{ACSMATCH}  =p1rev [p2rev]]
     {IPU} {WCS}                       [{WCSMATCH}               ]
```

CPU	specifies the CPU is the processor that is used
IPU	specifies the IPU is the processor that is used
p1file	specifies the 1- to 16-character name of the file containing the firmware loaded for a phase one processor. If a p2file is not specified, p1file is loaded regardless of the processor type.

p2file specifies the 1- to 16-character name of the file containing the firmware that is loaded for a phase two processor. When specified, either p1file or p2file is loaded depending on the processor's phase:

p1file	p2file	Processor phase	File loaded
Present	Not present	1	p1file
Present	Not present	2	p1file
Present	Present	1	p1file
Present	Present	2	p2file

ACS specifies the firmware is loaded into ACS

WCS specifies the firmware is loaded into WCS

ACSMATCH compares the current ACS revision number to the number specified by revnum

WCSMATCH compares the current WCS revision number to the number specified by revnum

p1rev is a 16-digit revision number. If p1rev and the current revision number do not match, a load does not take place and an error message is written to the LOD device. If a match occurs, the firmware is loaded regardless of the processor type.

p2rev is a 16-digit revision number. When specified, the firmware is loaded conditionally depending on the processor's phase:

p1rev	p2rev	Processor phase	Action
Present	Present	1	Revision number compared to p1rev. If matched, firmware is loaded. If not matched, no load occurs.
Present	Present	2	Revision number compared to p2rev. If matched, firmware is loaded. If not matched, no load occurs.
Present	Not present	1	Revision number compared to p1rev. If matched, firmware is loaded. If not matched, no load occurs.
Present	Not present	2	Revision number compared to p1rev. If matched, firmware is loaded. If not matched, no load occurs.

Examples

```
LOAD CPU ACS FILE=FIRM ACSMATCH=C67A303228200003
```

This compares the ACSMATCH revision number to the current ACS revision number. If the numbers match, the contents of file FIRM are loaded into the CPU's ACS.

```
LOAD IPU WCS FILE=FIRM
```

This loads the contents of file FIRM into the IPU's WCS.

12.5.8 MODE Directive

The MODE directive is used by DUMPACS to display the mode of a processor on the device assigned to LFC UT.

Syntax:

```
      { CPU }
MODE  { IPU }
```

CPU specifies the mode of the CPU is displayed

IPU specifies the mode of the IPU is displayed

12.5.9 PATCH Directive

The PATCH directive is used by LOADACS to modify the contents of ACS or WCS.

The PATCH directive can be continued across more than one input line by placing a hyphen (-) as the last significant character on the line to be continued. The hyphen must not be embedded in a value and must fall after the first value and before the SUM parameter.

Syntax:

```
PATCH  {CPU} [P1] {ACS}  LOC=addr VALUE=val[,val. . .][SUM=sum]
       {IPU} [P2] {WCS}
```

CPU	specifies the CPU is the processor that is used
IPU	specifies the IPU is the processor that is used
P1	specifies a phase one 32/67 processor
P2	specifies a phase two 32/67 processor
ACS	specifies the ACS is patched
WCS	specifies the WCS is patched
addr	is the address in ACS or WCS loaded with the contents of M.ACS
val	is a value to be patched
sum	is a number to be compared with the ones complement of the sum of the values. If sum does not equal the ones complement of the sum of the values, an error message is written.

If the specified phase and the actual phase mismatch, a warning message is sent to the LOD and the rest of the command group is skipped. Failure to conditionally patch does not disable ACS.

Examples

```
PATCH CPU ACS LOC=E30 VALUE=4F55D234, 3534AD24-
,4536 SUM=FFFFFFFFB0753B3C
```

The ones complement of the sum of the values is compared to the number in the SUM parameter. If the numbers match, the values are patched into the CPU's ACS.

Note: The hyphen and its position after the last digit in the third value. The comma separating the third and fourth values begins the new line.

```
PATCH IPU WCS LOC=1E30 VALUE=35F82
```

This patches location 1E30 in the IPU's WCS with the value 35F82.

```
PATCH CPU P1 ACS LOC=E30 VALUE=FEE, F1EF0EF00, FACE, FADE
  VERIFY;

PATCH CPU P2 ACS LOC=E3E VALUE=0,0, C0FFEE, D10DE
  VERIFY;

PATCH CPU ACS LOC=E29 VALUE=DEADDEAD, DEADDEAD
  VERIFY;

ENABLE ACS;
```

Directive groups one and two are conditional patches depending on the 32/67 processor's phase. Directive group three is unconditionally patched and directive group four passes control to ACS.

12.5.10 REVISION Directive

The REVISION directive is used by DUMPACS to display the revision microword on the device assigned to LFC UT.

Syntax:

$$\text{REVISION} \begin{Bmatrix} \text{CPU} \\ \text{IPU} \end{Bmatrix} \begin{Bmatrix} \text{PROM} \\ \text{ACS} \\ \text{WCS} \end{Bmatrix}$$

CPU	specifies the CPU is the processor that is used
IPU	specifies the IPU is the processor that is used
PROM	specifies the PROM is read
ACS	specifies the ACS is read
WCS	specifies the WCS is read

12.5.11 VERIFY Directive

The VERIFY directive is used by LOADACS to compare the contents of ACS or WCS with an internal buffer. The buffer contains the data that was written to ACS or WCS as the result of the previous directive.

If the verification fails, an error message is written.

Syntax:

```
VERIFY
```

12.6 Sample M.ACS File

The following is a sample of an M.ACS file.

```
COPY CPU;                                                   (Group 1)
PATCH CPU ACS LOC=E30 VALUE=4F55D234,35,34AD24-             (Group 2)
,4536 SUM=FFFFFFFFB0753B3C
VERIFY
ENABLE ACS;
LOAD IPU ACS FILE=IPUACS1 ACSMATCH=C67A300322820003         (Group 3)
ENABLE ACS;
```

When the sample file is executed, the phase one firmware in the CPU PROM is copied to the CPU ACS with the COPY directive. The semicolon at the end of the directive signals the end of a group in the M.ACS file.

The PATCH directive changes the contents of CPU ACS locations E30 to E33 to the values 4F55D234, 35, 34AD24, and 4536. The ones complement of the sum of these numbers is compared to FFFFFFFFB0753B3C.

The VERIFY directive compares the CPU ACS with the contents in the internal buffer. The buffer contains the M.ACS and ACS contents used in the PATCH directive.

The ENABLE directive sets the CPU to the ACS mode. The semicolon at the end of the command signals the end of the second group of directives.

The LOAD directive reads the PROM revision number. If the actual number does not match the revision number specified, LOADACS goes to the next group of directives. If the numbers match, the phase one firmware in system file IPUACS1 is loaded into the IPU ACS.

The last ENABLE directive sets the IPU to the ACS mode if the LOAD was successful. The semicolon at the end of the directive signals the end of the third group of directives.

12.7 Error Conditions and Messages

12.7.1 LOADACS Error Conditions and Messages

The following LOADACS error messages are written to the SLO device and the operator console.

```
                  { COPY  }
J.INIT: CANNOT    { LOAD  }  IPU - IPU NOT ONLINE
                  { PATCH }
```

This message displays when the IPU is not on-line or SYSGENed in and cannot be loaded. The CPU can be loaded if necessary.

J.INIT: UNRECOGNIZED COMMAND IN M.ACS, IGNORED:
directive

This message displays when there is an unrecognizable directive in M.ACS. The directive is displayed and LOADACS goes to the next line of M.ACS.

```
J.INIT:     CANNOT {LOAD  } {ACS}  FOR {CPU} -  INVALID ADDRESS
                   {PATCH } {WCS}      {IPU}

                 address IN filename
```

This message displays when firmware cannot be loaded because of an invalid address in a firmware file. The address and the file name containing the address are displayed. LOADACS goes to the next group of directives.

```
J.INIT:   CANNOT {LOAD } ACS FOR {CPU}-  NEW FIRMWARE HAS M/C TYPE type
                 {PATCH}         {IPU}
```

This message displays when the machine type in the firmware's revision number is not C67. The machine type of the new firmware is displayed. LOADACS goes to the next group of directives.

```
J.INIT:     CANNOT LOAD {ACS} FOR   {CPU}-  PRESENT REV NUMBER NOT AS
                        {WCS}       {IPU}

EXPECTED FROM     {ACSMATCH}COMMAND
                  {WCSMATCH}

                    PRESENT    =  revnumber
                    EXPECTED   =  revnumber
```

This message displays when the ACSMATCH or WCSMATCH directive is in effect and the revision number of the current firmware and the number specified by ACSMATCH or WCSMATCH do not match. The current firmware's revision number displays as PRESENT and the ACSMATCH or WCSMATCH revision number displays as EXPECTED. LOADACS goes to the next group of directives.

```
                        {LOAD }{ACS}      {CPU}                                {filename}
J.INIT: CANNOT          {PATCH}{WCS} FOR  {IPU}- CHECK SUM ERROR IN           {PROM    }
                        {COPY }
```

This message displays when a checksum error occurs during a LOAD, PATCH, or COPY. LOADACS goes to the next group of directives.

```
        {COPY }      {CPU}       {ACS}
J.INIT: {LOAD } TO   {IPU} FOR   {WCS} FAILED VERIFICATION,
        {PATCH}

ADDRESS addr DIFFERS        {PROM    }   = contents
                            {filename}

                            {ACS}        = contents
                            {WCS}
```

This message displays when a verification fails. The address of the failure is displayed, plus the contents of the differing firmware. LOADACS goes to the next group of directives.

```
J.INIT:     CANNOT LOAD   {ACS}  FOR  {CPU} - UNABLE TO ASSIGN filename
                          {WCS}       {IPU}
            ERROR RMxx
```

This message displays when the firmware file cannot be assigned. A decimal error number (RMxx) is returned by H.REMM. LOADACS goes to the next group of directives.

```
J.INIT:     CANNOT  {LOAD } {ACS}  FOR  {CPU} - INVALID VALUE
                    {PATCH} {WCS}       {IPU}

            value IN filename AT ADDRESS addr
```

This message displays when an invalid value is found in a firmware file. The file's name, the invalid value, and the address of the value are displayed. LOADACS goes to the next group of directives.

J.INIT: ENABLE COMMAND IS INVALID WHEN WCS ACTION IS SPECIFIED

This message displays when an ENABLE directive is used in a group of WCS directives. The ENABLE directive is valid for ACS only. Remove the ENABLE directive.

```
J.INIT:     CANNOT {LOAD }{CPU WCS}-  NO WCS PRESENT
                   {PATCH}{IPU    }
```

This message displays when there is no WCS to load. LOADACS goes to the next group of directives.

```
J.INIT:        CANNOT ENABLE {CPU}ACS  -  PREVIOUS ACTION TO THIS PROCESSOR FAILED
                             {IPU}
```

This message displays if a previous action to a processor failed. J.INIT cannot enable the ACS since that firmware has not been properly loaded. LOADACS goes to the next group of directives.

```
                   {COPY } {CPU}       {CPU}    {CPU}
J.INIT:     CANNOT {LOAD } {ACS} FOR   {IPU} -  {IPU}  IS IN ROMS IM MODE
                   {PATCH}
```

This message displays if the processor is in the ROMSIM mode, i.e., writes to ACS result in a System Check Trap. LOADACS goes to the next group of directives.

12.7.2 DUMPACS Error Conditions and Messages

The following DUMPACS error messages are written to the device assigned to LFC UT.

INVALID COMMAND

This message displays when the last directive entered is invalid. Enter a valid directive at the DMP prompt.

INVALID PROCESSOR MNEMONIC

This message displays when the last directive entered contains an invalid processor mnemonic. The valid mnemonics are CPU and IPU.

INVALID MEMORY TYPE

This message displays when the last directive entered contains an invalid memory type. Valid memory types are PROM, ACS, and WCS.

INVALID START ADDRESS

This message displays when the starting address specified in the DUMP directive is invalid. Reenter the directive with a new starting address.

INVALID END ADDRESS

This message displays when the ending address specified in the DUMP directive is invalid. Reenter the directive with a new ending address.

THIS MACHINE IS NOT A 32/67 - CAN NOT DO IT

This message displays when the directives can be used only on a 32/67 machine.

IPU NOT CONFIGURED IN SYSTEM

This message displays when the IPU was not configured in the system at SYSGEN time.

IPU MARKED OFFLINE

This message displays when the IPU is off-line. IPU directives are not executed until the IPU is marked on-line.

CAN NOT OPEN SLO

This message displays when DUMPACS cannot open LFC SLO to write required data. If the SLO default is used in this case, a system problem exists.

BAD ADDRESS addr FOUND IN FIRMWARE FILE

This message displays when an incorrect address is found in the firmware file. Correct and reassemble the firmware file, then reenter the directives.

UNABLE TO ASSIGN FILE - ERROR RMxx

This message displays when the firmware file cannot be assigned. A decimal error number (RMxx) is returned by H.REMM.

INVALID PARAMETER

This message displays when a parameter in the last directive is invalid. Correct the parameter and reenter the directive.

THERE IS NO WCS ON THIS PROCESSOR

This message displays when the processor used in the directive does not have WCS.

CHAPTER 13

VOLUME FORMATTER (J.VFMT)

The Volume Formatter (J.VFMT) formats volumes (discs). Volumes formatted by J.VFMT have the following data structures:

Volume descriptor
Resource descriptor allocation bit map (DMAP)
Descriptor for DMAP
File space bit map (SMAP)
Descriptor for SMAP
Descriptor for system image
General allocatable resource descriptors
Descriptor for allocatable resource descriptor
Root directory
Root directory descriptor
Deallocation file for DMAP (F-class only)
Descriptor for DMAP deallocation file
Deallocation file for SMAP (F-class only)
Descriptor for SMAP deallocation file
Media descriptor file (F-class only)
Descriptor for media descriptor file

13.1 General Description

J.VFMT operates in two distinct environments: a fully functional MPX-32 system and a starter system by the System Distribution Tape (SDT) when J.VFMT is activated from tape by SYSINIT.

When IPL is performed from the SDT, it is assumed file system support is not present in the form of a usable disc volume. To format a system volume, SYSINIT makes a static assignment indicating the system image is coming from tape, and a system image file is read from the SDT. Therefore, the IMAGE option for the FORMAT directive in J.VFMT must not be specified.

13.2 Logical File Code Assignments

Logical file codes required by J.VFMT are described in this section and summarized in Table 13-1.

13.2.1 Audit Trail (SLO)

As J.VFMT processes directives, it produces listed output for operations it performs. The LFC for the audit trail is SLO.

13.2.2 Directive Input (SYC)

Directive input is assigned to J.VFMT from logical file code SYC.

Table 13-1
Volume Formatter Logical File Code Assignments

Input/Output Description	**Logical File Code**	**Default Assignments for J.VFMT**
Audit Trail	SLO	$ASSIGN SLO TO LFC=UT
Directive Input	SYC	$ASSIGN SYC TO SYC

13.3 Using the Volume Formatter

The Volume Formatter formats volumes not currently mounted. Attempting to run J.VFMT on a mounted volume causes an abort.

Only one J.VFMT directive verb can be processed for each FMT prompt. Upon completion of the requested function, J.VFMT automatically exits and returns to the calling task. For example,

```
TSM>$J.VFMT
FMT>FORMAT DEV=DM0802 VOL=EXAM MAXRE=3000
     DEVICE=DM0802 -- VOLUME FORMATTING SUCCESSFULLY COMPLETED
TSM>
```

When the J.VFMT function is performed during an SDT boot, the next prompt displayed is:

```
FMT> FORMAT DEV=DM0800 VOL=EXAM MAXRE=3000
     DEVICE=DM0800 -- VOLUME FORMATTING SUCCESSFULLY COMPLETED
ENTER SYSTEM VOLUME CHANNEL AND SUBADDRESS:
```

When run in the batch mode, parameters must be specified in the first input string on the directive line.

In the interactive mode, prompts for more data after the first input string occur when:

- required parameters are not specified
- there is a syntax error in the directive line
- confirmation when the Boolean flag is turned on

When syntax errors occur in the interactive mode, the part of the directive line in error and an error message are displayed. The remainder of the directive line is ignored. However, directives and parameters in the directive line before the syntax error occurred will be processed and must not be respecified.

To recover from certain errors in the interactive mode, J.VFMT requires specific information obtained by error message output. In some cases, additional input is not accepted until requested input is supplied.

13.3.1 Directive Syntax Rules

The directive line has three parts:

. Verb
. Parameters
. Options

Blank spaces are legal delimiters between directive line components.

13.3.1.1 Verb

The verb is required as the first item on the directive line. The verb defines the action the utility is to take, (FORMAT, REPLACE, EXIT). The verb also determines the subset of options that are accepted as valid input.

13.3.1.2 Parameters

Parameters are required keywords followed by an equal sign and a value. They are entered on the directive line after the verb. Parameters specify what is affected by the action of the directive. Parameters have three types of values:

. Boolean - Values are true (T), false (F), yes (Y), no (N), on (ON), and off (OFF). Values true, yes, and on are synonymous. Values false, no, and off are synonymous.

. Numeric - Values are either decimal or hexadecimal numbers whichever is natural for that parameter. For example, radix can be stated using the radix indicators X and N, e.g., X'AB', N'18'.

. ASCII string - Values are expressed as strings containing alphanumeric characters.

13.3.1.3 Options

Options are keywords followed by an equal sign and a value. If specified, options are entered on the directive line after the parameters. Options are used as qualifiers, ACCESS=, CONFIRM=, IMAGE=. Where applicable, default values are provided. See individual directive descriptions for defaults.

13.3.2 Directive Line Continuations

Directive lines can be continued across as many lines as necessary by placing a hyphen (-) as the last significant character on the line. This results in a prompt for more input when in the interactive mode, and another read when in batch mode or when reading a select file. Each time a directive line is continued, the continuation character is replaced by a space. The only restriction is that single names, keywords, or an entire pathname must appear on the same line of input.

13.4 Accessing the Volume Formatter

The Volume Formatter can be accessed from batch or TSM in one of three ways:

```
$J.VFMT
$RUN J.VFMT
$EXECUTE J.VFMT
```

$RUN J.VFMT is valid from the system directory only.

When logical file code SYC is assigned to a terminal, the FMT> prompt is displayed.

13.5 Volume Formatter Directive Verbs

The following directive verbs are supported by the Volume Formatter. Valid abbreviations are indicated by underlining.

Directive	Description
EXIT	Terminates the Volume Formatter utility
FORMAT	Software formats a disc assuming an unformatted medium
NEWBOOT	Writes a new bootstrap process to an already formatted volume
REPLACE	Writes a new system image and bootstrap to an already formatted volume

13.5.1 EXIT Directive

The EXIT directive exits the Volume Formatter and returns control to TSM.

Syntax:

```
EXIT
```

13.5.2 FORMAT Directive

The FORMAT directive software formats an entire disc, assuming an unformatted medium. No checks are made for any existing data on the disc. The disc pack must already be hardware initialized before J.VFMT is run.

For all class F discs except floppy discs and memory discs, the diagnostic Media Verification Program must be run before J.VFMT. This marks bad blocks and reserved areas of the disc as allocated so they are not used.

Syntax:

```
FORMAT DEVICE=mmccss VOLUME=volname [ALOC=n]

[CONFIRM=bool] [DETACH=bool] [IMAGE=pathname] [MAXAU=n]

[MAXRES=n] [MAXROOT=n] [OWNER=name] [PROJECTGROUP=name]

[ACCESS= {OWNER       }([READ] [ADD] [DELENT] [TRAVERSE])] [ISIZE=N]
[        {PROJECTGROUP}                                 ]
[        {OTHER       }                                 ]

[BOOTFILE=pathname] [ECC=bool]
```

DEVICE=	specifies the disc device to be formatted mm a two character mnemonic: DM - moving head or memory disc FL - floppy disc DF - fixed head disc cc a two digit hexadecimal channel number ss a two digit hexadecimal subaddress
VOLUME=	specifies the name of the volume being formatted
ALOC=	specifies the decimal number of sectors per allocation unit. If not specified, a system default based upon the capacity of the disc is used. For information on allocation units, refer to Table 13.2.
CONFIRM=	if Y is specified, parameters are displayed before any action is taken and the J.VFMT prompt is displayed. Once all parameters are specified, this option must be reset to false (by specifying CON=N at the J.VFMT prompt). The formatting process then continues. This action is valid only in the interactive mode.
DETACH=	if Y is specified, parameters are displayed before any action is taken and J.VFMT is detached from the terminal and control of the terminal returns to TSM. Formatting then proceeds normally. When the formatting process completes, a message is displayed on the terminal if the user is logged on the system. This option is valid only in the interactive mode.
IMAGE=	specifies the pathname of the file containing the system image. If not specified and J.VFMT is activated from a disc volume, the volume is not bootable. This option must not be specified when J.VFMT is activated from a SDT.

MAXAU=	specifies the decimal number of allocation units to be used on the disc. The number depends on the parameter specified for the ALOC option of the FORMAT command, or the default granularity if ALOC is not specified. The value of MAXAU can be less than the actual number of allocation units physically available.
MAXRES=	specifies the decimal number of resource descriptors for which to reserve disc space. Every file defined on the volume requires one resource descriptor. If not specified, the default for memory discs or floppy discs is 70. All other devices default to 1000.
MAXROOT=	specifies the decimal number of entries in the root directory for which to reserve disc space. This number can be modified to accommodate various hashing algorithms. If not specified, the default is 100.
OWNER=	specifies the owner name to be associated with the volume. The owner name is recorded in the volume descriptor. If not specified, defaults to the owner name associated with the J.VFMT task.
PROJECTGROUP=	specifies the project group name to be associated with the volume. The project group name is recorded in the volume descriptor. If not specified, defaults to the project group name associated with the J.VFMT task.
ACCESS=	specifies the access attributes for the root directory being formatted. If not specified, the system-defined defaults are used. Contiguous open and close parentheses, such as ACCESS = ( ), indicates no access possible.
ISIZE=	specifies the number of blocks allocated for the system image. If not specified, the default is 400.
BOOTFILE=	specifies the pathname of the file containing the nonstandard bootstrap. The nonstandard bootstrap should be in absolute load module format. It cannot be larger than seven 192-word blocks in length including the load module preamble.
ECC=	allocates correctable data errors for user's use.

Table 13-2
Allocating Units

Unit Size and Type	Sectors per Allocation Unit
4MB Fixed Head	1
40MB Moving Head	2
80MB Moving Head	2
300MB Moving Head	4
Extended I/O Devices:	
Floppy Moving Head	1
Memory Disc	1
5MB Fixed Head	1
32MB Cartridge Module	2
40MB Moving Head	2
64MB Cartridge Module	2
80MB Moving Head	2
96MB Cartridge Module	2
160MB Moving Head	4
300MB Moving Head	4
340MB Moving Head	4
600MB Moving Head	10

13.5.3 NEWBOOT Directive

The NEWBOOT directive writes a nonstandard bootstrap process to an already formatted volume. For the volume to be bootable, it must have been formatted with the IMAGE option specified on the FORMAT directive line. See the FORMAT directive.

Syntax:

NEWBOOT DEVICE=mmccss VOLUME=volname BOOTFILE=pathname

[CONFIRM=bool]

DEVICE= specifies the two-character mnemonic (DM, FL, or DF), the two-digit hexadecimal channel number, and the two-digit hexadecimal subaddress.

VOLUME= specifies the name of the volume

BOOTFILE= specifies the pathname of the file containing the nonstandard bootstrap. The nonstandard bootstrap should be in absolute load module format. It cannot be larger than seven 192-word blocks in length including the load module preamble.

CONFIRM= specifies if confirmation is to be given before any formatting action is taken. If Y is specified, operator response is required to the confirmation prompt. If not specified, confirmation is not given. This option is valid only in the interactive mode.

13.5.4 REPLACE Directive

The REPLACE directive writes a new system image and standard bootstrap to an already formatted volume. This directive is used to modify data in the volume descriptor concerning the system image. The volume can already contain a system image.

Syntax:

```
REPLACE DEVICE=mmccss VOLUME=volname [CONFIRM=bool] [IMAGE=pathname]
[BOOTFILE=pathname]
```

DEVICE=	specifies the two-character mnemonic (DM, FL, or DF), the two-digit hexadecimal channel number, and the two-digit hexadecimal subaddress.
VOLUME=	specifies the name of the volume
IMAGE=	specifies the pathname of the file containing the system image. If not specified and J.VFMT is activated from a disc volume, the volume is not bootable. This option must not be specified when J.VFMT is activated from a SDT.
CONFIRM=	specifies if confirmation is to be given before any replacement action is taken. If Y is specified, operator response is required to the confirmation prompt. If not specified, confirmation is not given. This option is valid only in the interactive mode.
BOOTFILE=	specifies the pathname of the file containing the nonstandard bootstrap. The nonstandard bootstrap should be in absolute load module format. It cannot be larger than seven 192-word blocks in length including the load module preamble.

13.6 Volume Formatter Directive Parameters

The Volume Formatter supports three directive parameters: DEVICE, VOLUME, and BOOTFILE. Valid abbreviations are shown by underlining.

13.6.1 DEVICE Parameter

The DEVICE parameter specifies the disc device whose medium is to be formatted.

Syntax:

```
DEVICE=mmccss
```

mm	is a two-character mnemonic, i.e., DM (Moving Head Disc), FL (Floppy Disc), or DF (Fixed Head Disc)
cc	is the two-digit hexadecimal channel number
ss	is the two-digit hexadecimal subaddress

13.6.2 VOLUME Parameter

The VOLUME parameter specifies the volume name. The name is recorded in the volume descriptor.

Syntax:

```
    VOLUME=volname
```

volname   is a 1- to 16-character volume name

13.6.3 BOOTFILE Parameter

The BOOTFILE parameter specifies the pathname of a file containing a nonstandard bootstrap. The file should be in absolute load module format and cannot be larger than seven 192-word blocks in length (including the load module preamble).

Syntax:

```
    BOOTFILE=pathname
```

pathname   is the pathname of the file containing the nonstandard bootstrap

13.7 Volume Formatter Directive Options

The following directive options are supported by the Volume Formatter. Valid abbreviations are shown by underlining.

Option	Description
ACCESS	Specifies protection attributes which apply to the root directory of the volume being formatted
ALOC	Specifies the number of sectors per allocation unit
CONFIRM	Indicates if parameters are to be listed before any action is taken
DETACH	Indicates if J.VFMT is to be detached from the terminal, and control transferred to TSM, before completing the formatting process
ECC	Allocates correctable ECC data errors for user's use
IMAGE	Specifies the pathname of the file containing the system image to be written to the volume
ISIZE	Specifies the number of blocks allocated for the system image
MAXAU	Specifies the maximum number of allocation units to be used on the volume
MAXRES	Specifies the maximum number of resource descriptors on the volume

Option	Description
MAXROOT	Specifies the maximum number of root directory entries on the volume
OWNER	Specifies the volume owner name
PROJECTGROUP	Specifies the project group name

13.7.1 ACCESS Option

The ACCESS option specifies access/protection attributes that apply to the root directory of the volume being formatted. There are three catagories of users: OWNER, PROJECTGROUP, and OTHER. For every attribute specified, access is enabled. If the directive is input more than once, the old directive parameters are ignored and the new parameters are in effect.

Syntax:

```
ACCESS= { OWNER        } ([READ] [ADD] [DELENT] [TRAVERSE])
        { PROJECTGROUP }
        { OTHER        }
```

OWNER	specifies what access rights the owner has to the root directory
PROJECTGROUP	specifies what access rights the project group has to the root directory
OTHER	specifies what access rights others have to the root directory
READ	enables read access to the root directory
ADD	enables add access to entries in the root directory
DELENT	enables delete access to entries in the root directory
TRAVERSE	enables traverse access to the root directory

Note: If no parameters are specified, the system-defined defaults are used. A contiguous open and close parenthesis, such as ACCESS=( ), indicates no access is possible.

13.7.2 ALOC Option

The ALOC option specifies the number of sectors represented by one allocation unit. The size of the disc allocation bit map and the minimum file size (in blocks) are dependent upon this parameter. Any disc space allocation request is rounded to a multiple of this parameter.

Syntax:

```
ALOC=n
```

n   is a decimal number. If not specified or equal to zero, the system default is used. The system default is based upon the capacity of the disc as listed in Table 13-2.

Any allocation unit size that does not equal a factor of 20 (1,2,4,5,10,20) is forced to the next higher factor. If the unit size is greater than 20, it is forced to 20.

13.7.3 CONFIRM Option

The CONFIRM option displays all parameters before any formatting actions (allocations, writes, etc.) are taken. This option is valid only in the interactive mode. A Boolean flag is set/reset based on the Boolean parameter input for this option. Once this flag is turned on, formatting cannot proceed until the flag is turned off. This option allows the user to interactively manipulate the formatting parameters until they all have been satisfactorily specified and then proceed with the actual formatting of the disc.

Syntax:

```
CONFIRM=bool
```

bool   is one of the Boolean parameters: true, false, yes, no, on, or off. If true, yes, or on is specified, operator response is required to the confirmation prompt. If not specified, no confirmations are made.

13.7.4 DETACH Option

The DETACH option displays all parameters before any formatting action is taken. The Volume Formatter is then detached from the terminal and control is returned to TSM. The formatting process proceeds normally. Upon completion, a message is displayed on the terminal if the user is logged on the system.

This option is valid only in the interactive mode. A Boolean flag is set/reset based on the Boolean parameter input for this option.

Syntax:

```
DETACH=bool
```

bool   is one of the Boolean parameters: true, false, yes, no, on, or off. If not specified, terminal control is not transferred to TSM until the formatting process completes.

13.7.5 ECC Option

The ECC option allocates all correctable ECC data errors for the user's use.

Syntax:

```
ECC=bool
```

bool   is one of the boolean parameters: true, false, yes, no, on, or off. If not specified, correctable ECC data errors are deallocated as bad blocks.

13.7.6 IMAGE Option

The IMAGE option specifies the disc file containing the system image that is to be written to the volume. If a formatted volume does not contain a system image defined in the volume descriptor, the volume is not bootable. If a volume contains a system image defined in the volume descriptor, the volume becomes the system volume if IPL'd from. Otherwise, it is a user volume.

When J.VFMT is activated from an SDT, this option must not be specified.

Syntax:

    IMAGE=pathname

pathname specifies the pathname of the file containing the system image. If not specified, the volume is not bootable.

13.7.7 ISIZE Option

The ISIZE option establishes the number of blocks allocated for the system image.

Syntax:

    ISIZE=n

n is a decimal number. If not specified, the default is 400.

13.7.8 MAXAU Option

The MAXAU option specifies the number of allocation units used on the disc. The number is dependent upon the parameter specified for the ALOC option (or default granularity if ALOC is not specified). The numeric parameter specified for MAXAU may be less than the actual number of allocation units physically available.

Syntax:

    MAXAU=n

n is a decimal number. If not specified, the SYSGEN-defined default, (SJ.STBLS), is used.

13.7.9 MAXRES Option

The MAXRES option specifies the number of resource descriptors for which to reserve disc space. Every file defined on the volume requires one resource descriptor.

Syntax:

    MAXRES=n

n is a decimal number. If not specified, the default for memory discs or floppy discs is 70. All other devices default to 1000.

13.7.10 MAXROOT Option

The MAXROOT option specifies the number of entries in the root directory to reserve disc space. This number can be modified in order to accommodate various hashing algorithms.

Syntax:

MAXROOT=n

n is a decimal number. If not specified, the default is 100.

13.7.11 OWNER Option

The OWNER option specifies the owner name of the volume. The owner name is recorded in the volume descriptor.

Syntax:

OWNER=name

name is an owner name. If not specified, the default is the owner name associated with the J.VFMT task.

13.7.12 PROJECTGROUP Option

The PROJECTGROUP option establishes the project group name associated with the volume. The project group name is recorded in the volume descriptor.

Syntax:

PROJECTGROUP=name

name is a project group name. If not specified, the default is the project group name associated with the J.VFMT task.

13.8 Errors

The following error messages are issued by the Volume Formatter:

Code	Description
VF01	Error has occurred. See SLO file for description.
VF02	OPEN failure on audit trail device/file
VF03	EOF/EOM on audit trail device/file
VF04	I/O error on audit trail device/file

13.9 Examples

```
FORMAT DEV=DM0800 VOL=MASTER MAXRE=4000
```

This example formats a disc using the name MASTER. It is located on device DM0800, and space for 4000 resource descriptors is reserved.

```
NEWBOOT DEV=DM0800 VOL=NEW-
BOOTFILE=@SYSTEM(USER1)BOOTSTRAP-
CONF=Y
```

This example writes the nonstandard bootstrap specified by the BOOTFILE parameter to volume NEW located on device DM0800. All parameters will be displayed. Confirmation will have to be reset to false (i.e., CONF=N) to continue the formatting process.

```
REPLACE DEV=DM0800 VOL=MASTER IMA=@SYSTEM(USER1)BOOT
```

This example changes the system image on volume MASTER located on device DM0800 to the system image defined in file BOOT located on volume SYSTEM in directory USER1.

```
FORMAT DEV=DM0800 VOL=MASTER-
ACC=OW (R A D T)-
ACC=PR (R A )-
ACC=OT (R)
```

This example formats a disc using the name MASTER. It is located on device DM0800, the owner has complete access to the volume, the project group has read and add access to the volume, and other has read access to the volume.

Gould Inc., Computer Systems Division
6901 W. Sunrise Blvd.
P. O. Box 409148
Fort Lauderdale, FL 33340-9148
Telephone (305) 587-2900

GOULD
Electronics

Users Group Membership Application

USER ORGANIZATION: ______________________________

REPRESENTATIVE(S): ______________________________

____

____

ADDRESS: ______________________________

____

TELEX NUMBER: ______________ PHONE NUMBER: ______________

NUMBER AND TYPE OF GOULD CSD COMPUTERS: ______________________________

____

OPERATING SYSTEM AND REV. LEVEL: ______________________________

____

APPLICATIONS (Please Indicate)

1. EDP
   - A. Inventory Control
   - B. Engineering & Production Data Control
   - C. Large Machine Off-Load
   - D. Remote Batch Terminal
   - E. Other

2. Communications
   - A. Telephone System Monitoring
   - B. Front End Processors
   - C. Message Switching
   - D. Other

3. Design & Drafting
   - A. Electrical
   - B. Mechanical
   - C. Architectural
   - D. Cartography
   - E. Image Processing
   - F. Other

4. Industrial Automation
   - A. Continuous Process Control Op.
   - B. Production Scheduling & Control
   - C. Process Planning
   - D. Numerical Control
   - E. Other

5. Laboratory and Computational
   - A. Seismic
   - B. Scientific Calculation
   - C. Experiment Monitoring
   - D. Mathematical Modeling
   - E. Signal Processing
   - F. Other

6. Energy Monitoring & Control
   - A. Power Generation
   - B. Power Distribution
   - C. Environmental Control
   - D. Meter Monitoring
   - E. Other

7. Simulation
   - A. Flight Simulators
   - B. Power Plant Simulators
   - C. Electronic Warfare
   - D. Other

8. Other

Please return to:

Users Group Representative

Date: ______________

243-06-1 (1/86)

Gould Inc., Computer Systems Division Users Group. . .

The purpose of the Gould CSD Users Group is to help create better User/User and User/Gould CSD communications.

There is no fee to join the Users Group. Simply complete the Membership Application on the reverse side and mail to the Users Group Representative. You will automatically receive Users Group Newsletters, Referral Guide and other pertinent Users Group activity information.

Fold and Staple for Mailing

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 947 FT. LAUDERDALE, FL

POSTAGE WILL BE PAID BY ADDRESSEE

GOULD INC., COMPUTER SYSTEMS DIVISION
ATTENTION: USERS GROUP REPRESENTATIVE
6901 W. SUNRISE BLVD.
P. O. BOX 409148
FT. LAUDERDALE FL 33340-9970

Fold and Staple for Mailing

(Detach Here)

GOULD
Electronics